

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ**

*Мінухін С. В.*  
*Беседовський О. М.*  
*Знахур С. В.*

**МЕТОДИ І МОДЕЛІ ПРОЕКТУВАННЯ**  
**НА ОСНОВІ СУЧАСНИХ CASE-ЗАСОБІВ**

**Навчальний посібник**

**Харків. Вид. ХНЕУ, 2008**

УДК004.94(075.8)  
ББК32.973 – 01я73  
М62

Рецензенти: докт. техн. наук, професор кафедри штучного інтелекту Харківського національного університету радіоелектроніки *Бодянский Є. В.*; докт. техн. наук, професор кафедри інформаційних управляючих систем Харківського національного університету радіоелектроніки *Авраменко В. П.*

Рекомендовано до видання рішенням вченої ради Харківського національного економічного університету.

Протокол №7 від 29.02.2008 р.

**Рекомендовано Міністерством освіти і науки України  
як навчальний посібник для студентів вищих навчальних закладів  
(лист №1.4/18-Г-2267 від 31.10.2008 р.)**

**Мінухін С. В.**

М62      Методи і моделі проектування на основі сучасних CASE-засобів. Навчальний посібник / С. В. Мінухін, О. М. Беседовський, С. В. Знахур. — Харків: Вид. ХНЕУ, 2008. — 272 с. (Укр. мов.)

Розглянуто широке коло питань, присвячених викладанню та вирішенню завдань проектування інформаційних систем на основі сучасних CASE-засобів. Здійснено теоретико-методологічні засади структурно-функціонального проектування із застосуванням методології SADT, включаючи побудову моделей предметної галузі, основні стандарти структурної методології, аналіз інструментальних засобів для проектування систем на основі структурного підходу. Наведено приклади вирішення практичних завдань проектування окремих комплексів задач у пакетах BPWin та Business Studio.

Рекомендовано для студентів напряму підготовки "Комп'ютерні науки".

**ISBN 978-966-676-301-6**

УДК 004.94(075.8)  
ББК 32.973 – 01я73

© Харківський національний економічний університет, 2008  
© Мінухін С. В.  
Беседовський О. М.  
Знахур С. В.  
2008

## ВСТУП

Для того щоб успішно виконати проект, об'єкт проектування повинен бути, перш за все, правильно й адекватно описаний, тобто необхідно побудувати повноцінні та функціональні інформаційні моделі об'єкта проектування. Донедавна проектування інформаційних систем виконувалося головним чином на інтуїтивному рівні із застосуванням неформалізованих методів, які ґрунтувалися б на практичному досвіді, експертних оцінках і дорогих експериментальних перевірках якості функціонування подібних систем. Але природно, що під час розробки і функціонування інформаційних систем, потреби користувачів можуть змінюватися або уточнюватися, що ще більш ускладнює розробку та супровід.

У 1970 – 80-х роках при розробці інформаційних систем широко застосовувалася структурна методологія, що надає в розпорядження розробників суворі формалізовані методи опису ІС і схвалюваних технічних рішень. Ця методологія ґрунтувалася на наочній графічній техніці, інакше кажучи, для опису проекту використовувалися різного роду схеми і діаграми. Наочність і суворість засобів структурного аналізу дозволяла розробникам і майбутнім користувачам системи із самого початку неформально брати участь у її створенні, обговорювати й закріплювати розуміння основних технічних рішень. Проте широке застосування цієї методології та проходження її рекомендаціям при розробці конкретних проектів зустрічалося достатньо рідко, оскільки її практично неможливо реалізувати на належному рівні ручним, неавтоматизованим способом. Уручну дуже важко розробити і графічно представити суворі формальні специфікації системи, перевірити їх на повноту й несуперечність і тим більше змінити.

Якщо учасники проекту намагалися вдатися до ручної розробки, то перед ними виникали наступні проблеми:

- неадекватна специфікація вимог;
- нездатність виявляти помилки у проектних рішеннях;
- низька якість документації, що знижує експлуатаційні якості;
- затяжний цикл і незадовільні результати тестування.

Повинні були з'явитися спеціалізовані програмно-технологічні засоби для розробки проектів, зокрема, заснованих на інформатизації. Ними стали засоби, що реалізують CASE-технологію створення та

супроводу інформаційних систем. Термін CASE (Computer-Aided Software Engineering) сьогодні розуміється досить широко.

Первинне значення терміну, обмежене питаннями автоматизації розробки програмного забезпечення (ПЗ), в даний час придбало новий сенс, і тепер це поняття охоплює процес розробки складних інформаційних систем у цілому. Тепер під терміном CASE-засобу розуміються програмні засоби, що підтримують процеси створення й супроводу подібних систем, включаючи аналіз і формулювання вимог, проектування прикладного ПЗ (додатків) і баз даних, генерацію коду, тестування, документування, забезпечення якості, конфігураційне управління та управління проектом і т. д. CASE-засоби разом із системним ПЗ і технічними засобами утворюють повне середовище розробки.

Активні дослідження у сфері методології програмування привели до того, що програмування набуло рис системного підходу з розробкою і впровадженням мов високого рівня, методів структурного і модульного програмування, мов проектування і засобів їх підтримки, формальних і неформальних мов описів системних вимог і специфікацій і т. д.

Крім того, появі CASE-технології сприяли і такі чинники, як:

підготовка аналітиків і програмістів, сприйнятливих до концепцій модульного і структурного програмування;

широке впровадження і постійне зростання продуктивності комп'ютерів, що дозволили використовувати ефективні графічні засоби і автоматизувати більшість етапів проектування;

впровадження мережної технології, що надала можливість об'єднання зусиль окремих виконавців у єдиний процес проектування шляхом використання бази даних, що розділялася, містить необхідну інформацію про проект.

Таким чином, CASE-технологія є методологією проектування ІС, а також набір інструментальних засобів, що дозволяють у наочній формі моделювати наочну область, аналізувати цю модель на всіх етапах розробки і супроводу ІС і розробляти додатки відповідно до потреб користувачів. Велика частка CASE-засобів використовує методологію структурного (в основному) або орієнтованого аналізу і проектування, що використовують специфікації у вигляді діаграм або текстів для опису зовнішніх вимог, зв'язків між моделями системи, динаміки поведінки системи і архітектури програмних засобів.

Згідно з західними дослідженнями CASE-технологія потрапила в розряд найбільш стабільних інформаційних технологій.

Разом з тим необхідно зазначити наступне:

CASE-засоби не обов'язково дають негайний ефект; він може бути отриманий тільки через деякий проміжок часу;

реальні витрати на впровадження CASE-засобів зазвичай набагато перевищують витрати на їх придбання;

CASE-засоби забезпечують можливості для отримання істотної вигоди тільки після успішного завершення процесу їх впровадження.

Сучасні CASE-засоби охоплюють значну область підтримки чисельних технологій проектування інформаційних систем – від простих засобів аналізу та документування до повномасштабних засобів автоматизації, що покривають важливий життєвий цикл ПЗ.

Найбільш трудомісткими етапами розробки інформаційних систем є аналіз і проектування, в процесі яких CASE-засоби забезпечують якість схвалюваних технічних рішень і підготовку проектної документації. При цьому велику роль відіграють методи візуального представлення інформації. Вони припускає побудову структурних або інших діаграм в реальному масштабі часу, використання багатообразної колірної палітри, наскрізну перевірку синтаксичних правил. Графічні засоби моделювання дозволяють розробникам в наочному вигляді вивчати існуючу інформаційну систему, перебудувувати її відповідно до поставлених цілей і наявних обмежень.

У CASE-засоби потрапляють як досить дешеві системи для персональних комп'ютерів з обмеженими можливостями, так і дорогі системи для неоднорідних обчислювальних платформ і операційних середовищ. Так, сучасний ринок програмних засобів налічує близько 300 різних CASE-засобів, найбільш могутні з яких використовуються практично всіма провідними західними компаніями.

Зазвичай до CASE-засобів відносять будь-який програмний засіб, що автоматизує ту або іншу сукупність процесів життєвого циклу ПЗ і що володіє наступними особливостями:

графічні засоби для опису та документування ІС, що забезпечують зручний інтерфейс з розробником і розвивають його творчі можливості;

інтеграція окремих компонент CASE-засобів, що забезпечує керованість процесом розробки інформаційної системи;

використання спеціальним чином організованого сховища проектних метаданих (репозиторію).

Інтегрований CASE-засіб (або комплекс засобів, що підтримують повний життєвий цикл ПЗ) містить наступні компоненти:

репозиторій, що є основою CASE-засобу. Він повинен забезпечувати зберігання версій проекту та його окремих компонентів, синхронізацію надходження інформації від різних розробників при груповій розробці, контроль метаданих на повноту й несуперечність;

графічні засоби аналізу і проектування, які забезпечують створення та редагування ієрархічно пов'язаних діаграм (DFD, ERD і ін.), що створюють моделі інформаційної системи;

засоби розробки додатків, включаючи мови 4GL і генератори кодів;

засоби конфігураційного управління;

засоби документування;

засоби тестування;

засоби управління проектом;

засоби реінжинірингу.

Усі сучасні CASE-засоби можна класифікувати за типами й категоріями. Класифікація за типами відображає функціональну орієнтацію CASE-засобів на ті або інші процеси життєвого циклу. Класифікація за категоріями визначає ступінь інтегрованості за виконуваними функціями та включає окремі локальні засоби, вирішальні невеликі автономні завдання (tools), набір частково інтегрованих засобів, що охоплюють більшість етапів життєвого циклу інформаційних систем (toolkit) і повністю інтегровані засоби, що підтримують важливий життєвий цикл інформаційних систем і пов'язані загальним репозиторієм.

Крім цього CASE-засобу можна класифікувати за використовуваними методологіями і моделями систем і БД, ступені інтегрованості з СКБД.

Класифікація за типами в основному збігається з компонентним складом CASE-засобів і включає:

засоби аналізу (Upper CASE) – призначені для побудови й аналізу моделей наочної області (Design/IDEF (Meta Software), BPwin (Logic Works));

засоби аналізу і проектування (Middle CASE) – підтримують найбільш поширені методології проектування використовні для створення проектних специфікацій (Vantage Team Builder (Cayenne), Designer/2000 (ORACLE), Silverrun (CSA), PRO-IV (McDonnell Douglas),

CASE. Аналітик (Макропроджект)). Виходом застосування таких засобів є специфікації компонентів і інтерфейсів системи, архітектура системи, алгоритмів і структур даних;

засоби проектування баз даних, що забезпечують моделювання даних і генерацію схем баз даних (як правило, на мові SQL) для найбільш поширених СКБД. До них відносяться ERwin (Logic Works), S-Designor (SDP) і DataBase Designer (ORACLE). Засоби проектування баз даних є також у складі CASE-засобів Vantage Team Builder, Designer/2000, Silverrun і PRO-IV;

засоби розроблення додатків. До них відносяться: засоби 4GL (Uniface (Compuware), JAM (JYACC), PowerBuilder (Sybase), Developer/2000 (ORACLE), New Era (Informix), SQL Windows (Gupta), Delphi (Borland) і ін.) і генератори кодів, що входять до складу Vantage Team Builder, PRO-IV і частково – в Silverrun;

засоби реінжинірингу, що забезпечують аналіз програмних кодів і схем баз даних і формування на їх основі різних моделей і проектних специфікацій. Засоби аналізу схем БД і формування ERD входять до складу Vantage Team Builder, PRO-IV, Silverrun, Designer/2000, ERwin і S-Designor. В області аналізу програмних кодів найбільшого поширення набувають об'єктно-орієнтовані CASE-засоби, що забезпечують реінжиніринг програм на мові C++ (Rational Rose (Rational Software), Object Team (Cayenne)).

Допоміжні типи включають:

засоби планування та управління проектом (SE Companion, Microsoft Project і ін.);

засоби конфігураційного управління (PVCS (Intersolv));

засоби тестування (Quality Works (Segue Software));

засоби документування (SODA (Rational Software)).

Ураховуючи різноманітну природу CASE-засобів, було б помилково робити які-небудь твердження щодо реального задоволення тих або інших очікувань від їх упровадження.

Можна виділити наступні чинники, що ускладнюють визначення можливого ефекту від використання CASE-засобів:

широка різноманітність якості та можливостей CASE-засобів;

досить невеликий час використання CASE-засобів у різних організаціях і недолік досвіду їх застосування;

широка різноманітність у практиці впровадження різних організацій;

відсутність детальних метрик і даних для вже виконаних і поточних проектів;

широкий діапазон наочних областей проектів;

різний ступінь інтеграції CASE-засобів у різних проектах.

Для успішного впровадження CASE-засобів організація повинна володіти наступними якостями:

*технологія.* Розуміння обмеженості існуючих можливостей і здатність прийняти нову технологію;

*культура.* Готовність до впровадження нових процесів і взаємин між розробниками і користувачами;

*управління.* Чітке керівництво і організованість по відношенню до найбільш важливих етапів і процесів впровадження.

Для того щоб ухвалити зважене рішення щодо інвестицій в CASE-технологію, користувачі змушені проводити оцінку окремих CASE-засобів, спираючись на неповні та суперечливі дані. Ця проблема часто посилюється недостатнім знанням всіх можливих "підводних каменів" використання CASE-засобів.

Серед найбільш важливих проблем виділяють наступні:

достовірна оцінка віддачі від інвестицій в CASE-засоби скрутна, зважаючи на відсутність прийнятних метрик і даних за проектами і процесами розробки ПЗ; впровадження CASE-засобів може представляти тривалий процес і не принести негайної віддачі. Можливо навіть короткострокове зниження продуктивності в результаті зусиль, що витрачаються на впровадження;

відсутність повної відповідності між тими процесами та методами, які підтримуються CASE-засобами, і тими, що використовуються в даній організації, може привести до додаткових труднощів;

CASE-засоби часто важко використовувати в комплексі з іншими подібними засобами. Це пояснюється як різними парадигмами, підтримуваними різноманітними засобами, так і проблемами передачі даних і управління від одного засобу до іншого;

деякі CASE-засоби вимагають дуже багато зусиль для того, щоб виправдати їх використання в невеликому проекті, проте, можна отримати вигоду з тієї дисципліни, до якої зобов'язує їх застосування;

негативне відношення персоналу до впровадження нової CASE-технології може бути головною причиною провалу проекту.



Користувачі CASE-засобів повинні бути готові до необхідності довгострокових витрат на експлуатацію, частій появі нових версій і можливому швидкому моральному старінню засобів, а також постійним витратам на навчання і підвищення кваліфікації персоналу.

Грамотне, продумане і ґрунтовне використання CASE-технології здатне принести наступні переваги:

високий рівень технологічної підтримки процесів розробки та супроводу ПО;

позитивна дія на деякі або всі з перерахованих чинників: продуктивність, якість продукції, дотримання стандартів, документування;

прийнятний рівень віддачі від інвестицій у CASE-засоби.

## **Розділ 1**

### **Принципи структурно-функціонального проектування**

#### **1.1. Загальні принципи структурних методів і структурного аналізу**

Структурні методи є дисципліною системного аналізу й проектування, тобто присвячені аналізу складних систем і процесів [1].

Методи структурного аналізу й проектування визначають аналіз складності великих систем шляхом розчленовування наступним чином:

декомпозиції їх на частини (так звані "чорні скриньки");

ієрархічної організації цих частин.

Вигода при використанні чорних скриньок полягає в тому, що їх користувачам не потрібно знати, як вони працюють, а необхідно знати лише основні характеристики – так звані *входи й виходи*, а також їх призначення – *функцію*, яку вони виконують.

Отже, можна виділити наступні етапи спрощення складної системи чи об'єкта управління:

1. *Розбивка на чорні скриньки*. При цьому цей процес повинен задовольняти наступним критеріям:

кожна чорна скринька повинна реалізовувати одну функцію системи;

функція кожної чорної скриньки повинна бути зрозуміла незалежно від складності її реалізації;

зв'язок між чорними скриньками повинен вводитися тільки за наявності зв'язку між відповідними функціями системи;

зв'язки між чорними скриньками повинні бути простими, наскільки це можливо для забезпечення незалежності між ними.

*2. Ієрархія цих частин.* Для аналізу складної системи недостатньо розбивки її на частини: необхідно ці частини організувати певним чином, а саме – у вигляді множини ієрархічних структур.

*3. Використання графічних нотацій.* Аналіз вимог розроблювальної системи є найважливішим серед усіх етапів життєвого циклу (ЖЦ). Він впливає на всі наступні етапи, будучи в той же час найменш вивченим і зрозумілим процесом.

Проблеми, перед якими стає системний аналітик, взаємозалежні та характеризуються наступними твердженнями:

аналітикові складно одержати вичерпну інформацію для оцінки вимог до системи з погляду замовника;

замовник, у свою чергу, не має достатньої інформації щодо проблем обробки даних для того, щоб говорити про те, що є здійсненим, а що ні;

аналітик стикається з надмірною кількістю докладних відомостей як про предметну галузь, так і про нову систему;

специфікація системи не зрозуміла для замовника через обсяг технічних термінів;

у випадку зрозумілості специфікації для замовника вона буде недостатньо зрозумілою для проєктувальників і програмістів, які створюють систему.

*Структурним аналізом* прийнято називати такий метод дослідження системи, що починається з її загального огляду й потім деталізується, здобуваючи ієрархічну структуру із все більшим числом рівнів. Для таких методів характерна розбивка на рівні абстракції з обмеженням кількості елементів на кожному з рівнів (від 3 до 6 – 7); обмежений контекст, що включає лише істотні на кожному рівні деталі; дуальність даних і операцій над ними; використання суворих формальних правил запису; послідовне наближення до кінцевого результату.

Усі методології структурного аналізу базуються на ряді загальних принципів, частина з яких регламентує організацію робіт на початкових етапах ЖЦ системи або об'єкта управління, а частина використовується під час вироблення рекомендацій щодо організації робіт.

Використовуються два *основних* (базових) принципи: принцип "поділяй і пануй"; принцип ієрархічного упорядкування.

Перший із них є принципом вирішення складних проблем шляхом розбивки їх на безліч менших незалежних завдань, більш легких для їх розуміння й вирішення. Другий принцип визначає, що легше зрозуміти проблему, якщо вона розбита на частини, та декларує, що побудова цих частин та отримані результати також важливі для розуміння.

До *не основних принципів* структурного аналізу відносяться:

1) принцип абстрагування – полягає у виділенні істотних із деяких позицій аспектів системи й відволікання від несуттєвих з метою подання проблеми в простому загальному вигляді;

2) принцип формалізації – полягає в необхідності суворого методичного підходу до вирішення проблеми;

3) принцип приховування – полягає у приховуванні несуттєвої на конкретному етапі інформації: кожна декомпозована частина "знає" тільки необхідну їй інформацію;

4) принцип концептуальної спільності – полягає у використанні єдиної філософії на всіх етапах ЖЦ (структурний аналіз – структурне проектування – структурне програмування – структурне тестування);

5) принцип повноти – полягає в контролі на присутність зайвих елементів;

6) принцип несуперечливості – полягає в обґрунтованості погодженості елементів;

7) принцип логічної незалежності – полягає в концентрації уваги на логічному проектуванні для забезпечення незалежності від фізичного проектування;

8) принцип незалежності даних – полягає в тому, що моделі даних повинні бути проаналізовані й спроектовані незалежно від процесів їх логічної обробки, а також від їх фізичної структури й розподілу;

9) принцип структурування даних – полягає в тому, що дані повинні бути структуровані й ієрархічно організовані;

10) принцип доступу кінцевого користувача – полягає в тому, що користувач повинен мати засоби доступу до бази даних, які він може використати безпосередньо (без програмування).

Для цілей моделювання систем та об'єктів взагалі й використання структурного аналізу зокрема, використовуються три групи засобів, що визначають наступні компоненти:

*функції, які система повинна виконувати;*

*відносини між даними;*

*залежне від часу поведження системи (поведінковий аспект).*

Серед засобів вирішення цих завдань у методологіях структурного аналізу найбільш часто й ефективно застосовуваними є наступні [1]:

**DFD** (Data Flow Diagrams) – *діаграми потоків даних;*

**ERD** (Entity-Relationship Diagrams) – *діаграми "сутність – зв'язок";*

**STD** (State Transition Diagrams) – *діаграми переходів станів.*

Логічна DFD виконує наступні функції:

показує зовнішні (стосовно системи) джерела й стоки даних;

ідентифікує логічні функції (процеси) і групи елементів даних;

з'єднує одну функцію з іншими (за допомогою потоків);

ідентифікує сховища (накопичувачі) даних, до яких користувачами системи здійснюється доступ.

Структури потоків даних і визначення їх компонентів зберігаються й аналізуються у словнику даних. Зміст кожного сховища також зберігають у словнику даних, модель даних сховища розкривається за допомогою ERD. У випадку наявності реального часу засоби DFD доповнюються засобами опису залежного від часу поведження системи, що визначається на основі діаграм переходів станів – STD (рис. 1.1).

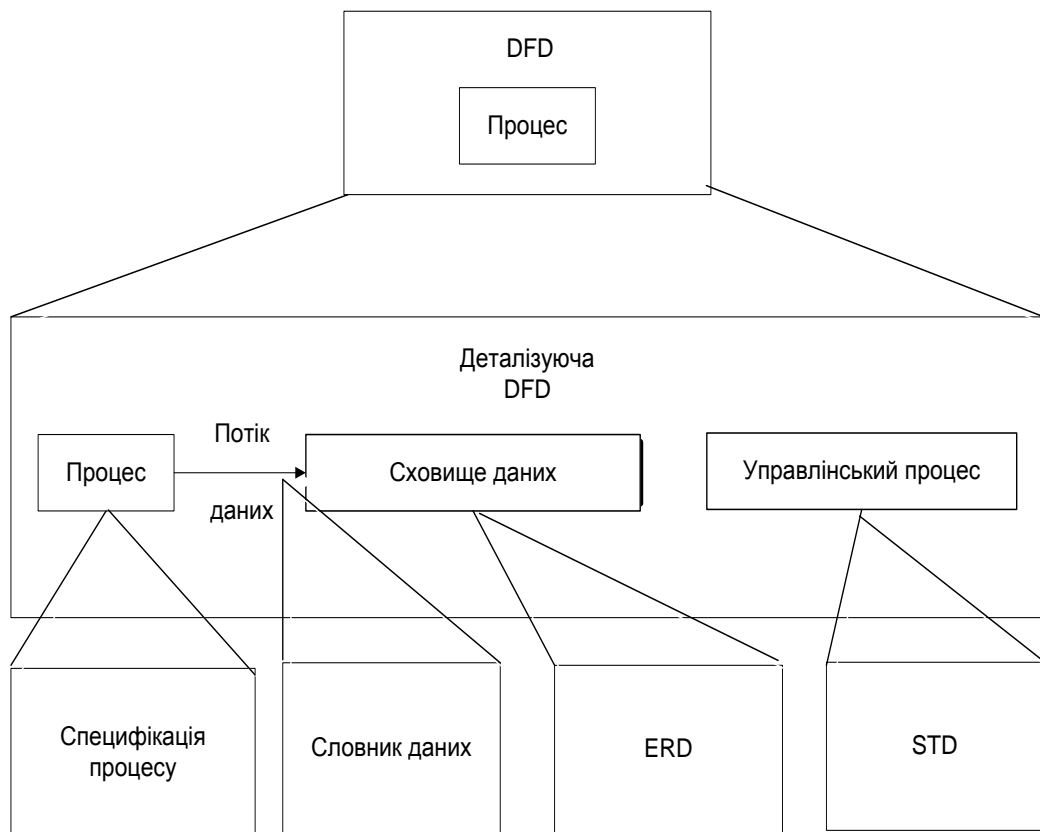


Рис. 1.1. Схема взаємодії діаграм у структурному аналізі

## 1.2. Засади структурно-функціональної методології

Методологія SADT розроблена Д. Россом і одержала подальший розвиток у відомій методології IDEF0 (Icam DEFINITION). Методологія SADT – це сукупність методів, правил і процедур, призначених для побудови *функціональної моделі* об'єкта чи системи. Функціональна модель SADT відображає функціональну структуру об'єкта, тобто виконані ними функції, дії і зв'язки між ними [1].

Основні елементи даної методології ґрунтуються на *наступних положеннях*:

1. *Графічне подання блокового моделювання*. Графіка блоків і дуг SADT-діаграми відображає функцію у вигляді блоку, а інтерфейси входу/виходу зображуються дугами, що відповідно входять у блок і виходять з нього. Взаємодія блоків один з одним описується за допомогою інтерфейсних дуг, котрі виражають "обмеження" блоків, які у свою чергу визначають: *коли, в якій послідовності і яким чином* функції виконуються й управляються.

2. *Обмежене деталювання.* Кількість рівнів декомпозиції повинна бути достатньою для досягнення мети розбудови проблеми чи завдання.

3. *Суворість і точність.* Виконання правил SADT вимагає достатньої суворості й точності, не накладаючи в той же час надмірних обмежень на дії аналітика.

*Правила SADT включають:*

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

зв'язаність діаграм за допомогою нумерації блоків;

унікальність міток і найменувань (відсутність повторюваних імен);

синтаксичні правила для графічних об'єктів (блоків і дуг);

поділ входів і управлінських дуг (правило визначення ролі даних на елементах діаграми);

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

Методологія SADT може використовуватися для моделювання широкого кола систем і визначення вимог і функцій, а потім для розробки системи, що задовольняє цим вимогам і реалізує ці функції. Для вже існуючих систем методологія SADT може бути використана для аналізу функцій, що виконуються системою, а також для визначення механізмів, за допомогою яких вони здійснюються.

### *Склад функціональної моделі*

Результатом застосування методології SADT є модель, яка складається з діаграм, фрагментів текстів і глосарію, що мають посилання один на одного. Діаграми – головні компоненти моделі, на якій показані всі функції ІС і інтерфейси за допомогою блоків і дуг. Місце з'єднання дуги із блоком визначає тип інтерфейсу. Керівна інформація входить у блок зверху, у той час як інформація, що піддається обробці, показана з лівого боку блоку, а результати виходу показані із правого боку. Механізм (людина, автоматизована система, засіб), що здійснює операцію, зображується дугою, що входить у блок знизу (рис. 1.2).

Однією з найбільш важливих особливостей методології SADT є поступове введення все більших рівнів деталізації в міру створення діаграм, що відображають структуру моделі [13].

На рис. 1.3 зображена базова структура SADT-моделі, компоненти якої можуть бути декомповані на іншій діаграмі (діаграмах) – діаграмі нижчого рівня декомпозиції. Кожна діаграма ілюструє "внутрішню побудову", складові блоку на батьківській діаграмі.

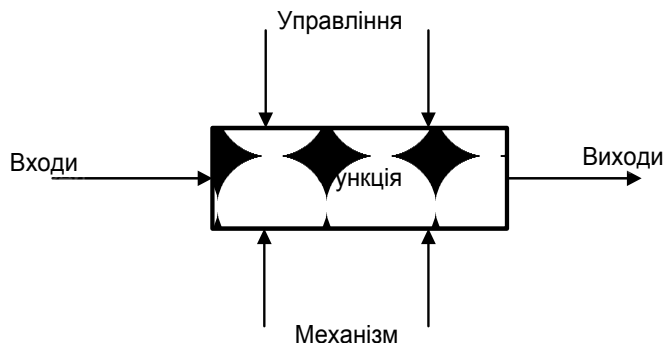


Рис. 1.2. Функціональний блок і інтерфейсні дуги

### *Ієрархія діаграм*

Побудова SADT-моделі починається з подання всієї системи у вигляді найпростішого компонента – одного блоку й дуг, що зображують інтерфейси з функціями поза системою, який має назву *контекстного*. Оскільки єдиний блок відображає всю систему як єдине ціле, ім'я, зазначене в блоці, є загальним. Це визначається й для інтерфейсних дуг – вони також показують *повний набір* зовнішніх інтерфейсів системи в цілому.

Потім блок, що зображує систему як єдиний модуль, деталізується на іншій діаграмі за допомогою декількох блоків, з'єднаних між собою інтерфейсними дугами. Ці блоки становлять основні *підфункції* реалізованої функції. Дана декомпозиція виявляє повний набір підфункцій, кожна з яких подана як блок, межі котрого визначені інтерфейсними дугами. Кожна інтерфейсна дуга визначає межу взаємодії з іншими дугами та блоками (підфункціями). Структура SADT декомпована подібним чином для більш детального подання.

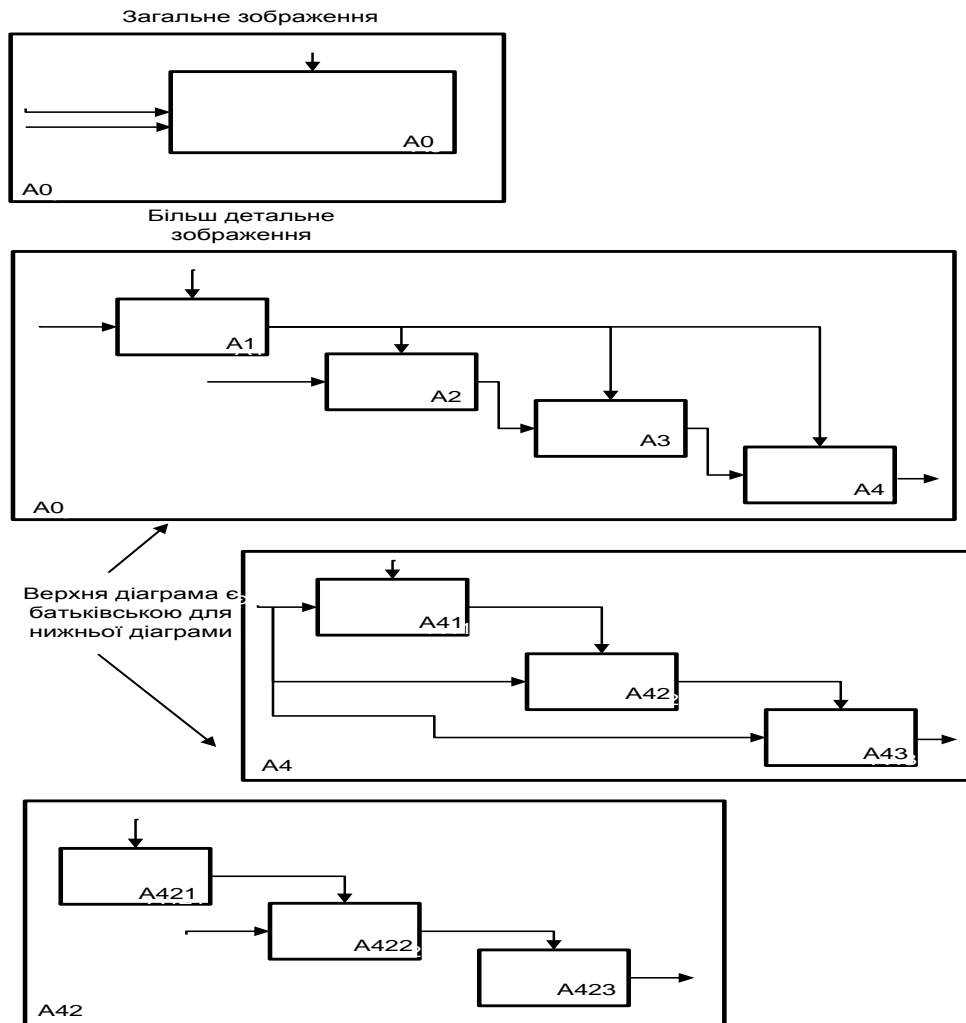


Рис. 1.3. Структура SADT-моделі й декомпозиція діаграм

У всіх випадках кожна підфункція може містити тільки ті елементи, які входять у загальну (результатну) функцію. Крім того, модель не може опустити які-небудь елементи: тобто, як уже зазначалося, батьківський блок і його інтерфейси забезпечують контекст. До нього не можна нічого додати, і з нього не може бути нічого вилучено.

Модель SADT є множиною діаграм із супровідною документацією, що розбивають складний об'єкт на складові частини, які зображені у вигляді блоків. Деталі кожного з основних блоків зображені у вигляді блоків на інших діаграмах. Кожна детальна діаграма є декомпозицією блоку з більш загальної діаграми (діаграми більш високого рівня). На кожному кроці декомпозиції більш загальна діаграма (даного рівня декомпозиції) називається батьківською для більш детальної (декомпозиційної) діаграми.



## Принципи декомпозиції

*Декомпозиція* – це процес створення діаграми, що деталізує певний блок і пов'язані з ним дуги. Результатом її є опис, який є "розламуванням" батьківського блоку на менші та більш приватні функції. Вона включає також *синтез*. Декомпозиція полягає в початковому відокремленні об'єкта на дрібніші частини й подальшому з'єднанні їх у детальніший опис об'єкта. Модель показує тільки результат взаємодії аналізу та синтезу.

За правилами методології SADT проєктувальник проводить спочатку аналіз і синтез системних об'єктів, записуючи, як саме піддалися розбиттю об'єкти, що входять у початковий об'єкт. Список даних починається зі всіх граничних дуг і їх ICOM-кодів і закінчується їх складовими. Потім проєктувальник проглядає список даних і, можливо, об'єднує ці складові, щоб виділити ті об'єкти, які виступатимуть як керівників.

Далі аналітик (проєктувальник) здійснює аналіз і синтез функцій системи, і робить це відповідно до списку даних. У процесі вільного об'єднання і введення нових керівних дуг створюється список функцій для подальшої деталізації. Наприклад, вказівки і креслення змушують проєктувальника: *вибрати функції, вибрати інструменти і підготувати робоче місце*. Потім ці функціональні частини об'єднуються в розумні (збалансовані) набори із 3 – 6 блоків. Важливо зазначити, що застосування правила "*від трьох до шести блоків*" змушує проєктувальника використовувати абстракцію і декомпозицію для поступового представлення деталей системи.

Після визначення функціональних частин список даних і список функцій використовуються для побудови чорнового варіанта діаграми. Знову виконуються аналіз і синтез, внаслідок чого формуються набори об'єктів, які представляються дугами, що сполучають блоки.

Така послідовність виконання декомпозиції має велике значення, оскільки, якщо декомпозиція виконана у такий спосіб, отримані блоки діаграми активізуються головним чином завдяки *дугам управління*. Отже, дуги управління впливають на якість і обґрунтованість результатної декомпозиції. Правило "від трьох до шести" блоків на одній діаграмі – особливість використання методології SADT. Вона визначається

можливістю сприйняття проектувальниками приблизно семи категорій, кожна з яких може містити близько семи окремих одиниць інформації.

Методологія SADT дотримується правила, вирішуючи як верхню межу кількості – шість блоків – поодинці на категорію. Ім'я блоку та його граничні дуги – це одиниці інформації, що поміщаються в категорії в процесі читання діаграми. Таким чином, SADT-діаграми створюються використовуючи раціональність розбиття і подальшого вивчення діаграм. Досвід показує, що діаграми з 4 – 5 блоків з не більше ніж п'ятьма дугами, що стосуються кожного блоку, є оптимальними за об'ємом інформації, які можна найефективніше використовувати.

### Стратегії декомпозиції

У процесі створення діаграм необхідно визначати стратегії декомпозиції.

Часто якнайкращою є *функціональна стратегія* декомпозиції – декомпозиція базується на функціональних взаєминах дій системи, тому що вона примушує проектувальника уважно обдумувати та вирішувати:

*що робить система;  
як вона працює.*

Крім того, у функціональних декомпозиціях віддають перевагу докладному показу необхідних обмежень на функції системи, а не їх послідовності.

*Декомпозиція відповідно до функцій*, які співробітники або організації виконують, використовується для створення системи описів, що фіксує взаємодію між дійовими особами у процесі їх роботи. Іноді взаємодія між функціями невелика, проте, дуже часто взаємозв'язки між функціями досить численні та складні. У цих випадках рекомендується використовувати цю стратегію тільки на початку роботи над моделлю системи, яку називають РЗ, – аббревіатура слів *people* (люди), *paper* (папери), *procedures* (процедури). Це допоможе зібрати початкову інформацію про систему, за допомогою якої можна створити більш обґрунтовану функціональну декомпозицію системи в цілому.

*Декомпозиція відповідно до вже відомих стабільних підсистем*. Вона приводить до створення набору моделей, по одній моделі на кожен підсистему або важливу компоненту. Потім для опису всієї системи

повинна бути побудована складена модель, яка об'єднує всі окремі моделі.

*Декомпозиція за життєвим циклом.* Деякі системи у процесі функціонування безперервно перетворюють свої входи в кінцевий продукт. Стратегія декомпозиції, заснована на відстежуванні циклу, так званого "життєвого циклу", для ключових входів системи є ефективною для опису таких процесів і об'єктів. Цю стратегію рекомендується застосовувати, коли метою системи є поліпшення одного з основних входів і можна визначити послідовні стадії поліпшення цього входу.

*Декомпозиція за фізичним процесом.* Результатом такого гатунку декомпозиції буде виділення функціональних стадій, етапів завершення або кроків виконання. Ця стратегія корисна при описі існуючих процесів (таких, наприклад, як робота промислового підприємства). Її застосування може привести до занадто послідовного опису системи, який повною мірою не враховуватиме обмеження, що диктуються функціями один одному. При цьому може виявитися прихованою послідовність управління. Дана стратегія рекомендується для випадків, коли метою моделі є опис фізичного процесу як такого.

### **Вибір стратегії декомпозиції**

Досвід показує, що розробці якісної діаграми декомпозиції може передувати декілька невдалих спроб. Перша спроба декомпозиції, в результаті якої створюється така діаграма, зазвичай призводить до появи зайвого детальній діаграмі. Ця початкова декомпозиція при детальному розгляді часто не відповідає меті моделі. На початку моделювання важливіша чіткість викладу, ніж його правильність, оскільки колективні знання експертів, проектувальників, замовників допоможуть розробити повноцінний загальний опис, який після деталізації задовольнятиме цілі моделі. Крім того, не дивлячись на адекватність початкової стратегії декомпозиції, відповідної повному життєвому циклу продуктів, всі діаграми моделі піддавалися перегляду для отримання правильного й узгодженого викладу цього конкретного сценарію.

## **Момент припинення процесу декомпозиції**

Постійна, але контрольована зміна діаграм моделі, може бути прикладом інженерної діяльності. Робота в методології SADT є *процесом*, оскільки вона включає послідовне поліпшення опису системи. Процес поліпшення моделі в деякий момент повинен бути припинений. SADT-моделі повинні не просто структурувати систему, а мати заданий ступінь точності їх уявлення. Тому декомпозиція в SADT припиняється, коли діаграми, що створюють нижній рівень моделі, достатньо деталізовані для досягнення мети моделі. Іншими словами, подальша декомпозиція не потрібна, якщо модель достатньо точна, щоб відповідати на всі питання, відповідні її цілі [1, 13].

Наприклад, метою моделі, що описує експериментальний механічний цех, є визначення обов'язків персоналу механічного цеху з тим ступенем подробиць, який достатній для опису цих обов'язків у навчальному керуванні. Це і є точним критерієм для даної моделі: процес моделювання припиняється, коли кожен блок може бути пояснений в одному параграфі тексту. Для цілей навчального керування один абзац пояснень розглядається як достатня точність. Тому, як і в інших прикладних дисциплінах, розробка SADT-моделі припиняється після досягнення необхідного *ступеня точності*.

### **1.3. Вивчення предметної галузі**

#### **Етапи моделювання опису наочної області**

##### ***Опитування***

Перше опитування служить точкою відліку у процесі моделювання. Щоб провести опит, аналітик спочатку вибирає якнайкраще джерело інформації (документ або конкретну людину), а потім організовує його "опитування". Мета опитування – отримання порції інформації, необхідної спершу або для продовження побудови певної частини моделі. Після першого опитування SADT-модель використовується для визначення тієї інформації, яку необхідно отримати в ході наступного опитування. Відповідно до ієрархії моделі може бути проведена послідовність опитувань для з'ясування все конкретніших деталей даної сфери.

## Джерела інформації

Джерелами інформації є експерти. Часто вони є якнайкращими джерелами, тому що їм знайомі поточні нюанси й недокументовані аспекти системи. Найважливіше – це те, що експертам відомі факти, які не відображені в документах або які важко пояснити. Ці факти іноді називають "казуальним знанням". Їх можна отримати тільки шляхом опитування експертів.

Щоб підготуватися до такого опитування, необхідно досліджувати інші джерела інформації, наприклад документи. Існує безліч різних стратегій для витягання інформації з цих джерел:

- читання документів;
- спостереження за виконуваними операціями;
- анкетування;
- використання власних знань;
- складання опису.

Документи є хорошим джерелом інформації, тому що вони найчастіше доступні і їх можна "опитувати" в зручному для себе темпі. Читання документів – прекрасний спосіб отримати первинне уявлення про систему і сформулювати питання до експертів і підтримувати бібліотеку документів. Для SADT-проектів такі документи можуть зберігатися в *бібліотеці проекту* і розповсюджуватися у вигляді невеликих робочих пакетів (тек).

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

Іноді досвідчені аналітики досліджують велику кількість систем певного типу (наприклад, виробництво, телефонна мережа, бухгалтерія). В ході багатьох проектів вони набувають фундаментальних знань у відповідній наочній сфері щодо певного класу систем. Такі знання дуже корисні, оскільки для багатьох різних систем існують загальні поняття. Ці

поняття можуть мати широке застосування. Якщо аналітик добре знає наочну сферу, то може сам стати джерелом інформації.

У процесі аналізу, незалежно від джерел інформації, проводяться опитування декількох типів. Вибір того або іншого типу залежить від виду необхідної інформації та поставленої мети:

- опитування для збору фактів;
- опитування для визначення проблем;
- наради для ухвалення рішень;
- діалоги – автор/читач.

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

## **Процес опитування**

Для опитування використовується підхід, що включає наступні три етапи: підготовку, проведення опитування і завершення.

### *Підготовка*

Підготовка до опитування набуває вирішального значення, якщо є тільки єдина можливість поговорити з експертом. Крім того, вона допоможе оптимізувати час, який проведете з джерелом інформації, і отримати надійний потік інформації.

Реалізується за допомогою наступної послідовності етапів:

- вибір потрібного співрозмовника;
- домовленість про зустріч;
- формулювання попередньої програми зустрічі;
- вивчення допоміжної інформації;
- узгодження дій з групою проектування.

*Вибір співрозмовника* є найважливішим кроком. Дуже часто аналітик не отримує необхідної інформації при неправильному виборі джерела. Необхідно вибрати представників відповідного ієрархічного рівня організації, правильні типи документів, кращі позиції для огляду, а

також експерта, що володіє потрібними знаннями та погоджувати цей вибір з іншими членами групи.

*Домовленість про зустріч.* Після вибору співрозмовника необхідно домовитися з ним про швидку, наскільки це можливо, зустріч: встановити мету зустрічі й обмеження на неї.

*Формулювання попередньої програми зустрічі.* Етапи: установка програми розмови; визначення кола обговорюваних проблем; формування конкретних питань, особливо тих, на які необхідно отримати відповіді для продовження роботи. В ході формування програми необхідно вивчити доступну початкову інформацію. Потрібно звернутися до відповідних документів і термінологічних довідників і використовувати SADT-модель, щоб простіше виділити коло питань.

*Узгодження дій з групою проектування.* Необхідно погоджувати приготування зі всією групою проектування. Правильна координація дозволяє зберегти час експерта й мінімізує дублювання дій авторами моделі.

### **Проведення опитування**

Під час проведення опитування найважливіше правильно організувати та підтримувати потік інформації від експерта до аналітика.

Починаючи розмову, потрібно не забути представитися і сформулювати мету зустрічі. Це допоможе уникнути непорозумінь і дасть розмові правильний напрям. Крім того, необхідно обговорити можливість ведення записів. Переконайте експерта в конфіденційності розмови і в тому, що згодом йому буде надана можливість внести поправки до ваших записів. Потім сформулюйте перше питання. Пам'ятайте, що перше питання часто задає тон всій розмові, тому добре продумайте його.

Збирайте інформацію, роблячи записи про все (про спеціальні терміни, взаємозв'язках між частинами системи і т. п.) і обмежуючи час бесіди. Запишіть SADT-функції і дані, спробуйте накидати діаграму. Підтримуйте потік інформації, ставлячи питання, які уточнюють і підтверджують відповіді. Питання, які можуть допомогти уточнити або підтвердити отриману інформацію, повинні бути сформульовані таким чином:

Чи можете навести приклад?

Коли це відбулося?

Чи є у цього правила винятку?

Чи можете ви навести які-небудь цифри для підтвердження ваших слів?

Перш за все не заперечуйте. Ніколи не ставте навідних питань або питань з короткими відповідями "так" чи ні". Натомість записуйте те, що вам говорять, і просіть підвести підсумок або дати пояснення. Ви отримаєте від опитування більше, якщо ви дасте експертові можливість говорити те, що він хоче сказати, а не те, що ви хочете почути.

Стежте за виникненням наступних ситуацій:

ви вже отримали достатньо інформації;

ви отримуєте великий обсяг невідповідної інформації;

велика кількість інформації вас пригнічує;

експерт починає втомлюватися;

у вас з експертом часто виникають конфлікти.

Будь-яка з цих причин – достатня підстава для завершення бесіди.

Коли ви вважаєте за потрібне закінчити опитування, завершуйте бесіду плавно. Стисло підсумуйте основні пункти і зробіть огляд отриманих відомостей, які можуть бути опущені або неправильно тлумачаться. Домовтеся про час наступної зустрічі, якщо вона потрібна, і отримаєте рекомендації для найближчих опитувань. Поінформуйте експерта, коли і як ви збираєтеся використовувати отриману інформацію і коли ви надішлете йому матеріал на рецензування.

## **Завершення**

Завжди оформляйте матеріали опитування відразу ж після зустрічі з експертом. У цьому випадку негайно виникає зворотний зв'язок, і ви мінімізуєте можливість втрати важливої інформації. Перегляньте і закінчіть ваші нотатки, а потім складіть SADT-госарій як засіб визначення нових понять і термінології. Потім накидайте діаграму, визначаючи, які ще слід задати питання і які сфери досліджувати. Якнайшвидше зробіть хороші копії цих діаграм і глосаріїв, сформууйте з них невеликий пакет матеріалів для рецензування (теку) і відправте її експертові.

Уміння проводити опитування так само важливо, як і уміння будувати хороші діаграми й моделі. Застосовуючи на практиці наведені



рекомендації, можна стати вмілішим інтерв'юєром. Чим краще ви проводите опитування, тим легше отримати базові знання, необхідні для ваших SADT-діаграм і SADT-моделей.

### **Основні етапи моделювання**

Перш ніж почати моделювання SADT-аналітик здійснює підготовку до нього, збирає інформацію, декомпозиє об'єкт і узагальнює цю декомпозицію. Підготовка включає:

*вибір мети моделі* (наприклад, опис того, як механічний цех проводить деталі), *вибір точки зору*, з якою буде представлена модель (наприклад, майстер, робочий);

*тип створеної моделі* (наприклад, модель "потокowego" процесу);

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

Таким чином, підготовка повинна максимально полегшити збір інформації.

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

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

### **Вибір мети та точки зору**

Мета й точка зору моделі визначаються на найпершій стадії створення моделі. Вибір мети здійснюється з урахуванням питань, на які

повинна відповісти модель, а вибір точки зору – відповідно до вибору позиції, з погляду якої описується система. Іноді мету й точку зору можна вибрати до того, як буде зроблена перша діаграма. Наприклад, мету моделі експериментального механічного цеху можна визначити наперед, тому що вона очевидна в постановці завдання: "зрозуміти обов'язки тих, хто працює в цеху так, щоб пояснити їх новому персоналу". Необхідно якомога раніше визначити мету й вибрати точку зору нової моделі.

Мета повинна бути сформульована точно, з декількох різних точок зору, перш ніж вибрати одну з них.

Іноді виявляється, що визначити мету й точку зору на самому початку моделювання надзвичайно важко. У такому випадку необхідно скласти списки даних і функцій і, можливо, намалювати діаграму A0. Зробивши це, можна "відчути" систему і встановити, чи описує її діаграма A0 з *потрібної точки зору*. В цьому випадку необхідно розробити декілька альтернативних A0-діаграм, перш ніж з'явиться достатня упевненість для того, щоб здійснити вибір правильної мети й точки зору.

### **Складання списку даних**

Списки об'єктів системи, що створюються в ході моделювання, в SADT прийнято називати "списками даних". Термін "дані" тут уживається як синонім слова "об'єкт". Отже, при обговоренні різних аспектів моделювання в SADT застосовуватимемо термін "список даних". Складання списку даних є початковим етапом створення кожної діаграми функціональної SADT-моделі.

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

У сучасних аналітичних методах дуже часто приділяється підвищена увага функціям у збиток даним. Починаючи зі складання списку даних, можна уникнути переходу до негайної функціональної декомпозиції. Списки даних допоможуть виконати глибший аналіз і при цьому не концентруватися на функціях системи, уникаючи пропусків, які часто виникають з упереджених уявлень про функціональні декомпозиції.

Крім того, можна приділити належну увагу даним і ідентифікувати обмеження, що визначають функціональну декомпозицію.

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

### **Складання списку функцій**

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

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

*Список функцій повинен знаходитися на одній сторінці із списком даних*. При цьому при складанні початкового списку не потрібно об'єднувати функції між собою. Натомість спочатку необхідно зосередитися на кожній конкретній функції та її відношенні до груп даних. Крім того, потрібно підбирати такі функції, які могли б працювати з найбільш загальними типами даних з вашого списку. Що стосується прикордонних функцій (функцій, які можуть виконуватися або системою,

або її оточенням), то спочатку дуже важко визначити, входять вони в модель чи ні.

Після цього необхідно об'єднати функції в "агрегати" – організація 3 – 6 функціональних угруповань. Основною вимогою є те, щоб ці угруповання мали один і той же рівень складності, містили приблизно однаковий "обсяг" функціональності та функції в кожній з них мали схожі операції й цілі. Об'єднання не завжди легко здійснити. Можна виявити, що на якомусь рівні моделі важко вибрати "якнайкращий" спосіб об'єднання функцій.

### **Процес моделювання**

Значною мірою успіх методології SADT пояснюється її графічною мовою, хоча не менш цінним є сам процес моделювання. Процес моделювання в SADT включає збір інформації про досліджувану область, документування отриманої інформації і представлення її у вигляді моделі та уточнення моделі за допомогою ітеративного рецензування. Крім того, цей процес підказує цілком певний шлях виконання узгодженої та достовірної структурної декомпозиції, що є ключовим моментом у кваліфікованому аналізі системи. SADT унікальна за своєю здатністю забезпечити як графічну мову, так і процес створення несуперечливої та корисної системи описів.

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

Процес моделювання в SADT є ітеративною послідовністю кроків, що приводять до точного опису системи. Висока ефективність цього процесу обумовлена його організацією, в основі якої лежить розподіл функцій, що виконуються учасниками створення SADT-проектів:

експерти є джерелами інформації, автори створюють діаграми та моделі, бібліотекар координує обмін письмовою інформацією, читачі рецензують і затверджують моделі, а Комітет технічного контролю приймає і затверджує модель.

### *1. Отримання знань у процесі опитування.*

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

### *2. Документування отриманих знань.*

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

### *3. Коректність моделі перевіряється у процесі ітеративного рецензування.*

Моделі створюються виходячи з дійсної ситуації і ці моделі проходять через серію послідовних поліпшень до тих пір, поки вони не представлятимуть реальний світ. Однією з основних компонент методології SADT є ітеративне рецензування, у процесі якого автор і експерт багато разів радяться (усно й письмово) щодо достовірності створюваної моделі. Ітеративне рецензування називається циклом "автор – читач".

Цикл "автор – читач" починається в той момент, коли автор ухвалює рішення розповсюдити інформацію про яку-небудь частину своєї роботи з метою отримання відгуку про неї. Матеріал для розповсюдження оформляється у вигляді "тек" – невеликих пакетів з результатами роботи, які критично обговорюються іншими фахівцями протягом певного часу. Зроблені письмові зауваження також поміщаються в теку у вигляді нумерованих коментарів. Теки із зауваженнями є, таким чином, зворотним зв'язком, який автори отримують на свою роботу. Читачі – це ті, хто читає і критикує створювану модель, а потім поміщає зауваження в теки. Їх робота можлива завдяки тому, що графічна мова SADT-діаграм дозволяє створювати діаграми і моделі, які можна легко і швидко читати.

Зазвичай окрема тека рецензується одночасно декількома читачами, і всі їх зауваження надходять до певного терміну до автора. Потім автор відповідає на кожне зауваження і узагальнює критику, що міститься в зауваженнях. За допомогою таких обговорень можна досить швидко обмінюватися ідеями. Таким чином, методологія SADT підтримує як паралельний, так і асинхронний перегляд моделі, що є найбільш ефективним способом розподілу роботи в колективі. Це показує, що моделювання в SADT є інженерною дисципліною, тому що ітеративна колективна діяльність – ознака інженерної діяльності. Це пов'язано з тим, що модель в SADT дуже рідко створюється одним автором. На практиці над різними частинами моделі можуть спільно працювати безліч авторів, тому що кожен функціональний блок моделі представляє окремий суб'єкт, який може бути незалежно проаналізований і декомпозований. Таким чином, модель сама координує роботу колективу авторів, тоді як процес моделювання SADT координує сумісне рецензування ідей, які виникають.

#### *4. Координація процесу рецензування.*

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

#### *5. Моделі використовуються після їх схвалення.*

SADT-моделі створюються з конкретною метою, і ця мета визначена на діаграмі A-0 моделі. У деякому сенсі ця мета визначає як

використовуватиметься модель. Таким чином, як тільки завершено створення моделі з необхідним рівнем деталізації та модель перевірена, вона може застосовуватися для досягнення поставленої мети. Наприклад, модель експериментального механічного цеху створена для опису діяльності різних працівників механічного цеху, хоча результуюча модель завжди призначалася як основа навчального допоміжного матеріалу для нового персоналу. Якщо ця модель точно описує роботу персоналу в цеху, але не може служити для підготовки навчального допоміжного матеріалу – вона даремна.

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

### **Побудова діаграми A0**

Початковий зміст діаграми декомпозиції першого рівня A0 забезпечують списки даних і функцій. Для правильного опису системи змісту треба надати форму.

У SADT це робиться за допомогою побудови низки діаграм.

Це здійснюється таким чином:

- 1) розташуванням блоків на сторінці;
- 2) малюванням основних дуг, що представляють обмеження;
- 3) малюванням зовнішніх дуг;
- 4) малюванням всіх дуг, що залишилися.

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

*Найдомінантніший блок* зазвичай розташовується у верхньому лівому кутку, а *найменш домінантний* – в нижньому правому. Це приводить до розташування, при якому більш домінантні блоки обмежують менш домінантні, утворюючи "ступінчасту" схему.

Домінування має найважливіше значення для чіткого представлення процесу. Наприклад, не має сенсу говорити про контроль за виконанням завдання до закінчення процесу виготовлення деталі.

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

Основними дугами, що представляють обмеження, завжди є зовнішні дуги, тобто дуги, що представляють дані, які надходять з безпосереднього оточення діаграми.

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

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

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

2) малювання основного потоку даних;

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

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

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

Для того, щоб надати деяку форму даним і функціям, краще за все зробити накидання (чернетка). У процесі роботи з чернеткою, ситуація починає роз'яснюватися. Те, що спочатку було не зрозумілим, стає



зрозумілим після закінчення накидування. При цьому часто доводиться перейменовувати дуги і блоки, закреслювати дуги, переміщати блоки. Тому аналітикові рекомендується спочатку робити нарис діаграми, а потім перемальовувати діаграму начисто, щоб уточнити своє розуміння, прояснити ситуацію і створити опис, який можуть подивитися інші.

### **Узагальнення діаграми A0**

Узагальнення є останнім важливим кроком початкового етапу моделювання. Для будь-якої SADT-діаграми є батьківська діаграма, що містить її контекст, де під контекстом розуміється блок з набором вхідних дуг, дуг управління і вихідних дуг. Верхня діаграма моделі (діаграма A0) не становить винятку. Контекстом для неї буде діаграма A-0, що є узагальненням усієї моделі.

Діаграма A-0 має декілька призначень:

по-перше, вона оголошує загальну функцію всієї системи;

по-друге, вона дає безліч основних типів або наборів даних, які використовує або проводить система;

по-третє, A-0-діаграма вказує взаємовідношення між основними типами даних, здійснюючи їх розмежування.

Таким чином, A-0-діаграма є загальним видом системи, що вивчається.

При створенні діаграми A-0 використовується інформація, вже зафіксована на діаграмі A0. Спочатку в центрі SADT-діаграми малюють один великий блок, назва якого збігається з назвою діаграми A0. У цей момент слід перевірити, чи адекватна назва відображає те, що робить система. Всі зовнішні дуги діаграми A-0 зображаються на діаграмі A-0, що входять у відповідну сторону блоку. При цьому потрібно перевірити, що назва кожної дуги описує те, чим обмінюються система та її середовище. Після того, як аналітик зобразить вхідні і вихідні дуги, потрібно перевірити точність опису потоку даних. Намалювавши дуги управління, переконайтеся, що саме вони управляють тим, як система перетворить вхідні дані у вихідні.

Після закінчення проведених робіт необхідно перевірити мету й точку зору моделі під основним блоком і звірити їх з тим, що представляється блоком і його дугами.

У процесі узагальнення аналітик переконується в тому, що воно допомагає прояснити опис системи, тому що при узагальненні

проглядаються мітки дуг для точнішого найменування даних, якими обмінюються система та її середовище. Крім того, під час узагальнення дуги часто об'єднуються для спрощення зображення моделі. В цьому випадку дуги розгалужуються на свої складові на діаграмі A0.

Побудова діаграми A-0 свідчить про закінчення початкового етапу моделювання. До цього моменту зроблена перша спроба узагальнити й описати основну діяльність системи та показати зв'язок системи з її середовищем. Не дивлячись на обмежене число описаних деталей, діаграми A-0 і A0 представляють закінчену картину, тому що вони відображають всі основні входи, управління, виходи і функції системи. Загальний вид системи, отриманий за допомогою діаграм A-0 і A0, – основна мета аналітика на початковому етапі побудови SADT-моделі.

### **Декомпозиція обмеженого об'єкта**

Декомпозиція моделі схожа на початковий етап моделювання, але простіше за нього. Чому? Тому що при декомпозиції моделі аналітик завжди знаходиться в контексті, визначеному блоком зі своїми дугами однієї з діаграм. Ця межа, яка називається межею об'єкта, визначена двома способами. По-перше, об'єкт, мета і точка зору кожної нової діаграми вже визначені на діаграмі AT. По-друге, кожен блок, декомпозований у нову діаграму, вже є обмеженим об'єктом. Іншими словами, він ідентифікує конкретну функцію і всі дані, які для неї потрібні або нею породжуються. Будувати діаграму, виходячи з цієї інформації, простіше, тому що список даних створюється на основі дуг, що входять у блок і виходять з нього, а також тому, що список функцій детально розкриває назву блоку. Процес декомпозиції обмеженого об'єкта складається з наступних кроків:

- 1) вибір блоку діаграми;
- 2) розгляд об'єкта, визначеного цим блоком;
- 3) створення нової діаграми;
- 4) виявлення недоліків нової діаграми;
- 5) створення альтернативних декомпозицій;
- 6) коригування нової діаграми;
- 7) коригування всіх пов'язаних з нею діаграм.

Кроки 1– 3 представляють творчу частину процесу. Здійснюючи їх, аналітик концентрує свої зусилля, пов'язані з виявленням нової

інформації про об'єкт, на більш високому рівні деталізації, щоб досягти чіткості викладу.

Кроки 4 – 7 складають етап саморецензування, в ході якого аналітик, створивши нову діаграму, перевіряє, яку вона несе інформацію і в якому вона знаходиться відношенні з батьківською діаграмою. Потім у створену діаграму і відповідно до пов'язаних з нею діаграм вносяться зміни, щоб досягти чіткості для інших.

### ***Вибір блоку***

Декомпозиція починається з читання діаграми A0 і визначення найзмістовнішого блоку. Це такий блок, декомпозиція якого виявить багато аспектів діаграми A0 і здійснюватиме великий вплив на майбутні декомпозиції інших блоків цієї діаграми. При виборі найзмістовнішого блоку слід враховувати домінування, функціональну складність і зрозумілість. Кращим не обов'язково буде блок, найбільш складний для розуміння, а той, який дозволить найглибше проникнути в суть даної системи.

### **Об'єкт, який визначається блоком**

Блок 2, *виконати завдання*, стає тепер самостійним об'єктом декомпозиції. Для виконання цієї декомпозиції спочатку оглянемо узагальнювальну діаграму і пригадаємо мету й точку зору моделі.

Потім складається список даних зі всіх дуг, що стосуються блоку, використовуючи ІСОМ-кодування для того, щоб не втратити які-небудь інтерфейсні дані. Наприклад, план *виконання завдання*, *верстати й інструменти*, а також брак входять в початковий список даних. Цей список тепер поліпшується завдяки декомпозиції первинних даних або введенню нових, тісно пов'язаних даних.

Далі на основі списку даних складаємо список функцій, дотримуючись функції, відповідної блоку верхньої діаграми. Наприклад, *вибрати інструменти*, *підготувати робоче місце*, *обробити на верстаті і зібрати і* визначити ступінь виконання завдання – дійсно є функціями, що здійснюється працівниками при виконанні завдання.

Необхідно прагнути обмежуватися розумним рівнем складності при об'єднанні функцій і даних: чотири – п'ять функціональних блоків, як правило, краще за все. Дуже багато даних і функцій часто містять дуже багато інформації. Це приводить до заплутаних діаграм, і, навпаки,

невелике число блоків дає дуже мало, і діаграма стає майже даремною. Якщо є баланс, необхідно перевірити, чи у всіх відносинах написані слова адекватні об'єкту, визначеному блоком і його граничними дугами на батьківській діаграмі.

### **Створення нової діаграми**

Нова діаграма на наступному рівні декомпозиції будується аналогічно діаграмам A0 і A-0: блоки розміщуються відповідно до домінування (тобто згідно з взаємним обмеженням блоків), потім створюються основні дуги, що представляють обмеження, потім зовнішні і, нарешті, внутрішні дуги. Таким чином, SADT-діаграми будуються відповідно до тієї інформації, яку вони несуть.

Наприклад, дуга *наступний крок завдання* є носієм істотної інформації: вона визначає, що потрібно зробити на наступному кроці завдання. Отже, оскільки вона обмежує решту всіх функцій, блок *визначити ступінь виконання, що породжує дугу наступний крок завдання*, є найдомінантнішим на цій діаграмі. *План виконання завдання*, очевидно, потрібний перш ніж що-небудь може відбутися, тому що вміст цієї зовнішньої дуги визначає послідовність кроків обробки. Дуга, дуже важлива, тому що керує. Крім того, аналітики прагнуть відобразити на діаграмі потоки інформації, особливо потоки інформації зворотного зв'язку. Зверніть увагу на те, що *результати обробки* є зворотний вхід від блоку *обробити на верстаті і зібрати до блоку визначити ступінь виконання завдання*. Це означає: *результат обробки* є *незавершене завдання*, яке отримується на кожному кроці виконання завдання, що повною мірою відповідає дійсності.

### **Виявлення інтерфейсних помилок**

У розглянутих моделях "збої" часто відбуваються в точках інтерфейсу. Для методології SADT інтерфейсними точками є місця з'єднання діаграм зі своїми батьківськими діаграмами. Тому кожен декомпозицію необхідно правильно сполучати з своїм батьком, використовуючи ISOM-мітки: перш ніж складати список даних, запишіть імена і ISOM-коди для всіх дуг, утворюючих межу. Виконавши декомпозицію, поверніться назад до початкового блоку батьківської діаграми і з'єднайте кожен зовнішню дугу нової діаграми з відповідною дугою, що стосується цього блоку. Це дозволить уникнути пропуску необхідного з'єднання.

## Принципи і прийоми розташування дуг

Дуги виражають зв'язки між блоками. Вони відображають відносини між блоками, незалежні від потенційного проходження.

Може трапитися, що, почавши малювати дугу, у вас виникнуть сумніви в тому, що вона потрібна на діаграмі. Не зобразивши дугу при першій розробки діаграми, ви ризикуєте зробити помилковий пропуск. Оскільки SADT-діаграми завжди перевіряються іншими фахівцями і оскільки відсутню дугу ніхто не зможе виявити, вилучення сумнівної дуги з діаграми – це прямий шлях до виникнення помилок і відсікання додаткових можливостей.

SADT-дуги – це набори об'єктів і тому вони потенційно можуть нести багато даних. Наприклад, *сировина* може бути просто сталевим бруском, а може бути: листами, рулонами, шматками дерева, пластика або різних металів. Досвідчені SADT-аналітики не зображають кожен об'єкт окремою дугою. Інколи можна давати визначення нової дуги в глосарії і, можливо, уточнювати її зміст при декомпозиції тих блоків, яких вона стосується.

### Перевірка діаграми

Процес декомпозиції в SADT зводиться до представлення кожного блоку діаграми за допомогою діаграми наступного рівня деталізації. Під час декомпозиції складають список даних і список функцій, об'єднують функції в блоки і використовують список даних при формуванні взаємозв'язків між блоками. В ході цього процесу аналітик здійснює такі кроки, як вибір блоку, вивчення його об'єкта та побудову нової діаграми.

*Метою аналітика* на цьому етапі є в першу чергу чітке сприйняття *суті декомповованої моделі*.

Побудувавши діаграму, спробуйте самостійно виявити її недоліки, перш ніж розсилати її для докладного рецензування. Досвідчені SADT-аналітики при декомпозиції блоку розділяють етап створення і етап критичного розгляду діаграми. Вони спеціально вибирають час для того, щоб досягнути її критичним поглядом, пам'ятаючи про те, що за декілька хвилин можна самому виявити ті помилки, які часто виявляються за допомогою зворотного зв'язку з читачами.

### Процес авторської перевірки

Процес авторської перевірки дає новий напрям роботі – визначення її якості. На етапі декомпозиції виникає діаграма, яка декомпозує блок і його дуги. Аналітик намагається пояснити об'єкт самому собі. Недивно, що результат може виявитися малодоступним для інших. У роботі, природно, з'являються жаргон і факти, які не чітко розуміються.

При критичній оцінці аналітик абстрагується від своєї роботи. Це дозволяє поглянути інакше на діаграму для того, щоб інформація, яку вона несе, стала доступною не тільки її авторові, але й іншим. Процес критичної оцінки здійснюється в наступному порядку:

- виявлення недоліків нової діаграми;
- створення альтернативних декомпозицій;
- коригування нової діаграми;
- коригування *всіх пов'язаних з нею діаграм*.

Часто в ході критичної оцінки виконують альтернативні декомпозиції, щоб перевірити, чи є початковий нарис кращим для передачі бажаній інформації. Крім того, перевіряють взаємозв'язки з батьківською й іншими діаграмами. Після цього до всіх діаграм вносяться необхідні зміни.

### **Виявлення недоліків нової діаграми**

Виявлення недоліків діаграми відбувається за схемою питання – відповідь, тому не можна чітко сформулювати правила її покрокової оцінки. Досвідчений SADT-аналітик, проте, у процесі оцінки діаграми задає певний набір питань щодо блоків, зв'язку з батьківською діаграмою і внутрішніх дуг. Відповідь на кожен із них дає напрям подальшим питанням. Таким чином, потік питань і відповідей управляє аналізом кожної частини діаграми.

### ***Питання про блоки***

Спочатку слід критично оцінити блоки діаграми. Визначимо функціональні аспекти діаграми, ставлячи запитання типу:

1. Чи представляють блоки змістовну декомпозицію функції?
2. Чи не виглядає діаграма заплутано?
3. Чи всі блоки відповідають точці зору моделі?
4. Чи несуть блоки достатній обсяг нової інформації?
5. Чи всі блоки мають однаковий рівень деталізації?

6. Чи відповідна складність всіх блоків?
7. Чи відображає кожен блок який-небудь аспект блоку батьківської діаграми?

### **Зв'язок з батьківською діаграмою**

Тепер поставимо запитання про зв'язок діаграми з її батьком. При цьому перевіримо, як діаграма вписується в модель.

1. Чи всі зовнішні дуги мають ISOM-коди?
2. Чи всі ISOM-коди сполучають дуги з одним і тим же значенням?
3. Чи доповнюють назви зовнішніх дуг інформацію, що повідомляється діаграмою?
4. Чи не суперечить сенс аналізованої діаграми сенсу батьківської діаграми?

### **Запитання про внутрішні дуги**

Запитаннями про внутрішні дуги зазвичай закінчують пошук помилок в діаграмі. Тепер після дозволу основних питань, слід проаналізувати деталі діаграми. До додаткових запитань відносяться:

1. Чи не дуже багато внутрішніх дуг?
2. Чи немає блоків без дуг управління?
3. Чи немає блоків без вихідних дуг?
4. Чи правильно відображають дуги, що представляють обмеження, домінування блоків?
5. Чи правильне рішення діаграми?
6. Чи всі важливі зворотні зв'язки відображені?
7. Чи всі помилкові ситуації враховані?

### **Створення альтернативних декомпозицій**

До цього моменту у вас накопичилося багато нової інформації про діаграму, яку ви накреслили. Скористайтеся цією інформацією розумно. Спробуйте провести альтернативні декомпозиції з використанням нових

фактів і подивіться, чи не вийде у вас точнішої діаграми. Навіть якщо вам не вдасться побудувати абсолютно нову діаграму, що доносить сенс краще, ви зможете виправити частину свого початкового опису.

### **Альтернативна декомпозиція та об'єднання функцій**

Іноді у аналітика виникають сумніви щодо блоків діаграми. На хорошій SADT-діаграмі блоки повинні володіти деякими важливими якостями:

виконувати певні функції;

мати однакову складність;

мати однаковий рівень деталізації;

з'єднуватися з іншими блоками діаграми;

впливати на управління, входи і виходи з певним сенсом;

*працювати разом з іншими блоками для виконання функції діаграми.*

Спробуйте об'єднати функції і дані інакше або складіть новий список функцій, якщо початковий набір блоків не дозволяє здійснити декомпозицію вдало. Ви можете це зробити і для того, щоб переконатися в правильності початкового розбиття. Наприклад, розділіть блок *обробити на верстаті і зібрати* на дві функції. Ви побачите, що це дуже просто, але при цьому обсяг нової інформації зовсім невеликий. У даному випадку початкове об'єднання обробки на верстаті і збірки в один блок на цьому рівні моделі є якнайкращим.

Застосовуючи ці прийоми, необхідно знати, що критерій якості для блоків достатньо суперечливий. Наприклад, добиваючись однакової складності блоків, можна ускладнити з'єднання між ними, а спрощення зв'язку між двома блоками може приховати який-небудь важливий на даному рівні деталізації факт. Основним чинником завжди повинен бути якнайкращий опис декомпозованого об'єкта. Побудова хороших блоків можлива тільки під час досягнення рівноваги між вимогами до складності з'єднання блоків і достатності рівня деталізації.

### **Альтернативне об'єднання та роз'єднання дуг**

Іноді можна виявити дві дуги, які починаються і закінчуються в одних і тих же місцях діаграми, тобто обидві дуги починаються і закінчуються в одних і тих же блоках. У цьому випадку подивіться на ці



дві дуги уважно. Може виявитися, що їх слід об'єднати в одну. Якщо ви можете придумати гарну назву, об'єднуючи назви цих дуг, об'єднайте їх. Якщо наявність двох дуг має певний сенс, не об'єднуйте їх. Об'єднання приховує деталі, тому не робіть це механічно, а зникнення деталей зашкодить діаграмі.

При аналізі можна виявити також дугу, що описує два абсолютно різних набору даних. У цьому випадку вивчіть дугу, щоб оцінити, чи приведе розділення її на дві до прояснення важливих для діаграми деталей. Будьте дуже обережні і прагніть зберегти рівновагу між прагненням до деталізації і збереженням наочності діаграми. Наприклад, дуги *статус завдання* і *незавершене завдання* не були об'єднані в одну вихідну дугу. Вони відображають різні типи даних (перша – поняття, друга – фізичну реальність) і кожна впливає на свою частину батьківської діаграми (*управляти виконанням, завдання і контролювати якість виконання* відповідно). Тому їх роздільне зображення дає чіткішу картину результатів роботи блоку *визначити ступінь виконання завдання*.

## Тестування

Хороший спосіб оцінки діаграми полягає в розгляді сценаріїв її роботи. Ви уявляєте собі можливу ситуацію і дивитесь, як працює діаграма в заданих умовах. У міру розвитку сценарію робіть позначки на діаграмі. Це дасть можливість завжди повторити сценарій, а інформація може допомогти при декомпозиції блоків цієї діаграми. Таким чином перевірите точність і зрозумілість розглянутого в діаграмі.

Розглянемо, що описує діаграма *виконати завдання*, починаючи обробляти нове завдання: вивчається *план виконання завдання* і вибирається *наступний крок завдання*. Це визначає, які вибрати *інструменти* і як підготувати *робоче місце*. Потім *сировина* і *брак* обробляються на *верстаті* і *збираються* і видаються *результати обробки*. Далі за цими результатами *визначається ступінь виконання завдання* і вибирається *наступний крок завдання*.

## Схематичне зображення декомпозиції наступного рівня

Ще один спосіб перевірки правильності діаграми – розкладання одного – два її блоків. (При цьому зберігайте свої накидання, щоб

полегшити майбутні декомпозиції цієї діаграми.) Деталізація деякої частини нової діаграми допоможе визначити збалансованість декомпозиції і виявити неузгодженість у розподілі функцій між новими блоками.

Наприклад, діаграма *виконати завдання* складається з чотирьох блоків. Іноді діаграму, що складається з чотирьох блоків, можна декомпонувати по-іншому, замінивши її діаграмою з п'яти – шести блоків, яка буде ненабагато складніша, але більш інформативна. Проте декомпонувавши будь-який з блоків діаграми *виконати завдання*, побачимо, що кожен з них досить змістовний. Не зважаючи на те, що на діаграмі *виконати завдання* чотири блоки, кожен із них сам по собі досить складний. Не має сенсу поєднувати або роз'єднувати ці блоки для створення альтернативної діаграми. Найкращий спосіб продовження цієї частини моделі полягає, мабуть, у збереженні цих чотирьох блоків з подальшою декомпозицією кожного з них.

### **Коригування нової діаграми**

Зазвичай, згаявши час на питання що стосується діаграми, тестування, виконання альтернативних креслень, автор коригує діаграму. В ході коригування стежте за правильним домінуванням, вибором назв блоків, інформативністю дуг і робіть пояснення.

### **Перевизначення блоків домінування**

Для того, щоб скласти перше враження про виконувани блоками функції, проектувальник діаграми прочитає по порядку їх найменування. Забезпечте йому правильне перше враження від діаграми, організувавши блоки так, щоб вони якомога точніше вказували на взаємний вплив. Один із поширених прийомів для цього полягає в розташуванні найбільш домінантного блоку у верхньому лівому кутку діаграми, а найменш домінантного – в нижньому правому. Пам'ятайте, що розташування блоків може полегшити або ускладнити проведення дуг. Іноді доводиться жертвувати розташуванням блоків в порядку спадання домінантності для простоти проведення дуг з метою швидкого отримання читаної діаграми. Наприклад, блоки *вибрати інструменти* і *підготувати робоче місце* були визначені як більш домінантні, ніж блок

*обробити на верстаті і зібрати*, через те, що вони часто повинні виконуватися до того, як верстати будуть відповідним чином використані. Тому, незважаючи на допоміжну роль по відношенню до блоку *обробити на верстаті і зібрати*, блоки *вибрати інструменти і підготувати робоче місце* є дуже важливими кроками, які передують обробці, що й відображено в нумерації блоків.

### **Змістовні назви блоків**

Для блоків зазвичай прагнуть вибрати змістовні назви. Проте в SADT немає необхідності виражати все за допомогою назв блоків, тому що про роботу блоку багато що повідомляють позначки дуг, що оточують його. Наприклад, *деталі, сировина і брак* перетворюються на *результати обробки*, відповідно до *наступного кроку завдання*. Таким чином, докладніша назва блоку не потрібна: воно може тільки ускладнювати розуміння.

### **Дуги, що передають інформацію**

Малюючи дуги, намагайтеся розташовувати їх акуратно, мінімізуючи число перетинів і максимізуючи простір між ними. Правильне графічне розташування сприяє підвищенню наочності та зрозумілості діаграми. Позначайте дуги чітко і точно. Хоча певна кількість слів передає інформацію краще, слід обмежувати себе. Використовуйте додаткове слово, якщо ситуація цього вимагає.

Викреслюючи дуги в порядку їх значущості, можна оцінювати їх в процесі малювання і уникати прагнення механічно приєднувати всі дуги до всіх блоків: спочатку зображуються дуги, що представляють обмеження, потім зовнішні дуги, потім основний шлях і, нарешті, зворотні зв'язки.

Рекомендується не використовувати як позначки дуг списки імен. Такі списки погіршують діаграму, тому що несуть менше інформації, ніж загальна назва всього набору даних. Хороша функціональна декомпозиція в SADT починається з хорошої декомпозиції даних. Відсутність загальної назви набору даних погіршує функціональну декомпозицію. Ретельно продумайте назви для наборів даних і

прогляньте списки даних і функцій попередніх діаграм, щоб полегшити вибір відповідних назв.

## Пояснення

Після закінчення побудови діаграми, поясніть її важливі аспекти за допомогою зауважень або додаткового матеріалу. Проте будьте обережні з поясненнями: не використовуйте їх як прикриття поганої побудови діаграм. Прояснійте тільки ті поняття, які не можна зобразити у вигляді блоків і дуг. З іншого боку, опис типових незавершених завдань істотно полегшить пояснення того, як вони повинні бути завершені.

## Виправлення взаємопов'язаних діаграм

Створення діаграми, відповіді на пов'язані з нею питання й її переробка забезпечують глибше розуміння батьківської діаграми і діаграм-нащадків побудованої діаграми. Зафіксуйте своє розуміння під час виправлення діаграми. Це саме той випадок, коли перенесення інформації знизу-вгору природним чином вписується в техніку декомпозиції. Аналітик змушений переносити інформацію на інші діаграми в трьох ситуаціях: під час зміни позначок зовнішніх дуг, під час появи нових зовнішніх дуг і під час перерозподілу функцій.

Перенесення необхідне в тому випадку, якщо змінилася назва зовнішньої дуги. Перенесення змінених позначок зовнішніх дуг негайно забезпечує надання батьківською діаграмою всіх даних, необхідних діаграмі-нащадкові. Перенесення необхідне також, коли на новій діаграмі з'являються нові вхідні або вихідні дуги. Ці нові дуги повинні, так або інакше, виникнути на батьківській діаграмі. Існує два методи: намалювати нові дуги на батьківській діаграмі або об'єднати дуги нової діаграми в одну і змінити відповідним чином позначку дуги на батьківській діаграмі. При цьому необхідно дотримувати правила з'єднання і розгалуження дуг.

Переміщення блоків становить найскладнішу ситуацію. Воно відбувається, коли функція (зазвичай на низькому рівні моделі) повинна з'явитися, але не з'являється на діаграмі, яку ви малюєте, а з'являється на іншій діаграмі моделі, або навпаки. Перенести таку функцію, представлену блоком і всіма його дугами, з однієї діаграми на іншу –

нелегка справа. Зазвичай це приводить до великих змін у позначках дуг, появі множини нових і зникненню деяких старих дуг. Іноді переміщення одного блоку веде до переміщення інших блоків на різні діаграми, викликаючи цілу хвилю змін. Як правило, переміщення блоку призводить до великої кількості технічно складної роботи і може призвести до помилок, якщо зміни не відстежуються досить ретельно.

#### **1.4. Концепція розробки ІС підприємства на основі структурного підходу**

Розробка складних інформаційних систем (ІС) таких, якими є ІС в адміністративно-управлінській діяльності підприємств (організацій, установ і т. д.; далі – ІС підприємств), неможлива без ретельно обдуманого методологічного підходу. Які етапи необхідно пройти, які методи і засоби використовувати, як організувати контроль за просуванням проекту і якістю виконання робіт – ці та інші питання вирішуються методологіями програмної інженерії.

У даний час існує ряд загальних методологій розробки ІС. Головна з них – єдина дисципліна роботи на всіх етапах життєвого циклу системи, облік критичних завдань і контроль їх рішення, застосування розвинених інструментальних засобів підтримки процесів аналізу, проектування та реалізації ІС.

Для різних класів систем використовуються різні методи розробки, які визначаються типом створюваної системи і засобами реалізації. Специфікації цих систем, в більшості випадків, складаються з двох основних компонентів – функціонального та інформаційного. За способом поєднання цих компонентів підходи до представлення інформаційних систем можна розбити на два основні типи – структурний і об'єктно-орієнтований.

В області створення систем автоматизації адміністративно-управлінської діяльності домінують структурні підходи, оскільки вони максимально пристосовані для взаємодії з користувачами (замовниками), фахівцями у сфері інформаційних технологій. Адекватними інструментальними засобами, що підтримують структурний підхід до створення інформаційних систем, є так звані CASE-системи автоматизації проектування.

Дана робота присвячена обговоренню питань застосування CASE-технології для структурного аналізу систем управління підприємств, демонстрації можливостей використання цієї технології на прикладі конкретної CASE-системи, а також розгляду проблем методичного забезпечення робіт із обстеження адміністративно-управлінської діяльності підприємств і відображенню отриманих результатів у вигляді відповідних CASE-моделей.

### **Концептуальні основи створення ІС підприємства. Основоположна концепція**

Основоположна концепція полягає в побудову сукупності логічних моделей наочної сфери за допомогою графічних методів структурного аналізу, які дали б можливість користувачам, аналітикам і розробникам отримати чітку загальну картину проекту, а також забезпечили б природний перехід до логічної моделі майбутньої ІС.

Об'єктом обговорення даного розділу є методи та інструментальні засоби, що з'явилися в результаті розвитку дисципліни структурного системного аналізу, а також переваги, які забезпечуються застосуванням вказаних методів і засобів.

З багатьох точок зору системний аналіз є найбільш важкою частиною процесу створення інформаційних систем. Маються на увазі не тільки технічні труднощі аналізу, хоча багато проектів вимагають, щоб аналітик володів глибокими знаннями у сфері сучасної технології обробки даних. Саме поєднання всіх труднощів робить системний аналіз такою складною та копіткою справою, маючи на увазі ще і той факт, що аналітик повинен грати роль посередника між замовником-користувачем і виконавцем-розробником.

Користувачі інтуїтивно розуміють свої проблеми, але не можуть пояснити їх, і, крім того, мають не чітке уявлення про те, яку користь можуть принести інформаційні технології, засновані на застосуванні комп'ютерів. Розробники з ентузіазмом говорять про існуючі можливості в сфері побудови систем обробки даних, але вони не мають інформації про те, що саме є якнайкращим для даного підприємства, установи або організації.

Аналітик повинен обрати золоту середину: обрати, що є в даний час можливим з погляду технології обробки даних, і що варто робити для даного конкретного підприємства.

Здійснення такого вибору, який був би прийнятним для всіх груп і витримав би перевірку часом, найскладніше завдання на етапі системного аналізу. Якщо виконати це завдання найкращим чином, то незалежно від того, наскільки важким виявиться проектування і розробка, створена система задовольнятиме вимоги даного підприємства. Якщо ж завдання буде виконано незадовільно, то не має значення наскільки добре пройде реалізація, оскільки створена система не буде тим, що дійсно необхідно даному підприємству, і витрати перевершать отримані переваги.

Структурний аналіз як сукупність методів постановки завдань проектування інформаційних систем через значну розмірність вирішуваних завдань, сам повинен спиратися на могутні засоби комп'ютерної підтримки, що забезпечує автоматизацію праці системних аналітиків – CASE-системи.

Хоча в даний час не існує загальноприйнятого визначення CASE, і зміст цього поняття зазвичай визначається переліком вирішуваних завдань, а також сукупністю вживаних методів і засобів, грубо можна сказати, що CASE є сукупністю методологій аналізу, розробки й супроводу складних систем, підтриману комплексом взаємопов'язаних засобів автоматизації. CASE – це інструментарій для системних аналітиків і програмістів, що дозволяє автоматизувати процеси аналізу, проектування та реалізації систем.

До справжнього моменту дисципліна CASE переросла в самостійний наукоємний напрям, що призвів за собою утворення могутньої CASE-індустрії, що об'єднала сотні фірм різної орієнтації. Серед них виділяються:

- фірми-розробники засобів аналізу і проектування програмного забезпечення;

- фірми-розробники спеціальних CASE-засобів, орієнтованих на вузькі наочні сфери застосування або на окремі етапи життєвого циклу систем;

- повчальні фірми, які організують семінари й курси підготовки фахівців, що надають практичну допомогу при використанні CASE-систем для розробки конкретних застосувань;

фірми, що спеціалізуються на випуску періодичних видань із CASE-тематики.

Основними користувачами CASE-систем є:

аналітичні центри державних, військових і комерційних організацій;  
банки і страхові компанії;

аудиторські та консалтингові фірми, що застосовують CASE-засоби для специфікації бізнес-процесів у системах управління виробництвом, комерційною діяльністю і фінансами з метою їх реорганізації та автоматизації;

компанії з розробки апаратного і програмного забезпечення систем обробки даних і, зокрема, інтегрованих систем, що інформаційно-управляють.

Існує думка, що CASE, разом із системами візуального програмування, є найбільш перспективним напрямом у програмній інженерії. З цим можна сперечатися, але те, що CASE – найбільший напрям, що динамічно розвивається, є в даний час незаперечним фактом. Практично не один серйозний американський або японський програмний проект не здійснюється без використання CASE-засобів. Відома методологія структурного системного аналізу SADT – Structured Analysis and Design Technique (точніше її підмножина IDEF0) прийнята як стандарт на розробку засобів програмного забезпечення Міністерством оборони США. Крім того, серед менеджерів і керівників комп'ютерних фірм є перевагою знання основи SADT і при обговоренні яких-небудь питань намалювати просту діаграму, що пояснює суть справи.

Архітектура більшості CASE-засобів заснована на парадигмі "методологія – метод – нотація – засіб". Методологія визначає критерії для оцінки й вибору проекту створюваної системи, етапи роботи та їх послідовність, а також правила розподілу і призначення методів. Методи – це систематичні процедури, які вживаються для генерації описів підсистем і функціональних компонентів системи з використанням відповідних нотацій. Нотації призначені для уявлення проектних даних про структуру системи і способи її функціонування, процеси, інформаційні потоки, накопичувачі та т. д. Засоби – інструментарій для підтримки й посилення методів. Інструментальні засоби підтримують роботу аналітиків при реалізації проекту в мережному інтерактивному режимі, вони сприяють організації проекту, забезпечують управління процесами аналізу і проектування.



## Життєвий цикл ІС

В основі діяльності зі створення й використання ІС лежить поняття життєвого циклу.

Життєвий цикл – це модель створення й використання ІС, що відображає її різні стани, починаючи з моменту виникнення необхідності в даному комплексі засобів і закінчуючи моментом його повного виходу з вживання у користувачів.

Досвід створення й використання замовлених ІС дозволяє умовно виділити наступні основні етапи їх життєвого циклу:

аналіз – визначення того, що повинна робити система;

проектування – визначення того, як система робитиме те, що вона повинна робити. Проектування це, перш за все, специфікація підсистем, функціональних компонентів і способів їх взаємодії в системі;

розробка – створення функціональних компонентів і підсистем окремо, з'єднання підсистем в єдине ціле;

тестування – перевірка функціональної та параметричної відповідності системи показникам, визначеним на етапі аналізу;

впровадження – установка і введення системи в дію;

супровід – забезпечення штатного процесу експлуатації системи на підприємстві замовника.

Етапи розробки, тестування і впровадження ІС позначаються єдиним осяжним терміном – реалізація.

Життєвий цикл утворюється відповідно до принципу спадного проектування і, як правило, носить ітераційний характер: реалізовані етапи, починаючи з найраніших, циклічно повторюються відповідно до змін вимог і зовнішніх умов, введення додаткових обмежень і т. п.

На кожному етапі життєвого циклу породжується певний набір технічних рішень і документів, що відображають їх, при цьому для кожного етапу результатними є документи і рішення, прийняті на попередньому етапі.

Існуючі моделі життєвого циклу визначають порядок виконання етапів у процесі створення ІС, а також критерії переходу від етапу до етапу.

Відповідно до цього найбільшого поширення набули три наступні моделі:

1. Каскадна модель – припускає перехід на наступний етап після повного завершення робіт попереднього етапу (характерна для військово-технічних проектів).

2. Поетапна ітераційна модель – модель створення ІС, припускає наявність циклів зворотного зв'язку між етапами. Перевага такої моделі полягає в тому, що міжетапні коригування забезпечують велику гнучкість і меншу трудомісткість порівняно з каскадною моделлю. Проте час життя кожного з етапів може розтягнутися на увесь період створення системи.

3. Спіральна модель – робить наголос на початкових етапах життєвого циклу: аналіз, попереднє й детальне проектування. Кожен виток спіралі відповідає поетапній моделі створення фрагмента або версії системи, на ньому уточнюються цілі й характеристики проекту, визначається його якість, плануються роботи наступного витка спіралі.

Невирішені питання й помилки, допущені на етапах аналізу і проектування ІС, породжують на подальших етапах складні, часто нерозв'язні проблеми і, в кінці кінців, призводять до провалу всього проекту.

Головна особливість сучасної індустрії замовлених ІС полягає в концентрації зусиль на двох початкових етапах її життєвого циклу – аналізі і проектуванні, при досить невисокій складності і трудовитратах на подальших етапах. Розглянемо етап аналізу детальніше.

### **Структурний аналіз**

Аналіз є першим етапом створення ІС, на якому вимоги замовника уточнюються, формалізуються й документуються. Фактично на цьому етапі дається відповідь на питання: "Що повинна робити майбутня система?". Саме тут знаходиться ключ до успіху всього проекту.

Метою аналізу є перетворення загальних розпливчатих знань про початкову наочну сферу в точні визначення і специфікації, а також генерація функціонального опису системи.

На цьому етапі специфікуються:

зовнішні умови роботи системи;

функціональна структура системи;

розподіл функцій між людиною і системою, інтерфейси;

вимоги до технічних, інформаційних і програмних компонентів системи;

умови експлуатації.

Розробка перерахованих вище специфікацій при створенні ІС, призначених для автоматизації управлінських процесів, в загальному випадку проходить чотири стадії.

Структурний аналіз починається з дослідження того, яка організована система управління підприємством, з обстеження функціональної й інформаційної структури системи управління. За наслідками обстеження аналітик на першій стадії аналізу будує узагальнену логічну модель початкової наочної сфери, що відображає її функціональну структуру, особливості основної діяльності та інформаційний простір, в якому ця діяльність відбувається. Використовуючи спеціальну термінологію, можна сказати, що аналітик будує модель "як є".

Друга стадія роботи, до якої притягуються зацікавлені представники замовника, а за необхідності й незалежні експерти, полягає в аналізі моделі "як є", виявленні її недоліків і вузьких місць, визначенні шляхів вдосконалення системи управління на основі виділених критеріїв якості.

Третя стадія аналізу, що містить елементи проектування, – створення вдосконаленої узагальненої логічної моделі, що відображає реорганізовану наочну сферу або її частину, яка підлягає автоматизації. Цю модель можна назвати моделлю "як треба".

Закінчується процес розробкою "карти автоматизації", що є моделлю реорганізованої наочної сфери, на якій позначені "межі автоматизації". Слід зазначити, що на практиці запропоновану загальну схему структурного аналізу і проектування, що включає стадію планування реорганізованої діяльності підприємства, доводиться зустрічати у край рідко. Таку роботу можуть виконати для замовників лише великі спеціалізовані консалтингові фірми, здатні підключити до роботи фахівців-експертів у тій сфері діяльності, яка підлягає автоматизації. В більшості випадків модель "як є" поліпшується системним аналітиком за рахунок усунення очевидних невідповідностей і вузьких місць, а отриманий таким чином варіант моделі розглядається надалі як модель "як треба".

Основним документом, що відображає результати робіт першого етапу створення ІС, є технічне завдання на проект (розробку), що містить,

окрім вищеперелічених визначень і специфікацій, також зведення про черговість створення системи, відомості про ресурси, що виділяються, директивні терміни проведення окремих етапів роботи, організаційні процедури і заходи щодо приймання етапів, захисту проектної інформації і т. д. Таким чином, автоматизація систем управління в тому вигляді як вона розуміється в даний час, є лише одним, найбільш просунутим завданням структурного аналізу. Іншими, не менш важливими і самостійними за характером отримуваних результатів, є завдання специфікації та реорганізації процесів, що протікають у системах управління. Очевидно, що цілеспрямоване рішення саме цих завдань у деяких випадках може привести до результатів, що дають відчутний економічний ефект без значних інвестицій у сферу автоматизації підприємства.

### **Принципи структурного аналізу**

Аналіз наочної сфери є найважливішим етапом серед всіх етапів життєвого циклу системи. Він здійснює істотний вплив на наступні етапи, будучи в той же час найменш вивченим і зрозумілим процесом. На цьому етапі, по-перше, необхідно зрозуміти, що передбачається зробити, а по-друге, задокументувати висунуті пропозиції, оскільки якщо проектні вимоги не зафіксовані і не зроблені доступними для учасників розробки, то вони начебто і не існують зовсім. При цьому мова, на якій формулюються результати аналізу, повинна бути достатньо проста і зрозуміла замовникові. У багатьох аспектах системний аналіз є найбільш важкою частиною процесу створення системи. Проблеми, з якими стикається системний аналітик, взаємопов'язані, і це є однією з головних причин їх важковирішуваності:

- аналітикові складно отримати вичерпну інформацію для оцінки вимог до системи з погляду замовника;

- замовник, у свою чергу, не має достатньої інформації про проблему обробки даних, щоб говорити, що є здійснимим, а що ні;

- аналітик стикається з надмірною кількістю докладних відомостей про наочну сферу і про нову систему;

- специфікація системи із-за обсягу і технічних термінів незрозуміла для замовника;

- у разі зрозумілості специфікації для замовника, вона буде недостатньою для розробників, що створюють систему.

Вирішення цих проблем може бути істотно полегшене за рахунок застосування сучасних структурних методів, серед яких центральне місце займають методології структурного аналізу.

Структурним аналізом прийнято називати метод дослідження системи за допомогою її графічного модельного уявлення, яке починається із загального огляду і потім деталізує, набуваючи ієрархічної структури із більшим числом рівнів. Для таких методів характерні:

розбиття на рівні абстракції з обмеженням числа елементів на кожному з рівнів (зазвичай від 3 до 9); обмежений контекст, що включає лише істотні на кожному рівні деталі;

дуальність даних і операцій з ними;

використання суворих формальних правил запису;

послідовне наближення до кінцевого результату.

Усі методології структурного аналізу базуються на ряді загальних принципів, частина з яких регламентує організацію робіт на початкових етапах життєвого циклу. Як два базових принципи використовуються наступні: принцип декомпозиції і принцип ієрархічного впорядкування. Дотримання зазначених принципів необхідно при організації робіт на початкових етапах життєвого циклу, незалежно від типу, що розробляється ІС, і використовуваної при цьому методології.

### **Застосування CASE-технологій при аналізі системи управління підприємством**

Нижче перераховані основні види й послідовність робіт, рекомендовані при побудові логічних моделей наочної сфери в рамках CASE-технології аналізу системи управління підприємством.

1. Проведення функціонального та інформаційного обстеження системи управління (адміністративно-управлінської діяльності) підприємства:

визначення організаційно-штатної структури підприємства;

визначення функціональної структури підприємства;

визначення переліку цільових функцій структурних елементів – підрозділів і посадовців;

визначення кола та черговості обстеження структурних елементів системи управління згідно зі сформульованими цільовими функціями;

обстеження діяльності виділених структурних елементів;

побудова FD-діаграми системи управління із зазначенням структурних елементів і функцій, реалізація яких моделюватиметься на DFD-рівні.

2. Розробка моделей діяльності структурних елементів і системи управління в цілому:

виділення безлічі зовнішніх об'єктів, що здійснюють істотний вплив на діяльність структурного елемента;

специфікація вхідних і вихідних інформаційних потоків;

виявлення основних процесів, що визначають діяльність структурного елемента і забезпечують реалізацію його цільових функцій;

специфікація інформаційних потоків між основними процесами діяльності, уточнення зв'язків між процесами та зовнішніми об'єктами;

оцінка обсягів, інтенсивності та інших необхідних характеристик інформаційних потоків;

розробка ієрархії діаграм потоків даних, які створюють функціональну модель діяльності структурного елемента;

об'єднання DFD-моделей структурних елементів в єдину модель системи управління підприємством.

3. Розробка інформаційних моделей структурних елементів і моделі інформаційного простору системи управління:

визначення суті моделі та їх атрибутів;

проведення атрибутного аналізу й оптимізація суті;

ідентифікація відносин між суттю і визначенням типів відносин;

дозвіл неспецифічних відносин;

аналіз і оптимізація інформаційної моделі;

об'єднання інформаційних моделей в єдину модель інформаційного простору.

Розробка пропозицій з автоматизації системи управління підприємства:

визначення меж автоматизації – складання переліку структурних елементів, що автоматизуються, розбиття процесів основної діяльності на автоматичні, автоматизовані і ручні;

складання переліку підсистем і окремих модулів, які входять до складу ІС, визначення способів їх взаємодії;

розробка пропозицій за черговістю проектування та реалізації підсистем і окремих модулів, які входять до складу ІС;

розробка вимог до засобів базового технічного забезпечення ІС;

розробка вимог до засобів базового програмного забезпечення ІС.

Логічна модель, що відображає діяльність системи управління підприємства й інформаційний простір, в якому ця діяльність протікає, є "знімком" положення справ (функціональна структура, ролі посадовців, взаємодія підрозділів, прийняті технології обробки управлінської інформації, автоматизовані і неавтоматизовані процеси і т. д.) на момент обстеження. Ця модель дозволяє зрозуміти, що робить і як функціонує підприємство з позицій системного аналізу, сформулювати пропозиції щодо поліпшення ситуації. Модель є не просто реалізацією початкових етапів роботи і підставою для формування технічного завдання на її подальші етапи. Вона є самостійним результатом, що має велике практичне значення.

Побудована модель є закінченим результатом з наступних причин:

1. Вона включає модель існуючої неавтоматизованої технології, прийнятої на підприємстві. Формальний аналіз цієї моделі дозволяє виявити вузькі місця в управлінні підприємством і сформулювати рекомендації з його поліпшення (незалежно від того, чи передбачається подальша розробка автоматизованої системи, чи ні).

2. Вона незалежна і відокремлювана від конкретних розробників, не вимагає супроводу і може бути безболісно передана іншим особам. Більш того, якщо з яких-небудь причин підприємство не готове до реалізації проекту в даний момент, модель може бути "покладена на полицю" до тих пір, поки в ній не виникне потреба.

3. Вона дозволяє здійснювати ефективне навчання нових працівників конкретним напрямом діяльності підприємства, оскільки відповідні технології містяться в моделі.

4. З її допомогою можна здійснювати попереднє моделювання перспективних напрямів діяльності підприємства з метою виявлення нових потоків даних, взаємодіючих процесів і структурних елементів.

5. Вона забезпечує розповсюдження накопиченого досвіду на інших підприємствах, дає можливість уніфікувати адміністративно-управлінську і фінансову діяльність цих підприємств.

Розвиток логічної моделі наочної сфери, її послідовне перетворення на модель цільової ІС дозволить інтегрувати перспективні пропозиції керівництва і провідних співробітників підприємства, експертів і системних аналітиків, сформулювати бачення нової, реорганізованої і автоматизованої діяльності підприємства.

## **Розділ 2. Стандарти структурного проектування систем сучасними CASE-засобами**

### **2.1. Типи CASE-засобів**

Виділяють наступні типи CASE-засобів:

засоби аналізу (Upper CASE) призначені для побудови і аналізу моделей наочної сфери (Design/IDEF (Meta Software), BPwin (Logic Works));

засоби аналізу і проектування (Middle CASE), які підтримують найбільш поширені методології проектування, що використовуються для створення проектних специфікацій (Vantage Team Builder (Cayenne), Designer/2000 (ORACLE), Silver-run (CSA), PRO-IV (McDonnell Douglas), CASE-аналітик);

засоби проектування баз даних (БД), що забезпечують моделювання даних і генерацію схем баз даних для найбільш поширених систем управління базами даних (БД). До них відносяться ERwin (Logic Works), S-Designor (SDP) і DataBase Designer (ORACLE). Засоби проектування баз даних є також у складі CASE-засобів Vantage Team Builder, Designer, Silverrun;

засоби розробки додатків: 4GL (Uniface (Compuware), JAM (JYACC), PowerBuilder (Sybase), Developer (ORACLE), New Era (Informix), SQL Windows (Gupta), Delphi (Borland) і ін.), і генератори кодів, що входять до складу Vantage Team Builder, Silverrun;

засоби реінжинірингу, що забезпечують аналіз програмних кодів і схем баз даних і формування на їх основі різних моделей та проектних специфікацій. Засоби аналізу схем БД і формування ERD входять до складу Vantage Team Builder, Silverrun, Designer, ERwin і S-Designor. У сфері аналізу програмних кодів найбільшого поширення набувають об'єктно-орієнтовані CASE-засоби, що забезпечують реінжиніринг програм на мові C++ (Rational Rose (Rational Software), Object Team (Cayenne)).

Більшість існуючих CASE-засобів заснована на методах структурного (функціонального) або об'єктно-орієнтованого аналізу і проектування, що використовують специфікації у вигляді діаграм або текстів для опису зовнішніх вимог, зв'язків між моделями системи,



динаміки поведінки системи та архітектури програмних засобів.

Існує більше 20 технологій проектування організаційно-технічних систем і декількох сотень інструментів, призначених для автоматизації цього процесу.

Серед давно відомих методів і мов моделювання бізнес-процесів можна назвати блок-схеми, орієнтовані графи, мережі Петрі, IDEF і SADT. Порівняно новими вважаються ARIS, UML, SPA, BPD, BPMN, XPDL, BPEL.

Пакет ARIS ToolSet – розраховано на велику кількість користувачів середовища опису і аналізу робочих процесів підприємств, що підтримує розробку складних гетерогенних інформаційних систем і супроводжує увесь цикл розробки (аналіз – проектування – реалізація). Застосування цих інструментальних засобів дозволяє скоротити тривалість етапу проектування при гарантованому рівні проектних рішень. У даному середовищі не накладається жорстких обмежень на послідовність опрацювання різних аспектів діяльності і надається ряд інших можливостей з опису даного підприємства.

BPwin – засіб функціонального моделювання, що використовує стандарт IDEF0-IDEF3, і ERwin – засіб концептуального моделювання БД на основі стандарту IDEF1X. Функціональна модель IDEF0 відображає функціональну структуру об'єкта, тобто здійснювані ним дії і зв'язки між цими діями. IDEF0 може застосовуватися для моделювання широкого кола систем і визначення вимог і функцій, а потім – для розробки системи, яка задовольняє ці вимоги і реалізує зазначені функції. BPwin також працює з організаційними діаграмами (organization charts), які дозволяють описати структуру підприємства і формуються на основі попередніх створених ролей. Крім того, в продукті BPwin 4.0 став можливим експорт моделі в систему імітаційного моделювання Arena (Systems Modeling Corp.). ERwin реалізує проектування схеми БД, генерацію її опису на мові цільової СУБД (ORACLE, Informix, Ingres, Sybase, DB/2, Microsoft SQL Server, Progress і ін.) і реінжиніринг існуючої БД. ERwin випускається в декількох різних конфігураціях, орієнтованих на найбільш поширені засоби розробки додатків 4GL. Версія ERwin/OPEN повністю сумісна із засобами розробки додатків PowerBuilder і SQLWindows і дозволяє експортувати опис спроектованої БД безпосередньо в репозитарій даних засобів. Для ряду засобів розробки

додатків (PowerBuilder, SQLWindows, Delphi, Visual Basic) виконується генерація форм і прототипів додатків. Мережна версія Erwin ModelMart забезпечує узгоджене проектування БД і додатків у рамках робочої групи.

Rational Rose – засіб автоматизації етапів аналізу і проектування ПЗ (програмного забезпечення), а також генерації кодів на різних мовах і випуску проектної документації. Rational Rose використовує методологію об'єктно-орієнтованого аналізу і проектування, засновану на підходах провідних фахівців у даній області – Рамбо і Джекобсона. Розроблена ними універсальна нотація для моделювання об'єктів (UML – Unified Modeling Language) є стандартом в області об'єктно-орієнтованого аналізу і проектування. Варіант Rational Rose визначається мовою, на якій генеруються коди програм (C++, Smalltalk, PowerBuilder, Ada, SQL Windows і ObjectPro). Основний варіант – Rational Rose/C++, що дозволяє розробляти проектну документацію у вигляді діаграм і специфікацій, а також генерувати програмні коди на C++. Крім того, Rational Rose містить засоби реінжинірингу програм, що забезпечують повторне використання програмних компонент у нових проектах.

У загальному вигляді усі CASE-засоби мають архітектуру, яка наведена на рис. 2.1.

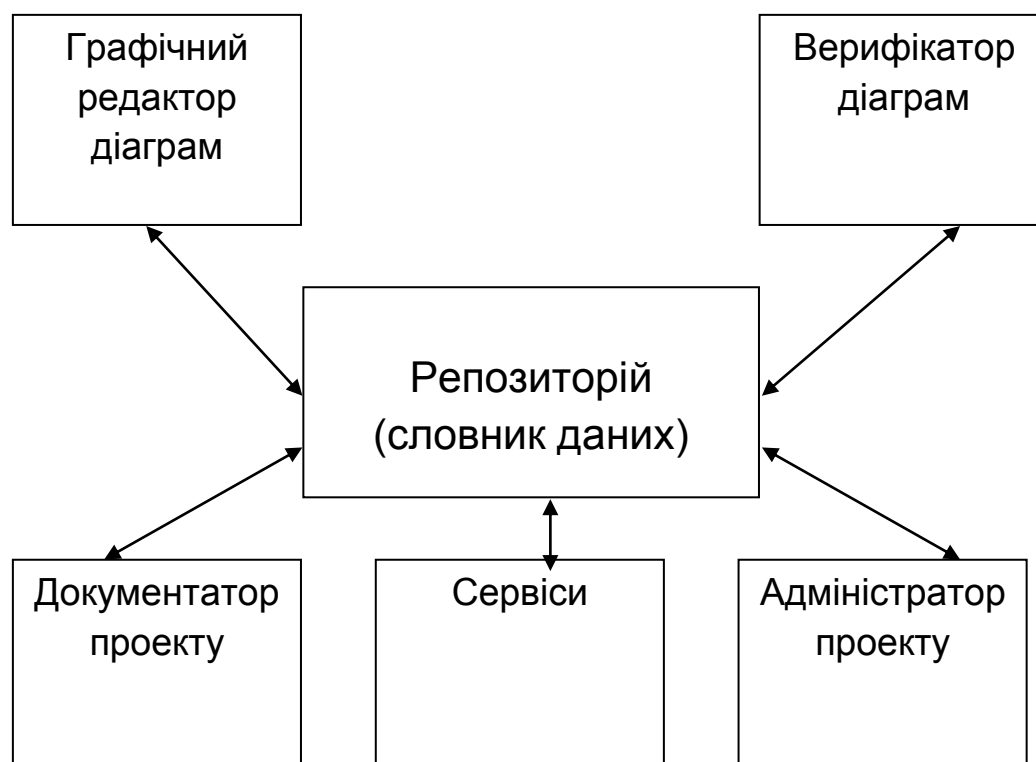


Рис. 2.1. Архітектура CASE-засобів

Також CASE-засоби використовують нотації найбільш поширених стандартів і методологій (наприклад, IDEF).

У рамках програми ICAM (ICAM – Integrated Computer Aided Manufacturing) була розроблена методологія IDEF (ICAM Definition), що дозволяє досліджувати структуру, параметри і характеристики виробничо-технічних і організаційно-економічних систем.

Загальна методологія IDEF складається з приватних методологій моделювання, заснованих на графічному уявленні систем:

IDEF0 – методологія функціонального моделювання. За допомогою наочної графічної мови IDEF0 система, перед розробниками і аналітиками у вигляді набору взаємопов'язаних функцій (функціональних блоків – в термінах IDEF0). Як правило, моделювання засобами IDEF0 є першим етапом вивчення будь-якої системи;

IDEF1 – методологія моделювання інформаційних потоків системи, що дозволяє відображати і аналізувати їх структуру і взаємозв'язки;

IDEF1X (IDEF1 Extended) – методологія побудови реляційних структур. IDEF1X відноситься до типу методологій "Сутність – взаємозв'язок" (ER – Entity – Relationship);

IDEF2 – методологія динамічного моделювання розвитку систем. У зв'язку зі складнощами аналізу динамічних систем від цього стандарту практично відмовилися, і його розвиток припинився на початковому етапі. Проте в даний час присутні алгоритми і їх комп'ютерні реалізації, що дозволяють перетворювати набір статичних діаграм IDEF0 на динамічні моделі, побудовані на базі "розфарбованих мереж Петрі" (CPN - Color Petri Nets);

IDEF3 – методологія документування процесів системи, яка використовується, наприклад, при дослідженні технологічних процесів на підприємствах. За допомогою IDEF3 описуються сценарій і послідовність операцій для кожного процесу. IDEF3 має прямий взаємозв'язок з методологією IDEF0: кожна функція (функціональний блок) може бути представлена у вигляді окремого процесу засобами IDEF3;

IDEF4 – стандарт дозволяє оптимізувати розробку додатків на основі об'єктно-орієнтованого підходу. Метод розроблений професіоналами об'єктно-орієнтованого дизайну і програмування. Найбільш важливою причиною його використання є той факт, що цей метод розглядає об'єктно-орієнтований дизайн як одну з частин розробки

складного програмного забезпечення. IDEF4 розширює процедури об'єктно-орієнтованого аналізу за рахунок фокусування на використанні графічного синтаксису і аспектах взаємодії важливих частин модельованої системи. Засоби IDEF4 дозволяють наочно відображати структуру об'єктів і закладені принципи їх взаємодії, тим самим представляючи можливість аналізувати і оптимізувати складні об'єктно-орієнтовані системи;

IDEF5 – методологія онтологічного дослідження складних систем. Застосовуючи методологію IDEF5, онтологію системи можна описати за допомогою певного словника термінів і правил, на підставі яких можуть бути сформовані достовірні твердження про стан даної системи в деякий момент часу. На базі цих тверджень формуються висновки про подальший розвиток системи і проводиться її оптимізація;

IDEF6 (Design Rationale Capture method) – стандартизації отримання знань про засіб моделювання, їх уявлення і використання при розробці систем управління підприємствами. Під знаннями розуміються причини, обставини, приховані мотиви, які обґрунтовують вибрані методи моделювання. Більшість методів моделювання фокусуються на моделях, а не на процесі їх створення. Метод IDEF6 акцентує увагу саме на процесі створення моделі;

IDEF8 – методологія розробки інтерфейсів взаємодії оператора та системи (призначених для користувача інтерфейсів). Сучасні середовища розробки призначених для користувача інтерфейсів більшою мірою створюють зовнішній вигляд інтерфейсу. IDEF8 фокусує увагу розробників інтерфейсу на програмуванні бажаної взаємної поведінки інтерфейсу і користувача на трьох рівнях: операції; сценарії взаємодії, на деталях інтерфейсу (які елементи управління пропонує інтерфейс для виконання операції);

IDEF9 (Business Constraint Discovery method) – методологія дослідження бізнес-обмежень, IDEF9 розроблено для визначення і аналізу обмежень, в умовах яких діє підприємство. Поява стандарту пов'язана з тим фактом, що при побудові моделей обмеженням, які впливають на протікання процесів на підприємстві, приділяється недостатня увага. Знання про основні обмеження і характер їх впливу залишаються неповними, неузгодженими. Це не обов'язково призводить до того, що побудовані моделі нежиттєздатні, однак їх реалізація зіткнеться з непередбаченими труднощами, внаслідок чого потенціал

буде не реалізований. У випадках, коли йдеться про вдосконалення структур або адаптації до змін, знання про існуючі обмеження мають критичне значення;

IDEF14 – методологія проектування комп'ютерних мереж, заснована на аналізі вимог, специфічних мережних компонентів, існуючих конфігурацій мереж. Також стандарт забезпечує підтримку рішень, пов'язаних з раціональним управлінням матеріальними ресурсами.

## **Стандарт IDEF0**

Методика функціонального моделювання IDEF0 є етапом розвитку графічної мови опису функціональних систем SADT (Structured Analysis and Design Technique). IDEF0 як стандарт був розроблений в 1981 р. в рамках програми автоматизації промислових підприємств, яка носила позначення ICAM (Integrated Computer Aided Manufacturing) і була запропонована департаментом військово-повітряних сил США. Сімейство стандартів IDEF успадкувало своє позначення від назви цієї програми (IDEF = ICAM DEFinition). У процесі практичної реалізації учасники програми ICAM зіткнулися з необхідністю розробки нових методів аналізу процесів взаємодії в промислових системах. При цьому, окрім вдосконаленого набору функцій для опису бізнес-процесів, однією з вимог до нового стандарту була наявність ефективної методології взаємодії в рамках "аналітик – фахівець". Новий метод повинен був забезпечити групову роботу над створенням моделі з безпосередньою участю всіх аналітиків і фахівців, зайнятих у рамках проекту. В результаті пошуку відповідних рішень з'явилася методологія функціонального моделювання IDEF0. Остання редакція IDEF0 була випущена в грудні 1993 р. Національним інститутом за стандартами і технологіями США (NIST).

Застосування IDEF базується на двох важливих чинниках:

IDEF0 дозволяє вирішувати задачі опису класифікації процесів у рамках інформаційних систем і систем менеджменту якості;

IDEF0 є стандартом для функціонального моделювання у ряді країн, що дає можливість її використання як єдиної мови опису.

Методологія IDEF0 заснована на наступних концептуальних положеннях [34]:

1. Модель – штучний об'єкт, що є відображенням (образ) системи і її компонентів. М моделює А, якщо М відповідає на питання відносно А. Модель розробляють для розуміння, аналізу і ухвалення рішень про реконструкцію або заміну того, що існує, або для проектування нової системи.

2. Система є сукупністю взаємопов'язаних і взаємодіючих частин, що виконують деяку корисну роботу. Частинами (елементами) системи можуть бути будь-які комбінації різноманітної суті, що включають людей, інформацію, програмне забезпечення, устаткування, вироби, сировину або енергію (енергоносії). Модель описує, що відбувається в системі, як нею управляють, яку суть вона відображає, які засоби використовує для виконання своїх функцій і що проводить.

3. Блокове моделювання і його графічне уявлення. Основний концептуальний принцип методології IDEF - представлення будь-якої системи, що вивчається, у вигляді набору взаємодіючих і взаємопов'язаних блоків, що відображають процеси, операції, дії (визначення – див. нижче), що відбуваються в системі, що вивчається. У IDEF0 все, що відбувається в системі і її елементах, прийнято називати функціями. Кожній функції ставиться у відповідність блок. На IDEF0-діаграмі, основним блоком є процес (прямокутник). Інтерфейси, за допомогою яких блок взаємодіє з іншими блоками або із зовнішнім по відношенню до модельованої системи середовищем, представляються інтерфейсними дугами (стрілками), що входять у блок або виходять з нього. Вхідні стрілки показують, які умови повинні бути одночасно виконані, щоб функція, що описується блоком, здійснилася.

4. Лаконічність і точність. Документація, що описує систему, повинна бути точною і лаконічною. Графічна мова дозволяє лаконічно, одно-значно і точно показати всі елементи (блоки) системи і всі відносини та зв'язки між ними, виявити помилкові, зайві або дублюючі зв'язки і т. д.

5. Передача інформації. Засоби IDEF0 полегшують передачу інформації від одного учасника розробки моделі (окремого розробника або робочої групи) до іншого. До таких засобів належать:

діаграми, які засновані на простій графіці блоків і стрілок, досить легко розуміються;

глосарій і супровідний текст для уточнення сенсу елементів діаграми;

послідовна декомпозиція діаграм, що будується за ієрархічним принципом, при якому на верхньому рівні відображаються основні функції, а потім відбувається їх деталізація та уточнення;

деревоподібні схеми ієрархії діаграм і блоків, що забезпечують опис моделі в цілому і вхідних в неї деталей.

6. Суворість і формалізм. Розробка моделей IDEF0 вимагає дотримання ряду суворих формальних правил, що забезпечують переваги методології щодо однозначності, точності й цілісності складних багаторівневих моделей: всі стадії й етапи розробки та коригування моделі повинні формально документуватися для того, щоб при її експлуатації не виникало питань, пов'язаних з неповнотою або некоректністю документації.

7. Ітеративне моделювання. Розробка моделі в IDEF0 є ітеративною процедурою. На кожному кроці ітерації розробник пропонує варіант моделі, який обговорюється, рецензується і у подальшому редагується, після чого цикл повторюється. Така організація роботи сприяє оптимальному використанню знань системного аналітика, що володіє методологією і технікою IDEF0, і знань фахівців – експертів в наочній сфері, до якої відноситься об'єкт моделювання.

8. Відокремлення "організації" від "функцій". При розробці моделей слід уникати початкової "прив'язки" функцій досліджуваної системи до існуючої організаційної структури об'єкта (підприємства, фірми, системи). Це допомагає уникнути суб'єктивної точки зору, нав'язаної організацією та її керівництвом. Організаційна структура повинна бути запропонована як результат використання (застосування) моделі. Порівняння результату з існуючою структурою дозволяє, по-перше, оцінити адекватність моделі, а по-друге – запропонувати рішення, спрямовані на вдосконалення цієї структури.

Основні визначення (поняття) методології та мови IDEF0 наведені в глосарії.

### **Синтаксис графічної мови IDEF0**

Набір структурних компонентів мови, їх характеристики і правила, що визначають зв'язки між компонентами, є синтаксисом мови. Компоненти синтаксису IDEF0 - блоки, стрілки, діаграми і правила. Блоки

представляють функції, сформульовані як діяльність, процес, операція, дія або перетворення. Стрілки представляють дані або матеріальні об'єкти, пов'язані з функціями. Правила визначають, як слід застосовувати компоненти. Діаграми забезпечують формат графічного і словесного опису моделей. Формат утворює основу для управління конфігурацією моделі [34, 37].

### **Блок**

Блок описує функцію. У середині кожного блоку поміщається його ім'я і номер. Ім'я повинно бути активним дієсловом або дієприкметниковим зворотом, що описує функцію. Номер блоку розміщується у правому нижньому кутку. Номери блоків використовуються для їх ідентифікації на діаграмі і у відповідному тексті. Розміри блоків повинні бути достатніми для того, щоб включити ім'я блоку. Блоки мають бути прямокутними, з прямими кутами. Блоки повинні бути намальовані суцільними лініями.

### **Інтерфейсна дуга (стрілка)**

Як показано на рис. 2.2, сегменти стрілок можуть бути прямими або ламаними; в останньому випадку горизонтальні й вертикальні відрізки стрілки сполучаються дугами, що мають кут  $90^\circ$ . Стрілки не становлять потік або послідовність подій, як в традиційних блок-схемах потоків або процесів, вони лише показують, які дані або матеріальні об'єкти повинні надійти на вхід функції для того, щоб ця функція могла виконуватися.

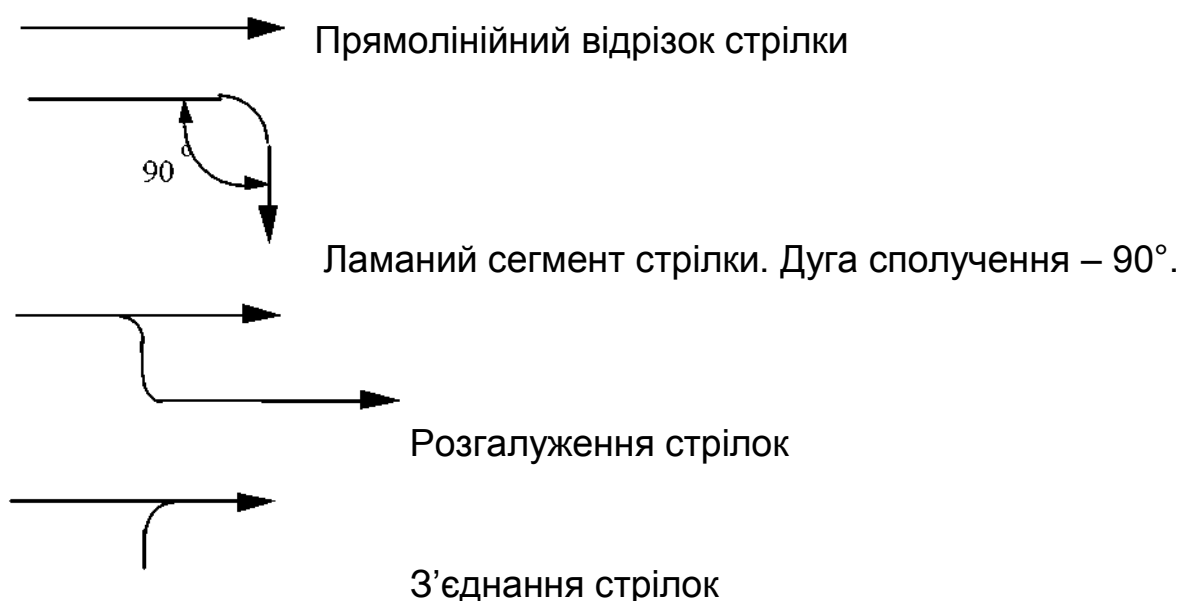


Рис. 2.2. Синтаксис стрілок



Ламані стрілки змінюють напрям тільки під кутом 90°. Стрілки повинні бути намальовані суцільними лініями різної товщини. Стрілки можуть складатися тільки з вертикальних або горизонтальних відрізків; відрізки, направлені по діагоналі не допускаються. Кінці стрілок повинні стосуватися зовнішньої межі функціонального блоку, але не повинні перетинати її. Стрілки мають приєднуватися до блоку на його сторонах. Приєднання в кутках не допускається.

### ***Семантика мови IDEF0***

Семантика визначає зміст (значення) синтаксичних компонентів мови і сприяє правильності їх інтерпретації. Інтерпретація встановлює відповідність між блоками і стрілками, з одного боку, і функціями та їх інтерфейсами – з іншого.

### **Семантика блоків і стрілок**

Оскільки IDEF0 є методологією функціонального моделювання, ім'я блоку, що описує функцію, повинне бути дієсловом або дієприкметниковим зворотом; наприклад, ім'я блоку "Виконати перевірку" означає, що блок з таким ім'ям перетворює неперевірені деталі на перевірені. Після привласнення блоку імені, до відповідних його сторін приєднуються вхідні, вихідні й такі стрілки, що управляють, а також стрілки механізму, що і визначає наочність і виразність зображення блоку IDEF0.

Щоб гарантувати точність моделі, слід використовувати стандартну термінологію. Блоки називають дієсловами або дієприкметниковими зворотами і ці імена зберігаються при декомпозиції Стрілки і їх сегментів, як окремі, так і зв'язані в "пучок", позначаються іменниками або оборотами іменника. Мітки сегментів дозволяють конкретизувати дані або матеріальні об'єкти, що передаються цими сегментами з дотриманням синтаксису розгалужень і злиття.

Кожна сторона функціонального блоку має стандартне значення з погляду зв'язку блок/стрілка. У свою чергу сторона блоку, до якої приєднана стрілка, однозначно визначає її роль. Стрілки, що входять в ліву сторону блоку – входи. Входи перетворюються або витратяться функцією, щоб створити те, що з'явиться на її виході. Стрілки, що входять в блок зверху визначають управління. Управління визначають умови, необхідні функції, щоб провести правильний вихід. Стрілки, що

залишають блок справа – виходи, тобто дані або матеріальні об'єкти, проведені функцією.

Стрілки, підключені до нижньої сторони блоку, становлять механізми. Стрілки, направлені вгору, ідентифікують засоби, що підтримують виконання функції. Інші засоби можуть успадковуватися з батьківського блоку. Стрілки механізму, направлені вниз, є стрілками виклику. Стрілки виклику позначають звернення з даної моделі або з даної частини моделі до блоку, що входить до складу іншої моделі або іншої частини моделі, забезпечуючи їх зв'язок, тобто різні моделі або різні частини однієї й тієї ж моделі можуть спільно використовувати один і той же елемент (блок). Стандартне розташування стрілок показано на рис. 2.3.

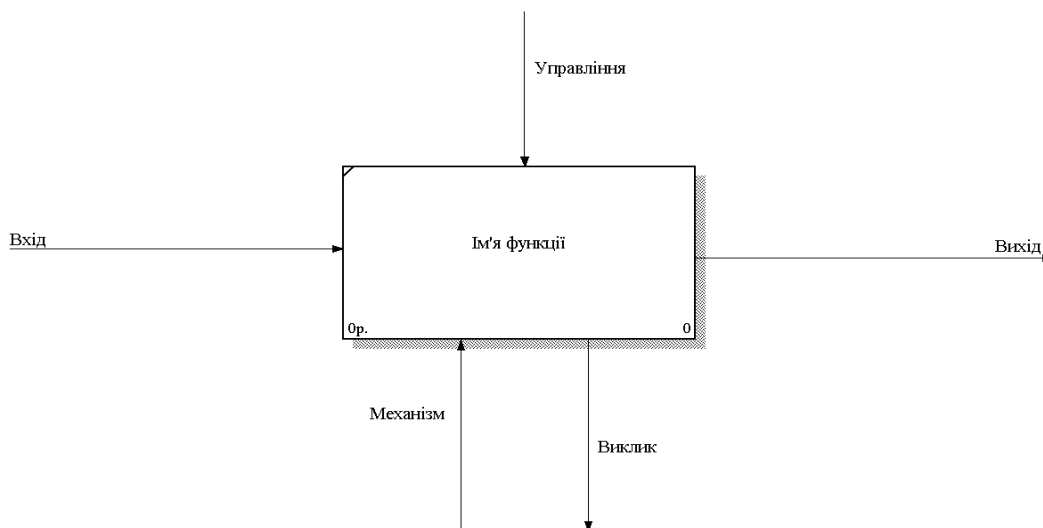


Рис. 2.3. Блок процесу та інтерфейсні дуги (стрілки)

Імена функцій повинні бути дієсловами або дієслівними зворотами. Приклади таких імен: обробляти деталі, планувати ресурси, проектувати систему. Стрілки ідентифікують дані або матеріальні об'єкти, необхідні для виконання функції або вироблювані нею. Кожна стрілка повинна бути помічена іменником або оборотом іменника, наприклад: специфікації, бюджет, директива, інженер-конструктор.

### ***Семантичні правила блоків і стрілок***

Ім'я блоку повинне бути активним дієсловом або дієслівним оборотом.

Кожна сторона функціонального блоку повинна мати стандартне відношення блок / стрілки:

вхідні стрілки повинні зв'язуватися з лівою стороною блоку;

стрілки, що управляють, повинні зв'язуватися з верхньою стороною

блоку;

вихідні стрілки повинні зв'язуватися з правою стороною блоку;

стрілки механізму (окрім стрілок виклику) повинні вказувати вгору і підключатися до нижньої сторони блоку;

стрілки виклику повинні вказувати вниз, підключатися до нижньої сторони блоку і позначатися посиланням на блок, що викликається.

Сегменти стрілок, за винятком стрілок виклику, повинні позначатися іменником або оборотом іменника, якщо тільки єдина мітка стрілки поза сумнівом не відноситься до стрілки в цілому.

Щоб пов'язати стрілку міткою, слід використовувати "тильду".

У мітках стрілок не повинні використовуватися наступні терміни: функція, вхід, управління, вихід, механізм, виклик.

## Діаграми IDEF0

IDEF0-моделі складаються з трьох типів документів [34, 37]: графічних діаграм, тексту і глосарію. Наведені документи мають перехресні посилання один на одного. Графічна діаграма – головний компонент IDEF0-моделі, що містить блоки, стрілки, з'єднання блоків і стрілок і асоційовані з ними відносини. Блоки представляють основні функції модельованого об'єкта. Ці функції можуть бути розбиті (декомпозовані) на складові частини і представлені у вигляді докладніших діаграм; процес декомпозиції продовжується, поки об'єкт не буде описаний на рівні деталізації, необхідної для досягнення мети конкретного проекту. Діаграма верхнього рівня забезпечує найбільш загальний або абстрактний опис об'єкта моделювання. За цією діаграмою слідує серія дочірніх діаграм, що дають детальніше уявлення про об'єкт.

Кожна модель повинна мати контекстну діаграму верхнього рівня, на якій об'єкт моделювання представлений єдиним блоком з граничними стрілками. Ця діаграма називається A–0 (A мінус нуль). Стрілки на цій діаграмі відображають зв'язки об'єкта моделювання з навколишнім середовищем. Оскільки єдиний блок представляє об'єкт, його ім'я – загальне для всього проекту. Це ж справедливо і для всіх стрілок діаграми, оскільки вони представляють повний комплект зовнішніх інтерфейсів об'єкта. Діаграма A–0 встановлює область моделювання і її межу. Приклад діаграми A–0 показаний на рис. 2.4.

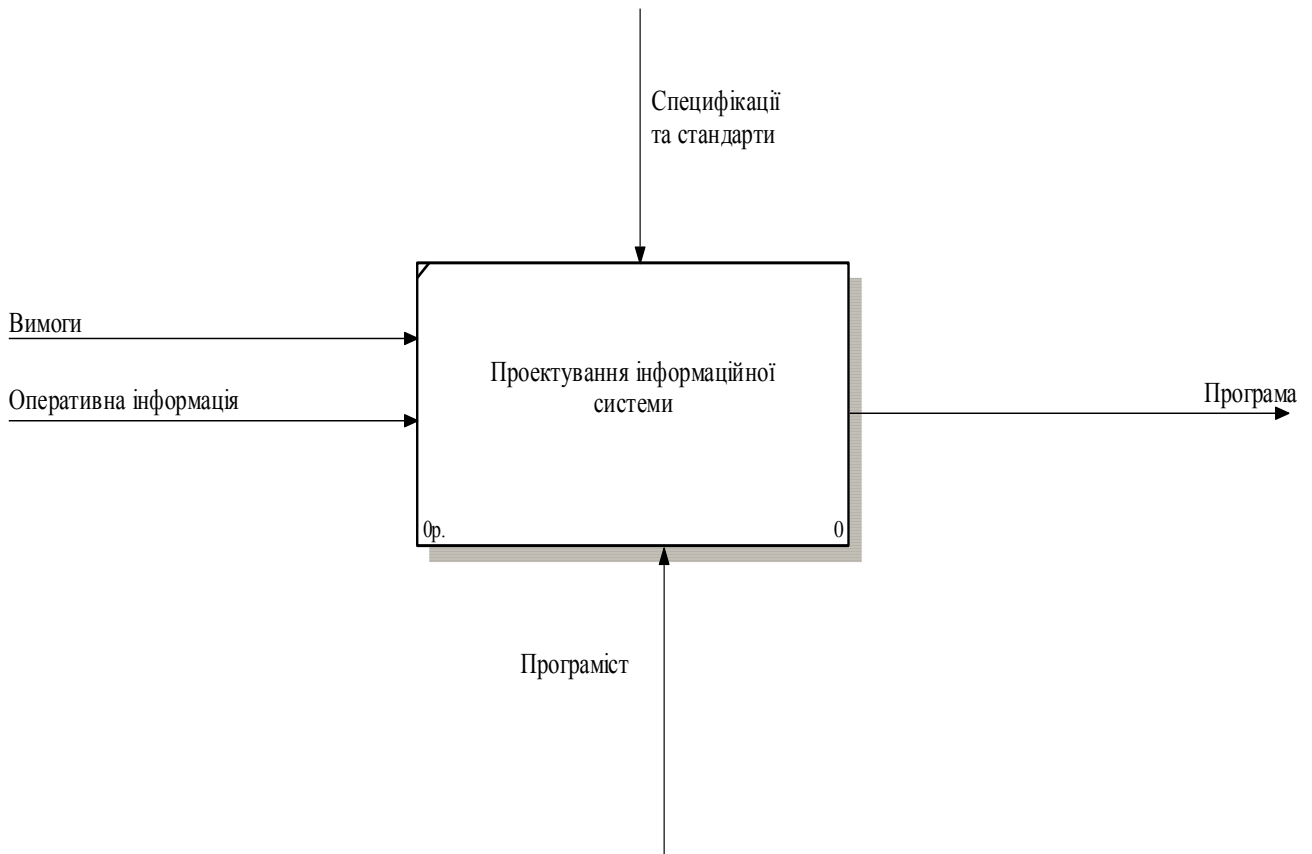


Рис. 2.4. Приклад контекстної діаграми

Контекстна діаграма А–0 також повинна містити короткі твердження, що визначають точку зору посадовця або підрозділу, з позицій якого створюється модель, і мета, для досягнення якої її розробляють. Ці твердження допомагають керувати розробкою моделі і ввести цей процес у певні рамки. Точка зору визначає, що і в якому розрізі можна побачити в межах контексту моделі. Зміна точки зору приводить до розгляду інших аспектів об'єкта. Аспекти важливі з однієї точки зору можуть не з'явитися в моделі, що розробляється з іншої точки зору на той же самий об'єкт.

Формулювання мети визначає причину створення моделі, тобто містить перелік питань, на які повинна відповідати модель. Найбільш важливі властивості об'єкта зазвичай виявляються на верхніх рівнях ієрархії; при декомпозиції функції верхнього рівня і розбиття її на підфункції, ці властивості уточнюються. Кожна підфункція, у свою чергу, декомпонується на елементи наступного рівня, і так відбувається поки не буде отримана релевантна структура, що дозволяє відповісти на

питання, сформульовані в меті моделювання. Кожна підфункція моделюється окремим блоком. Кожен батьківський блок детально описується дочірньою діаграмою на нижчому рівні. Всі дочірні діаграми повинні бути в межах області контекстної діаграми верхнього рівня.

Єдина функція, представлена на контекстній діаграмі верхнього рівня, може бути розкладена на основні підфункції за допомогою створення дочірньої діаграми. У свою чергу, кожна з цих підфункцій може бути розкладена на складові частини за допомогою створення дочірньої діаграми наступного, нижчого рівня, на якій деякі або всі функції також можуть бути розкладені на складові частини. Кожна дочірня діаграма містить дочірні блоки і стрілки, що забезпечують додаткову деталізацію батьківського блоку.

Дочірня діаграма, що створюється при декомпозиції, охоплює ту ж область, що й батьківський блок, але описує її детальніше. Таким чином, дочірня діаграма ніби вкладена у свій батьківський блок. Ця структура ілюструється на рис. 2.5. Батьківська діаграма – та, яка містить один або більше батьківських блоків. Кожна звичайна (неконтекстна) діаграма є також дочірньою діаграмою, оскільки за визначенням, вона детально описує деякий батьківський блок. Таким чином, будь-яка діаграма може бути як батьківською діаграмою (містити батьківські блоки), так і дочірня (детально описувати власний батьківський блок). Аналогічно блок може бути як батьківським (детально описуватися дочірньою діаграмою), так і дочірнім (що з'являється на дочірній діаграмі). Основне ієрархічне відношення існує між батьківським блоком і дочірньою діаграмою, яка його детально описує (рис. 2.5).

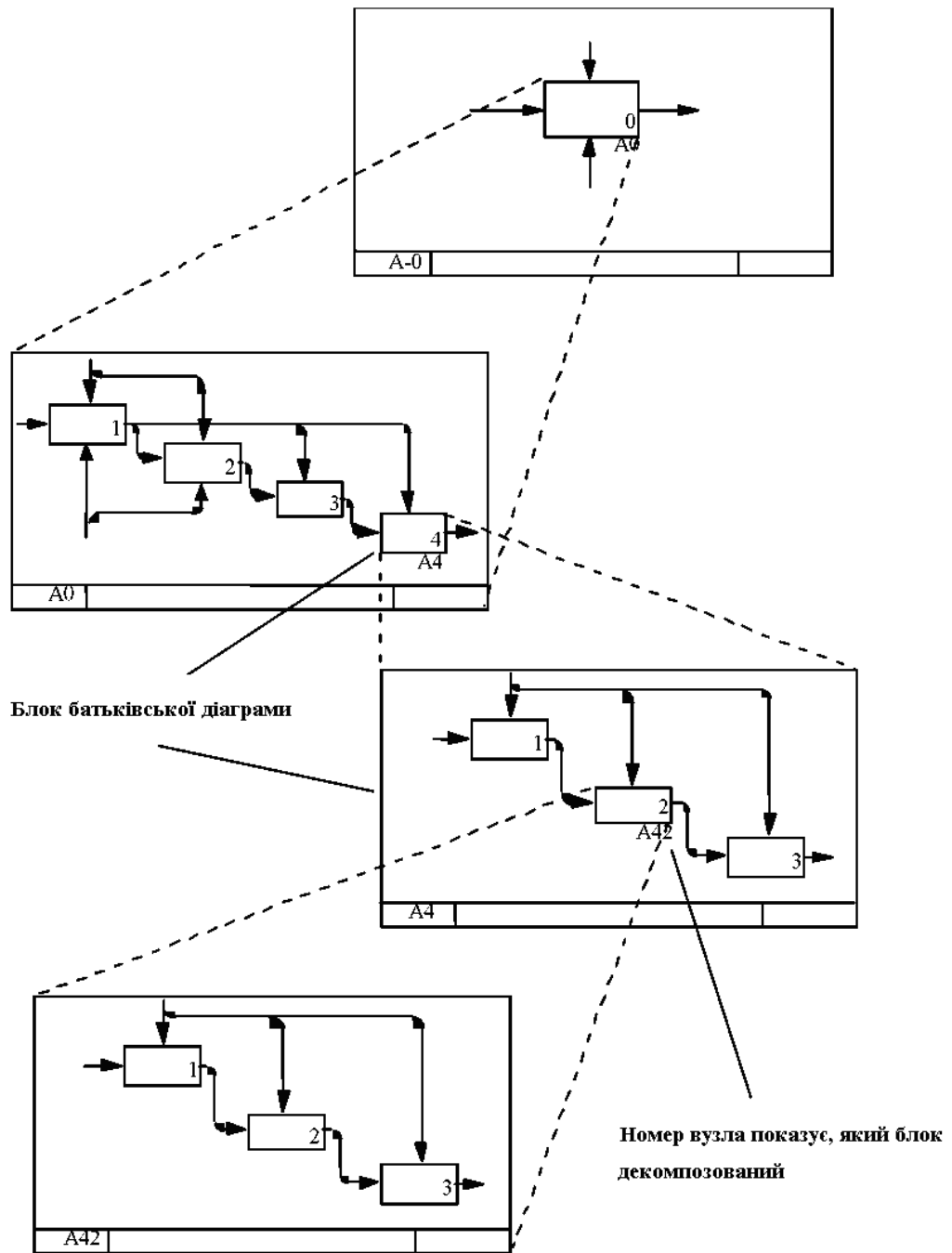


Рис. 2.5. Процес декомпозиції контекстної діаграми

Те, що блок є дочірнім і розкриває зміст батьківського блоку на діаграмі попереднього рівня, вказується спеціальним посилальним кодом, який вказано у правому нижньому куті блоку. Цей посилальний код може формуватися декількома способами, з яких найпростіший полягає в тому, що код, який починається з букви А (на ім'я діаграм А-0), містить цифри, що визначаються номерами батьківських блоків.

Наприклад, показані на рис. 2.6 коди означають, що діаграма є декомпозицією 1-го блоку діаграми, яка, у свою чергу, є декомпозицією 6-го блоку діаграми A0, а коди утворюються приєднанням номера блоку.

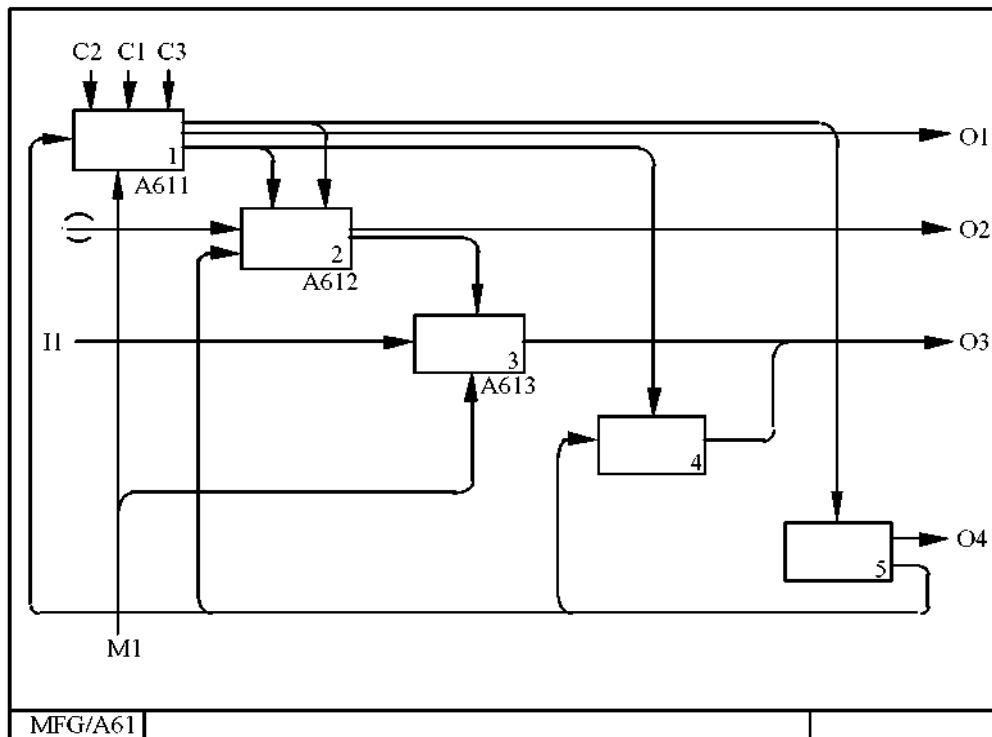


Рис. 2.6. Система кодування декомпованих діаграм

### Текст і глосарій

Діаграмі може бути поставлений у відповідність структурований текст, що є коротким коментарем до змісту діаграми. Текст використовується для пояснень і уточнень характеристик, потоків, внутрішньоблокових з'єднань. Текст не повинен використовуватися для опису і без того зрозумілих блоків і стрілок на діаграмах.

Глосарій призначений для визначення абревіатур (акронімів), ключових слів і фраз, що використовуються як імена і мітки на діаграмах. Глосарій визначає поняття і терміни, які повинні бути однаково зрозумілими всіма учасниками розробки і користувачами моделі, щоб правильно інтерпретувати її зміст.

**Діаграми - ілюстрації (FEO).** Ці діаграми використовуються як доповнення, що пояснюють специфіку змісту основних діаграм в тих випадках, коли це необхідно. Діаграма FEO не повинна підкорятися синтаксичним правилам IDEF0.

## Властивості діаграм

Стрілки на діаграмі IDEF0 описують дані або матеріальні об'єкти та одночасно задають обмеження (умови) для діаграми [34]. Вхідні та управляючі стрілки блоку описують умови, які повинні бути виконані для того, щоб реалізувалася функція, записана як ім'я блоку.

Рис. 2.7 ілюструє випадок, при якому "функція 3" може бути виконана тільки після отримання даних, вироблених "функцією 1" і "функцією 2".

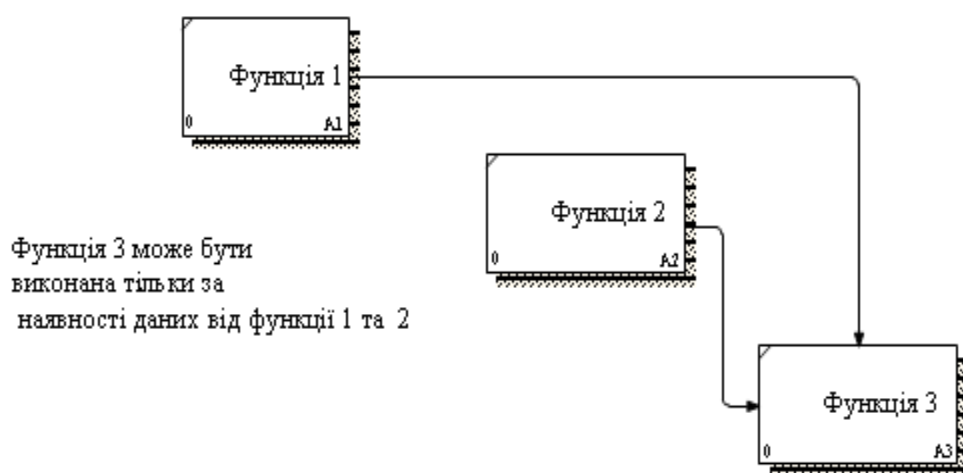


Рис. 2.7. Визначення умов виконання функції 3

### Розгалуження і злиття сегментів стрілок

Розгалуження і злиття стрілок покликане зменшити завантаженість діаграм графічними елементами (лініями). Щоб стрілки і їх сегменти правильно описували зв'язки між блоками - джерелами і блоками - споживачами, використовується апарат їх визначення. Мітки зв'язуються із сегментами за допомогою тильд. При цьому між сегментами виникають певні відносини, описані нижче.

Непомічені сегменти (рис. 2.8) містять всі об'єкти, вказані в мітці стрілки перед розгалуженням (всі об'єкти належать кожному із сегментів);



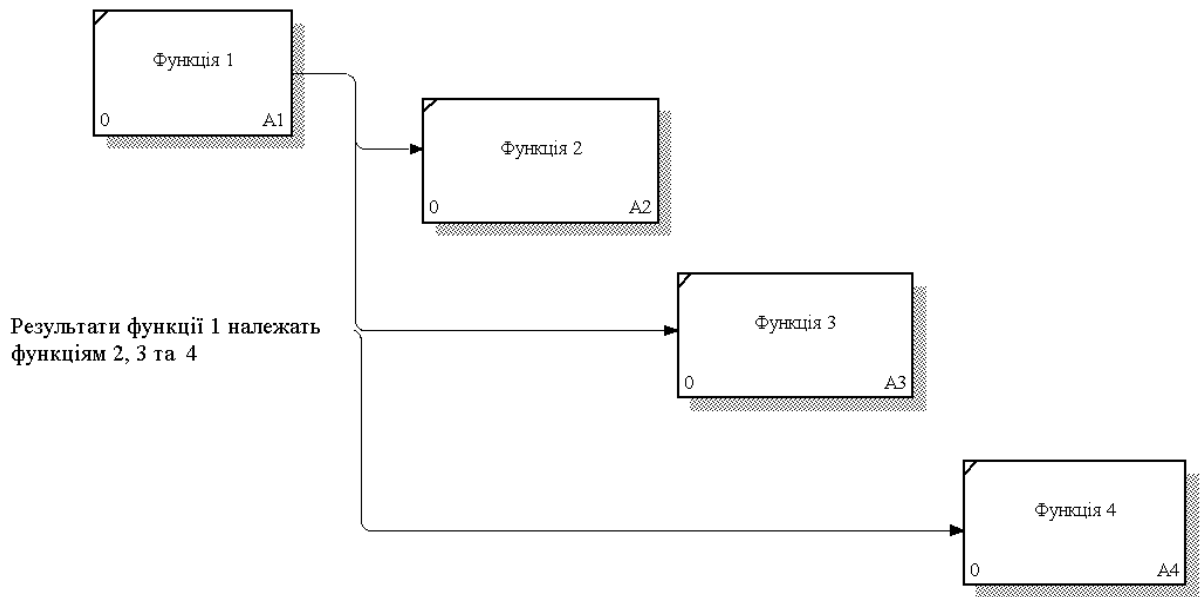


Рис. 2.8. **Всі об'єкти належать кожному із сегментів**

Сегменти, помічені після точки розгалуження (рис. 2.9), містять всі об'єкти вказані в мітці стрілки перед розгалуженням або їх частину, що описується міткою кожного конкретного сегменту.

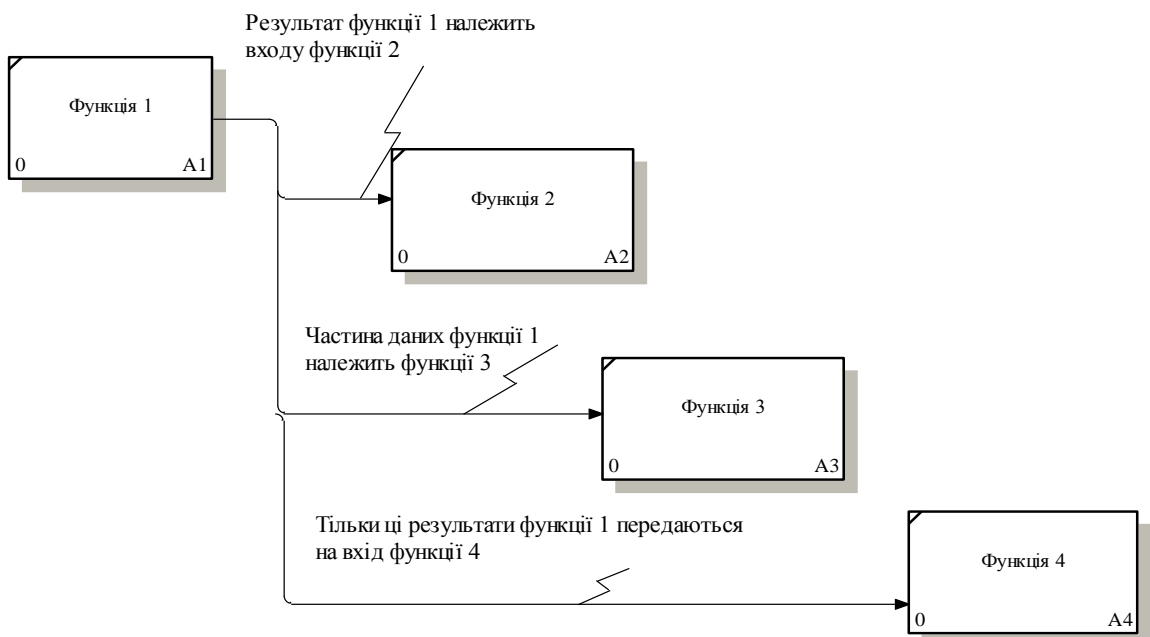


Рис. 2.9. **Опис мітки кожного конкретного сегменту**

При злитті непомічених сегментів об'єднаний сегмент стрілки містить всі об'єкти, що належать сегментам, які злилися і вказані в загальній мітці стрілки після злиття (рис. 2.10);

при злитті помічених сегментів (рис. 2.11) об'єднаний сегмент містить всі або деякі об'єкти, що належать сегментам, які злиті, і перераховані в загальній мітці після злиття; якщо загальна мітка після злиття відсутня, це означає, що загальний сегмент передає всі об'єкти, що належать сегментам, які були злиті.

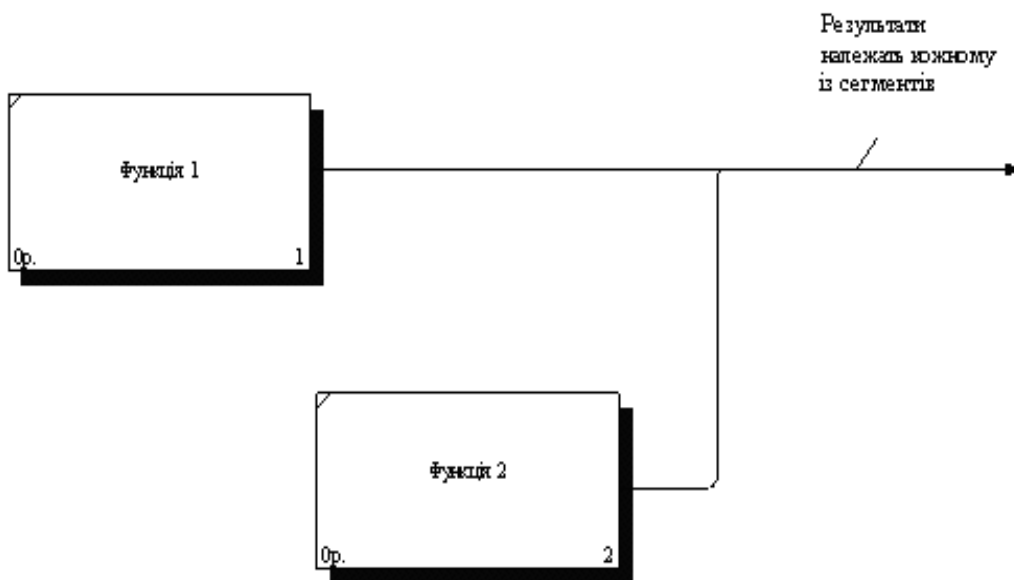


Рис. 2.10. Злиття непомічених сегментів

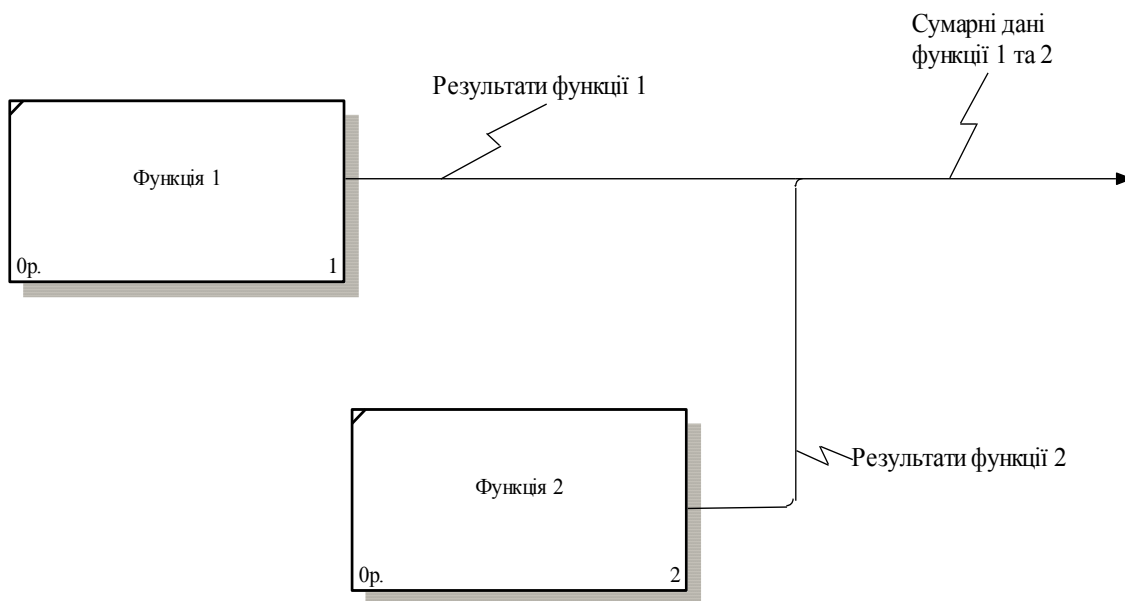


Рис. 2.11. Злиття помічених сегментів

### Відносини блоків на діаграмах

У методології IDEF0 існує 6 (шість) типів відносин між блоками в межах однієї діаграми:

- домінування;
- управління;
- вихід–вхід;
- зворотний зв'язок по управлінню;
- зворотний зв'язок по входу;
- вихід–механізм.

Передбачається, що блоки, розташовані на діаграмі вище і лівіше, "домінують" над блоками, які розташовані нижче і правіше. "Домінування" розуміється як вплив, який один блок здійснює на інші блоки діаграми.

Решту п'ять відносин описують зв'язки між блоками і зображаються відповідними стрілками.

Відносини управління "вихід–вхід" є простими, оскільки відображають прямі взаємодії, які зрозумілі й очевидні.

Відношення управління (рис. 2.12) виникає тоді, коли вихід одного блоку служить дією, що управляє, на блок з меншим домінуванням.

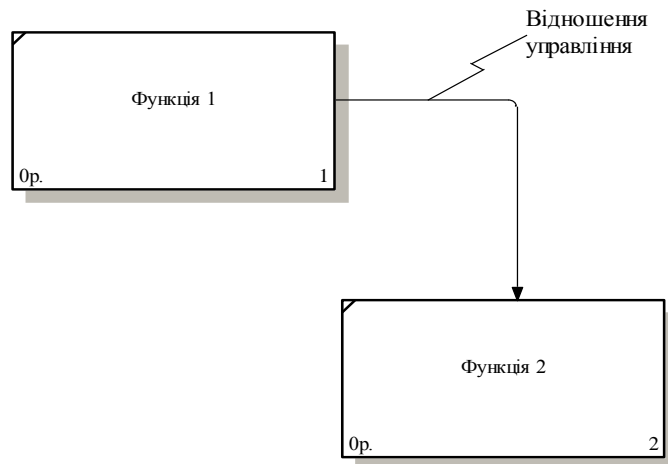


Рис. 2.12. Відношення управління

Відношення "вихід–вхід" (рис. 2.13) виникає при з'єднанні виходу одного блоку з входом іншого блоку з меншим домінуванням. Зворотний зв'язок по управлінню і зворотний зв'язок по входу є складнішими типами відносин, оскільки вони становлять ітерацію (вихід функції впливає на

майбутнє виконання інших функцій з великим домінуванням, що згодом впливає на початкову функцію).

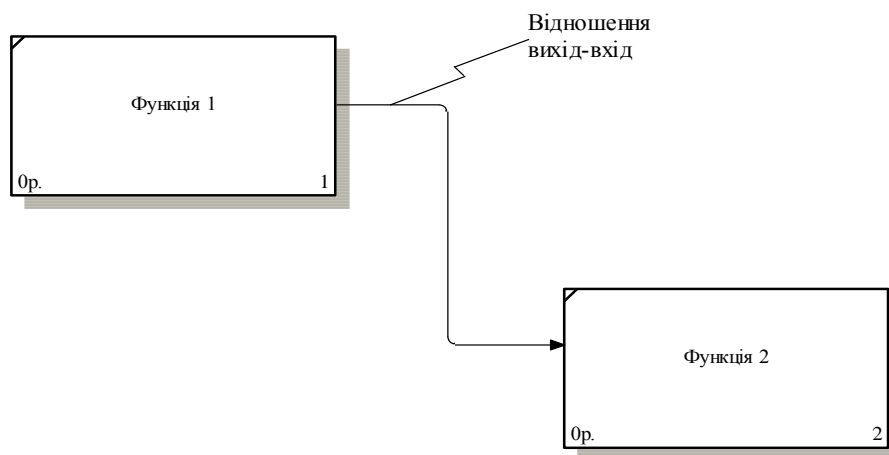


Рис. 2.13. Відношення вихід – вхід

Зворотний зв'язок по управлінню (рис. 2.14) виникає тоді, коли вихід деякого блоку формує результат, який є управлінням на блок з великим домінуванням.



Рис. 2.14. Зворотний зв'язок по управлінню

Відношення зворотного зв'язку по входу (рис. 2.15) має місце тоді, коли вихід блоку є входом іншого блоку з великим домінуванням.

Зв'язки "вихід–механізм" (рис. 2.16) відображають ситуацію, за якої вихід однієї функції є засобом досягнення мети для іншої.

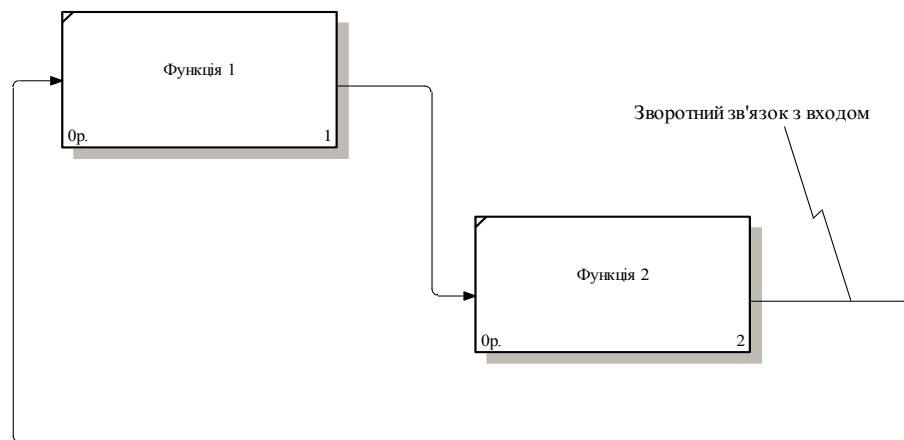


Рис. 2.15. Відношення зворотного зв'язку по входу

Зв'язки "вихід–механізм" виникають при відображенні в моделі процедур поповнення і розподілу ресурсів, створення або підготовки засобів для виконання функцій системи (наприклад, придбання або виготовлення необхідних інструментів і устаткування, навчання персоналу, фінансування, закупівля матеріалів).

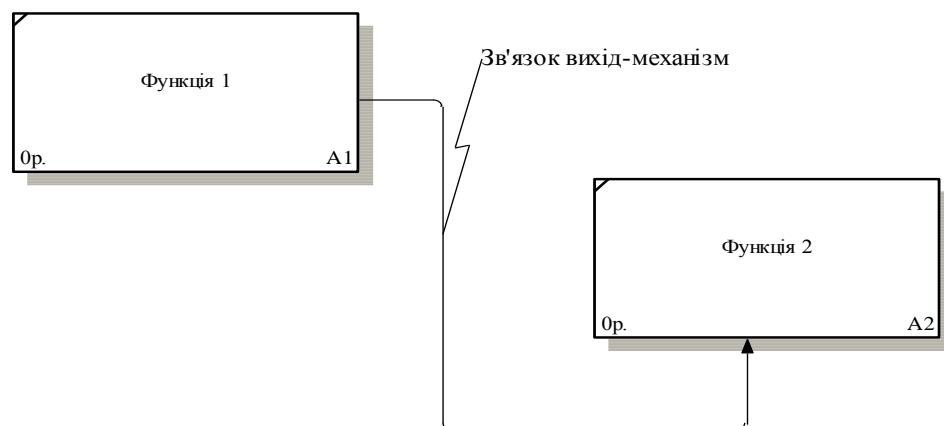


Рис. 2.16. Зв'язки "вихід–механізм"

*Відношення між блоками діаграми та іншими діаграмами  
(навколишнім середовищем)*

Усі описані вище відносини відображаються внутрішніми стрілками, тобто такими, у яких обидва кінці пов'язані з блоками діаграми.

Відношення між блоками діаграми та іншими діаграмами, що є по відношенню до даної діаграми навколишнім середовищем (оточенням), описуються граничними стрілками.

На звичайній (не контекстній) діаграмі граничні стрілки представляють входи, управління, виходи або механізми батьківського блоку діаграми. Джерело або споживачів граничних стрілок можна виявити тільки вивчаючи батьківську діаграму. Всі граничні стрілки на дочірній діаграмі, за винятком стрілок поміщених в тунель, повинні відповідати стрілкам батьківського блоку (рис. 2.17).

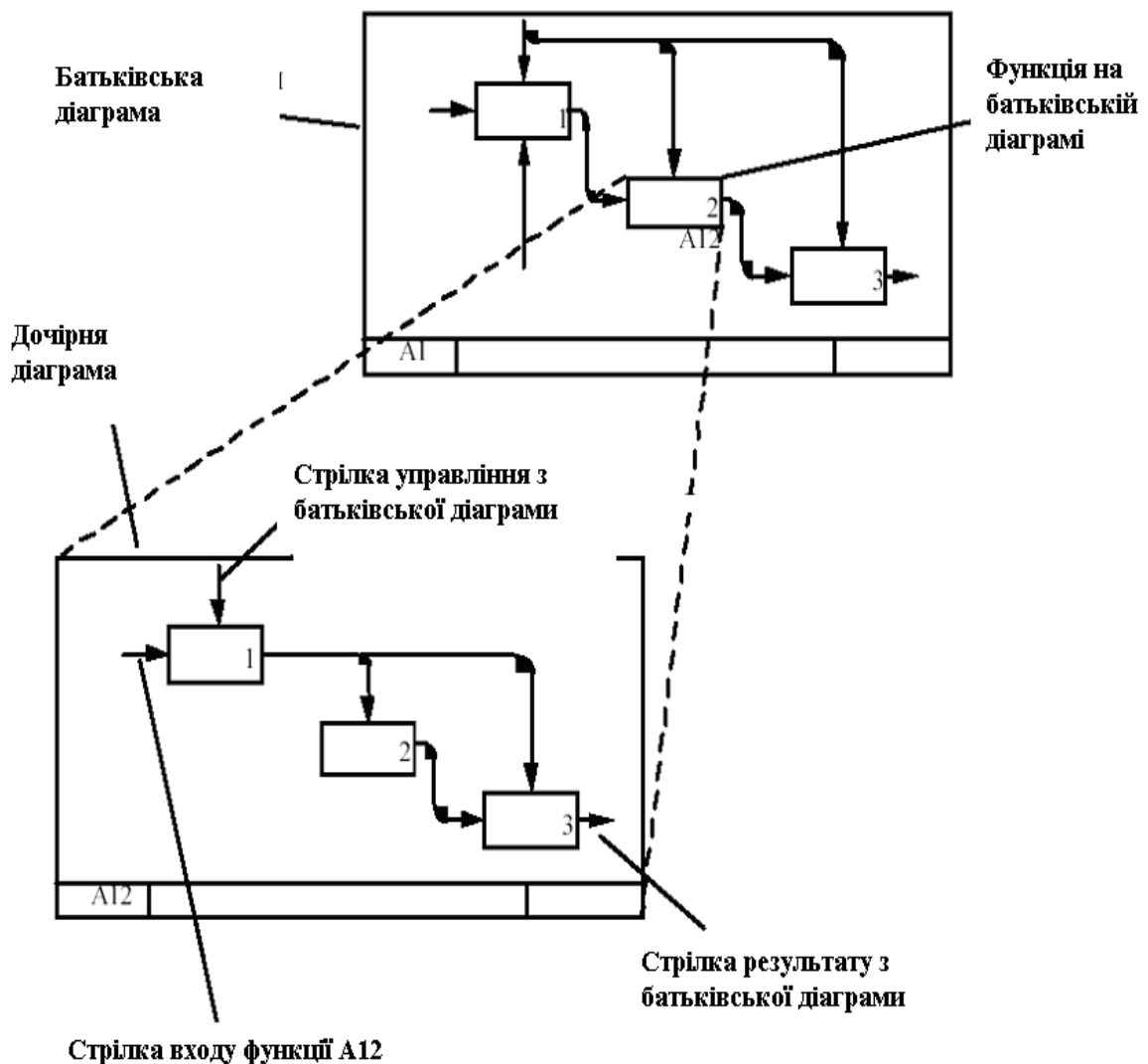


Рис. 2.17. Перехід стрілок (інтерфейсних дуг) на діаграмах з батьківської на дочірню

ISOM – кодування граничних стрілок.

ICOM – коди зв'язують граничні стрілки на дочірній діаграмі із стрілками батьківського блоку. Нотація, названа ICOM - кодом, визначає значення з'єднань [34, 37]. Букви I, C, O або M, написані біля незв'язаного кінця граничної стрілки на дочірній діаграмі ідентифікують стрілку як Вхід (Input), Управління (Control), Вихід (Output) або Механізм (Mechanism) в батьківському блоці. Буква слідує за числом, що визначає відносне положення точки підключення стрілки до батьківського блоку; це положення визначається зліва направо або зверху вниз. Наприклад, код "С3", написаний біля граничної стрілки на дочірній діаграмі, указує, що ця стрілка відповідає третій (вважаючи зліва) стрілці управління батьківського блоку.

Це кодування зв'язує кожен дочірню діаграму зі своїм батьківським блоком. Якщо блоки на дочірній діаграмі піддаються подальшій декомпозиції і детально описуються на дочірніх діаграмах наступного рівня, то на кожен нову діаграму призначаються нові ICOM - коди, що зв'язують граничні стрілки цих діаграм із стрілками їх батьківських блоків.

Іноді буквені ICOM - коди, що визначають ролі граничних стрілок (вхід, управління, механізм), можуть змінюватися при переході від батьківського блоку до дочірньої діаграми. Наприклад, стрілка управління в батьківському блоці може бути входом на дочірній діаграмі. Аналогічно, вхід батьківського блоку може бути управлінням для одного або більше дочірніх блоків. Приклади зміни ролей стрілок можна бачити на рис. 2.18.

Стрілки, поміщені в "тунель".

Тунель – круглі дужки на початку і/або в кінці стрілки. Тунельні стрілки означають, що дані, виражені цими стрілками, не розглядаються на батьківській діаграмі і/або на дочірній діаграмі.

Стрілка, поміщена в тунель там, де вона приєднується до блоку (рис. 2.19), означає, що дані, виражені цією стрілкою, не обов'язкові на наступному рівні декомпозиції.

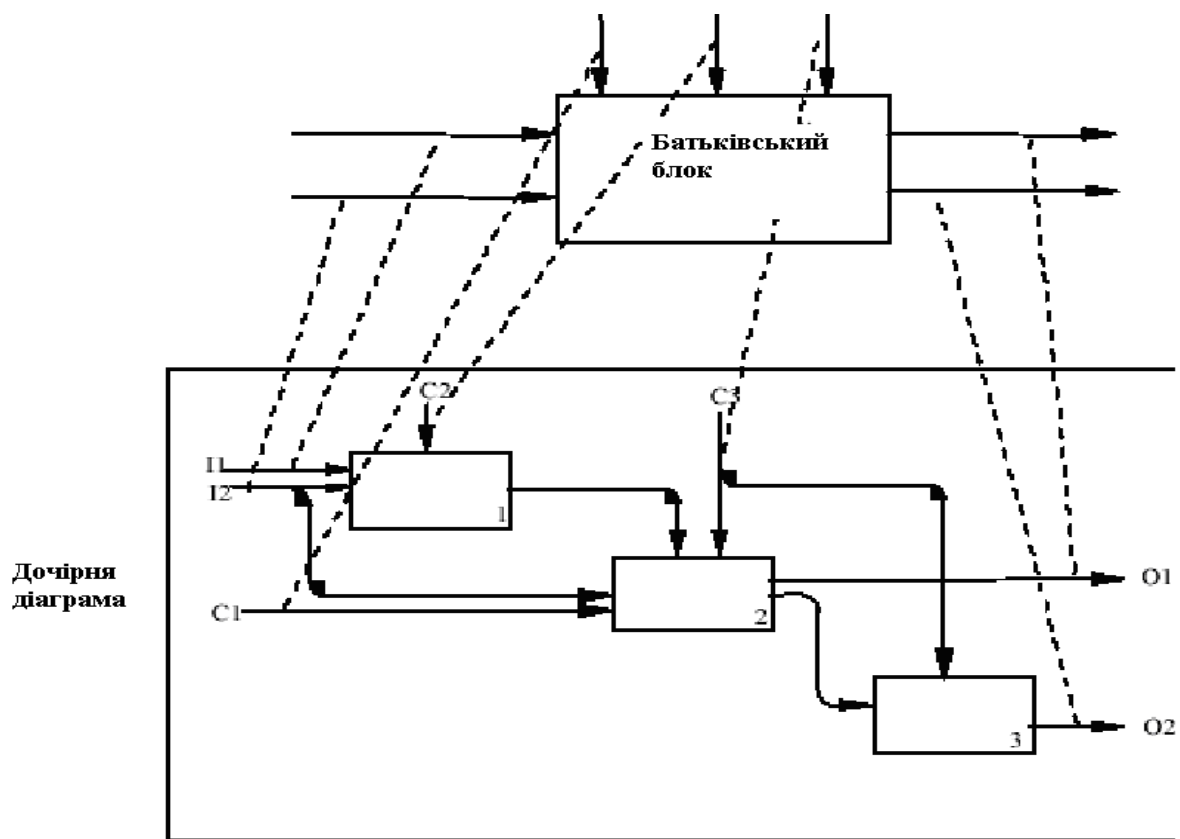


Рис. 2.18. Визначення кодів стрілок

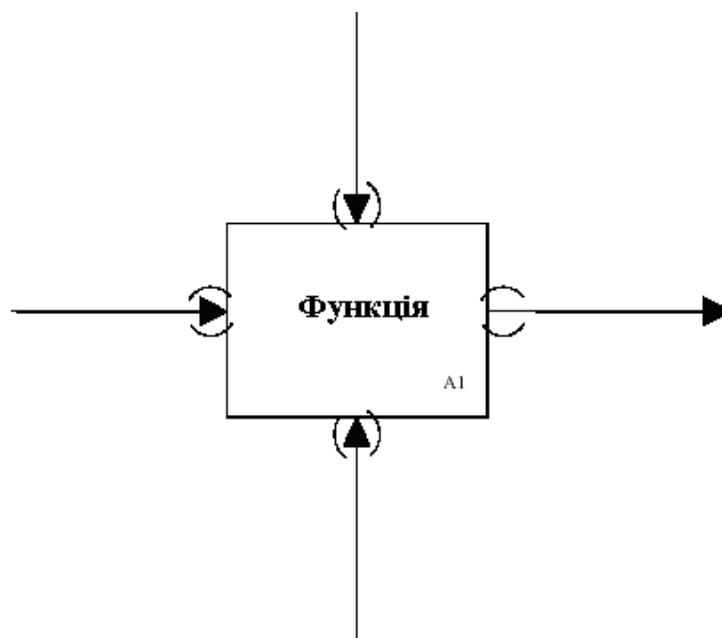


Рис. 2.19. Тунельні стрілки

Стрілка, що поміщається в тунель на вільному кінці (рис. 2.20) означає, що виражені нею дані відсутні на батьківській діаграмі.



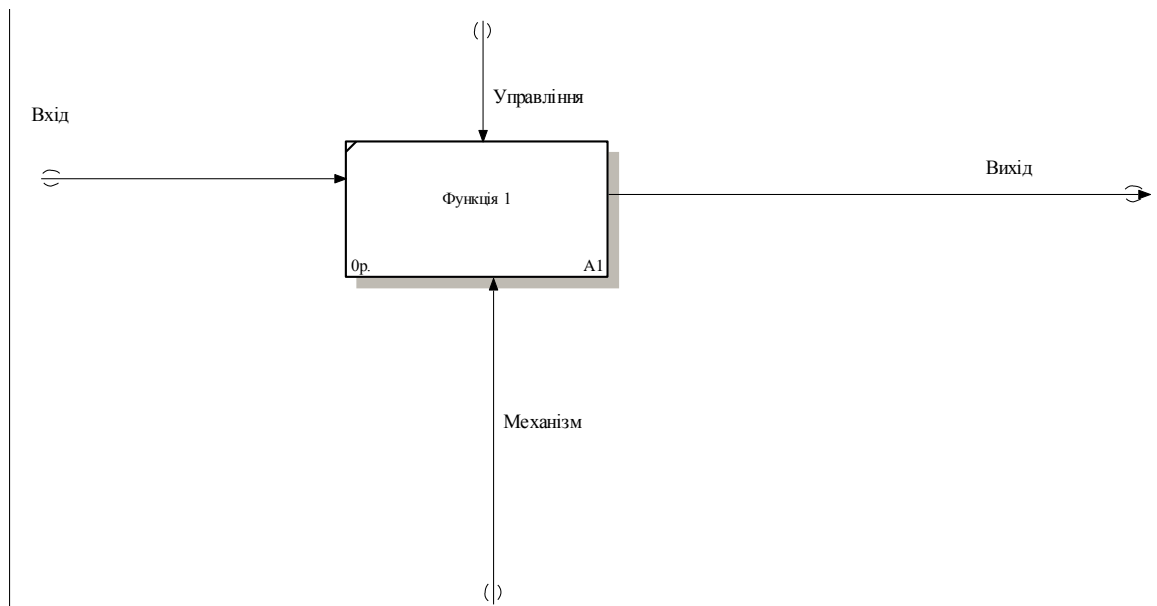


Рис. 2.20. Тунельні стрілки (діаграма нащадків)

Детальніше ця ситуація пояснюється на рис. 2.21.

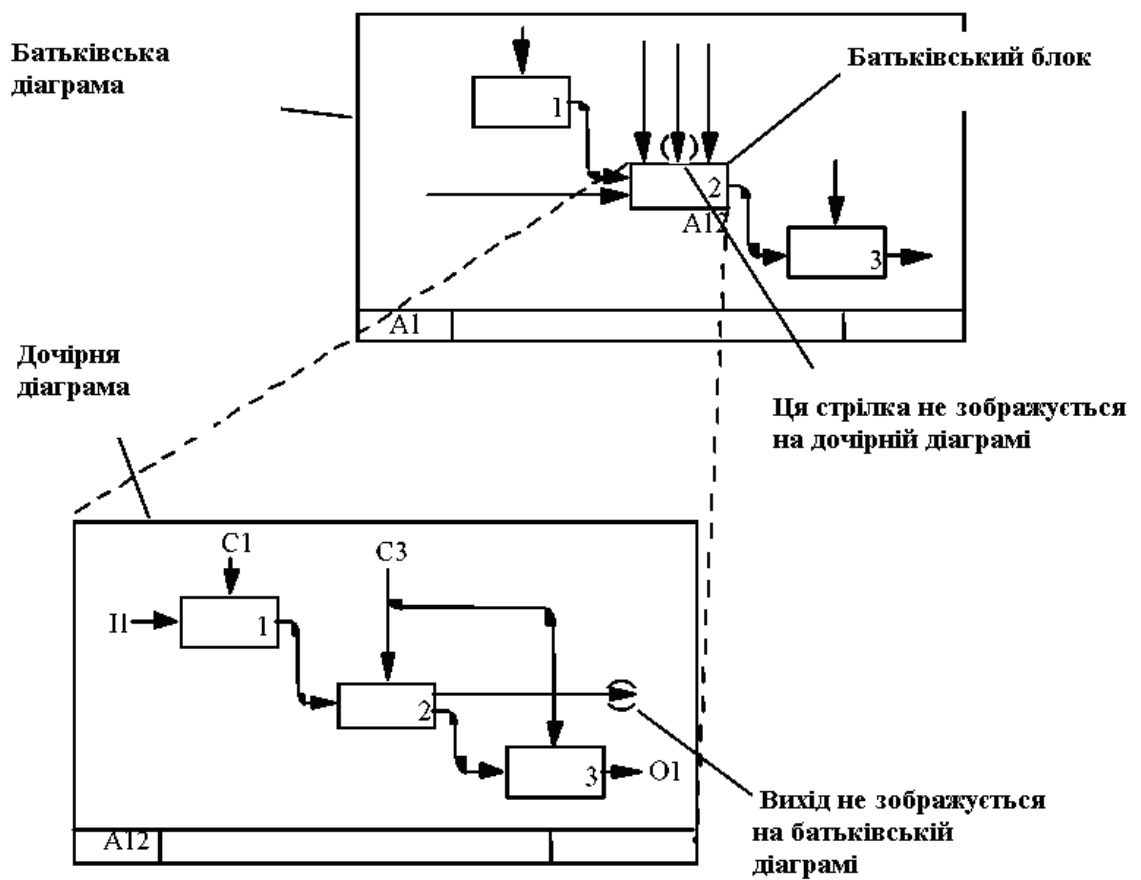


Рис. 2.21. Визначення тунельних стрілок

## Правила побудови діаграм

У складі моделі повинна бути присутня контекстна діаграма A–0, яка містить тільки один блок. Номер єдиного блоку на контекстній діаграмі A–0 повинен бути 0.

Блоки на діаграмі повинні розташовуватися по діагоналі – від лівого верхнього кута діаграми до правого нижнього в порядку привласнених номерів. Блоки на діаграмі, розташовані вгорі зліва "домінують" над блоками, розташованими внизу справа. "Домінування" розуміється як вплив, який блок надає на інші блоки діаграми. Розташування блоків на листі діаграми відображає авторське розуміння домінування. Таким чином, топологія діаграми показує, які функції здійснюють більший вплив на інші.

Неконтекстні діаграми повинні містити не менше трьох і не більше шести блоків [14, 34]. Ці обмеження визначає доступність для читання, розуміння і використання діаграми.

Діаграми з кількістю блоків, які менше трьох, викликають сумніви в необхідності декомпозиції батьківської функції. Діаграми з кількістю блоків більше шести складні для сприйняття читачами і викликають у автора труднощі при внесенні до неї всіх необхідних графічних об'єктів і міток.

Кожен блок неконтекстної діаграми отримує номер, що поміщається в правому нижньому кутку; порядок нумерації – від верхнього лівого до нижнього правого блоку (номери від 1 до 6).

Кожен блок, який декомпозується, повинен мати посилання на дочірню діаграму. Посилання (наприклад, вузловий номер, C-номер або номер сторінки) поміщається під правим нижнім кутом блоку.

Імена блоків (функцій) і мітки стрілок повинні бути унікальними. Якщо мітки стрілок збігаються, це означає, що стрілки відображають тотожні дані.

За наявності стрілок зі складною топологією доцільно повторити мітку для зручності її ідентифікації.

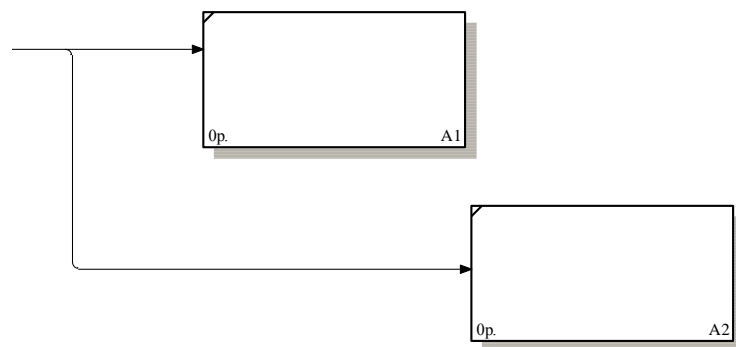
Слід забезпечити максимальну відстань між блоками і поворотами стрілок, а також між блоками і перетинами стрілок для полегшення читання діаграми. Одночасно зменшується вірогідність переплутати дві різні стрілки.

Блоки завжди повинні мати хоча б одну стрілку управління та одну вихідну стрілку, але можуть не мати вхідних стрілок.

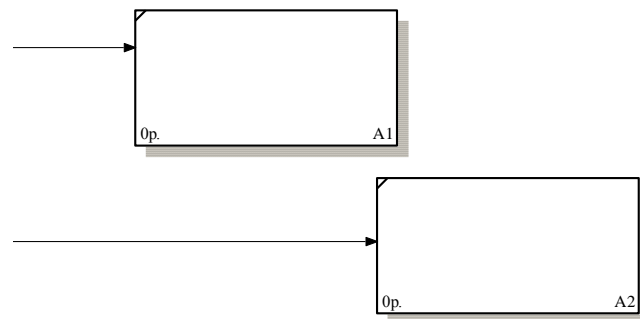
Якщо одні й ті ж дані служать і для управління, і для входу, викреслюється тільки стрілка управління. Цим підкреслюється характер даних і зменшується складність діаграми.

Максимально збільшена відстань між паралельними стрілками полегшує їх розміщення і читання й дозволяє прослідкувати шляхи стрілок.

Стрілки зв'язуються (зливаються), якщо вони представляють схожі дані і їх джерело не вказано на діаграмі (рис. 2.22, а,б).



а) Правильно



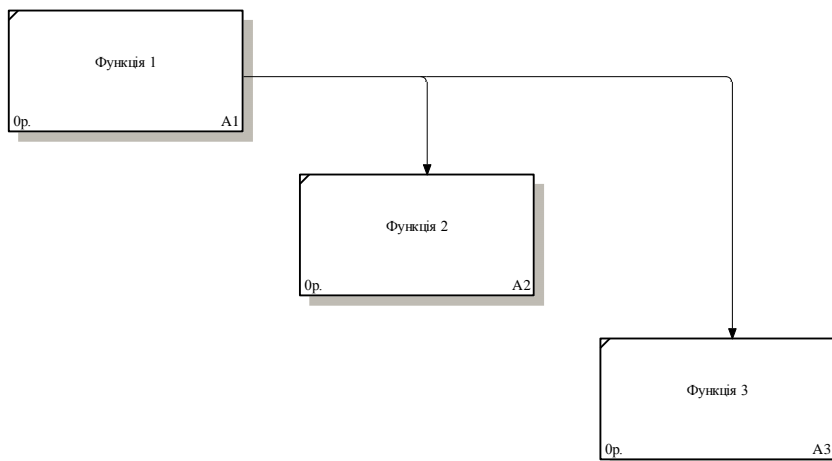
б) Неправильно

Рис. 2.22. Варіанти злиття стрілок

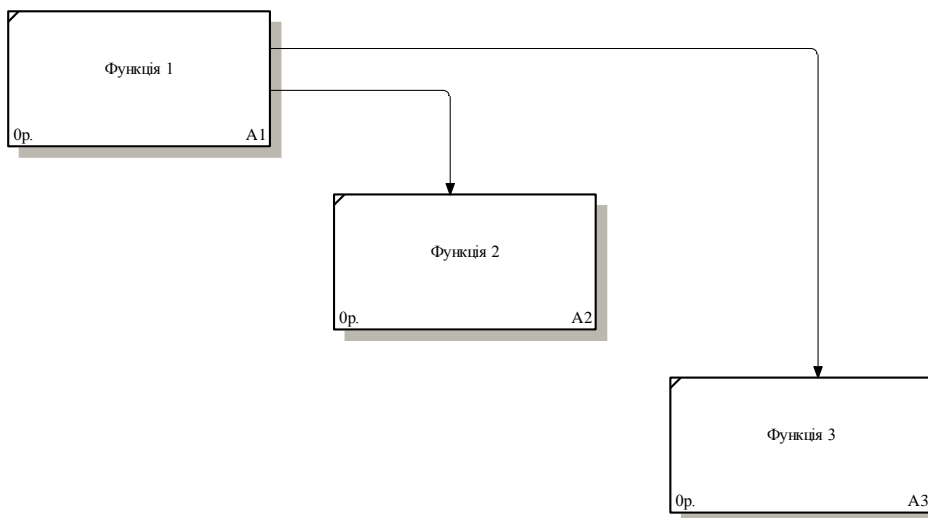
Стрілки об'єднуються, якщо вони мають загальне джерело або приймач, або вони представляють зв'язані дані. Загальна назва краще описує суть даних. Слід мінімізувати число стрілок, що стосуються кожної сторони блоку, якщо, звичайно, природа даних не дуже різноманітна (рис. 2.23, а,б).

Якщо можливо, стрілки приєднуються до блоків в одній і тій же позиції. Тоді з'єднання стрілок конкретного типу з блоками буде узгодженим і читання діаграми буде полегшеним.

При з'єднанні великого числа блоків необхідно уникати необов'язкових перетинів стрілок. Слід мінімізувати число петель і поворотів кожної стрілки.



а) Правильно



б) Неправильно

Рис. 2.23. Варіанти об'єднання стрілок

Коди привласнюються всім елементам моделі: діаграмам, блокам, стрілкам і приміткам. Вони можуть використовуватися в різних контекстах для точної вказівки на потрібний елемент моделі.

Основний код – вузловий номер, який з'являється там, де виконується декомпозиція функціонального блоку і створюється його докладний опис на дочірній діаграмі. Решта всіх посилальних кодів базується на вузлових номерах.

Кожному блоку на діаграмі привласнюється номер, що поміщається в нижньому правому внутрішньому кутку блоку. Ця система нумерації необхідна для однозначної ідентифікації блоків у межах діаграми і для генерації вузлових номерів. Ці номери використовуються також для посилань на блоки в тексті і глосарії.

На контекстній діаграмі A–0 єдиному блоку привласнюється номер 0 (нуль). На всіх інших діаграмах блоки нумеруються цифрами від 1 до 6, починаючи з верхнього лівого блоку (при їх діагональному розміщенні) і закінчуючи нижнім правим блоком. Якщо деякі блоки на діаграмі розміщені не по діагоналі, то спочатку нумеруються "діагональні" блоки (також починаючи з лівого верхнього блоку), а потім – "недіагональні" блоки, починаючи з нижнього правого проти годинникової стрілки.

#### *Вузлові номери*

Вузловий номер базується на положенні блоку в ієрархії моделі. Зазвичай вузловий номер формується додаванням номера блоку до номера діаграми, на якій він з'являється. Наприклад, вузловий номер блоку 2 на діаграмі A25 – A252. Всі вузлові номери IDEF0 починаються із великої букви, наприклад, "A". Коли батьківський блок детально описується дочірньою діаграмою, вузлові номери батьківського блоку і дочірньої діаграми збігаються.

Контекстні діаграми та дочірня діаграма верхнього рівня – виняток у вищезгаданій схемі вузлової нумерації. Кожна модель IDEF0 має контекстну діаграму верхнього рівня – діаграму A–0. Ця діаграма містить єдиний "вищий блок", який є унікальним батьком всієї моделі і несе унікальний номер 0 (нуль) і вузловий номер A0. Кожна модель IDEF0 повинна також мати принаймні одну дочірню діаграму, що містить декомпозицію блоку A0 на 3 ... 6 дочірніх блоків. Цим блокам привласнюються унікальні вузлові номери A1, A2, A3 ... A6. Таким чином, послідовність [A0, A1..., A2..., A3...] починає нумерацію вузлів для будь-якої моделі.

Наприклад, модель може мати наступні вузлові номери:

A-1 Додаткова контекстна діаграма.

A-0 Обов'язкова контекстна діаграма верхнього рівня (що містить вищий блок A0).

A0 Верхня дочірня діаграма.

A1, A2..., A6 Дочірні діаграми.

A11, A12..., A16..., A61..., A66 Дочірні діаграми.

A111, A112..., A161..., A611..., A666 Дочірні діаграми.

*Дочірні діаграми нижнього рівня*

Вузловий номер використовується також для позначення того, що блок декомпозиований. У цьому випадку вузловий номер, збігаючись із номером дочірньої діаграми, поміщається під правим нижнім кутом блоку на батьківській діаграмі.

### Дерево вузлів

Розроблена модель IDEF0 зі всіма рівнями структурною декомпозицією може бути представлена на єдиній діаграмі у вигляді дерева вузлів. Для зображення цього дерева немає стандартного формату. Єдина вимога полягає в тому, що вся ієрархія вузлів моделі повинна бути представлена наочно і зрозуміло [34, 37]. Приклад дерева вузлів показано на рис. 2.24.

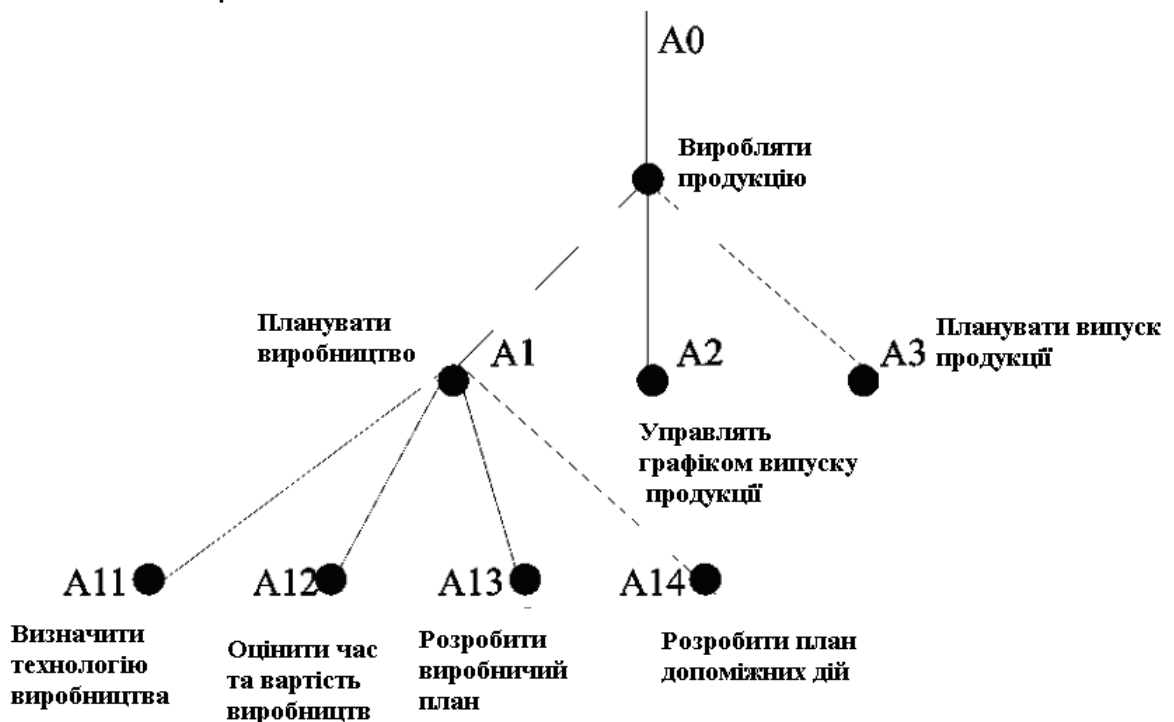


Рис. 2.24. Дерево вузлів

## Рекомендації щодо розробки функціональних моделей у середовищі IDEF0

Об'єктами функціонального моделювання і структурного аналізу (згідно з методологією IDEF0) є організаційно-економічні і виробничо-технічні системи. Згідно з основним положенням системного аналізу системою називається сукупність взаємодіючих об'єктів будь-якої природи, що володіє вираженою системною властивістю (властивостями), тобто властивістю, якої не має жодна з частин системи при будь-якому способі її розділення, і що не визначається з властивостей частин. Частини системи, що володіють власними системними властивостями, називаються підсистемами. Об'єднання декількох систем, що володіє системною властивістю, називають надсистемою або системою вищого порядку. Елементом системи є об'єкт з однозначно певними відомими властивостями.

Система (підсистема, елемент) мають входи і виходи. Будь-який елемент системи має принаймні один вхід і один вихід. Дія може полягати в передачі речовини, енергії, інформації або їх комбінації.

Наведені визначення корелюють з визначенням функціонального блоку IDEF0 з тією лише різницею, що в методології вхідні контакти підрозділяються на входи й управління.

Функціональний блок, який відображає модельовану систему в цілому (блок A0), так і блок на будь-якому рівні декомпозиції, є перетворюючими блоками. Перетворюючий блок – блок IDEF0 – діаграми дозволяє перетворити входи у виходи під дією управління за допомогою "механізмів". Перетворення – мета і результат роботи будь-якого блоку на діаграмі будь-якого рівня декомпозиції.

Перетворенню в блоці можуть піддаватися матеріальні й інформаційні об'єкти, створюючи відповідні потоки.

Матеріальний потік – безперервна або дискретна безліч матеріальних об'єктів, розподілена в часі.

Інформаційний потік – безліч інформаційних об'єктів, розподілена в часі.

Інформація, що бере участь у процесах, операціях, діях і діяльності в цілому, може бути класифікована на три групи: обмежувальна інформація; описова інформація; інформація управління.

Обмежувальна інформація міститься в законах, підзаконних актах, міжнародних, державних і галузевих стандартах, а також у спеціальних внутрішніх положеннях і документах підприємства, зокрема, в технічних вимогах, умовах, регламентах і т. п.

Описова інформація – опис атрибутів об'єкта, який трансформується у функціональному блоці. Міститься в кресленнях, технічних і інших описах, реквізитах документів.

Інформація управління – відомості про те, як, за яких умов і за якими правилами слід перетворити об'єкт (потік) на вході в об'єкт (потік) на виході блоку. Міститься в технологічних інструкціях, керівництві, директивах, параметрах запуску.

Документування процесів. Склад документів за процесами, використовуваних для їх подальшого менеджменту (планування, забезпечення, управління, поліпшення), включає два види документів:

- карта процесу;
- перелік процесів.

Для документування процесів використовується спеціальний бланк "Карта процесу", який розроблений таким чином, що поля, які містять робочу інформацію про процес, розташовані у верхній частині бланка, а поля, що містять ідентифікаційну інформацію, – в нижній частині бланка. У середній частині бланка розташовано поле, в яке заноситься опис процесу (у вигляді графічної діаграми або тексту) (рис. 2.25).

USED AT:	AUTHOR: Student PROJECT: 1	DATE: 10.03.2008 REV: 10.03.2008	<input checked="" type="checkbox"/> WORKING <input type="checkbox"/> DRAFT <input type="checkbox"/> RECOMMENDED <input type="checkbox"/> PUBLICATION	READER	DATE	CONTEXT: TOP
NOTES: 1 2 3 4 5 6 7 8 9 10						
NODE: A-0	TITLE: Проектування інформаційної системи				NUMBER:	

Рис. 2.25. Бланк "Карта процесу"



Бланк включає наступні поля:

Розділ "Робоча інформація":

поле "Автор/Дата/Проект". У цьому полі міститься інформація про автора діаграми, коли вона розроблена і до якого проекту відноситься. У полі "Дата" можуть міститися також дати подальших ревізії діаграми;

поле "Зауваження". У цьому полі читач відзначає зауваження, які він вносить до діаграми. Кожному зауваженню і коментарям до них привласнюється номер від 1 до 10. Відповідний номер закреслюється в полі "Зауваження". Ця процедура практично гарантує, що користувач і розробник не пропустять жодного зауваження, зробленого на діаграмі;

поле "Статус". У цьому полі відображаються поточні версії (стан) документа: "робоча", "чорнова", "рекомендовано", "публікація". Новим діаграмам завжди привласнюється "робоча" версія. Ця версія, як правило, містить багато зауважень. "Чорнова" версія – діаграма порівнянно з попередньою версією не змінилася. "Публікація" – це статус версії після розгляду і затвердження робочої діаграми;

поле "Контекст". У цьому полі вказується графічним або іншим чином рівень ієрархії даної діаграми в загальній структурі опису процесу.

Розділ "Ідентифікаційна інформація":

поле "Вершина". У цьому полі міститься код батьківського блоку, декомпозиція якого представлена на діаграмі;

поле "Найменування процесу". У цьому полі міститься назва процесу, представленого на діаграмі;

поле "Посилальний номер" ("Номер"). У цьому полі міститься посилальний номер процесу, представленого на діаграмі;

поле "Сторінка" ("Стор."). У цьому полі вказується номер сторінки в документі, до якого відноситься дана діаграма.

Для документування переліку процесів використовується спеціальний бланк "Перелік процесів". Бланк містить набір спеціальних полів. У поля верхньої частини бланка заноситься інформація про: розробника (авторові) документа; дату його створення; виправлення, що вносяться до документа; дати цих змін і інша інформація, необхідна для управління документацією на процеси.

У середній частині бланка розташовується інформація з опису процесів в організації. Списання процесу є рядком, що містить наступну інформацію:

поле "Сторінка" – номер сторінки, на якій знаходиться опис процесу;

поле "Вершина" – номер функціонального блоку;

поле "Найменування" – найменування функціонального блоку, що представляє процес;

поле "Посилальний номер" ідентифікаційний номер, привласнений даним / процесу;

поле "Статус" – статус опису процесу (Р – Робочий, Ч – Чорновий, П – Публікація).

Нижня частина бланка містить інформацію про найменування переліку процесів, а також посилальний номер переліку процесів.

Ефективне управління проектом опису процесу є також процесом, в ході якого координується робота розробників, експертів і керівництва організації.

Модель процесу включає:

збір інформації про досліджуваний процес;

документування отриманої інформації;

представлення інформації у вигляді моделі;

класифікацію процесу в рамках моделі;

уточнення моделі за допомогою ітеративного рецензування.

*Підготовчий етап.* На цьому етапі проводиться:

формулювання мети, точки зору про уявлення майбутніх моделей процесів і про їх передбачуване використання в майбутньому;

формування робочої групи з числа співробітників організації і залучених фахівців;

узгодження планів і термінів за проектом серед усіх учасників, призначення відповідальних виконавців за проектом, а також складання і затвердження термінів і бюджету за проектом.

*Порядок створення моделі.* На цьому етапі проводяться наступні роботи:

збір інформації (огляд документів, існуючий досвід, анкетування, спостереження за роботою співробітників в підрозділах і ін.);

документування отриманої інформації (проводиться робота із створення моделей процесів). Процес створення моделі здійснюється за допомогою методу декомпозиції. Для документування інформації про процес створюється діаграма А–0. Процес на цій діаграмі представлений

одним функціональним блоком, усередині якого розробник фіксує назву процесу;

побудова діаграм. Побудова діаграм починається з вершини A0 (але не A–0). Нижні рівні уточнюють структуру і зміст модельованого процесу, деталізують його, не розширюючи меж. При деталізації кожного блоку діаграми A0, необхідно детальніше відобразити те, що представлено на батьківському (попередньому в ієрархії) блоці;

перевірка коректності моделі. Побудовані моделі процесів проходять рецензію. Після рецензування всі зауваження надходять до розробника, який же узагальнює і вносить зміни.

*Порядок класифікації процесів.* Класифікація здійснюється в два етапи. На першому етапі розробник послідовно, діаграма за діаграмою, здійснює розмітку (маркування) ліній (інтерфейсних дуг) залежно від категорій об'єктів. На другому етапі розробник аналізує функціональні блоки. На підставі входів і виходів кожного блоку розробник ухвалює рішення про категорію процесів.

*Порядок ідентифікації процесів.* У процесі створення моделі розробник повинен дати всім функціональним блокам моделі найменування, а також коди вершин і посилальні номери.

*Порядок затвердження моделей.* Кожна модель створюється з певною метою, яка записана на діаграмі A–0 в назві процесу. Ця мета повинна бути досягнута. В процесі моделювання створюється робоча група фахівців, відповідальних за відповідність моделі призначенню.

Затверджені моделі можуть бути використані для подальшого аналізу і проектування інформаційних систем.

## **Моделювання даних**

Існують наступні методології моделювання [1, 13, 14]:

DFD (Data Flow Diagram) – методологія документування передачі та обробки інформації. Діаграми DFD зазвичай будуються для наочного зображення поточної роботи системи документообігу організації і, зокрема, можуть використовуватися як доповнення функціональної моделі бізнес-процесів, виконаної в IDEF0;

IDEF1 – цей стандарт розроблений як методологія вивчення і аналізу складу, структури і взаємозв'язків інформації в організації (модель AS IS), з метою виявлення потреб в її управлінні та вироблення

відповідних правил. При модернізації існуючих процесів стандарт може використовуватися як інструмент вивчення й аналізу складу і структури додаткових даних і правил управління інформацією, необхідних при функціонуванні підприємства в нових умовах (модель TO BE);

IDEF1X – є стандартом і методологією розробки реляційних баз даних. IDEF1X – не призначено для проведення динамічного аналізу за принципом "AS IS" - "TO BE". Ця методологія використовується, коли всі інформаційні ресурси вивчені за допомогою інших методів і ухвалено рішення про впровадження реляційної бази даних як основи або частини корпоративної інформаційної системи.

## Стандарт DFD

Для цілей моделювання систем використовуються три групи засобів, що ілюструють [41]:

- функції, які система повинна виконувати;
- відносини між даними;
- залежно від часу поведінку системи.

Серед засобів рішення даних задач в методологіях структурного аналізу найефективніше застосовуються наступні діаграми:

DFD (Data Flow Diagrams) – діаграми потоків даних спільно із словниками даних і специфікаціями процесів;

ERD (Entity-Relationship Diagrams) – діаграми "суть-зв'язок";

STD (State Transition Diagrams) – діаграми переходів станів.

Модель системи визначається як ієрархія діаграм потоків даних, що описують процеси перетворення інформації від моменту її введення в систему до видачі кінцевому користувачеві. Діаграми верхніх рівнів ієрархії – контекстні діаграми, задають межі моделі, визначаючи її оточення (зовнішні входи та виходи) і основні дані процеси. Контекстні діаграми деталізують за допомогою діаграм наступних рівнів.

### *Елементи DFD діаграм*

Основними елементами діаграм потоків даних є:

- зовнішня суть;
- процеси;
- накопичувачі даних;
- потоки даних.

Під зовнішньою суттю (External Entity) розуміється матеріальний об'єкт, що є джерелом або приймачем інформації. Як зовнішня сутність на DFD діаграмі можуть виступати замовники, постачальники, клієнти, склад, банк та інші. DFD методологія не оформлена як стандарт. З цієї причини в діаграмах потоків даних використовуються різні умовні позначення. На рис. 2.26 показані символи зовнішньої суті, використовувані в нотаціях "Yourdon and Coad Process Notation" і "Gane and Sarson Process Notation".

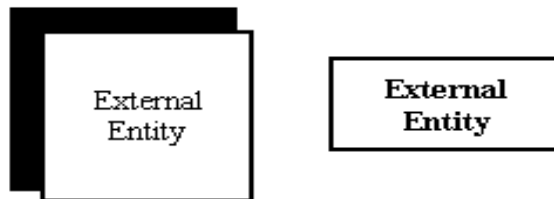


Рис. 2.26. Символи зовнішньої сутності

Визначення деякого об'єкта як зовнішня сутність указує на те, що він знаходиться за межами меж аналізованої інформаційної системи.

Процеси є перетворенням вхідних потоків даних у вихідні відповідно до певного алгоритму. У реальному житті процес може виконуватися деяким підрозділом організації, що виконує обробку вхідних документів і випуск звітів, окремим співробітником, програмою, встановленою на комп'ютері, спеціальним логічним пристроєм і тому подібне.

Процеси на діаграмі потоків даних зображаються, як показано на рис. 2.27.

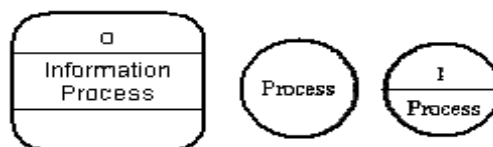


Рис. 2.27. Символи процесів

Номер процесу служить для його ідентифікації. У поле імені вводиться найменування процесу у вигляді пропозиції з дієсловом в невизначеній формі (обчислити, розрахувати, перевірити, визначити, створити, отримати) і пояснювальними іменниками, наприклад: "Надрукувати адресу одержувача", "Акцептувати рахунок".

Інформація в нижньому полі символу процесу вказує, який підрозділ організації, співробітник, програма або апаратний пристрій виконує даний процес. Якщо таке поле відсутнє, то подібна інформація може бути вказана в текстовій примітці.

На відміну від IDEF0 діаграм, в DFD діаграмах не використовуються стрілки управління для позначення правил виконання дії і стрілки механізмів для позначення необхідних ресурсів.

#### *Накопичувачі даних*

Накопичувачі даних призначені для зображення яких-небудь абстрактних пристроїв для зберігання інформації, яку можна туди у будь-який момент часу помістити або витягнути, незважаючи на їх конкретну фізичну реалізацію. Накопичувачі даних є яким-небудь прообразом бази даних інформаційної системи організації. Найбільш часто використовували символи позначення показані на рис. 2.28.



Рис. 2.28. Символи накопичувачів даних

Усередині символу вказується його унікальне в рамках даної моделі ім'я, найточніше, з погляду аналітика, що відображає інформаційну сутність, наприклад, "Постачальники", "Замовники", "Рахунки-фактури", "Накладні". Символи накопичувачів даних як додаткові елементи ідентифікації можуть містити порядкові номери.

#### *Потоки даних*

Потік даних визначає інформацію, що передається через деяке з'єднання (кабель, поштовий зв'язок, кур'єр) від джерела до приймача. На DFD діаграмах потоки даних зображаються лініями із стрілками, що показують їх напрям. Кожному потоку даних привласнюється ім'я, зміст, що відображає його. Приклад типової потокової діаграми показаний рис. 2.29.

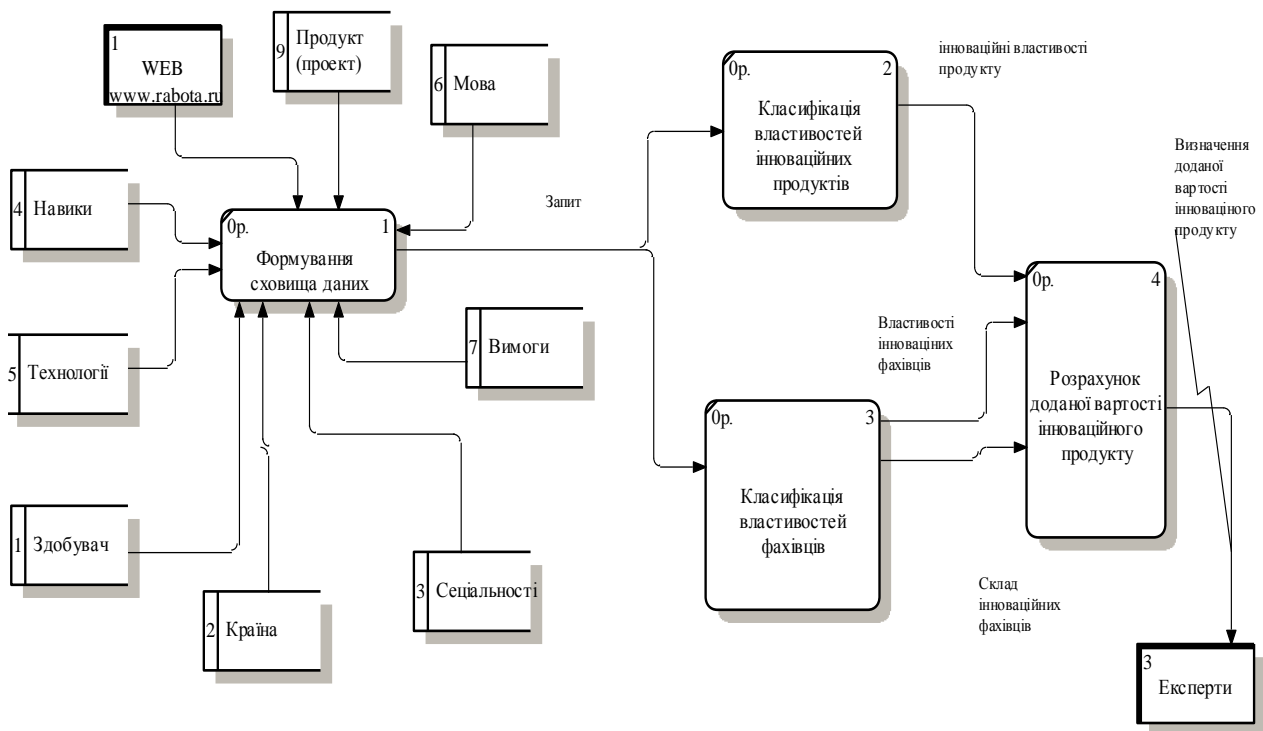


Рис. 2.29. Приклад діаграми потоків даних

*Ієрархія діаграм потоків даних*

Як наголошувалося раніше, діаграми потоків даних будуються за ієрархічним принципом [41]. Структура ієрархії DFD діаграм зображена на рис. 2.30.

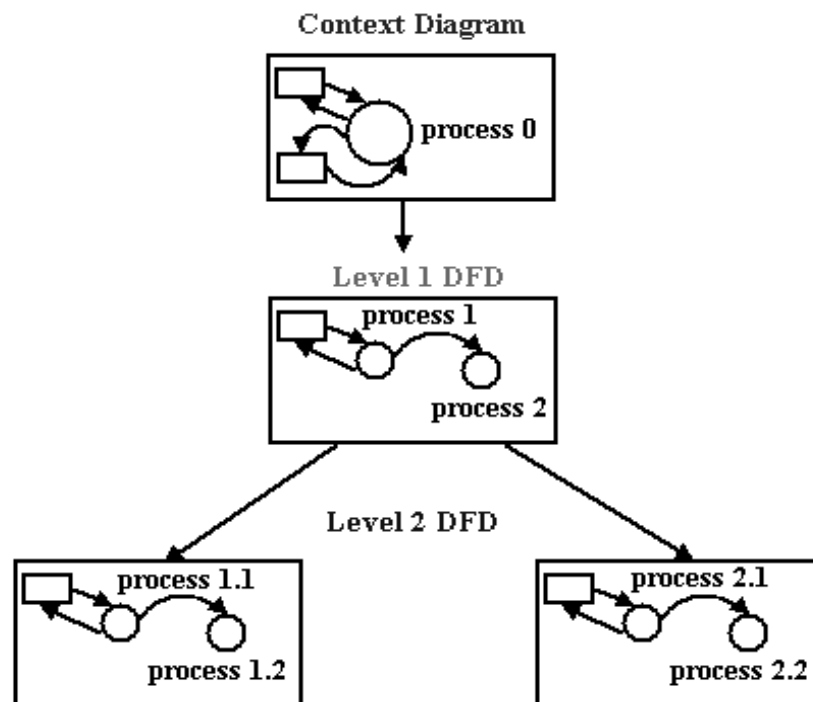


Рис. 2.30. Структура ієрархії DFD-діаграм

Контекстна діаграма верхнього рівня визначає межі моделі. Як правило, вона має зіркоподібну топологію, в центрі якої знаходиться головний процес, сполучений з приймачами і джерелами інформації, що є зовнішнім оточенням модельованої інформаційної системи

На першому рівні ієрархії показуються основні внутрішні процеси системи і відповідні їм зовнішня суть, накопичувачі і потоки даних.

Для кожного процесу діаграми першого рівня може бути проведена декомпозиція, яка, у свою чергу, також може бути розкрита детальніше. Декомпозиція процесів закінчується, коли досягнутий необхідний ступінь деталізації або процеси, що відображаються на черговому рівні діаграм, є елементарними і не можуть бути розбиті на дрібніші.

Логічна DFD показує зовнішні по відношенню до системи джерела і стоки (адресати) даних, ідентифікує логічні функції (процеси) і групи елементів даних, що пов'язують одну функцію з іншою (потоки), а також ідентифікує сховища (накопичувачі) даних, до яких здійснюється доступ. Структури потоків даних і визначення їх компонент зберігаються і аналізуються в словнику даних. Кожна логічна функція (процес) може бути деталізована за допомогою DFD нижнього рівня; коли подальша деталізація стає не корисною, переходять до виразу логіки функції за допомогою специфікації процесу. Вміст кожного сховища також зберігають в словнику даних, модель даних сховища розкривається за допомогою ERD. У разі наявності реального часу DFD доповнюється засобами опису залежної від часу поведінки системи, що розкриваються за допомогою STD. Ці зв'язки показані на рис. 2.31.

Перераховані засоби дають повний опис системи незалежно від того, чи є вона такою, що існує або розробляється з нуля. Таким чином, будується логічна функціональна специфікація – докладний опис того, що повинна робити система, звільнене наскільки це можливо від розгляду шляхів реалізації. Це дає проектувальникові чітке уявлення про кінцеві результати, які слід досягати.



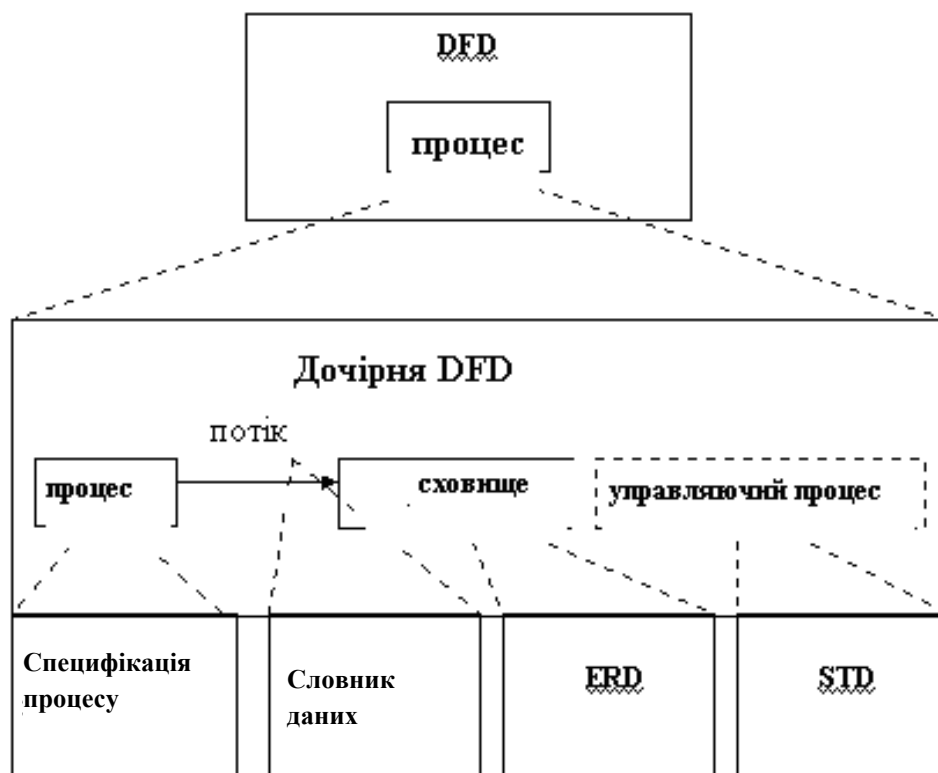


Рис. 2.31. Компоненти логічної моделі

Діаграми потоків даних (DFD) є основним засобом моделювання функціональних вимог проектованої системи. З їх допомогою ці вимоги розбиваються на функціональні компоненти (процеси) і представляються у вигляді мережі, пов'язаної потоками даних. Головна мета таких засобів – продемонструвати, як кожен процес перетворить свої вхідні дані у вихідні, а також виявити відносини між цими процесами [41].

На діаграмах функціональні вимоги представляються за допомогою процесів і сховищ, пов'язаних потоками даних.

Потоки даних є механізмами, що використовуються для моделювання передачі інформації (або навіть фізичних компонент) з однієї частини системи в іншу. Потоки на діаграмах зазвичай зображаються іменованими стрілками, орієнтація яких указує напрям руху інформації.

Іноді інформація може рухатися в одному напрямі, оброблятися і повертатися назад в її джерело. Така ситуація може моделюватися або двома різними потоками, або одним – двонаправленим.

Призначення процесу полягає в продукуванні вихідних потоків з вхідних відповідно до дії, імені процесу, що задається. Це ім'я повинне містити дієслово в невизначеній формі з подальшим доповненням

(наприклад, нарахувати заробітну плату). Крім того, кожен процес повинен мати унікальний номер для посилань на нього усередині діаграми. Цей номер може використовуватися спільно з номером діаграми для отримання унікального індексу процесу у всій моделі.

Сховище (накопичувач) даних дозволяє на певних ділянках визначати дані, які зберігатимуться в пам'яті між процесами. Фактично сховище представляє "зрізи" потоків даних в часі. Інформація, яку воно містить, може використовуватися у будь-який час після її визначення, при цьому дані можуть вибиратися у будь-якому порядку. Ім'я сховища повинне ідентифікувати його вміст і бути іменником. У разі, коли потік даних входить або виходить в/із сховища, і його структура відповідає структурі сховища, він повинен мати те ж саме ім'я, яке немає необхідності відображати на діаграмі.

Зовнішня сутність (або термінатор) представляє об'єкт (сутність) зовнішнього середовища системи, що є джерелом або приймачем системних даних. Її ім'я повинне містити іменник, наприклад, клієнт. Передбачається, що об'єкти, представлені такими вузлами, не повинні брати участі ні в якій обробці.

#### *Контекстна діаграма і деталізація процесів*

Декомпозиція DFD здійснюється на основі процесів: кожен процес може розкриватися за допомогою DFD нижнього рівня.

Важливу роль в моделі відіграє спеціальний вид DFD – контекстна діаграма, що моделює систему найбільш загальним чином. Контекстна діаграма відображає інтерфейс системи із зовнішнім світом, а саме, інформаційні потоки між системою і зовнішньою суттю, з якою вона повинна бути зв'язана. Вона ідентифікує цю зовнішню суть, а також єдиний процес, що відображає головну мету або природу системи наскільки це можливо. Контекстна діаграма встановлює межі аналізованої системи. Кожен проект повинен мати рівно одну контекстну діаграму, при цьому немає необхідності в нумерації єдиного її процесу.

DFD першого рівня будується як декомпозиція процесу, який присутній на контекстній діаграмі.

Побудована діаграма першого рівня також має безліч процесів, які у свою чергу можуть бути декомпозовані в DFD нижнього рівня. Таким чином, будується ієрархія DFD з контекстною діаграмою в корені дерева. Цей процес декомпозиції продовжується до тих пір, поки процеси можуть

бути ефективно описані за допомогою коротких (до однієї сторінки) специфікацій обробки (специфікацій процесів).

При такій побудові ієрархії DFD кожен процес нижчого рівня необхідно співвіднести з процесом верхнього рівня. Зазвичай для цієї мети використовуються структуровані номери процесів. Так, наприклад, якщо деталізуємо процес номер 2 на діаграмі першого рівня, розкриваючи його за допомогою DFD, що містить три процеси, то їх номери матимуть наступний вигляд: 2.1, 2.2 і 2.3. За необхідності можна перейти на наступний рівень, тобто для процесу 2.2 отримаємо 2.2.1, 2.2.2. і т. д.

*Декомпозиція даних і відповідні розширення діаграм потоків даних*

Індивідуальні дані в системі часто є незалежними. Проте іноді необхідно мати справу з декількома незалежними даними одночасно. Наприклад, в системі є потоки дерево, метал і нафта. Ці потоки можуть бути згруповані за допомогою введення нового потоку – ресурси. Для цього необхідно визначити формально потік ресурси як той, що складається з декількох елементів-нащадків. Таке визначення задається за допомогою форми Бекуса-Наура (БНФ) у словнику даних.

Зворотна операція розщеплювання потоків на підпотоки здійснюється з використанням групового вузла (рис. 2.32), що дозволяє розщепнути потік на будь-яке число підпотоків. При розщеплюванні також необхідно формально визначити підпотоки у словнику даних [14].

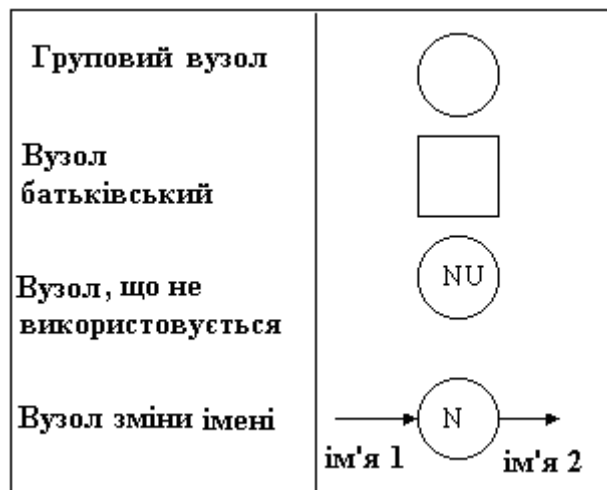


Рис. 2.32. Розширення діаграми потоків даних

Для забезпечення декомпозиції даних і деяких інших сервісних можливостей до DFD додаються наступні типи об'єктів:

1. Груповий вузол. Призначений для розщеплювання і об'єднання потоків. У деяких випадках може бути відсутнім (тобто фактично перероджуватися в точку злиття/розщеплювання потоків на діаграмі).

2. Вузол-нащадок. Дозволяє пов'язувати вхідні потоки, що виходять, між процесом, який деталізується, і детальної DFD.

3. Невживаний вузол. Застосовується в ситуації, коли декомпозиція даних проводиться в груповому вузлі, при цьому потрібні не всі елементи вхідного у вузол потоку.

4. Вузол зміни імені. Дозволяє неоднозначно іменувати потоки, при цьому їх вміст еквівалентний. Наприклад, якщо при проектуванні різних частин системи один і той же фрагмент даних отримав різні імена, то еквівалентність відповідних потоків даних забезпечується вузлом зміни імені. При цьому один із потоків даних є вхідним для даного вузла, а інший – вихідним.

Текст у вільному форматі може бути в будь-якому місці діаграми.

Головна мета побудови ієрархічної DFD полягає в тому, щоб зробити вимоги чіткими і зрозумілими на кожному рівні деталізації, а також розбити ці вимоги на частини з точно певними відносинами між ними. Для досягнення цього доцільно користуватися наступними рекомендаціями:

1. Розміщувати на кожній діаграмі від 3 до 6–7 процесів. Верхня межа відповідає людським можливостям одночасного сприйняття і розуміння структури складної системи з безліччю внутрішніх зв'язків, нижня межа вибрана з міркувань здорового глузду: немає необхідності деталізувати процес діаграмою, що містить усього один або два процеси.

2. Декомпозицію потоків даних здійснювати паралельно з декомпозицією процесів; ці дві роботи повинні виконуватися одночасно, а не одна після завершення іншої.

3. Вибирати чіткі, такі, що відображають суть справи, імена процесів і потоків для поліпшення розуміння діаграм, при цьому намагатися не використовувати аббревіатури.

4. Одноразово визначати функціонально ідентичні процеси на найвищому рівні, де такий процес необхідний, і посилатися на нього на нижніх рівнях.

5. Відокремлювати управлінські структури від оброблювальних структур (процесів), локалізувати управлінські структури.

Відповідно до цих рекомендацій процес побудови моделі розбивається на наступні етапи:

1. Розгалуження вимог і організація їх в основні функціональні групи.

2. Ідентифікація зовнішніх об'єктів, з якими система повинна бути пов'язана.

3. Ідентифікація основних видів інформації, що циркулює між системою і зовнішніми об'єктами.

4. Розробка контекстної діаграми, на якій основні функціональні групи представляються процесами, зовнішні об'єкти – зовнішньою суттю, основні види інформації – потоками даних між процесами та зовнішньою суттю.

5. Вивчення попередньої контекстної діаграми і внесення до неї змін за наслідками відповідей на питання, що виникають при цьому вивченні.

6. Побудова контекстної діаграми шляхом об'єднання всіх процесів попередньої діаграми в один процес, а також групування потоків.

7. Формування DFD першого рівня на базі процесів попередньої контекстної діаграми.

8. Перевірка основних вимог з DFD першого рівня.

9. Декомпозиція кожного процесу поточної DFD за допомогою деталізованої діаграми або специфікації процесу.

10. Перевірка основних вимог з DFD відповідного рівня.

11. Додавання визначень нових потоків у словник даних при кожній їх появі на діаграмах.

12. Паралельне (з процесом декомпозиції) вивчення вимог (у тому числі і тих, що знову надходять), ідентифікація процесів або специфікацій процесів, відповідних цим вимогам.

13. Після побудови двох-трьох рівнів проведення ревізії з метою перевірки коректності та поліпшення розуміння моделі.

### ***Словник даних***

Діаграми потоків даних забезпечують зручний опис функціонування компонент системи, але не забезпечують аналітика засобами опису деталей цих компонент, а саме, яка інформація перетвориться

процесами і як вона перетвориться. Для вирішення першого з перерахованих завдань призначені текстові засоби моделювання, що служать для опису структури перетворюваної інформації та отримали назву словників даних.

Словник даних є певним чином організований список всіх елементів даних системи з їх точними визначеннями, що дає можливість різним категоріям користувачів (від системного аналітика до програміста) мати загальне розуміння всіх вхідних і вихідних потоків і компонент сховищ. Визначення елементів даних у словнику здійснюються наступними видами описів:

- значень потоків і сховищ, зображених на DFD;

- композиції агрегатів даних, які рухаються упродовж потоків, тобто комплексних даних, які можуть розчленовуватися на елементарні символи (наприклад, адреса покупця містить поштовий індекс, місто, вулицю);

- композиції групових даних у сховищі;

- специфікацією значень і областей дії елементарних фрагментів інформації в потоках даних і сховищах;

- описом деталей відносин між сховищами.

#### *Зміст словника даних*

Для кожного потоку даних у словнику необхідно зберігати ім'я потоку, його тип і атрибути. Інформація з кожного потоку складається з ряду словникових статей, кожна з яких починається з ключового слова – заголовок відповідної статті, якому передуює символ "@".

За типом потоку в словнику міститься інформація, що ідентифікує:

- прості (елементарні) або групові (комплексні) потоки;

- внутрішні (що існують тільки усередині системи) або зовнішні (що пов'язують систему з іншими системами) потоки;

- потоки даних або потоки управління;

- безперервні (які приймають будь-які значення в межах певного діапазону) або дискретні (які приймають певні значення) потоки.

DFD-діаграми є ключовою частиною документа специфікації вимог. Кожен вузол – процес в DFD може розгортатися в діаграму нижнього рівня, що дозволяє на будь-якому рівні абстрагуватися від деталей (зазначимо, що структурні методології, орієнтовані на потоки управління, не володіють цією властивістю). Проектні специфікації будуються за DFD і їх специфікаціях автоматично.

Зазначимо, що DFD моделюють функції, які система повинна виконувати, але не визначають відношення між даними, а також про поведінку системи залежно від часу – для цих цілей методології використовують діаграми "суть-зв'язок" і діаграми переходів станів відповідно.

Головною відмінною рисою методології Гейна-Сарсона є наявність етапу моделювання даних, що визначає вміст сховищ даних (БД і файлів) в DFD в третій нормальній формі. Цей етап включає побудова списку елементів даних, розташованих у кожному сховищі даних; аналіз відносин між даними і побудова відповідної діаграми зв'язків між елементами даних; представлення всієї інформації за моделлю у вигляді зв'язаних нормалізованих таблиць. Крім того, методології відрізняються суто синтаксичними аспектами, так, наприклад, різні графічні символи, що представляють компоненти DFD.

Як приклад розглянемо верхній рівень функціональної моделі компанії, що займається розподілом товарів на замовлення. Замовлення піддаються вхідному контролю і сортуванню. Якщо замовлення не відповідає номенклатурі товарів або оформлений неправильно, то він анулюється з відповідним повідомленням замовника. Якщо замовлення не анулюване, то визначається, чи є на складі відповідний товар. У разі позитивної відповіді виписується рахунок до оплати і надається замовникові, під час вступу платежу товар надсилається замовникові. Якщо замовлення не забезпечене складськими запасами, то відправляється заявка на товар виробникові. Після надходження необхідного товару на склад компанії замовлення стає забезпеченим і повторює вищеописаний маршрут. При побудові даної моделі використана нотація Гейна–Сарсона [14] (рис. 2.33).

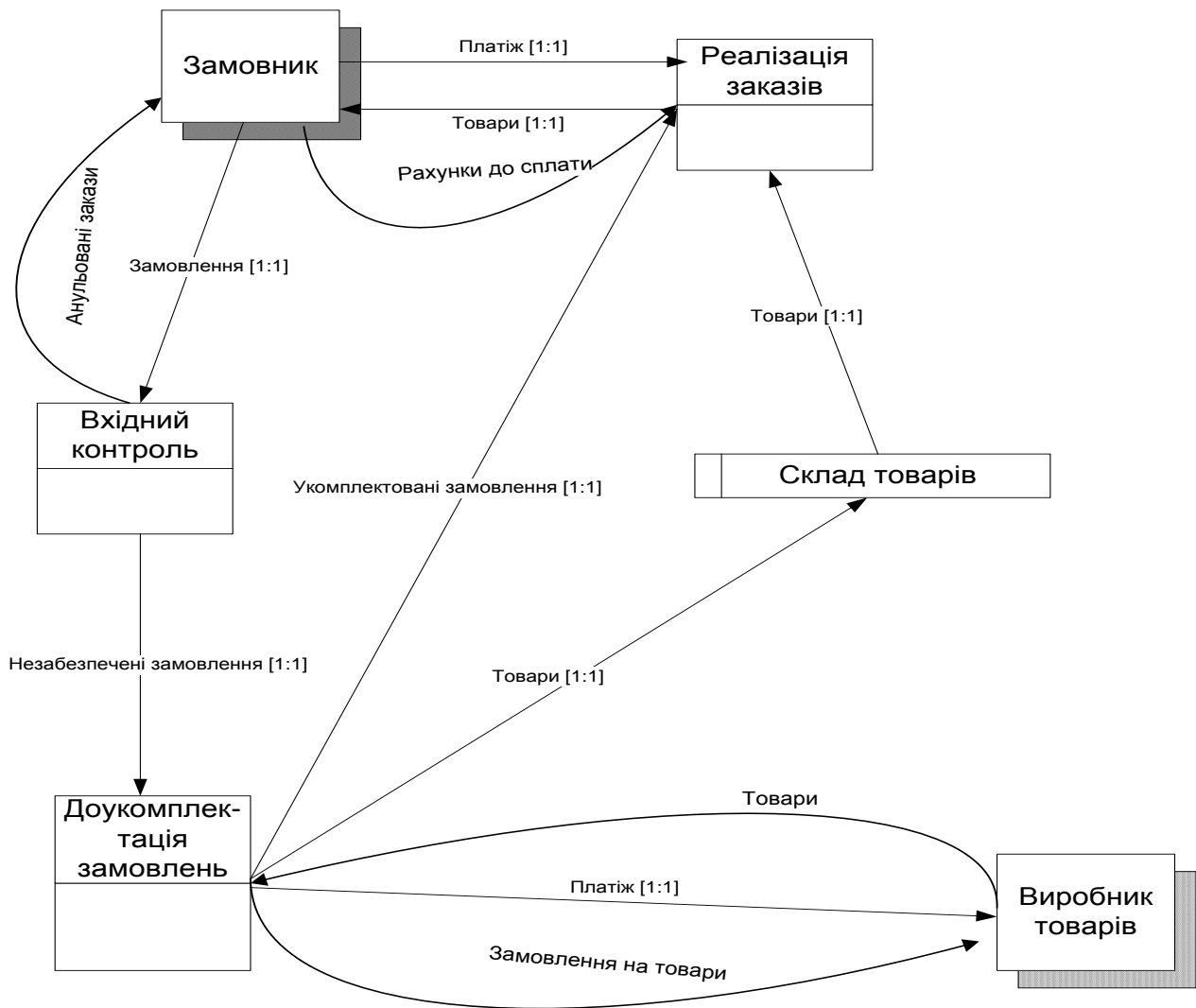


Рис. 2.33. Приклад діаграми Гейна–Сарсона

Таким чином, DFD є поетапним розбиттям функцій системи на підфункції. На першому етапі формується контекстна діаграма верхнього рівня, що ідентифікує межі системи і визначає інтерфейси між системою і оточенням. Потім після інтерв'ювання експерта наочної області формується список зовнішніх подій, на які система повинна реагувати. На наступному рівні деталізації аналогічна діяльність здійснюється для кожного з процесів.

### Стандарт IDEF1

Стандарт IDEF1 був розроблений як інструмент вивчення й аналізу складу використовуваної підприємством інформації і взаємозв'язків між



інформаційними потоками з метою їх структуризації, доповнення, визначення вимог до управління і вироблення відповідних правил. Таким чином, основним призначенням IDEF1 є визначення [38-40]:

яка інформація використовується в організації у процесі її діяльності;  
яким чином інформація збирається, зберігається і обробляється;  
які логічні зв'язки існують між інформаційними потоками;

які проблеми викликані відсутністю належного управління інформацією;

яка інформація і як повинна управлятися після внесення змін до бізнес-процесів.

#### *Концепція і базові поняття стандарту IDEF1*

Методологія IDEF1 є набором понять, правил і процедур, необхідних графічних і текстових засобів, а також табличних форм для створення інформаційних моделей.

Основними компонентами інформаційної моделі є:

діаграми – структурні зображення інформаційної моделі, що представляють, відповідно до набору правил, склад і логічні зв'язки використовуваних даних;

словник – значення кожного елементу моделі описується текстовим фрагментом.

Для розробника інформаційної моделі в IDEF1 важливими для розгляду є дві області:

реальний світ, що складається з матеріальних і нематеріальних об'єктів (таких, як люди, предмети, ідеї і ін.) з їх властивостями і взаємними зв'язками, які вивчаються через спілкування із співробітниками організації;

інформаційна область, що містить інформаційне зображення об'єктів реального світу.

Інформаційне зображення об'єкта – це зібрана, збережена і контрольована інформація про цей об'єкт. Одним із завдань моделювання є визначення мінімального набору характеристик, що дозволяють відобразити в інформаційній області всі стани і зміни об'єктів реального світу.

Базовим поняттям в методології IDEF1 є поняття сутності. Сутність визначається як реальний або абстрактний об'єкт, набір відмітних властивостей якого, що називається атрибутами, відомий. Кожна

сутність має ім'я і атрибути. Основними концептуальними властивостями сутності є:

стійкість – ім'я і набір атрибутів суті повинні бути незмінні;

унікальність – кожна сутність значеннями своїх атрибутів повинна відрізнятися від іншої сутності.

Прикладом фізичної суті може бути співробітник організації. Кожен співробітник має набір відмітних властивостей, таких, як ім'я, прізвище і по батькові, рік народження, посада, приналежність до певного підрозділу організації, табельний номер та інші. Безліч співробітників організації представляється безліччю суті з однаковим набором атрибутів. Таку безліч суті утворює клас сутності.

Атрибути є характерні властивості і ознаки об'єктів реального світу, що відносяться до певної сутності. Кожен атрибут має ім'я і значення. Прикладом імені атрибуту для суті співробітник є прізвище співробітника. Прикладом можливих значень атрибуту з ім'ям і прізвищем можуть бути: Іванов, Петров. Імена атрибутів, які є загальними для всіх екземплярів класу сутності, утворюють класи атрибутів [38-40].

Екземпляри сутності одного класу відрізняються один від одного комбінацією значень атрибутів. Клас атрибутів, за значеннями якого можна відрізнити одну сутність певного класу від іншої сутності того ж класу, називається ключовим класом атрибутів. Кожен клас сутності може мати один або декілька ключових класів атрибутів.

Екземпляри сутності одного класу можуть мати взаємні зв'язки або відношення з екземплярами сутності іншого класу. Відносини між сутностями описуються фразою в дієслівній формі. Прикладом відношення між сутністю "Співробітник" і "Відділ" може бути "працює в". Окремі екземпляри одного класу сутності можуть мати різні відношення з окремими екземплярами іншого класу сутності. Набір значень взаємних відносин між сутностями моделі складає клас відносин. Тобто клас відносин відображає можливі типи відносин між окремими екземплярами різних класів сутності.

Кожен клас має своє умовне графічне зображення на діаграмі інформаційної моделі згідно з методологією IDEF1.

### ***Фази розробки інформаційної моделі***

Для виконання моделювання повинна бути визначена команда, що складається з керівника проекту, аналітика – безпосереднього

розробника моделі, співробітників організації, що є джерелами інформації для побудови моделі, і рецензентів.

Роботу рекомендується розбивати на 5 фаз, кожна з яких повинна закінчуватися цілком певним вимірюваним результатом:

Фаза 0 – фаза визначення предмету дослідження і меж моделі;

Фаза 1 – на цій фазі визначаються класи сутності;

Фаза 2 – на цій фазі визначають класи відносин, що існують між визначеними на попередній фазі класами сутності;

Фаза 3 – предметом цієї фази є визначення класів ключів для кожного класу сутності і кожного класу атрибутів, який використовується класом ключів;

Фаза 4 – метою останньої фази є розподіл не ключових класів атрибутів за класами сутності і повний опис таких класів атрибутів.

Створення інформаційної моделі є циклічним ітераційним процесом, що складається із збору даних, побудові на їх основі моделі, усуненні зауважень рецензентів. У міру вивчення об'єкта дослідження і отримання додаткової інформації розробник моделі може неодноразово повертатися на попередні фази проектування, щоб внести зміни, уточнення та доповнення. Інформаційна модель повинна пройти комплексну перевірку, перш ніж на підставі її аналізу робитимуться висновки і прийматимуться рішення.

#### *Фаза 0*

На цій фазі вирішуються основні організаційні питання: визначаються предмет, цілі і межі моделювання, методи збору і джерела інформації, план виконання робіт і їх розподіл між виконавцями, які фіксуються у відповідних документах.

#### *Фаза 1*

Завданням цієї фази є визначення і опис класів сутності інформаційної моделі.

Вивчаючи документи, які використовуються у процесах діяльності організації, і опитуючи співробітників, аналітик формує пул класів сутності.

Після того, як класи сутності визначені, вони повинні бути описані. Тому наступним кроком цієї фази моделювання є формування глосарію або словника класів сутності.

#### *Фаза 2*

На цій фазі моделювання визначаються класи відносин, моделі, що існують між класами сутності. Відношення між класами сутності зображаються у вигляді діаграм. Перед побудовою діаграм рекомендується створити матрицю відносин, як показано на рис. 2.34.

		Сутність							
		1	2	3	4	5	6	7	8
1			X			X			
2	X				X	X			
3							X		
4			X						
5	X	X							
6				X					
7									
8									

Рис. 2.34. Матриця відносин

Використовуючи матрицю відносин, створюють діаграми класів сутності. IDEF1 діаграми містять зображення деякої кількості класів сутності, сполученої лініями, що представляють їх взаємні відносини. Як правило, в центрі діаграми розташовують зображення класу сутності, розгляду даної діаграми, що є предметом.

Для позначення класів відносин між класами сутності використовуються показані на рис. 2.35 лінії. Символи на кінцях ліній позначають, яка кількість екземплярів одного класу сутності можуть бути пов'язані з екземплярами іншого класу сутності. Таким чином, діаграми класів сутності створюють графічне зображення інформації, використуваної в організації. Модель представляє структуру інформації двояким чином – як безліч екземплярів сутності усередині кожного класу сутності і як безліч екземплярів відносин між класами сутності [40].

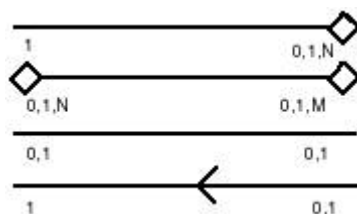


Рис. 2.35. Позначення класів відносин

Класи відносин повинні бути детально описані. Опис класів відносин стає частиною глосарію класів сутності.

### *Фаза 3*

Метою третьої фази є визначення класів ключів для кожного класу сутності.

Набори класів атрибутів розробником моделі групуються в пул класів атрибутів.

Класи атрибутів, також як раніше класи сутності і відносин, повинні бути детально описані. Аналізуючи властивості класів атрибутів, розробник моделі визначає ті, які використовуватимуться в класі ключів. Коли класи ключів визначені, розробник переходить до побудови діаграм класів атрибутів.

Як і в діаграмах класів сутності в діаграмах класів атрибутів увага фокусується на одному з класів сутності, зображення якого поміщується в центрі форми діаграми. Діаграма класу атрибутів може розглядатися як подальший розвиток діаграми класу сутності, оскільки вони відрізняються тільки інформацією, що міститься усередині блоку, що зображає клас сутності, – ключові класи й інші класи атрибутів використовуються як зміст блоку класу сутності.

### *Фаза 4*

На фазі 4 здійснюється розподіл класів атрибутів, які не можуть бути використані в класах ключів за відповідними класами сутності. Дії, що виконуються на четвертій фазі розробки моделі, багато в чому схожі з діями на попередній фазі.

У результаті виконання робіт четвертої фази розробник отримує структуровану інформаційну модель. Якщо дії на всіх фазах були виконані коректно, то кожен клас сутності буде представлений оптимальним набором інформації і кожна пара класів сутності, що спільно використовує клас відносин, точно відобразатиме взаємозалежність даних в моделі. Таким чином, IDEF1 інформаційна модель є формою представлення даних, яка полегшує розробку бази даних системи управління. Проте не можна сказати, що розробка інформаційної IDEF1 моделі є розробкою бази даних. IDEF1-модель представляє лише стійку інформаційну структуру і стійкий набір правил і визначень, з урахуванням яких може проводитися розробка бази даних.

## **Стандарт IDEF1X**

Методологія IDEF1X призначена для побудови концептуальної схеми логічної структури реляційної бази даних, яка була б незалежною від програмної платформи її кінцевої реалізації [38-40].

IDEF1X, також як і IDEF1, використовує поняття сутності, атрибутів, відносин і ключів. Мови графічного зображення моделей, використовуваних цими методологіями, також багато в чому схожі. Проте IDEF1X не розглядає об'єкти реального світу, а лише їх інформаційне відображення, оскільки до моменту розробки бази даних всі інформаційні ресурси організації повинні бути вивчені, необхідний набір даних для віддзеркалення її діяльності визначений і перевірений на повноту. Оскільки IDEF1X призначена для розробки реляційних баз даних, вона додатково оперує рядом понять, правил і обмежень, такими, як домени, уявлення, первинні, зовнішні й сурогатні ключі та інше, що прийшли з реляційної алгебри і в яких немає необхідності на етапах вивчення і опису діяльності організації. IDEF1X є стандартом (методологією) для розробки реляційних баз даних і використовує умовний синтаксис, спеціально розроблений для побудови концептуальної схеми структури даних, незалежної від кінцевої реалізації бази даних і апаратної платформи.

Основною перевагою IDEF1X порівняно з іншими численними методами розробки реляційних баз даних, такими, як ER і ENALIM є суворая стандартизація моделювання. Встановлені стандарти дозволяють уникнути різного трактування побудованої моделі, яка є значним недоліком ER.

Хоча термінологія IDEF1X практично збігається з термінологією IDEF1, існує ряд фундаментальних відмінностей в теоретичних концепціях цих методологій. Сутність в IDEF1X описує сукупність або набір екземплярів схожих за властивостями, але однозначно відрізняються один від одного однією або декількома ознаками. Кожен екземпляр є реалізацією сутності. Таким чином, сутність в IDEF1X описує конкретний набір екземплярів реального світу, на відміну від сутності в IDEF1, яка є абстрактним набором інформаційних відображень реального світу.

Рівні логічної моделі. Розрізняють три рівні логічної моделі, що відрізняються за глибиною представлення інформації про дані:

діаграма „сутність-зв'язок" (Entity Relationship Diagram, ERD);

модель даних, заснована на ключах (Key Based model, KB);  
повна атрибутивна модель (Fully Attributed model, FA).

Діаграма сутність-зв'язок є моделлю даних верхнього рівня. Вона включає сутність і взаємозв'язки, що відображають основні бізнес-правила наочної області. Така діаграма не дуже деталізована, в неї включаються основні сутності і зв'язки між ними, які задовольняють основні вимоги, що висуваються до ІС. Діаграма сутність-зв'язок може включати зв'язки багато до багатьох і не включати опис ключів. Як правило, ERD використовується для презентацій і обговорення структури даних з експертами наочної області.

Модель даних, заснована на ключах, є докладним представленням даних. Вона включає опис всієї суті та первинних ключів і призначена для представлення структури даних і ключів, які відповідають наочній області.

Повна атрибутивна модель – найбільш детальне представлення структури даних: представляє дані в третій нормальній формі і включає всю сутність, атрибути і зв'язки.

*Сутність і атрибути.* Основні компоненти діаграм IDEF1X – сутність, атрибути і зв'язки. Кожна сутність є безліччю подібних індивідуальних об'єктів, які називаються екземплярами. Кожен екземпляр індивідуальний і повинен відрізнятися від решти екземплярів. Атрибут виражає певну властивість об'єкта. З погляду БД (фізична модель) сутності відповідає таблиця, екземпляру сутності – рядок в таблиці, а атрибуту – колонка таблиці.

Побудову моделі даних припускає визначення сутності й атрибутів, тобто необхідно визначити, яка інформація зберігатиметься в конкретній суті або атрибуті. Сутність можна визначити як об'єкт, подію або концепцію, інформація про які повинна зберігатися. Сутність повинна мати найменування з чітким смисловим значенням, називатися іменниками в однині, не носити "технічних" найменувань і бути досить важливими для того, щоб їх моделювати. Назва сутності в однині полегшує надалі читання моделі. Фактично ім'я сутності дається на ім'я її екземпляра. Прикладом може бути сутність *Замовник* (але не *Замовники!*) з атрибутами *Номер замовника*, *Прізвище замовника* і *Адреса замовника*. На рівні фізичної моделі їй може відповідати таблиця *Customer* з колонками *Customer\_number*, *Customer\_name* і *Customer\_address*.

Кожен атрибут зберігає інформацію про певну властивість сутності, а кожен екземпляр сутності повинен бути унікальним. Атрибут або група атрибутів, які ідентифікують сутність, називається первинним ключем.

При встановленні зв'язків між сутністю атрибуту первинного ключа батьківської сутності мігрують як зовнішні ключі в дочірню сутність. Дуже важливо дати атрибуту правильне ім'я. Атрибути повинні називатися в однині і мати чітке смислове значення. Дотримання цього правила дозволяє частково вирішити проблему нормалізації даних уже на етапі визначення атрибутів. Наприклад, створення за сутністю Співробітник атрибуту Телефони співробітника має суперечності з вимогами нормалізації, оскільки атрибут повинен бути атомарним, тобто не містити множинних значень. Згідно з синтаксисом IDEF1X ім'я атрибуту повинне бути унікальним в рамках моделі (а не тільки в рамках сутності).

*Зв'язки.* Зв'язок є логічним співвідношенням між сутностями. Кожен зв'язок повинен називатися дієсловом або дієслівною фразою (Relationship Verb Phrases). Ім'я зв'язку виражає деяке обмеження або бізнес-правило і полегшує читання діаграми, наприклад:

Кожен клієнт <розміщує> Замовлення;

Кожне замовлення <виконується> Співробітником.

Зв'язок показує, які саме замовлення розмістив клієнт і який саме співробітник виконує замовлення.

На логічному рівні можна встановити ідентифікуючий зв'язок – один до багатьох, зв'язок – багато до багатьох і неідентифікуючий зв'язок багато до одного.

У IDEF1X розрізняють залежну і незалежну сутність. Тип сутності визначається її зв'язком з іншою суттю. Ідентифікуючий зв'язок встановлюється між незалежною (батьківський кінець зв'язку) і залежною (дочірній кінець зв'язку) сутностями. Залежна сутність зображається прямокутником з кутами, що округляють (сутність Замовлення на рис. 2.36).



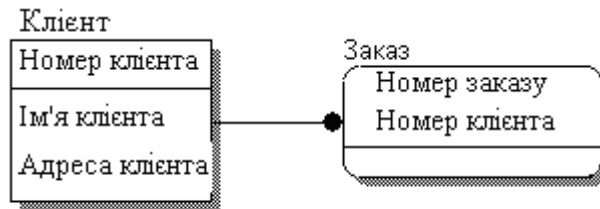


Рис. 2.36. Залежна та незалежна сутність

Екземпляр залежної сутності визначається тільки через відношення до батьківської сутності, тобто в структурі на рис. 2.36 інформація про замовлення не може бути внесена і не має сенсу без інформації про клієнта, який його розміщує. При встановленні ідентифікаційного зв'язку атрибути первинного ключа батьківської сутності автоматично переносяться до складу первинного ключа дочірньої сутності. Ця операція доповнення атрибутів дочірньої сутності при створенні зв'язку називається міграцією атрибутів. У дочірній сутності нові атрибути позначаються як зовнішній ключ – (FK) (рис. 2.36).

Далі при генерації схеми БД атрибути первинного ключа набудуть ознаки NOT NULL, що означає неможливість внесення запису в таблицю замовлень без інформації про номер клієнта.

При встановленні неідентифікаційного зв'язку (рис. 2.37) дочірня сутність залишається незалежною, а атрибути первинного ключа батьківської сутності мігрують до складу неключових компонентів батьківської сутності. Неідентифікуючий зв'язок служить для скріплення незалежної сутності.

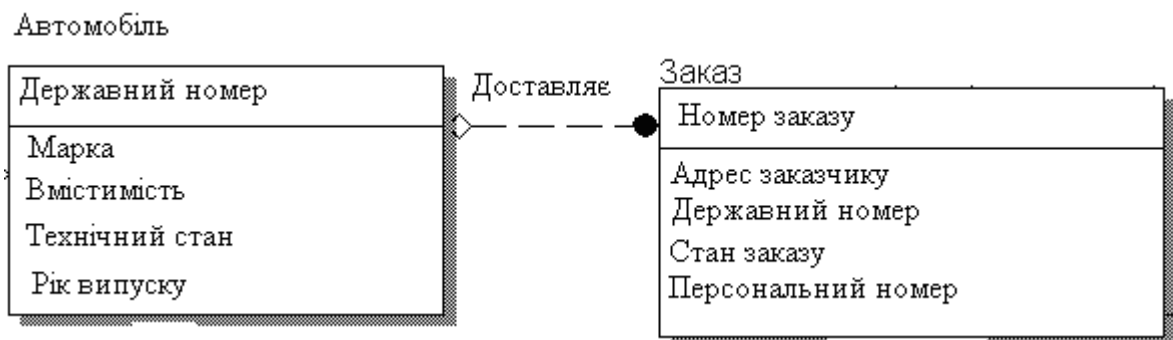


Рис. 2.37. Неідентифікуючий зв'язок

Екземпляр сутності Замовлення може існувати безвідносно до якого-небудь екземпляра сутності Автомобіль, тобто замовлення може

не перевозитися жодним автомобілем.

Ідентифікуючий зв'язок показується на діаграмі суцільною лінією з жирною точкою на дочірньому кінці зв'язку (рис. 2.37), що неідентифікуючий – пунктирною (рис. 2.37).

Потужність зв'язку (Cardinality) – служить для позначення відношення числа екземплярів батьківської сутності до екземплярів дочірньої.

Розрізняють чотири типи потужності:

загальний випадок, коли одному екземпляру батьківської сутності відповідають багато екземплярів дочірньої сутності не позначається яким-небудь символом;

символом Р позначається випадок, коли одному екземпляру батьківської сутності відповідають 1 або багато екземплярів дочірньої сутності (виключено нульове значення);

символом Z позначається випадок, коли одному екземпляру батьківської сутності відповідають 0 або 1 екземпляр дочірньої сутності (виключені множинні значення);

цифрою позначається випадок точної відповідності, коли одному екземпляру батьківської сутності відповідає наперед задане число екземплярів дочірньої сутності.

Зв'язок *один до багатьох* можливий тільки на рівні логічної моделі даних. На рис. 2.38 показано приклад зв'язку *багато до багатьох*. Лікар може приймати багато пацієнтів, пацієнт може лікуватися у декількох лікарів. Такий зв'язок позначається суцільною лінією з двома точками на кінцях.

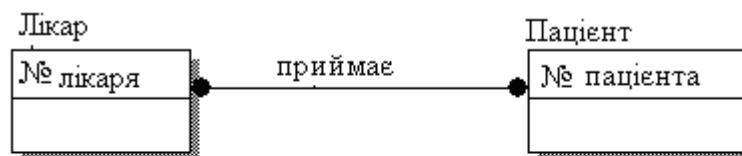


Рис. 2.38. Зв'язок багато до багатьох

Зв'язок *багато до багатьох* повинен називатися двома фразами – в обидві сторони (у прикладі "приймає/ лікується"). Це полегшує читання діаграми. Зв'язок на рис. 2.38 слід читати Лікар <приймає> Пацієнта, Пацієнт <лікується> у лікаря.

Типи суті та ієрархія спадкоємства. Як було зазначено вище, зв'язки визначають, чи є сутність незалежною або залежною. Розрізняють декілька типів залежних сутностей.

**Характеристична** – залежна дочірня сутність, яка пов'язана тільки з однією батьківською і по сенсу зберігає інформацію про характеристики батьківської сутності.

**Асоціативна** – сутність, пов'язана з декількома батьківською суттю. Така сутність містить інформацію про зв'язки сутності.

**Категоріальна** – дочірня сутність в ієрархії спадкоємства.

Ієрархія спадкоємства (або ієрархія категорій) є особливим типом об'єднання сутності, яка розділяє загальні характеристики. Наприклад, в організації працюють службовці, зайняті повний робочий день (постійні службовці) і сумісники. З їх загальних властивостей можна сформуванати узагальнену сутність (батьківський рівень) Співробітник (рис. 2.39), щоб представити інформацію, загальну для всіх типів службовців. Специфічна для кожного типу інформація може бути розташована в категоріальній сутності (нащадках) Постійний співробітник і Сумісник.

Зазвичай ієрархію спадкоємства створюють, коли декілька сутностей мають загальні за сенсом атрибути, або коли сутності мають загальні за сенсом зв'язки (наприклад, якби Постійний співробітник і Сумісник мали б схожий за сенсом зв'язок "працює в" з суттю Організація), або коли це диктується бізнес-правилами.

Для кожної категорії можна вказати дискримінатор – атрибут родового предка, який показує, як відрізнити одну категоріальну сутність від іншої (атрибут Тип на рис. 2.39).

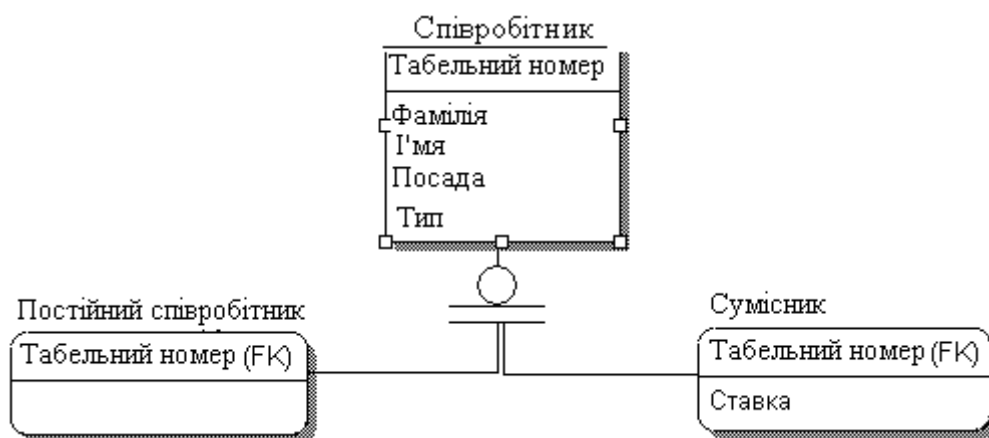


Рис. 2.39. Ієрархія спадкоємства

*Ключі.* Кожен екземпляр сутності повинен бути унікальний і відрізнятися від інших атрибутів.

Первинний ключ (primary key) – це атрибут або група атрибутів, що однозначно ідентифікує екземпляр сутності. Атрибути первинного ключа на діаграмі не вимагають спеціального позначення – це ті атрибути, які знаходяться в списку атрибутів вище за горизонтальну лінію. Вибір первинного ключа може виявитися непростим завданням, рішення якого може вплинути на ефективність майбутньої ІС. В одній сутності можуть опинитися декілька атрибутів або наборів атрибутів, що претендують на роль первинного ключа. Такі претенденти називаються потенційними ключами (candidate key).

Ключі можуть бути складними, тобто такими, що містять декілька атрибутів. Складні первинні ключі не вимагають спеціального позначення – це список атрибутів вищий за горизонтальну лінію.

Зовнішні ключі (Foreign Key) створюються автоматично, коли зв'язок сполучає суть: зв'язок утворює посилання на атрибути первинного ключа в дочірній сутності і ці атрибути утворюють зовнішній ключ в дочірній сутності (міграція ключа). Атрибути зовнішнього ключа позначаються символом (FK) після свого імені.

Залежна сутність може мати один і той же зовнішній ключ з декількох батьківської сутності. Сутність може також отримати один і той же зовнішній ключ кілька разів від одного й того ж батька через декілька різних зв'язків.

*Нормалізація даних.* Нормалізація – процес перевірки та реорганізації сутності й атрибутів з метою задоволення вимог до реляційної моделі даних. Нормалізація дозволяє бути упевненим, що кожен атрибут визначений для своєї сутності, значно скоротити обсяг пам'яті для зберігання інформації і усунути аномалії в організації зберігання даних. У результаті проведення нормалізації повинна бути створена структура даних, за якої інформація про кожен факт зберігається тільки в одному місці. Процес нормалізації зводиться до послідовного приведення структури даних до нормальних форм – формалізованих вимог до організації даних.

Відомо шість нормальних форм:

перша нормальна форма (1NF);

друга нормальна форма (2NF);

третя нормальна форма (3NF);  
нормальна форма Бойса – Кодда (посилена 3NF);  
четверта нормальна форма (4NF);  
п'ята нормальна форма (5NF).

Стандарти DFD, IDEF1, IDEF1X, ERD використані в методології Мартіна, яка надає загальну стратегію розробки інформаційних систем.

Підхід Мартіна базується на двох концепціях:

пошарового цілісного підходу до розробки інтегрованих застосувань, розвитку інформаційних систем, що базується на стратегічному плані;

первинній спрямованості на моделювання даних, а потім на функціональне моделювання.

Основні етапи підходу Мартіна:

етап стратегічного інформаційного планування починається з побудови стратегічного плану для бізнес-системи, що включає цілі і стратегії їх досягнення. Далі будується модель наочної області, що відображає існуючу специфіку і визначає основні бізнес-процеси й організаційну структуру бізнес-системи, а також визначається порядок розробки інформаційної системи. При моделюванні використовуються діаграми декомпозиції (ієрархічні деревоподібні структурні діаграми) і діаграми "суть-зв'язок" для представлення основних бізнес-процесів і структур даних відповідно.

На етапі аналізу основні бізнес-процеси, розроблені на етапі 1, використовуються для розбиття загального завдання на приватні, при цьому основна увага приділяється визначенню інформаційної та функціональної моделей для приватних завдань. При цьому діаграми "суть-зв'язок" трансформуються в нормалізовану модель даних, а діаграми декомпозиції розподіляються за підзадачами. Для представлення процесів служать DFD, діаграми декомпозиції, а для співвідношення даних і процесів, в яких ці дані використовуються, застосовуються матриці "сутність/процес".

На етапі логічного проектування базою для проектування є процеси, розроблені на етапі аналізу. Використовуючи методики спадної функціональної декомпозиції, проектується специфікації обробки в процесах і їх логічні структури даних. При цьому використовуються стандарти IDEF1, IDEF1X, що визначають типи сутності, їх атрибути і зв'язки, які деталізують логіку процесів. Для узгодження вимог користувача створюються прототипи призначених для користувача

інтерфейсів за допомогою схем екранів/звітів.

На етапі фізичного проектування і реалізації проводиться перетворення логічної моделі ІС у фізичну та її реалізація.

### **Стандарт IDEF3**

IDEF3 є стандартом документування технологічних процесів, що відбуваються на підприємстві, і надає інструментарій для наочного дослідження і моделювання їх сценаріїв. Виконання кожного сценарію супроводжується відповідним документообігом, який складається з двох основних потоків: документів, що визначають структуру й послідовність процесу (технологічних вказівок, описів стандартів і т. п.), і документів, що відображають хід його виконання (результатів тестів і експертиз, звітів про брак). Для ефективного управління будь-яким процесом необхідно мати детальне уявлення про його сценарій і структуру супутнього документообігу. Засоби документування і моделювання IDEF3 дозволяють виконувати наступні завдання [41]:

- документувати наявні дані про технології процесу, виявлені, скажімо, у процесі опиту компетентних співробітників, відповідальних за організацію даного процесу;

- визначати й аналізувати точки впливу потоків супутнього документообігу на сценарій технологічних процесів;

- визначати ситуації, в яких потрібне ухвалення рішення, що впливає на життєвий цикл процесу, наприклад зміна конструктивних, технологічних або експлуатаційних властивостей кінцевого продукту;

- сприяти ухваленню оптимальних рішень при реорганізації технологічних процесів;

- розробляти імітаційні моделі технологічних процесів за принципом "Як буде, якщо...".

Існують два типи діаграм в стандарті IDEF3, що представляють опис одного й того ж сценарію технологічного процесу в різних ракурсах. Діаграми відносяться до першого типу, називаються діаграмами Опису послідовності етапів процесу (Process Flow Description Diagrams, PFDD), а до другого – діаграмами стану об'єктів і їх трансформацій (Object State Transition Network, OSTN).

Два типи IDEF3 діаграм паралельні двом стратегіям опису. Діаграма опису процесів (Process Schematic) IDEF3 відображає

процесно-орієнтований погляд сценарію. Об'єктна діаграма (Object Schematics)

підтримує графічне відображення об'єктно-орієнтованої інформації. Об'єктна діаграма, яка описує одиничний сценарій, називається діаграма переходу (Transition Schematics). Діаграма, яка відображає додаткові об'єкти і зв'язки між об'єктами для забезпечення контекстних параметрів, називається розширеною діаграмою переходу (Enhanced Transition Schematics). Об'єктна діаграма, яка відображає об'єктно-орієнтовану інформацію складного сценарію називається просто Об'єктна діаграма (Object Schematics).

Приклад діаграми PFDD наведено на рис. 2.40.

Діаграма опису процесів IDEF3 використовується для збору даних, управління і відображення процесно-орієнтованих даних. Діаграма забезпечує графічний засіб, який допомагає експертам наочної області і аналітикам з різних наочних областей управляти процесами отримання знань.

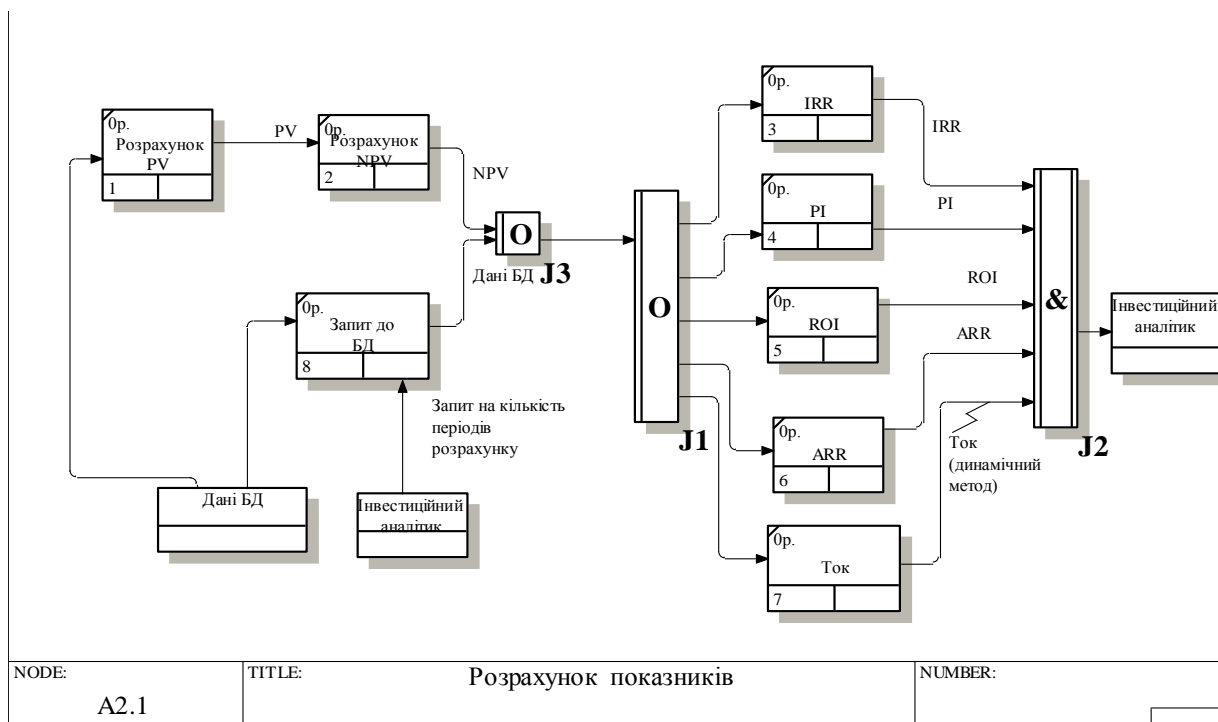


Рис. 2.40. PFDD діаграма

## Процесно-орієнтований погляд

Діаграма включає дані про події та об'єкти, які беруть участь у цих подіях і зв'язках, що впливають на подію [1].

Процесно-орієнтований опис створюється системно з використанням основних компонувальних модулів мови побудови IDEF3. Ці компонувальні модулі мають спеціальну семантику. Процес з погляду власника представлений схематично як помічені прямокутники, які пронумеровані від 1 до 10. Кожен прямокутник представляє помітний інформаційний блок про події, рішення або процеси. Таким чином, блоки (прямокутники) представляють тип подій. Такі події описуються нейтральним терміном "модулі одиниці поведінки" (units of behavior – UOB). Кожен UOB представляє реальний процес. Інформація, записана в UOB, включає ім'я блоку (частіше за всю дієслівну підставу), яке показує, що представляє UOB, імена об'єктів, які беруть участь у процесі їх властивості і зв'язки, встановлені між об'єктами. Стрілки, які називаються зв'язками, сполучають блоки (прямокутники). Робота UOB, з якої виходить стрілка, повинна бути закінчена перш ніж почнеться робота UOB, в яку стрілка входить. Перехрестя – це точка в процесі, де він розбивається на декілька шляхів або де декілька шляхів об'єднуються. Перехрестя представляють обмеження (або ефект обмеження) активаційної логіки процесу.

IDEF3 дозволяє користувачеві збирати дані в різних рівнях абстракції за допомогою механізму декомпозиції. Декомпозиція забезпечує метод організації детальнішого опису UOB. Схема декомпозиції використовує ті ж синтаксичні правила, що й для сценаріїв, і вони створюються за допомогою тих же елементів IDEF3. UOB може мати будь-який номер на різних рівнях декомпозиції. Можна використовувати більше, ніж одну декомпозицію для однієї UOB для представлення різних точок зору і забезпечення детальнішого опису процесу. Кожен процес представлений полем, що відображає назву процесу. Номер ідентифікатора процесу призначається послідовно. У правому нижньому кутку UOB елемента розташовується посилання (IDEF0/USER або інші) і використовується для вказівки посилань або на елементи з функціональної моделі IDEF0, або для вказівки на відділи або конкретних виконавців, які виконуватимуть зазначену роботу.



Зв'язки використовуються для позначення відносин між функціональними елементами UOB. Для відображення тимчасової послідовності виконання сценаріїв у діаграмах опису процесу використовуються два основні типи зв'язків: зв'язки старшинства і відносні зв'язки (рис. 2.41).

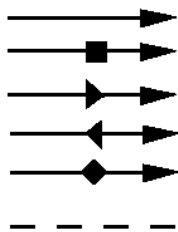


Рис. 2.41. Типи зв'язків (стрілки)

Для опису специфічних відносин між елементами призначено чотири додаткових типи зв'язків – стримуваних зв'язків старшинства. Використання в IDEF3 діаграмах опису процесу різних типів зв'язків дає можливість користувачам методу фіксувати додаткову інформацію про специфіку відносин між елементами діаграми. IDEF3 елемент діаграми опису процесу зв'язок необхідний для зв'язку елементів діаграми та опису динаміки процесів, що відбуваються. Зв'язки старшинства виражають тимчасові відносини старшинства між елементами діаграми. При цьому перший елемент повинен завершитися перш ніж почне виконуватися наступний. Графічно стрілка передування (старшинства) відображається **суцільною лінією з одиночною стрілкою**.

Використання відносного зв'язку вказує на той факт, що між взаємодіючими елементами діаграми опису процесу існують відносини невизначеного типу. Відносні зв'язки графічно зображаються як **пунктирні лінії**.

Тип зв'язку *потік об'єктів* запропонований розробниками CASE-засобів, що підтримують моделювання в стандарті IDEF3. Графічно цей зв'язок зображається як **суцільна лінія з подвійною стрілкою**. Значення зв'язку *потік об'єктів* наступне: між UOB елементами відбувається передача об'єкта, причому перший елемент UOB повинен завершитися перш ніж почне виконуватися наступний.

Перехрестя використовуються для відображення логіки відносин між безліччю подій і тимчасової синхронізації активізації елементів

діаграм IDEF3. Розрізняють перехрестя для злиття (Fan-in Junction) і розгалуження (Fan-out Junction) стрілок (рис. 2.42).

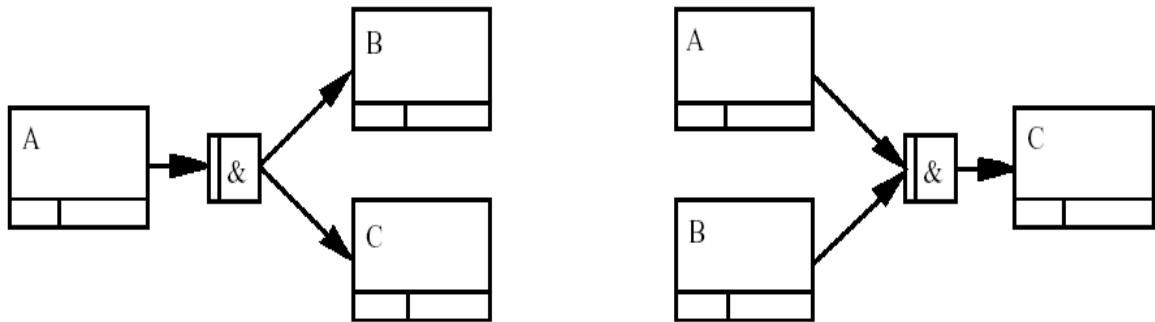


Рис. 2.42. Перехрестя для злиття (Fan-in Junction) і розгалуження (Fan-out Junction) стрілок

Перехрестя не може використовуватися одночасно для злиття і для розгалуження. При внесенні перехрестя до діаграми необхідно вказати тип перехрестя. Тип перехрестя визначає логіку і тимчасові параметри відносин між елементами діаграми. При внесенні перехрестя до діаграми необхідно вказати тип перехрестя. Класифікація можливих типів перехресть наведена в табл. 2.1 [41].

Таблиця 2.1

### Класифікація можливих типів перехресть

Позначення	Найменування	Сенс у разі злиття стрілок (Fan-in Junction)	Сенс у разі розгалуження стрілок (Fan-out Junction)
	Asynchronous AND	Усі попередні процеси повинні бути завершені	Усі наступні процеси повинні бути запуснені
	Synchronous AND	Усі попередні процеси завершені одночасно	Усі наступні процеси запускаються одночасно
	Asynchronous OR	Один або декілька попередніх процесів повинні бути завершені	Один або декілька попередніх процесів повинні бути запуснені
	Synchronous OR	Один або декілька попередніх процесів завершуються одночасно	Один або декілька наступних процесів запускаються одночасно

<b>X</b>	XOR (Exclusive OR)	Тільки один попередній процес завершений	Тільки один наступний процес запускається
----------	--------------------	--	---

Усі перехрестя в PFDD діаграмі нумеруються, кожен номер має префікс "J".

Графік запуску – це візуальне відображення тимчасової послідовності виконання UOB елементів (рис. 2.43).

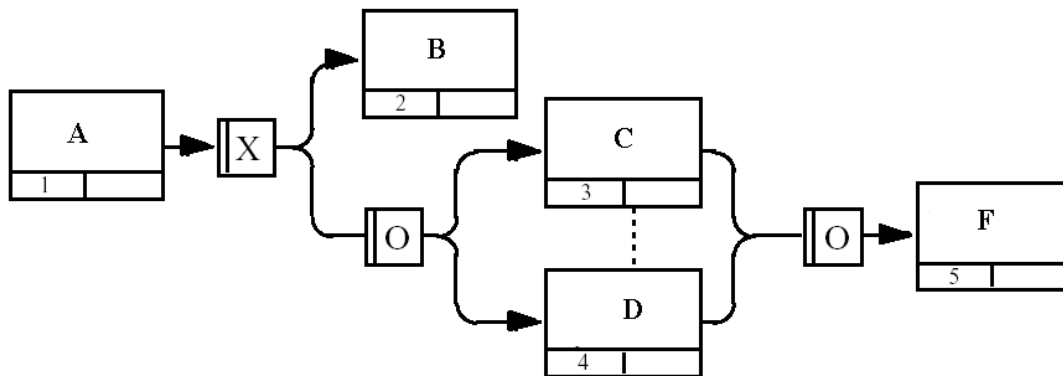


Рис. 2.43. Графік запуску

Елемент референт – це елемент посилання (рис. 2.44). Референти розширюють межі розуміння діаграми і спрощують конструкцію опису (тим самим виключають неоднозначність). Референти використовуються як в IDEF3 діаграмах опису процесу, так і об'єктних діаграмах OSTN.

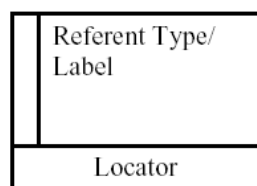


Рис. 2.44. Позначення референта

Референти призначені для:

звернення до функціонального елемента UOB, без дублювання його визначення;

передачі управління або організації поворотних циклів;

організації зв'язку між IDEF3 діаграмами опису процесу і OSTN об'єктними діаграмами.

Кожен тип референта може використовуватися як в IDEF3 діаграмі опису процесу, так і в об'єктній діаграмі OSTN. Проте найбільш продуктивно референти використовуються в IDEF3 діаграмах опису процесу (рис. 2.45).

ТИП РЕФЕРЕНТА	ПОЗНАЧКА РЕФЕРЕНТА
UOB	UOB Label
SCENARIO	Scenario Label
TS	Transition Schematic Label
GO-TO (used only in process schematics)	UOB Label  Scenario Label  Junction Type (i.e., &, O, or XOR)

**Рис. 2.45. Типи референта**

Крім поділу на типи, методологія IDEF3 визначає два види референтів за способом запуску.

Розподіл на референти "Запустити і Продовжити" і "Запустити і Чекати" дозволяє описати часові межі виконання референта. Так, використання референта "Запустити і продовжити" вказує, що згаданий референт повинен лише ініціалізуватися (активізуватися) раніше, ніж буде завершено роботу елемента IDEF3, який викликав референта.

Використання референта "Запустити і Чекати" вказує, що згаданий референт повинен активізуватися і завершитися перш ніж елемент, який його викликав, завершить свою роботу.

Якщо використовується референт "Запустити і Продовжити", який має тип UOB, SCENARIO або GO-TO, то на виході такого елемента не може використовуватися стрілка старшинства.

Якщо тип референта UOB, то найменування цього референта повинне бути ідентичне найменуванню елемента UOB, який заздалегідь визначений. Якщо референт UOB використовується в діаграмі опису

процесу і прикладений до елементу діаграми, то під час виконання UOB елементу здійснюється активізація відповідного UOB елементу (причому тимчасових обмежень із завершенням UOB елементу, що викликається, немає). Якщо UOB референт прикладений до стрілки, яка зв'язує елементи стану об'єкта в діаграмі об'єкта, то виконання згаданого в референті UOB повинно початися перш ніж почне змінюватися стан об'єкта. Якщо використовується референт типу Scenario, то відповідно його назва повинна збігатися з назвою сценарію, на який посилається вищезгаданий референт. Під час використання Scenario референта в діаграмі опису процесу під час виконання UOB елементу здійснюється активізація викликаного сценарію (причому тимчасових обмежень із завершенням сценарію, що викликається, немає). Сценарій же, що викликається, виконується на всю "глибину" декомпозиції.

Приклад використання референту наведено на рис. 2.46.

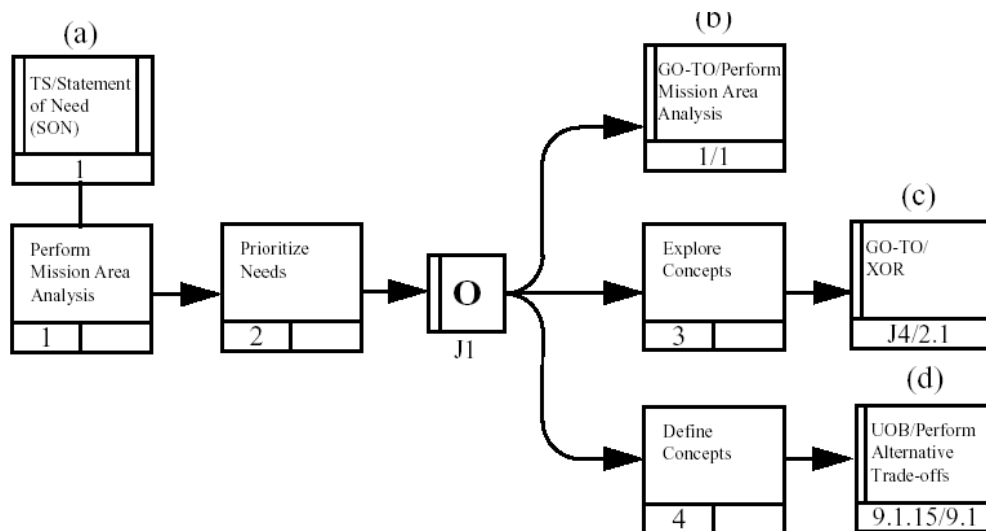


Рис. 2.46. Приклад використання референта

Елемент "примітка" може використовуватися як у діаграмах опису процесу, так і об'єктних діаграмах OSTN. Цей елемент може бути прикладений до функціонального елементу UOB, перехрестя, зв'язку, об'єкта або референта. Елемент "примітка" призначений для:

Ідентифікації і підкреслення участі специфічних об'єктів або відносин, пов'язаних з функціональним елементом UOB, зв'язком або переходом.

Приєднання прикладів, об'єктів і т. п. (наприклад, екранних форм).

Відображення спеціальних умов, уточнень з'єднання або обмежень, пов'язаних з елементами діаграм.

Примітки можуть використовуватися для забезпечення додаткової інформації у процесі моделювання, для приєднання до діаграм ілюстрацій, тексту, екранних форм, коментарів і т. д. Примітки надають можливість виразити ідеї або концепції замість використання відносних зв'язків.

Поле примітки розділене на два розділи. Верхня частина елемента використовується для ідентифікації примітки і містить ідентифікатор примітки, складений з номера елемента, для якого робиться примітка, і номера примітки з префіксом N (наприклад, J1/N1). Нижня частина примітки називається поле примітки, і призначена безпосередньо для тексту, малюнка і т. п. примітки. Стандартом IDEF3 не визначені які-небудь обмеження на форму і склад змісту поля примітки, хоча групою розробників моделі можуть бути визначені деякі угоди для вирішення яких-небудь цілей.

Методологія IDEF3 дає можливість представляти процес у вигляді ієрархічно організованої сукупності діаграм. Діаграми складаються з декількох елементів опису процесу IDEF3, причому кожен функціональний елемент UOB потенційно може бути деталізований на іншій діаграмі. Таке розділення складних комплексних процесів на його структурні частини називається декомпозицією. Декомпозиція формує межі для опису процесу і кожен UOB елемент розглядається як формальна межа деякої частини цілої системи, яка описує процес (рис. 2.47).

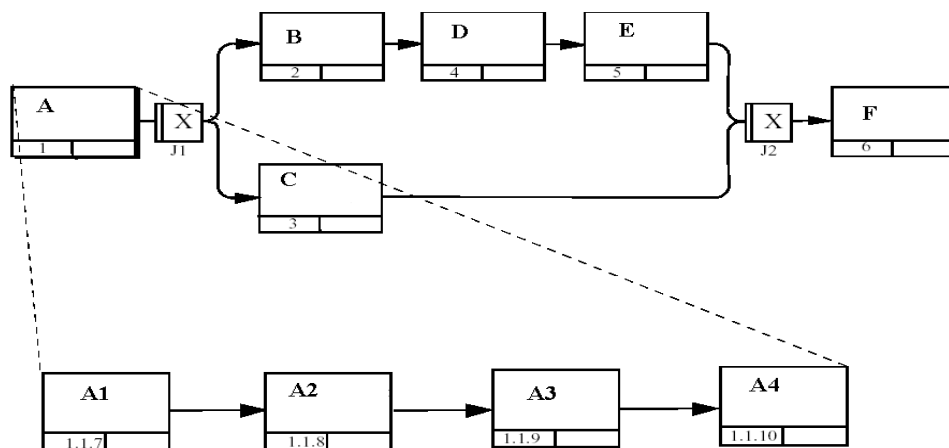


Рис. 2.47. Приклад декомпозиції діаграми

Номери UOB дочірніх діаграм мають накрісну нумерацію. Застосування принципу декомпозиції в IDEF3 дозволяє структурно описувати процеси з будь-яким необхідним рівнем деталізації.

## Стандарт IDEF5

Основною характерною межею онтологічного аналізу є розділення реального світу на складові і класи об'єктів (at its joints) і визначення їх онтологій, або ж сукупності фундаментальних властивостей, які визначають їх зміни та поведінку. Методологія IDEF5 забезпечує наочне представлення даних, отриманих у результаті обробки онтологічних запитів в простій природній графічній формі [42].

Онтологічний аналіз зазвичай починається зі складання словника термінів, який використовується при обговоренні та дослідженні характеристик об'єктів і процесів, складових даної системи, а також створення системи точних визначень цих термінів. Крім того, документуються основні логічні взаємозв'язки між відповідними введеними термінами, поняттями. Результатом цього аналізу є онтологія системи, або ж сукупність словника термінів, точних їх визначень взаємозв'язків між ними.

Таким чином, онтологія включає сукупність термінів і правила, згідно з яким ці терміни можуть бути скомбіновані для побудови достовірних тверджень про стан даної системи в деякий момент часу. Крім того, на основі цих тверджень можуть бути зроблені відповідні висновки, що дозволяють вносити зміни до системи, для підвищення ефективності її функціонування.

У будь-якій системі існує дві основні категорії предметів сприйняття, такі, як самі об'єкти, складові системи (фізичні й інтелектуальні) і взаємозв'язки між цими об'єктами, що характеризують стан системи. В термінах онтології, поняття взаємозв'язку, однозначно описує або, іншими словами, є точним дескриптором залежності між об'єктами системи в реальному світі, а терміни є, відповідно, точними дескрипторами реальних об'єктів.

При побудові онтології в першу чергу відбувається створення списку або бази даних дескрипторів і за допомогою їх, якщо їх набір достатній, створюється модель системи. Таким чином, на початковому етапі повинні бути виконані наступні завдання:

- створення й документування словника термінів;

- опис правил і обмежень, згідно з якими на базі введеної термінології формуються достовірні твердження, що описують стан системи;

побудова моделі, яка на основі існуючих тверджень дозволяє формувати необхідні додаткові твердження.

При розгляді кожної системи існує величезна кількість тверджень, що достовірно відображають її полягання в різних розрізах, а побудована онтологічним способом модель повинна вибирати з них найбільш корисні для ефективного розгляду в тому або іншому контексті. Додатково ця модель допомагає описувати поведінку об'єктів і відповідну зміну взаємозв'язків між ними, або, іншими словами, поведінку систему. Таким чином, онтологія є яким-небудь словником даних, що включає і термінологію, і модель поведінки системи.

Процес побудови онтології, згідно з методологією IDEF5, складається з п'яти основних дій [42]:

*1. Вивчення і систематизація початкових умов*

Ця дія встановлює основні цілі та контексти проекту розробки онтології, а також розподіляє ролі між членами проекту.

*2. Збір і накопичення даних*

На цьому етапі відбувається збір і накопичення необхідних початкових даних для побудови онтології.

*3. Аналіз даних*

Ця стадія полягає в аналізі та угрупованні зібраних даних і призначена для полегшення побудови термінології.

*4. Початковий розвиток онтології*

На цьому етапі формується попередня онтологія на основі відібраних даних.

*5. Уточнення та затвердження онтології*

Завершальна стадія процесу.

Для підтримки процесу побудови онтологій в IDEF5 існують спеціальні онтологічні мови: схематична мова (Schematic LANGUAGE-SL) і мова доопрацювань і уточнень (Elaboration LANGUAGE-EL). SL є наочною графічною мовою, спеціально призначеною для викладу компетентними фахівцями в даній області системи основних даних у формі онтологічної інформації. Ця нескладна мова дозволяє природним чином представляти основну інформацію в початковому розвитку онтології та доповнювати, існуючі онтології новими даними. EL є структурованою текстовою мовою, яка дозволяє детально характеризувати елементи онтології.

Мова SL дозволяє будувати різноманітні типи діаграм і схем у



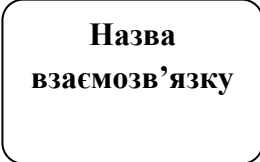





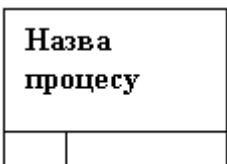
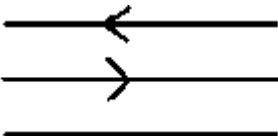



IDEF5. Основна мета всіх цих діаграм – наочно та візуально представляти основну онтологічну інформацію.

Не зважаючи на схожість, що здається, семантика й позначення схемної мови SL істотно відрізняється від семантики і позначень інших графічних мов. Справа в тому, що частина елементів графічної схеми SL може бути змінена або зовсім не братися до уваги мовою EL. Причина цього полягає в тому, що основною метою застосування SL є створення лише допоміжної структурованої конструкції онтології, і графічні елементи SL не несуть достатньої інформації для повного уявлення та аналізу системи, тим самим вони не призначені для збереження при кінцевому етапі проекту. Ретельний аналіз, забезпечення повноти представлення структури даних, отриманих у результаті онтологічного до-слідження, є завданням застосування мови EL. Позначення елементів стандарту IDEF5 наведено у табл. 2.2.

Таблиця 2.2

### Опис елементів стандарту IDEF5

Позначення класів, окремих елементів	Позначення взаємозв'язків і зміни стану	Позначення процесів, з'єднань і перехресть
<p>Позначення класу:</p>  <p>Позначення окремого елемента:</p> 	<p>Позначення первинних взаємозв'язків: Взаємозв'язок багатьох з багатьма</p>  <p>2) Взаємозв'язок двох класів</p>  <p>Позначення вторинних взаємозв'язків між двома класами:</p>  <p>Позначення зміни стану:</p> <p>Повільна зміна </p> <p>Швидка зміна </p> <p>Миттєва зміна </p>	<p>Позначення процесу</p>  <p>Позначення з'єднань:</p>  <p>Позначення перехресть:</p> 

Як правило, найбільш важливі та помітні залежності між об'єктами завжди є переважними, коли конкретні люди висловлюють свої знання й думки, що стосуються тієї або іншої системи. Подібні взаємозв'язки значним чином описуються мовами IDEF5. Усього існує чотири основні види схем, які наочно використовуються для накопичення інформації про онтологію в досить прозорій графічній формі.

*Діаграма класифікації.* Діаграма класифікації забезпечує механізм для логічної систематизації знань, накопичених під час вивчення системи. Існує два типи таких діаграм: діаграма суворої класифікації (Description Subsumption – DS) і діаграма природної або видової класифікації (Natural Kind Classification – NKC). Основна відмінність діаграми DS полягає в тому, що визначальні властивості класів вищого і всіх подальших рівнів є необхідною і достатньою ознакою приналежності об'єкта до того або іншого класу. Приклад такої діаграми – класифікація багатокутників за кількістю кутів. Визначальною властивістю кожного дочірнього класу додатково є кількість кутів в багатокутнику. Очевидно, знаючи цю визначальну властивість для будь-якого багатокутника, можна однозначно віднести його до того або іншого дочірнього класу. За допомогою діаграм DS, як правило, класифікуються логічні об'єкти.

Діаграми природної класифікації або ж діаграми NKC, навпаки, не припускають того, що властивості класу є необхідною і достатньою ознакою для приналежності до них тих або інших об'єктів. У цьому вигляді діаграм визначення властивостей класу є більш загальним. Приклад такої діаграми також наведений на рис. 2.48.

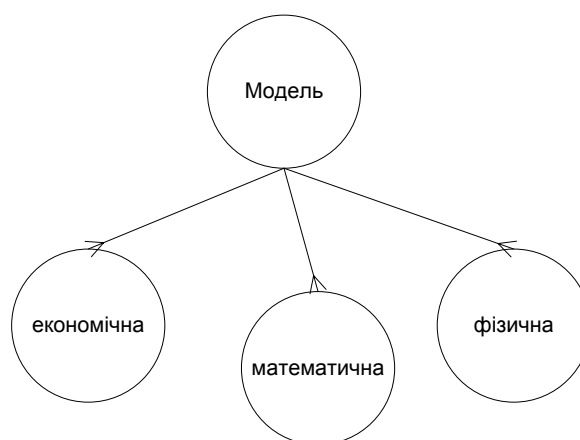
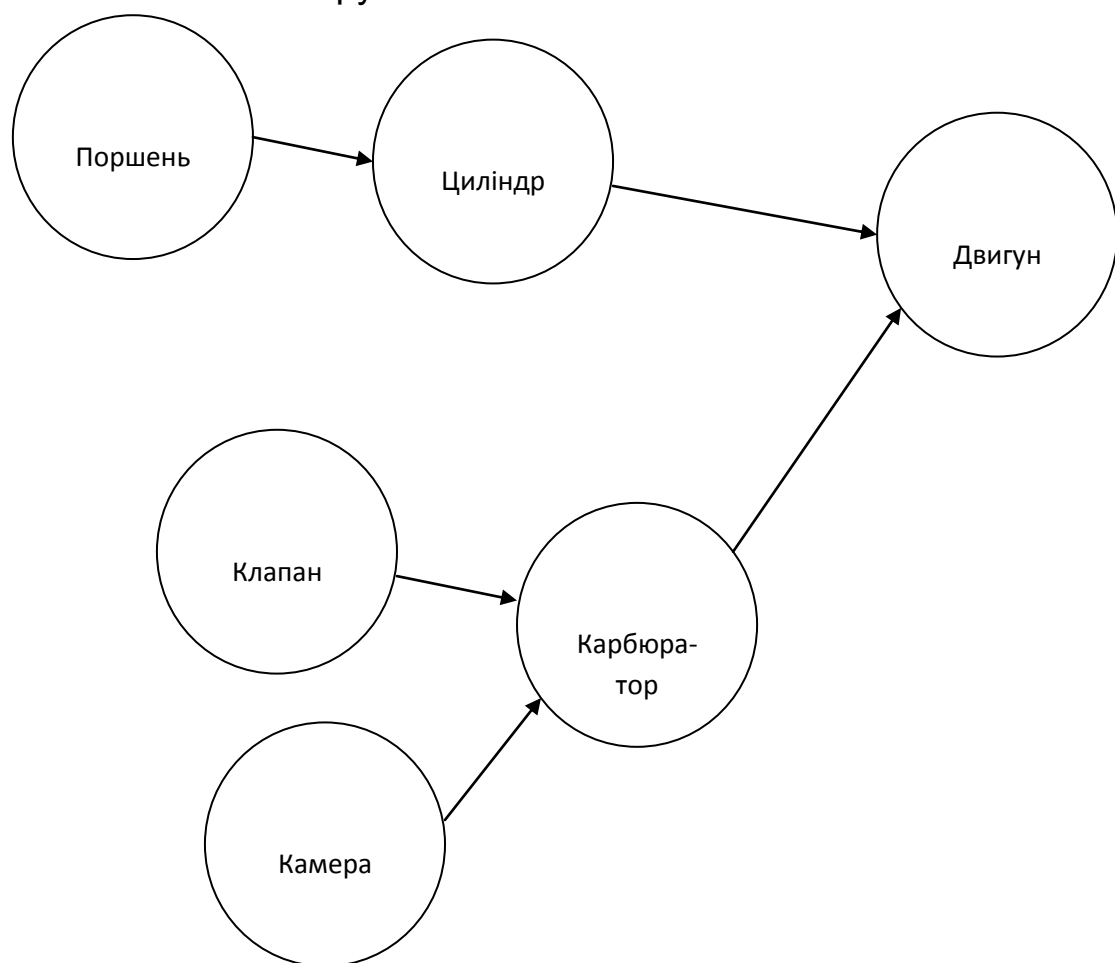


Рис. 2.48. Діаграма природної класифікації

*Композиційна схема.* Композиційні схеми (Composition Schematics) є механізмом графічного представлення складу класів онтології і фактично є інструментами онтологічного дослідження за принципом "Що з чого складається". Зокрема, композиційні схеми дозволяють наочно відобразити склад об'єктів, що відносяться до того або іншого класу. На рис. 2.49 зображена композиційна схема двигуна, що відноситься до класу бензинових двигунів. У даному випадку двигун є системою, до якої ми застосовуємо методи онтологічного дослідження. За допомогою композиційної схеми ми наочно документуємо, що двигун складається з циліндрів та карбюратора. Кожен циліндр має поршень. Карбюратор включає клапан та камеру.



**Рис. 49. Композиційна схема**

Схеми взаємозв'язків (Relation Schematics) дозволяють розробникам візуалізувати та вивчати взаємозв'язки між різними класами об'єктів у системі. В деяких випадках схеми взаємозв'язків використовуються для відображення залежностей між самими ж

класовими взаємозв'язками. Мотивацією для розвитку подібної можливості є те тривіальне правило, що всі знову розроблені концепції завжди базуються на існуючих і вивчених. Це тісно узгоджується з теорією Новака і Гоуена (Novak & Gowin), суть якої полягає в тому, що вивчення будь-якої системи часто походить від приватного до загального, тобто відбувається дослідження і дослідження нової приватної інформації, що впливає на кінцеві характеристики більш загальної концепції, до якої ця інформація мала пряме відношення. Виходячи з цієї гіпотези, природним чином вивчення нового або такого, що погано розуміється взаємозв'язку, є співвідношення її з достатньо вивченим взаємозв'язком для дослідження характеристик їх співіснування.

*Діаграма стану об'єкта.* Діаграма стану об'єкта (Object State Schemantic) дозволяє документувати той або інший процес з погляду зміни стану об'єкта. У процесах, що відбуваються, можуть відбутися два типи зміни об'єкта: об'єкт може змінити свій стан або клас. Між цими двома видами змін по суті не існує принципової різниці: об'єкти, що відносяться до певного класу К в початковому стані протягом процесу можуть просто перейти до його дочірнього або просто спорідненого класу[1].

Наприклад, отримана у процесі нагрівання тепла вода, вже відноситься не до класу ВОДА, а до його дочірнього класу ТЕПЛА ВОДА. Проте при формальному описі процесу, щоб уникнути плутанини, доцільно розділяти обидва види змін, і для такого розділення використовується позначення наступного вигляду: "клас: стан". Наприклад, тепла вода описуватиметься таким чином: "вода: тепла", холодна, - "вода: холодна" і т. д.. Таким чином, діаграми полягання в IDEF5 наочно представляють зміни стану або класу об'єкта протягом усього ходу процесу. Приклад такої діаграми наведений на рис. 2.50.

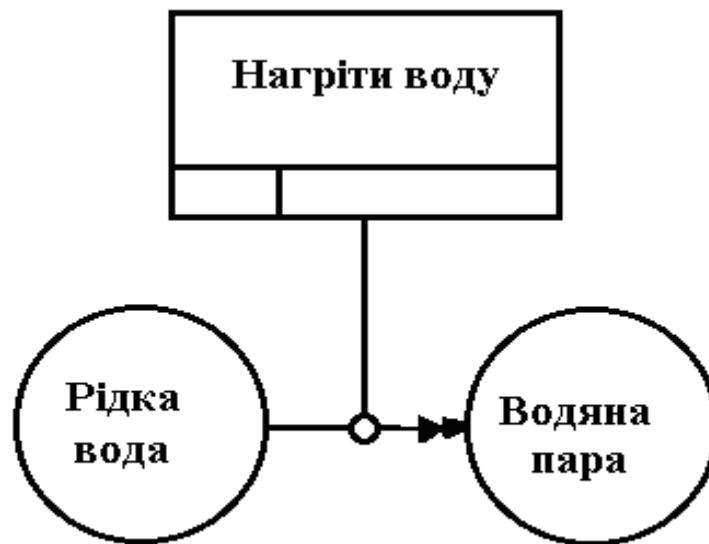


Рис. 2.50. Діаграма станів

Будова і властивості будь-якої системи можуть бути ефективно досліджені та задокументовані за допомогою наступних засобів: словника термінів, які використовуються при описі характеристик об'єктів, і процесів, що мають відношення до даної системи, точних і однозначних визначень всіх термінів цього словника і класифікації логічних взаємозв'язків між цими термінами.

### Стандарт IDEF9

IDEF9 метод є одним з методів опису специфікацій вимог [43]. Зокрема, метод IDEF9 призначений для визначення та аналізу умов та обмежень у бізнес-системах. Основним стимулом для розвитку даного методу є той факт, що умови та обмеження, які впливають на структуру підприємства, як правило, погано визначені й досліджені. Організація може функціонувати (що часто й трапляється), не знаючи про умови та обмеження, які діють у її системі. Проте, якщо існує завдання щодо поліпшення функціонування системи виробництва, підвищення його ефективності та адаптації підприємства до нових ринкових умов, знання про вимоги мають критично важливе значення. Кожен об'єкт у системі виробництва має свої власні вимоги, і їх виконання впливає на виконання завдання виробництва в цілому. Відповідно, якщо є необхідність змінити поведінку системи (наприклад, для підвищення ефективності виробництва), необхідно знати, які вимоги повинні бути

задоволені в першу чергу. В результаті використання IDEF9 системні вимоги описуються в спеціалізованому каталозі. Завдяки цьому завжди можна перевірити, які системні вимоги задоволені, і, відповідно, змінити умови роботи для отримання необхідного результату. Знаючи системні вимоги можна також пояснити, чому ця система працює таким або іншим чином, а також визначити, як поліпшити її роботу. Тому IDEF9 є незамінним інструментом бізнес-інженера. Вимоги визначають тип з'єднання, які можуть бути встановлені між об'єктами системи, між процесами, між об'єктами та процесами. Вимоги викладені в заявах. Прикладом вимоги-заяви є: "Тільки керівники департаментів, уповноважені підписувати накладні". Прикладом вимоги заява про процес: "Перевезення здійснюються тільки за наявності підписаної накладної" або "Максимальна вантажопідйомність підйомника складає 500 кг".

Таким чином, необхідно формулювати певні вимоги для функціонування системи. Крім того, в систему включено опис цілей роботи цієї системи. Це необхідно для визначення тільки тих вимог, об'єктів і процесів, які мають важливе значення для системи.

Вимоги поділяються на:

внутрішні, які здійснюють вплив усередині системи;

зовнішні, які впливають на вхід системи або які визначають зовнішню дію на систему;

міжсистемні, які дозволяють з'єднати системи.

Крім того, вимоги поділяються на сприятливі й обмежувальні в контексті даних умов. Обмежувальні вимоги є очевиднішими, і описуються у вигляді наступних симптомів проблеми: перевищення витрат, низька якість. Такі симптоми називаються доказами існуючих вимог у системі.

Результатом впливу вимоги на систему називається наслідок цієї вимоги. Наслідки можуть бути прямими й непрямими, ненавмисними, бажаними і небажаними в рамках даного контексту.

Для ефективного визначення вимог і управління ними необхідно швидко виявляти потреби та реагувати на типові проблеми, наприклад:

вартість вимоги виконання перевищує вимога вартості;

існуюча вимога більше не відповідає цілям підприємства;

вимога провокує непередбачені та небажані наслідки;

об'єкти системи, відповідальні за виконання вимог, не здатні

забезпечити їх виконання.

Системи, які переобтяжені застарілими або неприйнятними вимогами, втрачають продуктивність і витрачають ресурси на виконання цих вимог. Тому необхідно постійно здійснювати ревізії вимог.

Потенційними користувачами методу є підприємці, керівники організацій, фахівці й аналітики. Знання системних вимог може допомогти у реінжинірингу бізнес-процесів (РБП), управлінні якістю (TQM), в стратегічному плануванні.

Процедура визначення, підтвердження та подальшого вдосконалення вимоги складається з наступних етапів:

збір інформації (збір) – припускає збір даних про систему та отримання свідоцтв про вимоги;

класифікація – визначення системи умов, об'єктів, процесів і видів зв'язку;

гіпотези формування – вибір вимог-кандидатів на основі отриманих даних і доказів.

вимога реалізації – визначення кращих вимог-кандидатів;

вимога перевірки діяльності – перевірка виконання вимог із залученням зацікавлених осіб.

вимога вдосконалення – модернізація, поліпшення вимог і деталізація характеристик вимог.

Відповідно до стандарту IDEF9 вимоги відображаються у вигляді наочних схем [43]. Вимоги на схемах відображаються у вигляді прямокутників із закругленими кінцями, об'єкти і процеси у вигляді прямокутників. Зв'язки між об'єктами системи зображені у вигляді ліній. Зв'язки із вказівкою на зобов'язуючі обставини представлені у вигляді лінії з чорним кругом у кінці. Використовуються також вузли у вигляді квадратів для визначення типу з'єднання: AND, OR, X. На рис. 2.51 показано основні блоки стандарту IDEF9.

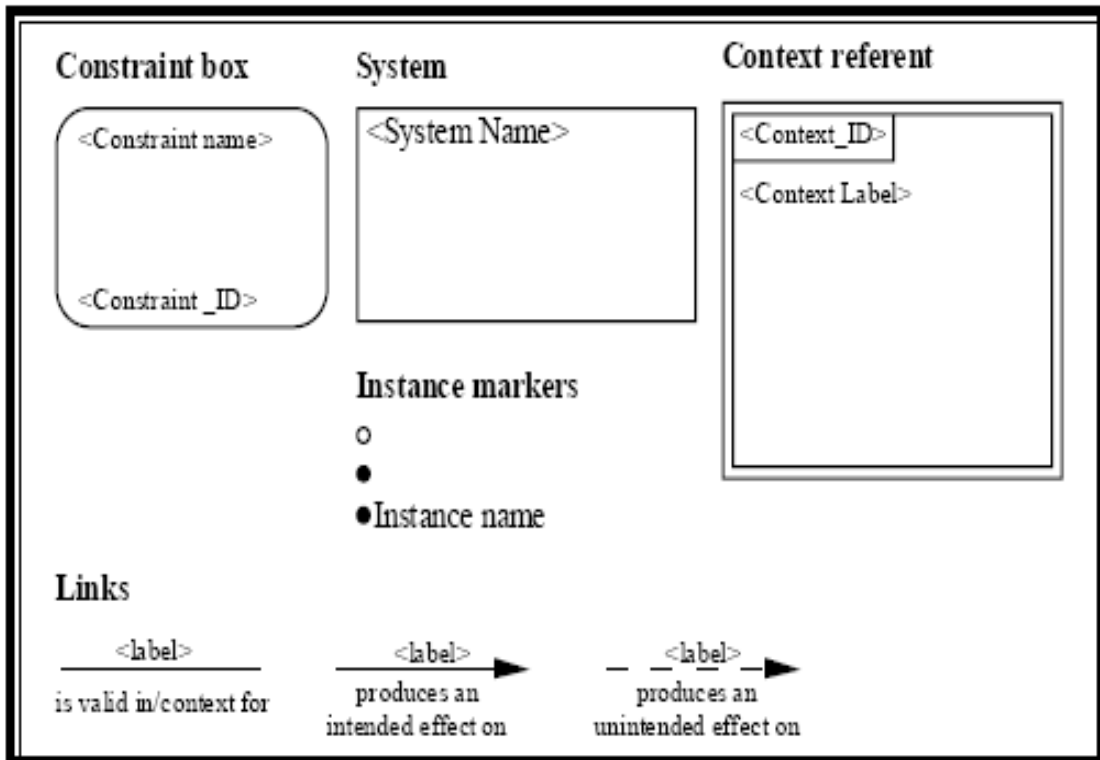


Рис. 2.51. Блоки опису вимог

Наприклад, є наступна вимога: "Документи для оплати споживаної електроенергії повинні бути підписані начальником відділу енергетики, головним бухгалтером". Відповідна схема матиме наступний вигляд (рис. 2.52).

Подвійний прямокутник представляє контекст завдання, прямокутник із закругленими кінцями показує саму вимогу, а прямокутники представляють об'єкти, які впливають на вимогу виконання. З'єднання AND визначає вимогу, в якій вказано, що рахунок повинен бути підписаний кожним із трьох суб'єктів (об'єктів).



Рис. 2.52. Опис вимоги на основі стандарту IDEF9



Для побудови схеми аналізу вимог щодо цілей використовуються наступні блоки (рис. 2.53).

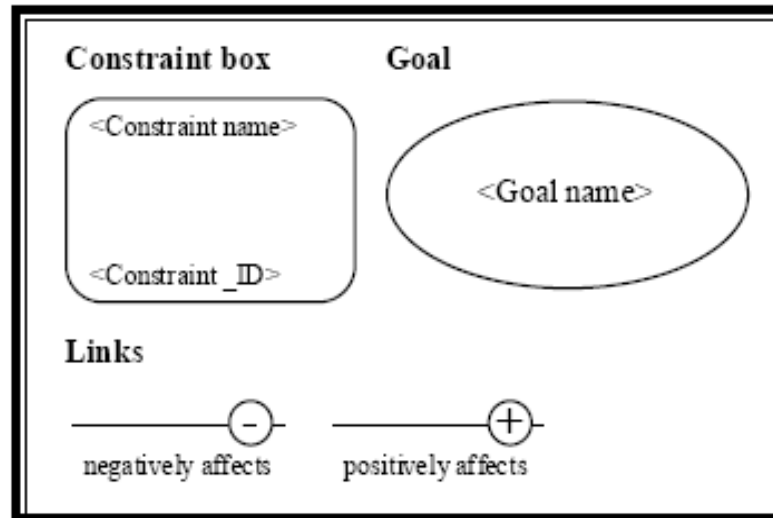


Рис. 2.53. Основні блоки опису цілей у стандарті IDEF9

Мета описується в блоку-овалі. Визначення мети повинно бути зрозумілим і конкретним, наприклад: "Підвищити ефективність праці робітників цеху", "Максимізувати прибуток підприємства".

Під час побудови схеми зліва описуються негативні фактори (симптоми, докази), а з справа – позитивні. В цілому схема відповідає принципам SWOT - аналізу (рис. 2.54).

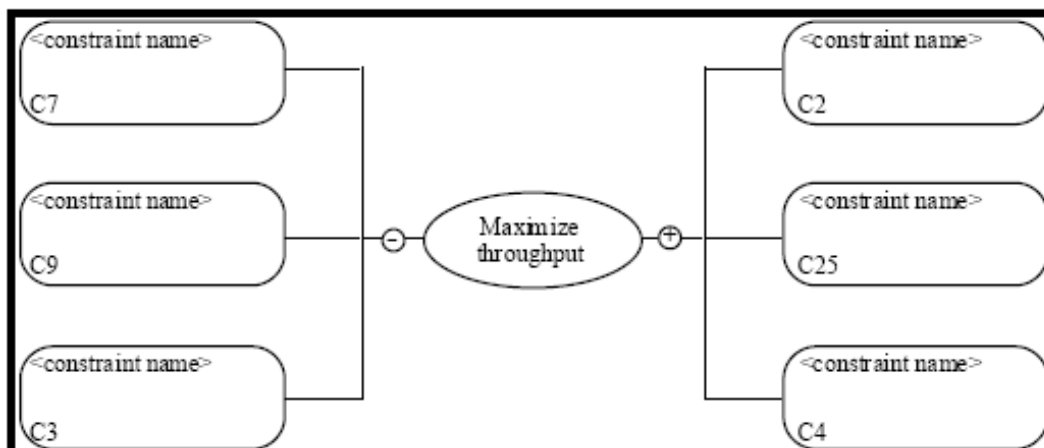


Рис. 2.54. Побудова схеми опису цілей

Слід зазначити, що ефективне управління змінами, значно сприяло документуванню бізнес-вимог. Визначення факторів та обмежень, які

впливають на досягнення мети дозволяє системно підійти до процедури визначення вимог щодо системи, яка проектується, або змінюється.

## **2.2. Альтернативна методологія опису предметної області проектування – об'єктно-орієнтоване проектування**

Ідеї системного підходу та їх реалізація в об'єктно-орієнтованій методології є базою сучасного проектування й управління складними системами. Такі поняття, як система, структура, стан, ієрархія, подія прийшли з системного аналізу і доповнені поняттями класу, об'єкту, атрибуту, інкапсуляції, відносин узагальнення, агрегації та іншими стали основою парадигми об'єктно-орієнтованого проектування (ООП), який широко використовується в сучасних автоматизованих системах. Ідеї ООП втілені в основних мовах, таких, як Express або UML.

Серед мов, які використовуються в системах CASE-засобів на стадії концептуального проектування складних систем, домінуюче положення має мова Unified Modeling Language (UML), що підтримується і розвивається міжнародним консорціумом OMG (Object Management Group). Мова UML призначена для опису, візуалізації та документування об'єктно-орієнтованих систем у процесі їх розробки, в першу чергу їх програмного забезпечення.

Розробка моделі додатку за допомогою мови UML починається з побудови діаграм використання (use case diagram). Ці діаграми характеризують функціональність створюваної системи з позицій користувача і служать для відображення взаємодії користувачів з проектованою системою. На діаграмах в овалах зазначені варіанти використання, тобто ті функції, які повинна виконувати система. Користувачі зображені у вигляді стилізованих фігурок, ними можуть бути не тільки люди, але й будь-які зовнішні утворення, що користуються послугами проектованої системи. Завдяки діаграмам використання визначається й узгоджується зовнішня функціональність системи і у результаті формується технічне завдання на розробку цієї системи.

Далі розробляються діаграми взаємодії "користувач-система", при цьому виявляються необхідні об'єкти додатку, будуються діаграми класів, формується компонентна структура програмного забезпечення.

Для зображення класів ООП використовують прямокутники, які поділяються на секції. У верхній секції записують ім'я класу, в середній – атрибути класу і в нижній – процедури класу.

Класи і їх відносини складають основу діаграм класів (class diagram). Зв'язки (асоціації) в цих діаграмах показують лініями між зв'язаними класами, причому на кінцях лінії можна вказати характер відношення ("один до одного", "один до багатьох" і т. п.). Відносини залежності, тобто впливи одного класу на інший, зображають стрілкою з пунктирною лінією, направленою до залежний елемент (залежність можна виявити після зміни опису підлеглого елементу, якщо змінюється опис елементу, що впливає). Якщо відношенням зв'язані рівноправні елементи, то така асоціація зображається суцільною лінією, якщо відношенням зв'язано більше двох класів, то в діаграму додається ромбоподібна зв'язка.

Окремі випадки асоціацій – узагальнення та агрегація. Відношення узагальнення (спадкоємство) зображають суцільною лінією, незафарбованою стрілкою, що закінчується, біля батьківського елементу. Відношення агрегації (відношення "частина – ціле") показують такою ж лінією, але з ромбоподібною стрілкою, що закінчується біля елементу "ціле". Ромбоподібна стрілка зафарбовується, якщо частини не можуть існувати без цілого, тобто якщо при ліквідації класу "ціле" ліквідовуються і всі його "частини". На основі діаграм класів можна надалі отримати імітаційну модель на об'єктно-орієнтованій мові програмування.

Діаграми взаємодії об'єктів (interaction diagrams) відносяться до діаграм процесів, що відображають поведінковий аспект моделювання. Діаграми взаємодії представлені діаграмами послідовностей і кооперації. Окрім них до діаграм процесів відносяться діаграми станів і діяльності.

У діаграмах послідовностей (sequence diagram), які також називаються діаграмами сценаріїв, відбивається послідовність подій, що полягають у діях одного об'єкта на деякий інший об'єкт. У цих діаграмах об'єкти зображаються прямокутниками і розташовуються кожен у своїй вертикальній колонці діаграми. Від кожного об'єкта паралельно осі часу йдуть їх так звані лінії життя. Кожна подія зображається горизонтальною лінією із стрілкою від лінії життя об'єкта, що посилає повідомлення, до лінії життя об'єкта, що приймає повідомлення. Над цими лініями можливе розташування пояснюючого тексту. Лінії розташовуються одна над іншою в порядку, в якому здійснюються події.

Слід зазначити, що в діаграмах взаємодії фігурують об'єкти, а не класи, це наголошується підкресленням імені об'єкта усередині

прямокутника об'єкта.

У діаграмах кооперації (collaboration diagram) об'єкти, представлені прямокутниками, які зв'язані між собою лініями, що зображують повідомлення (потік управління). Повідомлення впорядковані за часом появи. Біля лінії вказується порядковий номер повідомлення, напрям потоку і, можливо, деякі інші пояснення.

Діаграма станів (statechart diagram) є граф переходу станів, відомий за використанням у багатьох застосуваннях, але зображуваний за правилами мови UML. За допомогою діаграми станів моделюється послідовність подій, що відбуваються в системі.

Вершини графа переходу станів відповідають станам і в UML зображуються прямокутниками із вказівкою усередині прямокутників імен станів і, можливо, списків внутрішніх дій, допустимих в даному стані. Дуги графа відповідають переходам з одного стану в інший і зображуються лініями із звичайними стрілками. Біля лінії може бути записане ім'я події і/або вказані дії, що виконуються при переході. Перехід спрацьовує після виконання внутрішніх дій відповідного стану. Після імені події можна в прямих дужках записати умову – булеву функцію. Наприклад, перехід може спрацювати тільки в тому випадку, якщо функція А приймає значення true.

Діаграми діяльності (activity diagram) близькі за своєю семантикою до діаграм станів. Розрізняються вони тим, що в діаграмах діяльності кожній вершині графа відповідає деяка елементарна дія і перехід у новий стан відбувається після закінчення цієї дії. Вершини зображуються прямокутниками з округлими бічними сторонами, переходи – лініями із звичайними стрілками, переходи з декількох вершин в одну подальшу (переходи типу злиття – join) або з однієї вершини в декілька подальших (переходи типу розділення – fork) – потовщеними короткими лініями, так само як зображуються переходи в мережах Петрі. Перехід за умовою в одну з альтернативних вершин зображається за допомогою ромба, з якого виходять дуги переходів до альтернативних вершин.

У UML використовуються також діаграми компонентів і розгортання, які застосовуються для моделювання фізичної організації інформаційної системи.

Наприклад, до компонентів програмної системи можуть відноситися програмні модулі, бібліотеки, файли. У діаграмах розгортання показують розподіл класів за апаратними засобами.

Прикладом програмної системи, що підтримує мову UML, є система Rational Rose компанії Rational Software.

## **Розділ 3**

### **Інструментальні засоби структурного проектування**

#### **3.1. Класифікація CASE-засобів**

Вручну дуже важко розробити і графічно представити формальні специфікації системи, перевірити їх на повноту й несуперечність, і тим більше змінити. Якщо все ж таки вдається створити сувору систему проектних документів, то її переробка при появі серйозних змін складна та трудомістка. Якщо учасники проекту намагалися вдатися до ручної розробки, то перед ними виникали наступні проблеми:

- неадекватна специфікація вимог;
- нездатність виявляти помилки у проектних рішеннях;
- низька якість документації, що знижує експлуатаційні якості;
- затяжний цикл і незадовільні результати тестування.

Сучасні CASE-засоби охоплюють велику сферу підтримки численних технологій проектування інформаційних систем – від простих засобів аналізу і документування до повномасштабних засобів автоматизації, що покривають весь життєвий цикл програмного забезпечення (ПЗ).

Найбільш трудомісткими етапами розробки інформаційних систем є аналіз і проектування, у процесі яких CASE-засоби забезпечують якість технічних рішень, що приймаються, і підготовку проектної документації. При цьому велику роль відіграють методи візуального представлення інформації. Це передбачає побудову структурних або інших діаграм у реальному масштабі часу, використання різноманітної колірної палітри, наскрізну перевірку синтаксичних правил. Графічні засоби моделювання дозволяють розробникам в наочному вигляді вивчати існуючу інформаційну систему, перебудовувати її відповідно до поставлених цілей і наявних обмежень.

Зазвичай до CASE-засобів відносять будь-який програмний засіб, що автоматизує ту або іншу сукупність процесів життєвого циклу ПЗ і має наступні характеристики [4]:

могутні графічні засоби для опису й документування ІС, що забезпечують зручний інтерфейс з розробником і розвивають його творчі можливості;

інтеграція окремих компонент CASE-засобів, що забезпечує керованість процесом розробки інформаційної системи;

використання спеціальним чином організованого сховища проектних метаданих (репозиторія).

Інтегрований CASE-засіб (або комплекс засобів, що підтримують повний життєвий цикл ПЗ) містить наступні компоненти:

репозиторій, що є основою CASE-засобу. Він повинен забезпечувати зберігання версій проекту і його окремих компонентів, синхронізацію надходження інформації від різних розробників при груповій розробці, контроль метаданих на повноту і несуперечність;

графічні засоби аналізу і проектування, котрі забезпечують створення і редагування ієрархічно зв'язаних діаграм (DFD, ERD і ін.), що створюють моделі інформаційної системи;

засоби розробки додатків, включаючи мови 4GL і генератори коду;

засоби конфігураційного управління;

засоби документування;

засоби тестування;

засоби управління проектом;

засоби реінжинірингу.

Усі сучасні CASE-засоби можна класифікувати за типами та категоріями. Класифікація за типами відображає функціональну орієнтацію CASE-засобів на ті або інші процеси життєвого циклу. Класифікація за категоріями визначає ступінь інтегрованості за функціями, що виконуються, і включає окремі локальні засоби, котрі вирішують невеликі автономні завдання (tools), набір частково інтегрованих засобів, що охоплюють більшість етапів життєвого циклу інформаційних систем (toolkit) і цілком інтегровані засоби, що підтримують весь життєвий цикл інформаційних систем і зв'язані загальним репозиторієм. Окрім цього, CASE-засоби можна класифікувати за методологіями, що використовуються, і моделях систем і БД; ступенем інтегрованості із СКБД; доступним платформам.

Класифікація за типами в основному збігається з компонентним складом CASE-засобів і включає [1]:

засоби аналізу (Upper CASE), призначені для побудови та аналізу моделей предметної області (Design/IDEF (Meta Software), VPwin (Logic Works));

засоби аналізу та проектування (Middle CASE), котрі підтримують найбільш поширені методології проектування й використовуються для створення проектних специфікацій (Vantage Team Builder (Cayenne), Designer/2000 (ORACLE), Silverrun (CSA), PRO-IV (McDonnell Douglas), CASE.Аналитик (Макропроджект)). Виходом таких засобів є специфікації компонентів і інтерфейсів системи, архітектура системи, алгоритмів і структур даних;

засоби проектування баз даних, що забезпечують моделювання даних і генерацію схем баз даних (як правило, на мові SQL) для найбільш поширених СКБД. До них відносяться ERwin (Logic Works), S-Designor (SDP) і DataBase Designer (ORACLE). Засоби проектування баз даних є також у складі CASE-засобів Vantage Team Builder, Designer/2000, Silverrun і PRO-IV;

засоби розробки додатків. До них відносяться засоби 4GL (Uniface (Compuware), JAM (JYACC), PowerBuilder (Sybase), Developer/2000 (ORACLE), New Era (Informix), SQL Windows (Gupta), Delphi (Borland) і ін.) і генератори коду, що входять до складу Vantage Team Builder, PRO-IV і частково – в Silverrun;

засоби реінжинірингу, що забезпечують аналіз програмних кодів і схем баз даних і формування на їх основі різних моделей і проектних специфікацій. Засоби аналізу схем БД і формування ERD входять до складу Vantage Team Builder, PRO-IV, Silverrun, Designer/2000, ERwin і S-Designor. В області аналізу програмних кодів найбільшого поширення набувають об'єктно-орієнтовані CASE-засоби, що забезпечують реінжиніринг програм на мові C++ (Rational Rose (Rational Software), Object Team (Cayenne)).

Допоміжні типи включають:

засоби планування та управління проектом (SE Companion, Microsoft Project і ін.);

засоби конфігураційного управління (PVCS (Intersolv));

засоби тестування (Quality Works (Segue Software));

засоби документування (SODA (Rational Software)).

## 3.2. Засоби аналізу (Upper CASE)

### 3.2.1. AllFusion Process Modeler компанії LogicWorks

Обираючи продукт, котрий підтримує методологію IDEF0, бізнес-аналітик бере на себе зобов'язання виконувати суворі угоди вибраної методології. При цьому він отримує:

перевірену десятиріччями в різних сферах методологію моделювання і аналізу діяльності підприємства автоматизовану систему, котра: а) контролює синтаксис розробки моделі, що підтримує угоди для методології IDEF0; б) формує звіти, що представляють в зрозумілому і зручному для людини вигляді осмислену інформацію, що міститься в моделі, зокрема завдяки підтримці зазначених вище синтаксичних угод.

Завдяки суворому синтаксису, засоби моделювання, засновані на IDEF0, маючи описаний за цим стандартом бізнес-процес, як звіт видадуть [18]:

перелік ролей, необхідних для функціонування підприємства при використанні майбутньої системи автоматизації;

заготовки для проектування оргштатної структури підприємства;

заготовки для написання інструкцій щодо виконання якої-небудь роботи і частково для складання посадових інструкцій для працівника, який виконує ту або іншу роль, та інше.

Продукт AllFusion Process Modeler (що раніше носив ім'я BPwin) – засіб моделювання бізнес-процесів, призначений для вирішення численних проблем, що виникають у бізнесі.

AllFusion Process Modeler – це засіб моделювання, що допомагає в процесі аналізу, надає підтримку при створенні документації та дозволяє підвищувати ефективність складних бізнес-процесів. Моделі процесів дозволяють формувати докладну документацію, до якої увійдуть дані про необхідні операції, а також інформацію про методи управління зазначеними операціями та необхідні ресурси. Таким чином користувач отримує вичерпне уявлення про методики виконання різних операцій, від організації технологічних процесів у невеликих відділах до комплексної діяльності в масштабах усього підприємства.

Основні переваги інструменту [18]:



словарниковий інтерфейс дозволяє швидко і без зайвих зусиль вводити інформацію про моделі і управляти нею. Простий у використанні інтерфейс надає чудові можливості заповнення моделей;

вирішення AllFusion Process Modeler автоматизують більшість операцій у процесі створення моделей, надаючи семантичну суворість, необхідну для досягнення бажаного результату. "Підсвічування" об'єктів дозволяє розробникам уникнути багатьох помилок, що виникають при моделюванні;

розробники можуть налаштовувати параметри AllFusion Process Modeler для того, щоб забезпечити збір важливої для бізнесу інформації. Вказана інформація миттєво стає доступною в рамках редактора звітів AllFusion Process Modeler і може бути експортована в інші додатки (наприклад, в Microsoft Word або Excel);

завдяки підтримці діаграм Swim Lane користувачі отримують у своє розпорядження ефективний механізм візуалізації і оптимізації складних процесів. Зазначені діаграми допомагають організовувати процеси та дозволяють проглядати інформацію про процеси, ролі і повноваження;

продукт AllFusion Process Modeler передбачає можливість явного визначення ролей, які відповідають за опис і класифікацію завдань у рамках окремого процесу;

продукт AllFusion Process Modeler підтримує три основні стандарти, що використовуються для моделювання бізнес-процесів (IDEF0), технологічних процесів (IDEF3) і потоків даних (DFD) в нотації Йордана-ДеМарко;

AllFusion Process Modeler надає повну підтримку функціонально-вартісного аналізу (ABC), крім того рішення оптимізоване для проведення аналізу процесів. Засоби створення вичерпних звітів і двосторонній інтерфейс, забезпечений інструментами на базі технології ABC, дозволяють компаніям без зусиль запроваджувати стратегії управління, засновані на виконанні різних операцій;

конструктор шаблонів звітів (RTB) – це механізм, який використовується у AllFusion ERwin Data Modeler і AllFusion Process Modeler. Зазначений механізм дозволяє створювати вичерпні звіти і всеосяжні web-сайти. Розробники можуть формувати шаблони звітів, котрі використовуються до будь-якої моделі. Принцип "визначивши одного разу – використовуй багато разів" дозволяє організаціям встановлювати нові стандарти складання звітів;

моделювання допомагає розробникам вивчати в динаміці ефект від внесених змін. Будь-які сценарії можуть бути протестовані перед їх впровадженням, що гарантує вибір оптимального рішення.

### **3.3. Засоби аналізу і проектування (Middle CASE)**

#### **3.3.1. Vantage Team Builder**

VantageTeam Builder фірми CADRE (відомий раніше як WESTMOUNT I-CASE Yourdon) – один із найбільш могутніх на російському ринку засобів розробки інформаційних систем. Заснований на структурному підході, він дозволяє вести розробку паралельно за трьома напрямками – побудова моделі даних, розробка моделі поведінки системи (функціональної моделі) і проектування інтерфейсу системи.

VantageTeam Builder фірми CADRE дозволяє використовувати метод структурного проектування Yourdon'a з деякими додатковими інструментами.

Відповідно до реалізованого в VantageTeam Builder методу робота над проектом розбивається на чотири фази: Аналіз, Архітектура (Глобальне проектування), Дизайн (Детальне проектування) і Програмування [7].

У фазі аналізу VantageTeam Builder надає засоби розробки діаграм Потоків даних, що використовуються як для опису взаємодії системи із зовнішнім світом (контекстна діаграма), так і для визначення структури процесу обробки інформації (діаграми потоків даних нижчих рівнів). За бажанням можливе формалізоване задавання вимог до системи, що розробляється, у вигляді Списку подій. У цьому випадку забезпечується контроль відповідності контекстної діаграми і Списку подій. Крім того, забезпечується контроль правильності декомпозиції діаграм при переході з рівня на рівень.

Для розробки моделі даних пропонується досить широкий спектр засобів, що включає діаграми Структури даних, діаграми Стосунків сутностей і текстовий опис у формі Бекуса-Науера. Різні типи діаграм порівнюються між собою на несуперечність.

На фазі Архітектури системи можна розробити архітектуру обчислювального комплексу і уточнити апаратне оснащення робочих місць системи. Визначається розподіл завдань між обчислювальними

засобами, а також потоки даних між обчислювальними засобами, завданнями (окремими виконуваними модулями) і процесами обробки інформації у складі одного завдання. Створюються специфікації (формалізовані описи) процесів обробки інформації нижнього рівня. За необхідності можливе введення керівних потоків між процесами обробки інформації та розробка їх структури за допомогою спеціальних діаграм. Для опису дій керівних потоків можливе використання діаграм Зміни станів.

Для опису необхідної структури бази даних використовуються діаграми Стосунків сутностей в нотації Чена. Вони дозволяють вказувати різні типи реляційних стосунків між таблицями (загальне, тотальне, слабке, рекурсивне), зв'язки різної потужності (1-1, 1-N, N-M), а також різні суб- і супертипи, асоційовані сутності і зв'язки з атрибутами.

На фазі Архітектури починається визначення принципів побудови інтерфейсу системи з використанням діаграм Послідовності екранних форм. Вони дозволяють вказувати як умови переходів між екранними формами, так і дії, що виконуються при цьому.

На фазі дизайну здійснюється остаточне відпрацювання моделі даних, функціональної моделі і проектування інтерфейсу системи за допомогою вже згадуваних діаграм, а також ряду спеціальних типів діаграм, що дозволяють однозначно сформулювати вимоги до інтерфейсу і програми.

Розробка структури бази даних передбачає повний опис усіх атрибутів сутності (полів таблиць).

В основу розробки додатку покладені інтегровані діаграми Послідовності екранних форм. У фазі дизайнера в цих діаграмах для кожної форми вказується ім'я діаграми Змісту екранної форми і ім'я Структурної схеми програмного модуля, що реалізує відповідний процес обробки інформації.

Діаграма Змісту екранних форм дозволяє вказати таблиці та їх поля, представлені у екранній формі, спосіб їх уявлення (список – повноекранна форма), а також основні функціональні можливості (доступність таблиць на запис або лише для читання інформації).

VantageTeam Builder в комплекті з генератором GRINDERY дозволяє використовувати один і той же проект для генерації додатків на таких різних мовах, як Informix-4GL, NewEra і SUPERNOVA. Це, зокрема,

дозволяє спростити перехід від Informix-4GL до NewEra і самоосвоєння об'єктних підходів до розробки додатків.

Vantage Team Builder забезпечує виконання наступних функцій [7]:

проективання діаграм потоків даних, "сутність-зв'язок", структур даних, структурних схем програм і послідовностей екранних форм;

проективання діаграм архітектури системи – SAD (проективання складу і зв'язку обчислювальних засобів, розподіл завдань системи між обчислювальними засобами, моделювання стосунків типу "клієнт-сервер", аналіз використання менеджерів транзакцій і особливостей функціонування систем у реальному часі);

генерація коду програм на мові 4GL цільової СУБД з повним забезпеченням програмного середовища і генерація SQL-коду для створення таблиць БД, індексів, обмежень цілісності і процедур, що зберігаються;

програмування на мові C з вбудованим SQL;

управління версіями та конфігурацією проекту;

багатокористувальницький доступ до репозиторію проекту;

генерація проектної документації за стандартними та індивідуальними шаблонами;

експорт і імпорт даних проекту у форматі CDIF (CASE Data Interchange Format).

При побудові всіх типів діаграм забезпечується контроль відповідності моделей синтаксису методів, що використовуються, а також контроль відповідності однойменних елементів і їх типів для різних типів діаграм.

При побудові DFD забезпечується контроль відповідності діаграм різних рівнів декомпозиції. Контроль за правильністю верхнього рівня DFD здійснюється за допомогою матриці списків подій (ELM). Для контролю за декомпозицією складових потоків даних використовується декілька варіантів їх опису: у вигляді діаграм структур даних (DSD) або в нотації БНФ (форма Бекуса-Наура).

Для побудови SAD використовується розширена нотація DFD, що дає можливість вводити поняття процесорів, завдань і периферійних пристроїв, що забезпечує наочність проектних рішень.

При побудові моделі даних у вигляді ERD виконується її нормалізація і вводиться визначення фізичних імен елементів даних і таблиць, які використовуватимуться в процесі генерації фізичної схеми

даних конкретної СКБД. Забезпечується можливість визначення альтернативних ключів сутності і полів, що складають додаткові точки входу в таблицю (поля індексів), і потужності стосунків між сутностями.

Для підготовки проектної документації можуть використовуватися видавничі системи FrameMaker, Interleaf або Word Perfect. Структура і склад проектної документації можуть бути настроєні відповідно до заданих стандартів. Налаштування виконується без зміни проектних рішень.

### **3.3.2. Designer/2000 компанії Oracle**

До складу Oracle| Designer/2000 включені наступні модулі [6]:

а) інструментарій аналізу наочної області:

*Process Modeler* – засіб аналізу ділової активності організації. Дозволяє створити модель структури організації і прив'язати до цієї моделі функції, що здійснюються в різних підрозділах, і інформаційні потоки між функціями. Містить елементи бізнес-аналізу.

*Dataflow Diagrammer* – в цьому інструменті на базі DFD-діаграм деталізуються функції, що описані в *Process Modeler*. Використовується нотація Йордона – ДеМарко.

*Function Hierarchy Diagrammer* – цей модуль автоматично вибудовує ієрархії функцій, визначених у двох попередніх інструментах, є також можливість створювати прототипи функцій.

*Entity Relationships Diagrammer* – інструмент моделювання даних (діаграми "сутність-зв'язок"), якими оперують функції, визначені в *Dataflow Diagrammer*. Використовується нотація Баркера.

*Matrix Diagrammer* – інструмент для дослідження зв'язків між функціями і даними;

б) генератори структур:

*Database Wizard* – генерує реляційні структури з ER-діаграм.

*Application Wizard* – генерує ієрархію програмних модулів кінцевого додатка обробки даних на основі ієрархії функцій. При цьому може одночасно генеруватися декілька взаємопов'язаних підсистем для різних підрозділів однієї організації. Під час генерації автоматично виявляються

однакові з погляду використання інформаційних об'єктів функції, які можуть бути об'єднані в одному модулі;

в) інструментарій проектування додатка:

*Data Diagrammer* – інструмент для доопрацювання реляційних структур даних на основі нотації Баркера.

*Module Structure Diagrammer* – інструмент для управління структурою програмних модулів готового застосування. Тут визначаються типи модулів (меню, екранна форма, звіт) і їх ієрархії викликів.

*Module Data Diagrammer* – засіб для проектування екранного інтерфейсу програмного модуля на основі даних, що використовуються ним. Дозволяє без програмування досить гнучко управляти зовнішнім виглядом і поведінкою модуля, що генерується;

г) генератори даних і коди:

*Server Generator* – генерує базу даних на основі реляційних моделей.

*генератори коду* – на основі моделей, побудованих в *Module Data Diagrammer*, дозволяє створити початковий код для Visual Basic, C, Java, а також інструментів середовища Oracle Developer/2000 (Oracle Forms, Oracle Reports). У останньому випадку можливе циклічне доопрацювання додатка: у прототип додатка, що був згенерований, у Developer/2000 вносяться зміни, які видно для Designer/2000 і не втрачаються при повторній регенерації.

*Preferences Navigator* – засіб управління перевагами при генерації програмних модулів. Дозволяє встановлювати численні опції (наприклад, зовнішній вигляд елементів екранного інтерфейсу) як для проекту в цілому, так і для кожного модуля окремо.

Відповідно до загальної архітектури цієї CASE-системи виділяються наступні основні етапи процесу розробки системи [6]: моделювання та аналіз ділової діяльності, розробка концептуальних моделей предметної області, проектування прикладної системи та реалізація.

Перший етап пов'язаний з моделюванням і аналізом процесів, що описують діяльність організації, технологічні особливості роботи. Метою є побудова моделей існуючих процесів, виявлення їх недоліків і можливих джерел удосконалення. Цей етап не є обов'язковим у разі, коли існуюча технологія й організаційні структури чітко визначені, добре зрозумілі і не вимагають додаткового вивчення та реорганізації.

На другому етапі розробляються детальні концептуальні моделі предметної області, що описують інформаційні потреби організації, особливості функціонування і т. д. Результатом є моделі двох типів –

інформаційні, такі, що відображають структуру та загальні закономірності предметної області, функціональні – описують особливості завдань, що вирішуються.

На наступній стадії, етапі проектування, на підставі концептуальних моделей виробляються технічні специфікації майбутньої прикладної системи – визначається структура і склад бази даних, специфікується набір програмних модулів. Первинний варіант проектних специфікацій може бути отриманий автоматично за допомогою спеціальних утиліт на підставі даних концептуальних моделей.

На етапі реалізації створюються програми, що відповідають усім вимогам проектних специфікацій. Використання генераторів додатків, що входять до складу DESIGNER/2000, дозволяє повністю автоматизувати цей етап, істотно скоротити терміни розробки системи і підвищити її якість і надійність.

Відповідно до загальної архітектури інструментальні засоби, що входять до складу DESIGNER/2000, розбиваються на наступні компоненти [6]:

- засоби доступу до репозиторію;
- засоби управління репозиторієм;
- засоби аналізу ділової діяльності;
- засоби концептуального моделювання;
- засоби проектування системи;
- генератори додатків.

Засоби моделювання процесів, що входять до складу DESIGNER 2000 повністю відповідають вимогам наочності, виразності. Використання засобів мультимедіа, включаючи візуалізацію, звуковий супровід, відеозображення, анімацію, дозволяє додатково підвищити виразність моделей процесів і забезпечує можливість досліджувати їх динамічні характеристики.

Загальна модель ділової діяльності представляється у вигляді сукупності діаграм, кожна з яких описує окремий процес у вигляді його розбиття на взаємозв'язані один з одним кроки або підпроцеси.

Діаграма будується із стандартних елементів, основними з яких є:

*базовий процес* – процес, що визначає загальний контекст для всіх підпроцесів даної діаграми, тобто той головний процес, який описується даною діаграмою.

*Крок процесу* – певна частина діяльності в рамках базового процесу; згодом будь-який крок може бути основою для нового базового процесу та нової діаграми.

*Сховища* – призначені для представлення деякого інформаційного фонду або матеріального складу.

*Потоки* – описують передачу інформації або матеріальних об'єктів між двома кроками процесів або між процесом і сховищем.

*Організаційні одиниці* – представляють структуру підприємства або фірми; допускається ієрархічна структура організаційних підрозділів без обмеження на рівень вкладеності.

*Події* – можуть бути вхідні та вихідні; служать для зв'язку різних діаграм процесів.

У міру уточнення моделі класифікуються кроки процесів (введення даних, прийняття рішень, видача звіту), типізуються потоки (потік даних, матеріальний потік, тимчасовий потік), сховища (сховище даних, матеріальний склад) і ін. елементи. Для кожного типу можна використовувати своє графічне уявлення, забарвлення.

З будь-яким окремим кроком процесу або з потоком можна зв'язати спеціальне зображення у вигляді ікони, звуковий супровід, підвищуючи виразність і наочність всієї діаграми в цілому. Для детальнішого розгляду окремий крок процесу можна також пов'язати із заздалегідь заготовленим відеокліпом, що показує, як здійснюється відповідний технологічний етап у реальному житті.

Крім того, для кожного типу об'єктів або для окремого індивідуального об'єкта можна задати цілий спектр різноманітних кількісних параметрів, включаючи, наприклад, часові витрати і ресурси, необхідні для виконання того або іншого кроку процесу або для реалізації деякого потоку. Після цього за допомогою спеціальної процедури анімації можна "пожвавити" діаграму, побачити поведінку моделі в динаміці з урахуванням введених тимчасових затримок і витрат ресурсів, заданих для окремих складових. Таке "імітаційне" моделювання наочно показує, як розподіляється процес за часом, а також дає уявлення про динамічну залежність між різними підпроцесами. В разі паралельних кроків процесів спеціальний механізм "аналіз критичних ділянок" виявляє "вузькі" місця в технологічних ланцюжках і визначає можливі способи удосконалення діяльності.



Уся інформація, що вводиться в модель, може бути експортована в стандартні пакети електронних таблиць, такі, як Microsoft Excel і Lotus-1-2-3. Це є корисним для глибшого складного аналізу часових і вартісних витрат, для представлення результатів у вигляді спеціальних графіків і діаграм.

Засоби концептуального моделювання, що входять до складу DESIGNER/2000 є сукупністю графічних редакторів, що забезпечують підтримку інформаційних і функціональних моделей концептуального рівня. До складу цих засобів входять:

- графічний редактор ER-діаграм;
- графічний редактор ієрархії функцій;
- графічний редактор діаграм потоків даних.

Кожен з цих редакторів (діаграмерів) забезпечує зручні засоби роботи з діаграмами певного типу, що відповідають усім вимогам сучасного графічного інтерфейсу.

Усі специфікації проекту системи розробляються на основі моделей концептуального рівня і повинні забезпечувати виконання всіх вимог і обмежень, що містяться в них. Початковий варіант специфікацій можна сформувати автоматично за допомогою спеціальних утиліт, до яких відносяться:

- утиліта автоматичної генерації специфікацій бази даних за ER-моделлю;

- процедура генерації специфікацій модулів за ієрархією функцій і потоками даних.

За допомогою утиліти автоматичної генерації проекту бази даних за ER-діаграмою створюють специфікації майбутньої бази даних – її склад таблиць, перелік стовпців кожної з них, уточнення характеристик стовпців, опис обмежень цілісності і т. д.

При генерації схеми бази даних за ER-моделлю кожній сутності ставиться у відповідність таблиця, атрибутам – стовпці таблиць, а для кожного зв'язку створюються додаткові стовпці і визначаються обмеження цілісності типу зовнішніх ключів (foreign key constraint). Підтипи сутності в ER-моделі реалізуються в схемі бази даних у вигляді однієї загальної таблиці або у вигляді сукупності декількох – по одній на кожен підтип.

За ієрархією функцій і діаграм потоків даних можуть автоматично згенерувати специфікації програмних модулів прикладної системи, що

описують спільні характеристики модулів, з якими таблицями працює кожен з них, можливі дії з цими таблицями і т. д. При цьому аналізується діаграма ієрархії функцій і для кожної елементарної функції (нижньої в ієрархії) створюється специфікація модуля. Тип модуля встановлюється за певними правилами відповідно до вказаного для функції способу роботи з сутністю. Наприклад, якщо відомо, що функція використовує сутність лише для читання інформації, то вважається, що їй повинен відповідати модуль типу звіт. Передбачена також можливість "об'єднувати" деякі однотипні функції, тобто створювати один модуль відразу для декількох різних елементарних функцій. Основою для такого "об'єднання" служить також характер використання функціями сутності – якщо дві функції працюють з однією й тією ж сутністю і однаково їх використовують, то це може вважатися підставою створення для них одного спільного модуля. Розробникові надається можливість регулювати рівень "об'єднання" функцій шляхом вибору одного з наявного набору критеріїв.

Для роботи з будь-якими специфікаціями рівня проектування в DESIGNER/2000 передбачений цілий набір графічних редакторів, кожен з яких орієнтований на специфікації певного типу.

Генератори, що входять до складу DESIGNER/2000, розбиваються на дві групи [6]:

- генератор сервера;

- генератори клієнтської частини.

Генератор серверної частини автоматично будує за специфікаціями бази даних тексти програм на мові SQL, використовуючи всі засоби визначення баз даних, включаючи тригери, процедури, що зберігаються, і т. д..

Генератори клієнтської частини забезпечують автоматичне формування текстів програмних модулів за їх специфікаціями, записаними в репозиторії.

Текст програми, що генерується, за необхідності можна додатково доопрацювати і доповнити, користуючись безпосередньо відповідними інструментальними засобами розробки "нижнього рівня", що входять до складу DEVELOPER/2000. В цьому випадку з'являється деяка небезпека невідповідності кінцевої версії програми специфікаціям проекту, що особливо незручно при можливих змінах проекту, наприклад, схеми бази даних. Така зміна вимагає регенерації модулів з урахуванням нових

таблиць і взаємозв'язків, а, з іншого боку, виконання повторної генерації "знищить" стару версію програми зі всіма введеними "вручну" додатковими фрагментами. Для такої ситуації в DESIGNER/2000 передбачений спеціальний режим роботи генераторів – регенерація. При регенерації відбувається не заміна старого програмного тексту на новий, а його "редагування", коли по можливості вносяться лише необхідні зміни і не коректуються окремі фрагменти. При цьому можна управляти рівнем змін і заборонити модифікувати певні фрагменти програми (наприклад, все, що відноситься до опису розташування полів або тригери заданого типу і т. п.).

З додаткових засобів, що входять до складу генераторів, особливий інтерес представляють утиліти, що дозволяють використовувати DESIGNER/2000 не тільки для нових розробок, що починаються з етапу постановки завдання, але і для готових прикладних систем, які були спроектовані і реалізовані без використання CASE-засобів. Для такої ситуації передбачена можливість реінжинірингу системи – автоматичного створення за діючою версією додатку його опису в репозиторії, тобто специфікацій структури бази даних, програмних модулів типу екранних форм і звітів, ER-моделі. Отримані описи можуть бути використані для аналізу можливостей і виявлення недоліків існуючої версії додатка, а згодом модифіковані відповідно до нових вимог і умов функціонування системи. На основі відредагованих специфікацій проводиться генерація нової версії системи.

Designer/2000 забезпечує графічний інтерфейс при розробці різних моделей (діаграм) предметної області. В процесі побудови моделей інформація про них заноситься в репозиторій. До складу Designer/2000 входять наступні компоненти [6]:

Repository Administrator – засоби управління репозиторієм (створення та видалення додатків, управління доступом до даних з боку різних користувачів, експорт і імпорт даних);

Repository Object Navigator – засоби доступу до репозиторію, що забезпечують багатовіконний об'єктно-орієнтований інтерфейс доступу до всіх елементів репозиторію;

Process Modeller – засіб аналізу та моделювання ділової діяльності, що ґрунтується на концепціях реінжинірингу бізнес-процесів (BPR – Business Process Reengineering) і глобальної системи управління якістю (TQM – Total Quality Management);

Systems Modeller – набір засобів побудови функціональних і інформаційних моделей проекрованої ІС, що включає засоби для побудови діаграм "сутність-зв'язок (Entity-Relationship Diagrammer)", діаграм функціональних ієрархій (Function Hierarchy Diagrammer), діаграм потоків даних (Data Flow Diagrammer) і засіб аналізу та модифікації зв'язків об'єктів репозиторію різних типів (Matrix Diagrammer);

Systems Designer – набір засобів проектування ІС, що включає засіб побудови структури реляційної бази даних (Data Diagrammer), а також засоби побудови діаграм, що відображають взаємодію з даними, ієрархію, структуру і логіку додатків, що реалізовується процедурами, які зберігаються, на мові PL/SQL (Module Data Diagrammer, Module Structure Diagrammer і Module Logic Navigator);

Server Generator – генератор описів об'єктів БД ORACLE (таблиць, індексів, ключів, послідовностей і т. д.). Окрім продуктів ORACLE, генерація і реінжиніринг БД може виконуватися для СУБД Informix, DB/2, Microsoft SQL Server, Sybase, а також для стандарту ANSI SQL DDL і баз даних, доступ до яких реалізується за допомогою ODBC;

Forms Generator (генератор додатків для ORACLE Forms). Додатки, що генеруються, включають різні екранні форми, засоби контролю даних, перевірки обмежень цілісності та автоматичні підказки. Подальша робота з додатком виконується в середовищі Developer/2000;

Repository Reports – генератор стандартних звітів, інтегрований з ORACLE Reports, котрий дозволяє русифікувати звіти, а також змінювати структурне представлення інформації.

Репозиторієм Designer/2000 є сховище всіх проектних даних і може працювати в багатокористувальницькому режимі, забезпечуючи паралельне оновлення інформації декількома розробниками. В процесі проектування автоматично підтримуються перехресні посилання між об'єктами словника і можуть генеруватися більше 70 стандартних звітів про предметну область. Фізичне середовище зберігання репозиторію – база даних ORACLE.

### **3.3.3. Silverrun**

CASE-засіб Silverrun американської фірми Computer Systems Advisers, Inc. (CSA) використовується для аналізу і проектування ІС бізнес-класу і орієнтовано більшою мірою на спіральну модель ЖЦ. Воно

застосовується для підтримки будь-якої методології, заснованої на роздільній побудові функціональної та інформаційної моделей (діаграм потоків даних і діаграм "сутність – сутність-зв'язок").

Настроювання на конкретну методологію забезпечується вибором необхідної графічної нотації моделей і набору правил перевірки проектних специфікацій. У системі є готові налаштування для найбільш поширених методологій [14]: DATARUN (основна методологія, яка підтримується Silverrun), Gane/Sarson, Yourdon/DeMarco, Merise, Ward/Mellor, Information Engineering. Для кожного поняття, введеного в проект, є можливість додавання власних описів. Архітектура Silverrun дозволяє нарощувати середовище розробки в міру необхідності.

CASE-система Silverrun складається з наступних інструментів [14]:

BPM – модуль побудови моделей бізнес-процесів у формі діаграм потоків даних (DFD-діаграм). Підтримує нотації Yourdon/DeMarco, Gane/Sarson, Ward/Mellor і багато інших. Даний інструмент дозволяє автоматично перевірити цілісність побудованої моделі, причому список критеріїв перевірки визначається користувачем (наприклад, відсутність імен у елементів моделі, потоки даних типу "сховище – сховище" або "зовнішня сутність – зовнішня сутність" і т. д.). У модулі BPM забезпечена можливість роботи з моделями великої складності: автоматична перенумерація, робота з деревом процесів (включаючи візуальне перетягання гілок), від'єднання і приєднання частин моделі для колективної розробки.

ERX – модуль концептуального моделювання даних (побудова діаграм "сутність-зв'язок"). Цей модуль має вбудовану експертну систему, що дозволяє створити коректну нормалізовану модель даних за допомогою відповідей на змістовні питання про взаємозв'язок даних. Можлива автоматична побудова моделі даних з описів структур даних. Аналіз функціональних залежностей атрибутів дає можливість перевірити відповідність моделі вимогам третьої нормальної форми та забезпечити їх виконання. Перевірена модель передається в модуль RDM.

RDM – модуль реляційного моделювання, дозволяє створювати деталізовані моделі "сутність-зв'язок", призначені для реалізації в реляційній базі даних. У цьому модулі документуються всі конструкції, пов'язані з побудовою бази даних: індекси, тригери, процедури, що зберігаються, і т. д. Можливість створювати підсхеми відповідає підходу

ANSI SPARC до представлення схеми бази даних. На мові підсхем моделюються як вузли розподіленої обробки, так і призначені для користувача уявлення. Цей модуль забезпечує проектування й повне документування реляційних баз даних.

Менеджер репозиторію робочої групи (WRM – Workgroup Repository Manager) застосовується як словник даних для зберігання спільної для всіх моделей інформації, а також забезпечує інтеграцію модулів Silverrun в єдине середовище проектування.

Для автоматичної генерації схем баз даних у Silverrun існують мости до найбільш поширених СКБД: Oracle, Informix, DB2, Ingres, Progress, SQL Server, SQLBase, Sybase. Для передачі даних в засоби розробки додатків є мости до мов 4GL: JAM, PowerBuilder, SQL Windows, Uniface, NewEra, Delphi. Всі мости дозволяють завантажити в Silverrun RDM інформацію з каталогів відповідних СУБД або мов 4GL. Це дозволяє документувати, перепроєктувати або переносити на нові платформи бази даних і прикладні системи, що вже знаходяться в експлуатації. При використанні моста Silverrun розширює свій внутрішній репозиторій специфічними для цільової системи атрибутами. Після визначення значень цих атрибутів генератор додатків переносить їх у внутрішній каталог середовища розробки або використовує при генерації коди на мові SQL. Таким чином можна повністю визначити ядро бази даних з використанням всіх можливостей конкретної СКБД: тригерів, процедур, що зберігаються, обмежень посилальної цілісності. При створенні додатку на мові 4GL дані, перенесені з репозиторію Silverrun, використовуються або для автоматичної генерації інтерфейсних об'єктів, або для швидкого їх створення вручну.

#### **3.3.4. CASE.Аналитик**

CASE.Аналитик є практично єдиним в даний час конкуренто-спроможним російським CASE-засобом функціонального моделювання й реалізує побудову діаграм потоків даних відповідно до методології DFD. Його основні функції [1;12]:

- побудова та редагування DFD;
- аналіз діаграм і проектних специфікацій на повноту і несуперечність;
- отримання різноманітних звітів за проектом;

генерація макетів документів відповідно до вимог ГОСТ 19.XXX і 34.XXX.

База даних проекту реалізована у форматі СУБД Paradox і є відкритою для доступу.

За допомогою окремого програмного продукту (Catherine) виконується обмін даними з CASE-засобом ERwin. При цьому з проекту, який виконується в CASE.Аналитике, експортується опис структур даних і накопичувачів даних, які за певними правилами формують опис сутностей і їх атрибутів.

### **3.3.5. Power Designer компанії Sybase**

До складу Power Designer входять наступні модулі [1]:

Process Analyst – засіб для функціонального моделювання, підтримує нотації Йордона-Демарко, Гейна-Сарсона та ін. Є можливість описати елементи даних (імена, типи, формати), пов'язані з потоками даних і сховищами даних. Ці елементи передаються на наступний етап проектування, причому сховища даних можуть бути автоматично перетворені в сутності.

Data Analyst – інструмент для побудови моделі "сутність-зв'язок" і автоматичної генерації на її основі реляційної структури. Початкові дані для моделі "сутність-зв'язок" можуть бути отримані з DFD-моделей, створених в модулі Process Analyst. У ER-діаграмах допускаються лише бінарні зв'язки, завдання атрибутів у зв'язків не підтримується. Підтримуються діалекти мови SQL приблизно для 30 реляційних СКБД, при цьому можуть згенерувати таблиці, уявлення, індекси, тригери і т. д. В результаті породжується SQL-сценарій (послідовність команд CREATE), виконання якого створює спроектовану схему бази даних. Є також можливість встановити з'єднання із СУБД через інтерфейс ODBC. Інші можливості: автоматична перевірка правильності моделі, розрахунок розміру бази даних, реінжиніринг (побудова модельних діаграм для вже існуючих баз даних) і т. д.

Application Modeler – інструмент для автоматичної генерації прототипів програм обробки даних на основі реляційних моделей, побудованих в Data Analyst. Може бути отриманий код для Visual Basic, Delphi, а також для таких систем розробки в архітектурі "клієнт-сервер", як PowerBuilder, Uniface, Progress і ін. Генерація коду здійснюється на

основі шаблонів, відповідно управляти генерацією можна за рахунок зміни відповідного шаблону.

### **3.3.6. AllFusion Component Modeler**

AllFusion Component Modeler (раніше: Paradigm Plus) – CASE-засіб для проектування, візуалізації та підтримки якісних інформаційних систем. Забезпечуючи розширену підтримку спільного проектування і багаторазового використання компонентів моделі, Component Modeler істотно збільшує продуктивність команди розробників. Component Modeler спрощує створення стратегічно важливих, багатоланкових додатків масштабу підприємства, здатних адаптуватися до змінних потреб бізнесу.

Відмітні особливості продукту [18]:

легкість у використанні уніфікованої мови моделювання (UML). Component Modeler забезпечує повну підтримку UML, об'єктної мови моделювання для документування, специфікації та побудови додатків, заснованих на компонентах. Візуальне моделювання допомагає розробникам працювати зі складними моделями, розуміти вплив змін, що вносяться, і забезпечувати відповідність розробки вимогам кінцевого користувача;

пряме і зворотне проектування. Component Modeler підтримує синхронізацію проектування та реалізації додатку при будь-якій кількості змін початкового коду і ітерацій проектування, без маркерів коду або втрати даних, можливих при звичайному рішенні, заснованому на XML;

гнучкість перенесення інформації. Поєднання XML (як джерела даних) і шаблонів XSL забезпечує незвичайну гнучкість перенесення інформації як в Component Modeler, так і з нього. Крім того, підтримується обмін інформацією між продуктами сімейства ADvantage;

інтелектуальний редактор діаграм. Component Modeler включає можливість "Intelligent Relationship", яка спрощує створення великомасштабних моделей, скорочуючи кількість операцій, необхідних для графічного представлення зв'язків;

Model Expert Engine покращує якість моделювання за рахунок підтримки UML для проектування систем реального часу;

підказка сигнатури. Component Modeler забезпечує спливаючі вікна з підказками сигнатури UML для кожної зміни або створення об'єкта;



віти, що настроюються, публікуються в Web. Component Modeler генерує звіти, що настроюються, які сприяють взаєморозумінню між членами команди розробників. Завдяки повній підтримці шаблонів XML і XSL ці звіти можна генерувати в різних форматах і швидко публікувати в Web. Component Modeler також включає широко документований API, доступний з будь-якої стандартної скриптової мови.

Інтеграція Component Modeler з ERwin Data Modeler дозволяє організаціям з легкістю використовувати повторно і застосовувати в процесі групової розробки компоненти баз даних і додатків, забезпечуючи відповідність між моделями баз даних і компонентів.

Сумісне програмне забезпечення (ПЗ): Microsoft Visual J++, IBM VisualAge, Microsoft Visual C++, AllFusion Harvest Change Manager (Computer Associates), AllFusion ERwin Data Modeler (Computer Associates), ADvantage Joe (Computer Associates) та ін.

### ***3.3.7. BusinessObjects Enterprise***

BusinessObjects Enterprise забезпечує ефективне управління спеціалізованими інструментальними засобами для кінцевих користувачів (бізнес-звітами, динамічними запитами, аналітичними даними та інструментами управління продуктивністю) на надійній, масштабованій і відкритій сервісно-орієнтованій архітектурі [20].

Платформа бізнес-аналізу BusinessObjects Enterprise включає спеціальні технологічні сервіси для забезпечення максимальної ефективності використання. Ця платформа надає інтегровані засоби забезпечення безпеки, адміністрування та аудиту, що полегшують управління системою, а також засоби швидкого впровадження і налаштування, що спрощує роботу IT-адміністраторів і дозволяє їм спрямовувати свої зусилля на вирішення завдань бізнесу.

Працівники на всіх рівнях будь-якої організації несуть відповідальність за прийняття рішень, які повинні зміцнювати спільний успіх бізнесу. Для цього працівникам необхідний доступ до важливої інформації про діяльність організації, можливості аналізу і спільної роботи за цією інформацією з постачальниками, партнерами і замовниками.

BusinessObjects Enterprise поєднує можливості забезпечення кінцевих користувачів всіма необхідними засобами аналізу і створення

звітності з можливостями гнучкого управління системою, дозволяючи адміністраторам упевнено розгортати і стандартизувати необхідні BI (Business Intelligence, бізнес-аналіз)-рішення. BusinessObjects Enterprise забезпечує [20]:

- масштабовану і адаптивну сервісно-орієнтовану архітектуру;
- ефективну роботу кінцевого користувача;
- просте в розгортанні і управлінні рішення.

BI-платформа є одним з основних компонентів всієї IT-інфраструктури. Вона забезпечує можливість доступу до корпоративної інформації та підтримку прийняття рішень. Ця платформа гарантує користувачам можливість відстежувати і розуміти ділову інформацію, а також управляти нею. Тому платформа має бути побудована на основі високодоступної і ефективною архітектури для обробки, управління, аналізу та доставки важливої інформації широкому колу користувачів.

При використанні BusinessObjects Enterprise для побудови BI-системи, організації отримують можливість швидкого створення систем для вирішення конкретних бізнес-завдань, а також можливість подальшого розвитку цих систем у міру зростання своїх вимог до бізнес-аналізу.

Платформа BusinessObjects Enterprise спроектована для забезпечення масштабованості, надійності, відмовостійкості, розширюваності та безперервної доступності в режимі 24/7 (24 години в день, 7 днів на тиждень). BusinessObjects Enterprise враховує важливість масштабного розгортання, підтримка Unicode і сумісна з операційними системами Microsoft Windows, Sun Solaris, IBM AIX, HP-UX і Linux. Таким чином, організації можуть почати з одного BI-проекту на одній операційній платформі, а далі поступово стандартизувати BI-рішення в масштабі всього підприємства з використанням різних операційних платформ.

У зв'язку з необхідністю розповсюдження аналітичної інформації як усередині, так і за межами організації, ключовим завданням стало забезпечення масштабованості BI-платформи. Платформа BusinessObjects Enterprise володіє необхідною масштабованістю для забезпечення по-треб зростаючого числа користувачів, здатна обробити кількість інформації, що збільшується, і здатна бути масштабованою як на одній обчислювальній машині, так і у складі кластера, зберігаючи високий рівень продуктивності.

BusinessObjects Enterprise надає кінцевим користувачам доступ до достовірної інформації, яку можна безпосередньо інтегрувати в портали, в зовнішні мережі, в аналітичні системи і в додатки для управління ефективністю діяльності організацій. Інноваційні інструменти, такі, як Encyclopedia і Process Tracker протягом усього процесу прийняття рішень дозволяють кінцевим користувачам здійснювати контроль, управління і взаємодію за допомогою інтуїтивного зрозумілого інтерфейсу, реалізованого на базі Web.

BusinessObjects Enterprise XI включає новий порталний інтерфейс для бізнес-аналізу, призначений для кінцевих користувачів, – BusinessObjects InfoView. InfoView полегшує доступ до інформації і спрощує її розуміння, що сприяє більшій ефективності роботи користувачів з даними. Портал InfoView надає єдиний Web-інтерфейс, що дозволяє отримати доступ і працювати з будь-яким типом BI-ресурсів, включаючи звіти, аналітику, інформаційні панелі (dashboard), системи показників ефективності (scorecard) і стратегічні карти.

Тематичні дискусії допомагають користувачам зрозуміти бізнес-контекст інформації, що міститься в документах. Це дозволяє проникнути в суть проблеми і додає упевненість в рішеннях, що приймаються. Функціональність тематичних дискусій повністю інтегрована у всі елементи середовища BusinessObjects Enterprise. Використовувати дискусії можна як у межах порталу InfoView, так і в рамках інструментальних панелей з управління продуктивністю і систем сукупних показників. Таким чином, користувачі можуть брати участь у дискусіях і обговорювати своє розуміння проблеми в межах будь-якого документа, з яким вони працюють.

До складу BusinessObjects Enterprise включені [20] засоби обробки інформації (підготовки звітів) за розкладом для документів Business Objects Web Intelligence і Crystal Reports, а також для EPM-метрик, доступних для кінцевих користувачів і адміністраторів. Інтегровані можливості обробки інформації за розкладом у поєднанні з системою безпеки, заснованої на ролях користувачів, дозволяють технічним фахівцям і користувачам швидко та легко доставляти інформацію потрібним адресатам. Розклад може бути складений на щоденній і щотижневій основі, а також відповідно до щомісячного або поквартального бізнес-календаря. Користувачі можуть поширювати документи за допомогою електронної пошти або направляти їх у

системний накопичувач, виводити на системний принтер або викладати у вигляді файлу з метою подальшої взаємодії.

Платформа BusinessObjects Enterprise тісно інтегрована з продуктами Microsoft Office, включаючи можливість публікації та управління документами Word, Excel і PowerPoint в системному репозиторії. За допомогою цього продукту користувачі зможуть просто й ефективно вставляти у свої документи, таблиці і презентації точні та оновлювані бізнес-аналітичні дані. Для забезпечення спільного прийняття рішень такі документи можуть бути надані відповідно до правил безпеки іншим користувачам.

Можливість швидко та ефективно привести систему в робочий стан дає співробітникам IT-відділу більше часу на роботу з постійно змінними вимогами кінцевих користувачів. Просте управління системою за допомогою повнофункціональної консолі управління через Web забезпечує адміністраторів можливістю видаленої конфігурації системи за допомогою будь-якого Web-браузера. Платформа BusinessObjects Enterprise є могутнім набором BI-сервісів, оснащених широким спектром інструментів (Software Development Kit, SDK) для розробки Java-, .NET-додатків і Web-сервісів, що забезпечує простоту використання цієї платформи розробниками. Розробники зможуть використовувати SDK-інструменти для тісної інтеграції засобів бізнес-аналізу в додатки і портали в рамках всієї організації.

BusinessObjects Enterprise – повнофункціональна масштабована платформа бізнес-аналізу, яка забезпечує користувачів всією необхідною інформацією і допомагає провести глибокий аналіз даних і виявити приховані в них тенденції і закономірності, що дозволить прийняти своєчасні ефективні бізнес-рішення, розширивши таким чином конкурентні переваги організації.

### **3.4. Засоби проектування баз даних**

#### **3.4.1. ERwin Data Modeler компанії LogicWorks**

ERwin Data Modeler – засіб розробки структури бази даних (БД) [5]. ERwin поєднує графічний інтерфейс Windows, інструменти для побудови ER-діаграм, редактори для створення логічного та фізичного опису моделі даних і прозору підтримку провідних реляційних СУБД і

настільних баз даних. За допомогою ERwin можна створювати або проводити зворотне проектування (реінжиніринг) баз даних.

Керівники проектів можуть за допомогою ERwin Data Modeler ретельно задокументувати структуру БД, отримати звіти презентаційної якості та забезпечити ефективне управління проектом, використовуючи середовище для проектування AllFusion Model Manager (раніше: ModelMart).

Можливі дві точки зору на інформаційну модель і, відповідно, два рівні моделі. Перший – логічний (точка зору користувача) – описує дані, задіяні в бізнесі підприємства. Другий – фізичний – визначає представлення інформації в БД. ERwin об'єднує їх в єдину діаграму, що має декілька рівнів уявлення.

Оскільки ERwin Data Modeler підтримує роботу з БД на фізичному рівні, враховуючи особливості кожної конкретної СКБД, адміністратори БД можуть з його допомогою максимально підвищити продуктивність інформаційної системи. Розробники за допомогою ERwin Data Modeler можуть спочатку, використовуючи візуальні засоби, описати схему БД, а потім автоматично згенерувати файли даних для вибраної реляційної СУБД (пряме проектування). Автоматично генеруються також тригери, що забезпечують посиальну цілісність БД. ERwin Data Modeler підтримує нотації проектування даних IDEF1X, IE і Dimensional [5].

Користувач описує структуру даних візуально. Він задає прообразами реляційних таблиць, сутності з їх атрибутами і за допомогою миші "натягує" між ними зв'язки, які є прототипами реляційних стосунків.

ERwin Data Modeler дозволяє за вже існуючими файлами БД відновлювати логічну структуру даних. Це називається зворотним проектуванням. Воно дозволяє, по-перше, переносити структуру БД (без даних) з однієї СУБД в іншу і, по-друге, досліджувати старі проекти. Цей процес найбільш поширений при переході з однієї технології на іншу (з файл-сервер на клієнт-сервер), а також при зміні сервера БД. На основі моделі даних надається можливість створювати звіти, які дозволяють істотно спростити процес документування технічного проекту.

ERwin підтримує пряме і зворотне проектування 20 типів баз даних різних виробників, від настільних до реляційних СУБД і спеціалізованих СКБД, призначених для створення сховищ даних.

Відмітні особливості ERwin Data Modeler [5]:

*Архітектура рівня проектування (Design Layer).* Забезпечується гнучкість генерації моделей даних, яка повністю відповідає потребам організації. Продукт підтримує роздільні логічні та фізичні моделі, разом із змішаними логічними/фізичними моделями. Зберігається інформація про відношення та хронологію всього процесу проектування, що дозволяє користувачеві швидко визначати вплив змін, зроблених на одному рівні, на наступний рівень.

*Технологія трансформації (Transform Technology).* Фізична структура бази даних рідко відповідає оригінальній логічній структурі. Для досягнення прийнятної продуктивності сучасні eBusiness-додатки вимагають денормалізації таблиць. Технологія трансформації ERwin Data Modeler дозволяє реалізувати цей тип змін, разом з тим підтримуючи цілісність структури оригіналу.

*Визначення стандартів (Defining Standards).* ERwin Data Modeler забезпечує визначення і подальшу підтримку стандартів за допомогою словника доменів (Domain Dictionary), редактора стандартів іменування (Naming Standards Editor) і редактора стандартів типів даних (Datatype Standards Editor). Словник доменів містить атрибути, котрі використовуються повторно і забезпечує використання несуперечливих імен і визначень протягом проектування бази даних. Редактор стандартів іменування дозволяє користувачам створювати словник допустимих слів, скорочень і правил іменування. Редактор стандартів типів даних дає можливість користувачам визначати стандарти для типів даних. Допускається використання як визначуваних користувачем, так і типів даних конкретної СКБД.

*Управління великими моделями.* ERwin Data Modeler полегшує управління моделями великих підприємств за рахунок використання предметних областей (Subject Areas) і екранів, що зберігаються (Stored Displays). Предметні області надають індивідуальним проектувальникам можливість сфокусованого погляду, розділяючи модель на дрібніші і за рахунок цього полегшує керування підмножинами. Екрани, що зберігаються, надають множинні графічні представлення моделі або її предметних областей, тим самим полегшуючи обмін інформацією між спеціалізованими групами користувачів.

*Повне порівняння (Complete Compare).* Ця технологія автоматизує синхронізацію моделі та бази даних. Вона порівнює модель з базою даних, відображає відмінності і дозволяє користувачеві вибрати, які

відмінності необхідно перемістити в модель, а які згенерувати в базі даних. Якщо зміни моделі переміщені в базу даних, автоматично генерується скрипт, що змінює базу даних.

*Генерація схеми бази даних.* У ERwin Data Modeler включені оптимізовані шаблони тригерів тригерів, і міжплатформена макромова, що підтримує налаштування тригерів і процедур, що зберігаються. Виходячи з фізичної структури моделі, генеруються повні визначення наступних елементів бази даних відповідно до цільової СКБД: бази даних /табличні простору, таблиці та уявлення; стовпці з обмеженнями за замовчуванням і обмеженнями доменів; первинні ключі зовнішні ключі та індекси; процедури, що зберігаються, і код тригерів та ін.

*Проектування сховищ і вітрин даних.* ERwin Data Modeler надає специфічну техніку моделювання для проектування сховищ даних, – такі як розмірне моделювання за схемою "зірки" або "сніжинки" – додаючи упевненість проектувальникам, що сховище даних оптимізоване як за продуктивністю, так і за аналітичними можливостями. Крім того, ERwin Data Modeler здатний збирати й документувати широкий спектр інформації про сховище даних, включаючи джерела даних, логіку трансформації даних і правила управління даними.

Бази даних: Oracle, InterBase, Ingres, Microsoft SQL Server, Clipper, ODBC, DB2, dBASE, Paradox, FoxPro, Rdb, HIRDB, Red Brick Warehouse, Informix, SAS, SQL Anywhere, Microsoft Access, SQL Base, Teradata, Sybase.

Засоби розробки: Delphi, PowerBuilder, Visual Basic, Oracle Designer і багато інших.

CASE-засоби: Rational Rose, AllFusion Process Modeler (раніше: BPwin) і Oracle Designer (експорт-імпорт), AllFusion Model Manager (раніше: ModelMart), AllFusion Component Modeler (раніше: Paradigm Plus).

ERwin Data Modeler не прив'язаний до технології якої-небудь конкретної фірми, що постачає СУБД або засоби розробки. Він підтримує різні сервери баз даних і настільні СУБД, а також може звертатися до бази даних через ODBC.

Можливості ERwin Data Modeler доповнює могутня лінійка продуктів ADvantage від Computer Associates. ADvantage – лінійка продуктів для підтримки всіх стадій розробки інформаційних систем, багато в чому аналогічна лінійці продуктів Rational Software.

Центральною частиною ADvantage є лінійка взаємно інтегрованих CASE-засобів AllFusion Modeling Suite. Вона включає ERwin Data Modeler, Process Modeler (раніше: BPwin) для моделювання бізнес-процесів, Data Model Validator (раніше: ERwin Examiner) для перевірки моделей баз даних і Component Modeler (раніше: Paradigm Plus) для моделювання додатків і генерації об'єктного коду. Data Model Validator доповнює функціональність ERwin Data Modeler, дозволяючи шукати помилки в моделях ERwin Data Modeler і в структурі баз даних, попутно навчаючись моделюванню завдяки режиму підказок.

У ERwin існують два рівні уявлення та моделювання – логічний і фізичний. Вибір між логічним і фізичним рівнем відображення здійснюється через лінійку інструментів або меню. Всередині кожного з цих рівнів є наступні режими відображення [5]:

режим "сутності" – усередині прямокутників відображається ім'я сутності (для логічної моделі) або ім'я таблиці (для фізичного представлення моделі); служить для зручності огляду великої діаграми або розміщення прямокутників сутності на діаграмі;

режим "визначення сутності" служить для презентації діаграми іншим людям;

режим "атрибути". При переході від предметної області до моделі потрібно вводити інформацію про те, що складає сутність. Ця інформація вводиться шляхом задавання атрибутів (на фізичному рівні – колонок таблиць). У цьому режимі прямокутник – сутність ділиться лінією на дві частини – у верхній частині відображаються атрибути (колонки), складові первинного ключа, а в нижній – решта атрибутів. Цей режим є основним при проектуванні на логічному і фізичному рівнях;

режим "первинні ключі" – усередині прямокутників – сутностей показуються лише атрибути/колонки, складові первинного ключа;

режим "піктограми". Для презентаційних цілей кожній таблиці може бути поставлена у відповідність піктограма (bitmap);

режим "показ дієслівної фрази". На дугах зв'язків показуються дієслівні фрази, що зв'язують сутність (для логічного рівня) або імена зовнішніх ключів (для фізичного рівня).

Орієнтація ERwin на засоби 4GL дозволяє задати для майбутніх додатків більшість параметрів, безпосередньо пов'язаних з базою даних, вже на стадії проектування інформаційної моделі.



### **3.4.2. S-Designor**

S-Designor є CASE-засобом для проектування реляційних баз даних. За своїми функціональними можливостями і вартістю він близький до CASE-засобу ERwin, але відрізняється нотацією, яка використовується на діаграмах.

Це графічний інструмент для проектування структури реляційних баз даних. S-Designor реалізує методологію інформаційного моделювання, засновану на представленні інформаційних об'єктів і взаємозв'язків між ними у вигляді ER-діаграми ("сутність-зв'язок"). Використовується нотація IE (Information Engineering).

S-Designor реалізує стандартну методологію моделювання даних і генерує опис БД для таких СКБД, як ORACLE, Informix, Ingres, Sybase, DB/2, Microsoft SQL Server і ін. [8]. Для існуючих систем виконується реінжиніринг БД.

S-Designor сумісний з рядом засобів розробки додатків (PowerBuilder, Uniface, TeamWindows і ін.) і дозволяє експортувати опис БД в репозиторії даних засобів. Для PowerBuilder виконується також пряма генерація шаблонів додатків.

При проектуванні в S-Designor використовується дворівневий підхід. Перший рівень – концептуальна модель даних (ER-діаграма). Другий рівень – фізична модель даних. При переході на другий рівень S-Designor автоматично генерує відповідну фізичну модель даних для заданої СКБД, зважаючи на специфіку останньої [8].

Особливість реалізації циклу розробки в S-Designor полягає в тому, що він дозволяє виконувати "наскрізне проектування". Це означає, що на основі розробленої концептуальної моделі можна автоматично згенерувати фізичну та після цього виконати генерацію структури БД. При зворотному проектуванні послідовність дій прямо протилежна. Можна отримати фізичну модель на основі структури БД і після цього автоматично згенерувати концептуальну модель. На кожному етапі можна вносити зміни в моделі концептуального та фізичного рівнів [8].

S-Designor підтримує широкий спектр засобів розробки 4GL. Підтримка засобів розробки 4GL полягає в тому, що на етапі проектування моделі даних забезпечується можливість проектувати відображення елементів об'єктів, пов'язаних з базою даних.

У S-Designor є можливості як імпорту розширених атрибутів у репозиторій цільового засобу розробки, так і експорту з репозиторію в модель. Розширені атрибути можна зберігати у файлах і згодом при розробці нової моделі даних завантажувати спроектовані раніше розширені атрибути в нову модель.

### **3.5. Засоби розробки додатків**

#### **3.5.1. Uniface**

Uniface – продукт фірми Compuware (США) – є середовищем розробки великомасштабних додатків в архітектурі "клієнт-сервер" і має наступну компонентну архітектуру [1]:

Application Objects Repository (репозиторій об'єктів додатків) містить метадані, які автоматично використовуються рештою компонентів упродовж життєвого циклу ІС (прикладні моделі, описи даних, бізнес-правила, екранні форми, глобальні об'єкти та шаблони). Репозиторій може зберігатися в будь-якій з баз даних, які підтримуються Uniface;

Application Model Manager підтримує прикладні моделі (ER-моделі), кожна з яких є підмножиною загальної схеми БД з погляду даного додатку, і включає відповідний графічний редактор;

Rapid Application Builder – засіб швидкого створення екранних форм і звітів на базі об'єктів прикладної моделі. Він включає графічний редактор форм, засоби прототипування, відладки, тестування і документування. Реалізований інтерфейс з різноманітними типами віконних елементів управління (Open Widget Interface) для існуючих графічних інтерфейсів – MS Windows (включаючи VBX), Motif, OS/2. Універсальний інтерфейс представлення (Universal Presentation Interface) дозволяє використовувати одну й ту ж версію додатка в середовищі різних графічних інтерфейсів без зміни програмного коду;

Developer Services (служби розробника) – використовуються для підтримки великих проектів і реалізують контроль версій (Uniface Version Control System), права доступу (розмежування повноважень), глобальні модифікації і т. д.. Це забезпечує розробників засобами паралельного проектування, вхідного і вихідного контролю, пошуку, перегляду, підтримки і видачі звітів за даними системи контролю версій;

Deployment Manager (управління розповсюдженням додатків) – засоби, що дозволяють підготувати створений додаток для розповсюдження, встановлювати і супроводжувати його (при цьому платформа користувача може відрізнятись від платформи розробника). У їх склад входять мережні драйвери і драйвери СКБД, сервер додатків, засоби розповсюдження додатків і управління базами даних. Uniface підтримує інтерфейс практично зі всіма відомими програмно-апаратними платформами, СКБД, CASE-засобами, мережними протоколами та менеджерами транзакцій;

Personal Series (персональні засоби) – використовуються для створення складних запитів і звітів у графічній формі (Personal Query і Personal Access – PQ/PA), а також для перенесення даних в такі системи, як WinWord і Excel;

Distributed Computing Manager – засіб інтеграції з менеджерами транзакцій Tuxedo, Encina, CICS, OSF DCE.

Підтримуються наступні СКБД: DB2, VSAM і IMS; PolyServer забезпечує також взаємодію з ОС MVS.

Середовище функціонування Uniface – всі основні UNIX-платформи і MS Windows.

### **3.5.2. JAM**

Пакет JAM має модульну структуру і складається з наступних компонент [15]:

ядро системи. Ядро є закінченим модулем і дозволяє повністю розробляти додатки; всі інші модулі є додатковими для ядра і самостійно використовуватися не можуть;

JAM/DBi – модуль інтерфейсу до СКБД. Для кожної СКБД, котра підтримується JAM`ом, існує спеціалізований модуль. Наприклад, JAM/DBi-Oracle, JAM/DBi-Informix, JAM/DBi-ODBC і т. д.;

JAM/RW – модуль генератора звітів;

JAM/CASEi – модуль інтерфейсу до CASE верхнього рівня (наприклад, CASE структурного аналізу і дизайну). Для кожної CASE, котра підтримується JAM`ом, існує спеціалізований модуль. Наприклад, JAM/CASEi-TeamWork, JAM/CASEi-Innovator і т. д.;

JAM/TPi – модулі інтерфейсу до Моніторів Транзакцій (MT). Існує два варіанти цих модулів – TPi-Server і TPi-Client. Для кожного MT,

котрий підтримується JAM`ом, існують спеціалізовані модулі. Наприклад, JAM/TPi-Server TUXEDO і т. д.

### Інтерфейс до CASE структурного аналізу і дизайну JAM/CASEi

Одним із основних завдань CASE є розробка структури БД. Інформація про структуру БД зберігається в репозиторії CASE. Модуль JAM/CASEi дозволяє здійснити обмін інформацією між Візуальним Репозиторієм Об'єктів JAM і репозиторієм CASE аналогічно тому, як структура БД імпортується в Репозиторій JAM безпосередньо з БД. Відмінність полягає в тому, що у разі інтерфейсу до CASE цей обмін є двонаправленим, тобто інформацію з Репозиторію JAM можна експортувати в репозиторій CASE.

На рис. 3.1 наведена схема взаємодії JAM і CASE з використанням модуля JAM/CASEi [15].

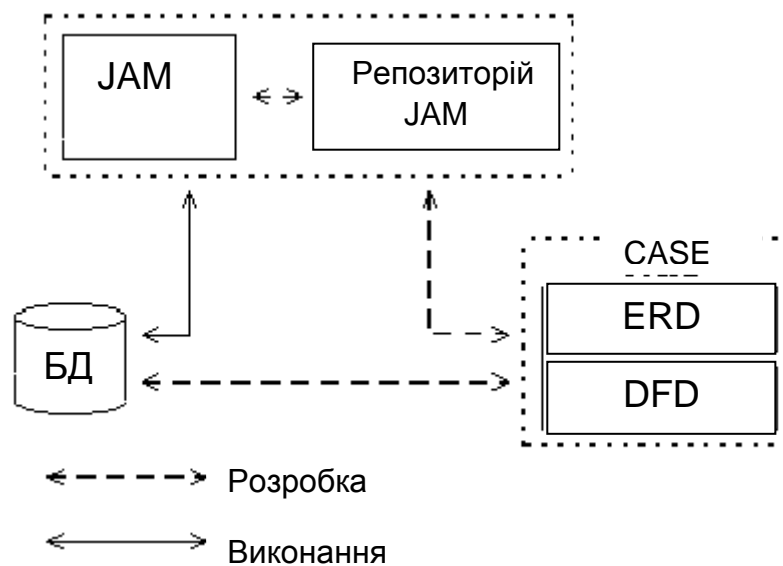


Рис. 3.1. Взаємодія JAM, CASE і СКБД

Модуль JAM/CASEi дозволяє здійснювати імпорт/експорт не лише в розділ ERD репозиторію CASE, але і в розділ DFD.

Окрім модуля JAM/CASEi, фірма-виробник поширює модуль JAM/CASEi Developer's Kit. За допомогою цього модуля можна самостійно розробити інтерфейс (тобто спеціалізований модуль JAM/CASEi) для конкретного інструменту CASE, якщо готового модуля JAM/CASEi для даної CASE-системи ще не існує.

Ядро системи включає наступні основні компоненти [15]:

редактор екранів. До складу редактора екранів входять: середовище розробки екранів, візуальний репозиторій об'єктів, власна СУБД JAM-JDB, менеджер транзакцій, відладчик, редактор стилів;

редактор меню;

набір допоміжних утиліт;

засоби виготовлення промислової версії додатка.

Міст (інтерфейс) SILVERRUN-RDM <-> JAM реалізує взаємодію між CASE-засобом Silverrun і JAM (перенесення схеми бази даних і екранних форм додатка). Даний програмний продукт має 2 режими роботи:

прямий режим (Silverrun-RDM->JAM) призначений для створення об'єктів CASE-словника і елементів репозиторію JAM на основі представлення схем в SILVERRUN-RDM. У цьому режимі міст дозволяє, виходячи з представлення моделей даних інтерфейсу в SILVERRUN-RDM, проводити генерацію екранів і елементів репозиторію JAM. Міст перетворює таблиці і стосунки реляційних схем RDM в послідовність об'єктів JAM відповідних типів. Методика побудови моделей даних інтерфейсу в SILVERRUN-RDM припускає застосування механізму підсхем для прототипування екранів додатка. За описом кожної з підсхем RDM міст генерує екранну форму JAM;

зворотний режим (JAM->Silverrun-RDM) призначений для перенесення модифікацій об'єктів CASE-словника в реляційну модель SILVERRUN-RDM.

### **3.5.3. PowerBuilder**

PowerBuilder – це об'єктно-орієнтований інструмент для професійної розробки додатків в середовищі клієнт/сервер, що дозволяє колективам розробників легко і швидко створювати графічні додатки, які мають доступ до баз даних і іншої корпоративної інформації, що зберігається локально або на мережних серверах [19].

PowerBuilder включає набір інструментів, що забезпечують всебічну підтримку розробки додатків: Інтелектуальний SQL (SQL Smart), Зручні об'єкти (Object Easy), Колективну розробку (Enterprise Enabled) і Інтегроване середовище проектування (Developer Designed).

Платформа для розробки додатку в середовищі клієнт/сервер повинна включати всі необхідні технології для створення додатків

масштабу підприємства. PowerBuilder надає відмінний набір базових інструментів для роботи, проте існує цілий ряд продуктів, які часто використовуються для розробки інформаційних систем, наприклад, системи контролю початкових текстів, системи проектування структур баз даних, додаткові бібліотеки класів і системи автоматизованого тестування, а також різні сервери баз даних, мережні продукти, системи управління документами і т. д.

Багато провідних виробників беруть участь у програмі CODE, інтегруючи свої продукти в середовище PowerBuilder. Партнерів умовно можна розділити на декілька категорій, залежно від функцій "програмного продукту-партнера" [19]:

тестування додатків особливо важливо при швидкому переході до розробок в середовищі клієнт/сервер. Для ефективного тестування додатків використовуються спеціалізовані інструменти для планування, розробки і виконання тестів додатків (TeamTest (Software Quality Automation), QA Partner (Mercury Interactive), PowerRunner (Segue Software), Automated Test Facility (Softbridge));

управління всіма технічними й організаційними аспектами підтримки проектів з розробки інформаційних систем. Із середовищем розробки PowerBuilder тісно інтегровані: ER-Modeller (Chen & Associates), Exceleator (Intersolv), System Engineer (LBMS), ERwin/ERX (Logic Works), Visible Analyst Workbench (Visible Systems);

бібліотеки класів і елементів управління для зберігання спільних компонент, які часто повторно використовуються (Object Start (Greenberg & Russel), POWERTOOL (PowerServ), PowerClass (ServerLogic), Formula One, ImageStream, First Impression (Visual Tools));

інтерфейс до баз даних (Allbase/SQL, Image/SQL (Hewlett Packard), DB2/2, DB2/6000 (IBM), Online (Informix), SQL Server (Microsoft), Oracle Server (Oracle), SQL Server, SQL Anywhere (Sybase), EDA/SQL (Information Builders));

підтримка тривірневої (у більш загальному випадку, багаторівневої) архітектури (Visual DCE (Gradient Technologies), Distributed Computing Intergator for TopEnd (Tangent International), Distributed Computing Intergator for Tuxedo (Tangent International), EncinaBuilder (Transarc));

управління даними та потоками інформації в розподіленому середовищі масштабу підприємства (Notes (Lotus Development));

інтеграція ІС для великих підприємств PowerBuilder, не вдаючись до низькорівневого програмування (Extra! (Attachmate Corp), Rumba (Wall Data));

інтеграція систем управління документами і зображеннями з додатками на PowerBuilder (WORKFLO (FileNet Corp), Wang OpenImage (Wang Laboratories), Watermark Discovery Edition (Watermark Software));

прямі зв'язки з середовища PowerBuilder до головних систем управління об'єктами та контролю версій для управління розробкою об'ємних додатків (PVCS (Intersolv), Endeavor (Legent Corp), RCS (Mortice Kern Systems)).

#### **3.5.4. Developer/2000**

Developer/2000 забезпечує розробку переносних додатків, що працюють в графічному середовищі Windows, Macintosh або Motif. У середовищі Windows інтеграція додатків Developer/2000 з іншими засобами реалізується через механізм OLE і елементи VBX. Взаємодія додатків з іншими СУБД (DB/2, DB2/400, Rdb) реалізується за допомогою засобів ORACLE Client Adapter для ODBC, ORACLE Open Gateway і API.

#### **3.5.5. New Era**

Комплекс Informix-NewEra включає наступні інструментальні засоби [3]:

інструменти візуального програмування – генератор вікон і генератор складових органів управління;

компілятори й інтерпретатор мови NewEra;

компілятор файлів повідомлень і довідок;

генератор додатків;

інтерактивний символний відладчик програм на мові NewEra;

електронна документація, доступна в інтерактивному режимі;

набір прикладів елементарних додатків (рецептів);

демонстраційні бази даних;

допоміжний продукт Informix-NewEra ViewPoint Pro, який містить інструменти адміністрування баз даних і репозиторію даних NewEra; засоби для побудови простих додатків без програмування на SQL або інших мовах.

Найважливіші характеристики Informix-NewEra [3]:

багатоплатформність, кросплатформність. Informix-NewEra працює в системах MS Windows, OSF/Motif і Macintosh. Додатки або компоненти додатків, розроблені на одній платформі, можуть працювати на інших платформах;

об'єктно-орієнтований характер інструменту. Переваги даного підходу – повторне використання коду, бізнес-моделювання, хороша пристосовність додатків до змінних вимог, простота супроводу, інтегрованість з бібліотеками від незалежних постачальників. Допустиме використання зовнішніх бібліотек, розроблених на C і C++;

підтримка разом з об'єктно-орієнтованими розробками традиційного структурного програмування;

підтримка групових розробок. Програмісти спільно використовують бібліотеки, файли початкових кодів і ресурсів. Можливе застосування системи управління версіями і конфігураціями PVCS (Intersolv);

стилістична одноманітність призначеного для користувача інтерфейсу забезпечується за рахунок механізму спадкоємства і підтримки репозиторію даних. У репозиторії зберігаються такі елементи інтерфейсу, як колірне і шрифтове оформлення, маски введення і формати виведення даних, правила верифікації, заголовки та мітки стовпців, котрі використовуються при виводі, і т. п.;

відкритість створюваних додатків по відношенню до СКБД, що використовується. У комплект постачання входять бібліотеки взаємодії із СУБД Informix, а також із СКБД, доступними через інтерфейс ODBC. Визначений об'єктний інтерфейс для створення інших аналогічних бібліотек доступу до СКБД;

спадкоємність по відношенню до Informix-4gl. Завдяки істотній сумісності мов NewEra і 4gl, можливе перенесення додатків 4gl в середовище Informix-NewEra;

можливість створення розподілених багатоланкових додатків, в яких обробка даних відокремлена від обслуговування інтерфейсу з користувачем.

### **3.5.6. Delphi**

Delphi дозволяє розширювати функціональні можливості середовища, в якому працює розробник за допомогою так званих



"відкритих інтерфейсів". Такий підхід дозволяє використовувати Delphi вже в ролі загального ядра набору інструментальних засобів на всіх етапах створення прикладних систем – починаючи з CASE-систем і закінчуючи генерацією документації за проектами, що створюються, з повною їх інтеграцією в IDE [13].

Ряд виробників програмних продуктів заявили про підтримку ними Delphi на достатньо високому рівні інтеграції (маючи на увазі, наприклад, для CASE-систем, не тільки генерацію коду відповідно до синтаксису Object Pascal, але й доступ до таких продуктів безпосередньо з IDE). Як приклад можна привести компанію Popkin Software (виробника CASE-засобу System Architect). Відомий ряд систем контролю версій – Intersolv PVCS і MKS Source Integrity, здатен працювати з Delphi (32-розрядна версія PVCS входить в постачання Delphi Client/Server Suite 2.0) і, наприклад, моніторами транзакцій (існує досвід взаємодії з Novell Tuxedo та ін.).

### **3.6. Засоби реінжинірингу**

#### **3.6.1. Rational Rose**

В основі роботи Rational Rose лежить побудова різного роду діаграм і специфікацій, що визначають логічну та фізичну структури моделі, її статичні й динамічні аспекти. До їх числа входять діаграми класів, станів, сценаріїв, модулів, процесів [16].

У складі Rational Rose можна виділити 6 основних структурних компонентів: репозиторій, графічний інтерфейс користувача, засоби переглядання проекту (browser), засоби контролю проекту, засоби збирання статистики і генератор документів. До них додаються генератор коду (індивідуальний для кожної мови) і аналізатор для C++, котрий забезпечує реінжиніринг – відновлення моделі проекту за початковими текстами програм.

Репозиторій є об'єктно-орієнтованою базою даних. Засоби перегляду забезпечують "навігацію" за проектом, зокрема, переміщення за ієрархіями класів і підсистем, перемикання від одного вигляду діаграм до іншого і т. д. Засоби контролю і збирання статистики дають можливість знаходити та усувати помилки у міру розвитку проекту, а не після завершення його опису. Генератор звітів формує тексти вихідних документів на основі інформації, що міститься в репозиторії.

Засоби автоматичної генерації коду програм на мові C++, використовуючи інформацію, що міститься в логічній і фізичній моделях проекту, формують файли заголовків і файли описів класів і об'єктів. Скелет програми, що створюється таким чином, може бути уточнений шляхом прямого програмування на мові C++. Аналізатор коду C++ реалізований у вигляді окремого програмного модуля. Його призначення полягає в тому, щоб створювати модулі проектів у формі Rational Rose на основі інформації, що міститься у визначених користувачем початкових текстах на C++.

В процесі роботи аналізатор здійснює контроль правильності початкових текстів і діагностику помилок. Модель, отримана в результаті його роботи, може цілком або фрагментарно використовуватися в різних проектах.

Аналізатор володіє широкими можливостями налаштування по входу і виходу. Наприклад, можна визначити типи початкових файлів, базовий компілятор, задати, яка інформація має бути включена у модель, що формується, і які елементи вихідної моделі слід виводити на екран. Таким чином, Rational Rose/C++ забезпечує можливість повторного використання програмних компонент.

У результаті розробки проекту за допомогою CASE-засобу Rational Rose формуються наступні документи [16]:

- діаграми класів;
- діаграми станів;
- діаграми сценаріїв;
- діаграми модулів;
- діаграми процесів;
- специфікації класів, об'єктів, атрибутів і операцій;
- заготовки текстів програм;
- модель програмної системи, що розробляється.

Засоби Rose надають розробникам:

проектування систем – кодогенерація. Дозволяє намальовану модель перетворити в опис на конкретній мові програмування. Підтримується: C++, Ada, Java, Basic, Xml, Oracle. Також до Rose сторонніми компаніями розробляються спеціальні мости до тих мов програмування, що не входять в стандартне постачання, наприклад, до Delphi;

можливості зворотного проектування – реінжинірингу, коли готову інформаційну систему (наприклад, на C++) або базу даних (на Oracle) "закачують" в Rose з метою отримання наочної візуальної (структурної) моделі;

round-trip engineering – поєднує можливості перших двох підходів, коли створюється система, а після проходження деякого часу еволюційного періоду (доопрацювань) піддається знов реінжинірингу і знову кодогенерації.

## **3.7. Засоби планування і управління проектом**

### **3.7.1. SE Companion**

Інструментальний засіб SE Companion є середовищем, в якому реалізований електронний варіант методології DATARUN. Він дозволяє [4,12]:

- створити гіпертекстовий опис методології у вигляді ієрархії опису стадій, етапів і операцій розробки;

- створити гіпертекстовий опис всіх методів і методик реалізації процесів ЖЦ ПЗ;

- виділити з гіпертекстового опису ієрархію процесів ЖЦ ПЗ для планування і управління процесом створення ПЗ (ієрархію робіт);

- змінювати гіпертекстові описи ЖЦ і методів так, як це необхідно розробникові, іншими словами, проводити авторизацію методології і відстежувати ці зміни в ієрархії робіт, призначеній для управління проектом;

- прив'язати до процесів ЖЦ інструментальні засоби підтримки цих процесів і забезпечити виклик інструментальних засобів з відповідних екранів гіпертекстового довідника;

- забезпечити проглядання гіпертекстових екранів опису методів, що використовуються з інструментальних засобів;

- забезпечити підтримку процесу управління розробкою, зокрема, за рахунок взаємодії з засобом планування робіт MS Project, оцінювання трудомісткості проекту, відстежування виконання робіт, створення графіків робіт і ін.

### **3.7.2. AllFusion Model Manager**

Пакет AllFusion Model Manager (раніше відомий як ModelMart) – це масштабоване багатокористувальницьке середовище моделювання, що надає розробникам широкі можливості спільної роботи. Рішення AllFusion Model Manager дозволяє організувати взаємодію між всесвітньо відомими продуктами AllFusion ERwin Data Modeler і AllFusion Process Modeler від компанії Computer Associates International, Inc. (CA).

Командам розробників потрібні інструментальні засоби, що полегшують передачу інформації між працівниками в контрольованому середовищі. AllFusion Model Manager є одним з найнадійніших багатокористувальницьких середовищ моделювання, призначеним для виконання великомасштабних завдань, що вимагають координації дій різних учасників проектів.

Переваги AllFusion Model Manager [18]:

- забезпечення групової роботи з AllFusion ERwin Data Modeler і AllFusion Process Modeler;

- ефективніша спільна робота розробників моделей;

- підтримка сумісного використання моделі колективом розробників;

- підвищення рівня зв'язності і міри повторного використання елементів моделі;

- забезпечення відповідності стандартам моделювання організації;

- забезпечення розробників моделей засобами для визначення дії змін, що вносяться до моделі, і селективного відкату до попередньої версії.

Покращення процесу моделювання за рахунок введення підмоделей, контролю версій і синхронізації проектування.

Надання менеджерам можливості аналізу спільної роботи працівників з подальшим наданням різних рівнів повноважень.

Основні функції AllFusion Model Manager наведені в табл. 3.1:

### Функції AllFusion Model Manager

Функція	Опис	Переваги для користувачів
1	2	3
Менеджер версій ( <b>Version Manager</b> )	Менеджер версій точно визначає відмінності між версіями моделі та відстежує зміни з найвищим рівнем деталізації	Користувачі можуть відмінити будь-яку окрему зміну від попередньої версії без втрати проміжних оновлень
Менеджер вирішення конфліктів ( <b>Conflict Resolution Manager</b> )	Ця функція визначає конфлікти між змінами розробниками, що працюють і вносяться паралельно, на рівні об'єктів в межах моделі і дозволяє користувачам вибірково комбінувати результати	Завдяки точному уявленню про наслідки змін, що вносяться іншими проектувальниками проміжних оновлень, і будь-які конфлікти, які витікають з цих змін, розробники моделей можуть заощадити час і уникнути зайвих витрат і зусиль
Утиліта синхронізації ( <b>Synchronizer</b> )	Утиліта синхронізації дозволяє користувачам AllFusion ERwin Data Modeler і AllFusion Process Modeler спільно використовувати метадані	Ця функція дозволяє інформаційним системам організації оптимально підтримувати бізнес-процеси. Такі елементи, як сутність (Entities), атрибути (Attributes) і визначення (Definitions) можуть спільно використовуватися цими інструментами в обох напрямках

Закінчення табл. 3.1

1	2	3
Підмоделі <b>(Sub-Modeling)</b>	Підмоделі дозволяють користувачеві логічно ділити велику модель на декілька менших предметних областей з кращими можливостями для управління. У міру оновлення підмоделі також змінюється і велика початкова модель	Підмоделі полегшують розробку завдяки використанню більш керованих вузькоспеціалізованих моделей і зменшують кількість системних ресурсів, необхідних для завантаження і вивантаження даних при роботі з моделлю в режимі check in / check out (реєстрація / контроль)
Менеджер управління змінами <b>(Change Control Manager)</b>	Під час відкритої сесії моделювання користувач може включити цю функцію для відображення наслідків змін, що вносяться до моделі	Менеджер управління змінами дозволяє користувачам чітко бачити наслідки їх додавань в модель і вибірково застосовувати або скасовувати ці зміни
Злиття моделей <b>(Model Merge)</b>	Злиття моделей об'єднує дві моделі в одну	Це дозволяє користувачам погоджувати інформацію між об'єктами, що дублюються
Проглядання звітів <b>(Report Browser)</b>	Report Browser генерує звіти за наслідками аналізу наслідків по всьому вмісту бази, включаючи численні бібліотеки та моделі	Це дозволяє користувачам для будь-якої моделі, що зберігається в AllFusion Model Manager, використовуючи зручний графічний інтерфейс, запускати вбудовані звіти або створювати свої власні
Аналіз впливу змін <b>(Impact Analysis)</b>	Користувачі можуть проводити оцінку впливу змін на модель після остаточного збереження цих змін у моделі	Аналіз впливу змін дозволяє користувачам переглядати зроблені протягом однієї сесії зміни, і надає можливість відмінити ці зміни до стану, що передувало оновленню центральної моделі

### 3.7.3. AllFusion Model Navigator

AllFusion Model Navigator – інструмент, що надає доступ "тільки для читання" до моделей, створених за допомогою AllFusion ERwin Data Modeler (раніше: ERwin) і AllFusion Process Modeler (раніше: BPwin).

Model Navigator служить доповненням до цих програмних продуктів, забезпечуючи співробітникам, що не беруть участь у процесі моделювання, доступ до інформації, що міститься в моделях (для створення презентацій, аналізу і розробки додатків) [18].

AllFusion Model Navigator надає широкі можливості для спільного використання моделей користувачами, які не беруть прямої участі в проектуванні баз даних або моделюванні бізнес-процесів. AllFusion Model Navigator дозволяє розробникам різних додатків використовувати інформацію про взаємозв'язок таблиць баз даних для створення ефективних форм введення даних і OLAP-запитів. Застосовуючи AllFusion Model Navigator, наприклад, кожен користувач може працювати зі всією інформацією, що міститься в моделі даних AllFusion ERwin Data Modeler.

AllFusion Model Navigator – ефективний комунікаційний інструмент. Він підтримує стандартні можливості перегляду, навігації, друку і генерації звітів в AllFusion ERwin Data Modeler і AllFusion Process Modeler. При створенні презентацій користувачі можуть маніпулювати різними предметними областями і екранними формами, виділяти елементи діаграм шрифтами і кольором, збільшувати, зменшувати або переміщати об'єкти діаграм.

AllFusion Model Navigator дозволяє створювати і роздруковувати будь-які звіти AllFusion ERwin Data Modeler і AllFusion Process Modeler. Користувачі можуть безпосередньо роздруковувати діаграми, використовуючи стандартні діалоги друку.

При розробці додатків AllFusion Model Navigator дозволяє розглядати моделі даних як з клієнтського, так і з серверного боку. Розробники знатимуть, які стовпці присутні в кожній таблиці бази даних, які вказані ключі, обмеження і правила перевірки достовірності, і як взаємопов'язані різні таблиці бази даних. Крім того, розробники зможуть проглядати інформацію про атрибути клієнтського представлення таблиць в AllFusion ERwin Data Modeler, і, таким чином, визначати, як відображати і форматовувати дані в стовпцях.

## 3.8. Засоби конфігураційного управління

### 3.8.1. PVCS Version Manager

Мета конфігураційного управління (КУ) – забезпечити керованість і контрольованість процесів розробки і супроводу ПЗ. Для цього необхідна точна і достовірна інформація про стан ПЗ і його компонент в кожен момент часу, а також про всі передбачувані і виконані зміни.

Найбільш поширеним засобом КУ є PVCS фірми Intersolv (США), що включають ряд самостійних продуктів: PVCS Version Manager, PVCS Tracker, PVCS Configuration Builder і PVCS Notify [2].

**PVCS** Version Manager призначений для управління всіма компонентами проекту і ведення планомірної багатоверсійної і багатоплатформеної розробки силами команди розробників в умовах однієї або декількох локальних мереж. Поняття "проект" трактується як сукупність файлів. У процесі роботи над проектом проміжний стан файлів періодично зберігається в архіві проекту, ведуться записи про час збереження, відповідність один одному декількох варіантів різних файлів проекту. Окрім цього, фіксуються імена розробників, відповідальних за той або інший файл, склад файлів проміжних версій проекту і ін. Це дозволяє повернутися при необхідності до якого-небудь з попередніх станів файлу (наприклад, при виявленні помилки, яку в даний момент важко виправити).

**PVCS** Version Manager призначений для використання в робочих групах. Система блокувань, реалізована в PVCS Version Manager дозволяє запобігти одночасному внесенню змін до одного й того ж файлу. В той же час PVCS Version Manager дозволяє розробникам працювати з власними версіями загального файлу з напівавтоматичним дозволом конфліктів між ними.

Доступ до архівів PVCS Version Manager можливий не тільки через сам Version Manager, але і із понад 50 інструментальних засобів, зокрема MS Visual C і MS Visual Basic, Uniface, PowerBuilder, SQL Windows, JAM, Delphi, Paradox і ін.

Результатом роботи PVCS Version Manager є створений засобами файлової системи репозиторій, що зберігає в компактній формі всі робочі версії програмного продукту разом з необхідними коментарями і помітками.



### **3.8.2. PVCS Tracker**

Іншим засобом конфігураційного управління є PVCS Tracker – спеціалізована надбудова над офісною електронною поштою, призначена для обробки повідомлень про помилки в продукті, доставці їх виконавцям і контролю за виконання. Інтеграція з PVCS Version Manager дає можливість пов'язувати з повідомленнями ті або інші компоненти проекту. Звітні можливості PVCS Tracker включають безліч різновидів графіків і діаграм, що відображають стан проекту і процесу його відладки, зрізи за різними компонентами проекту, розробниками і тестувальниками. З їх допомогою можна наочно показати поточний стан роботи над проектом і її тимчасові тенденції.

Персонал, що працює з PVCS Tracker ділиться на п'ять груп залежно від їх обов'язків: користувачі, розробники, група тестування і контролю якості, група технічної підтримки і супроводу, управлінський персонал. Цим п'яти групам персоналу відповідають п'ять зумовлених груп PVCS Tracker [2]:

користувачі (Submitters) – мають обмежені права на внесення зауважень і повідомлень про помилки в базу даних PVCS Tracker;

розробники (Development Engineers) – мають право проводити основні операції з вимогами і зауваженнями в базі даних PVCS Tracker. Якщо розробники діляться на підгрупи, то для кожної підгрупи можуть бути задані окремі списки прав доступу;

тестувальники (Quality Engineers) – мають право проводити основні операції з вимогами і зауваженнями;

супровід (Support Engineers) – мають право вносити будь-які зауваження, вимоги і рекомендації в базу даних, але не мають прав щодо розподілу робіт і зміни їх пріоритетності та термінів виконання;

керівники (Managers) – мають право розподіляти роботи між виконавцями і ухвалювати рішення про їх належного виконання. Керівникам різних груп можуть задані різні права доступу до бази даних PVCS Tracker.

На додаток до цих п'яти зумовлених груп існує група адміністратора бази даних і 11 додаткових груп, які можуть бути налаштовані відповідно до специфічних посадових обов'язків працівників, які користуються PVCS Tracker.

Вимога або зауваження, яка надходить до PVCS Tracker проходить чотири етапи обробки:

реєстрація – внесення зауваження в базу даних;

розподіл – призначення відповідального виконавця і термінів виконання;

виконання – усунення зауваження, яке у свою чергу може викликати додаткові зауваження або вимоги на додаткові роботи;

приймання – приймання робіт і зняття їх з контролю або направлення на доопрацювання.

Для отримання змістовної інформації про хід розробки, PVCS Tracker дозволяє отримувати три типи статистичних звітів: частотні, тренди і діаграми розподілу.

Частотні звіти містять інформацію про частоту зауважень, що надходять, за одну годину тестування програмного продукту. Проте універсального частотного звіту не існує, оскільки на оцінку якості впливають тип методів тестування, серйозність виявлених помилок і значення дефектних модулів для функціонування всієї системи. Мале число фатальних помилок, що приводять до повної зупинки розробки, гірше за велике число зауважень до зовнішнього вигляду інтерфейсу користувача. Отже, частотні звіти мають бути налаштовані на виявлення якого-небудь конкретного аспекту якості для того, щоб їх можна було використовувати для прогнозування закінчення робіт над проектом.

Тренди містять інформацію про зміни того або іншого показника в часі і характеризують стабільність і безперервність процесу розробки. Вони дозволяють відповісти на питання:

чи встигає група розробників впоратися з зауваженнями, що надходять;

чи покращується якість програмного продукту і яка динаміка цього процесу;

як вплинуло те або інше рішення (збільшення числа розробників, введення змінного графіка, впровадження нового методу тестування) на роботу групи і т. п.

Діаграми розподілу – найбільш різноманітні та корисні форми звітів для здійснення оперативного керівництва. Вони дозволяють відповісти на запитання: який метод тестування ефективніший, які модулі викликають найбільше число нарікань, хто з розробників краще може

впоратися з конкретним типом завдань, чи немає перекосу в розподілі робіт між виконавцями, чи немає модулів, тестуванню яких було приділено недостатньо уваги і т. д.

**PVCS** Tracker призначений для використання в робочих групах, об'єднаних в загальну мережу. В цьому випадку центральна база або проект PVCS Tracker знаходиться на загальнодоступному сервері мережі, доступ до якого реалізується за допомогою ODBC-драйверів, що входять до складу PVCS Tracker. Головною особливістю PVCS Tracker порівняно із звичайним додатком СУБД є його здатність автоматично повідомляти користувача про надходження інформації, що його цікавить або відноситься до його компетенції, і гнучка система розподілу повноважень усередині робочої групи. За необхідності PVCS Tracker може використовувати для повідомлення окремих членів групи через електронну пошту.

**PVCS** Tracker підтримує групову роботу в локальних мережах і взаємодіє зі СУБД dBase, ORACLE, SQL Server і SYBASE за допомогою ODBC.

**PVCS** Tracker може бути інтегрований з будь-якою системою електронної пошти, що підтримує стандарти VIM, MAPI або SMTP.

**PVCS** Version Manager і PVCS Tracker оточені допоміжними компонентами: PVCS Configuration Builder і PVCS Notify.

### ***3.8.3. PVCS Configuration Builder***

**PVCS** Configuration Builder призначений для збирання остаточного продукту з компонент проекту. PVCS Configuration Builder дозволяє описувати процес збирання як на стандартній мові MAKE, так і на власній внутрішній мові, що має істотно великі можливості. PVCS Configuration Builder дозволяє здійснювати збирання програмного продукту на підставі файлів, що зберігаються в репозиторії PVCS Version Manager.

Звичайна процедура збирання програмного продукту за допомогою PVCS Configuration Builder складається з трьох кроків:

- будується файл залежностей між початковими модулями;
- до отриманого файлу вносяться зміни з метою його налаштування та оптимізації;
- здійснюється збирання програмного продукту з початкових модулів.

### **3.8.4. PVCS Notify**

**PVCS** Notify забезпечує автоматичне розсилання повідомлень про помилки в базі даних пакету PVCS Tracker за робочими станціями призначення. При цьому використовується офісна система електронної пошти cc:Mail або Microsoft Mail. PVCS Notify розширює можливості PVCS Tracker і використовується тільки спільно з ним.

**PVCS** Notify настроюється з середовища PVCS Tracker. Налаштування включає визначення інтервалу часу, через який PVCS Notify перевіряє вміст бази даних, визначення критеріїв відбору записів для розсилання повідомлень, визначення списків адрес для розсилання. Після налаштування, PVCS Notify починає роботу в автономному режимі, автоматично розсилаючи повідомлення про зміни в базі даних PVCS Tracker.

**PVCS** Notify призначений для використання у великих робочих групах, частина членів яких хоча і доступна тільки через засоби електронної пошти, проте повинна мати оперативну інформацію про вимоги на зміну програмного продукту, зауваженнях, помилках, ході і результатах його тестування.

Результатом роботи PVCS Notify є оформлені відповідно до одного із стандартів поштової повідомлення, готові для розсилання за допомогою системи електронної пошти.

### **3.8.5. ClearQuest**

ClearQuest – це засіб управління запитами на зміну (Change Request Management – CRM), спеціально розроблений з урахуванням динамічної і складної структури процесу розробки ПЗ. ClearQuest призначений для відстежування і управління будь-яким типом дій, що приводить до змін протягом усього життєвого циклу продукту, допомагаючи тим самим створювати якісне ПЗ більш передбаченим чином [11].

ClearQuest є багатомодульним додатком, який власними засобами створює базу даних проекту і заносить в неї всі зміни. Він підтримує СУБД провідних виробників, зокрема Oracle, Microsoft і Sybase. Модулі є складовою частиною CQ і поставляються разом з ним одним комплектом:

ClearQuest. Основний модуль роботи з продуктом. Містить в собі засоби щодо внесення та документування дефектів (як окремої проблеми управління змінами), побудови графіків, таблиць, запитів, звітів;

ClearQuest Web. Web- версія продукту, яка здатна забезпечити повну функціональність для тих користувачів, які працюють віддалено через Інтернет. CQWEB часто користуються при терміновому внесенні невеликих змін до бази, перебуваючи віддалено. Даний модуль можуть використовувати для документування змін (дефектів) клієнти або служба технічної підтримки;

ClearQuest Designer. Спеціальний модуль, керує схемами баз даних CQ, користувачами і зовнішнім виглядом CQ;

ClearQuest MultiSite. Розширення для підтримки управління змінами в регіонально віддалених командах. Необхідність в CQMS виникає при роботі над складними проектами, коли команди тестувальників і розробників знаходяться на значному віддаленні, а рівень інтеграції має бути вище, ніж просто Web-доступ.

Основні завдання, що вирішуються за допомогою ClearQuest [11]:  
управляти змінами, що виникають в ході процесу розробки ПЗ;  
оптимізувати шлях проходження запитів на зміни, а також пов'язані з ним форми і процедури;

через World Wide Web підтримувати зв'язок усередині команд, розділених територіально;

запроваджувати надійний і перевірений процес CRM, або змінити вже існуючий процес для задоволення специфічним вимогам;

за допомогою багатих можливостей графічного представлення інформації і звітів візуально аналізувати отриманий прогрес проекту;

інтегруватися з засобами конфігураційного управління, такими, як Rational's ClearCase, дозволяючи створювати зв'язки між запитами на зміну та розвитком коду.

Основна інформаційна одиниця продукту – дефект – це знайдена помилка в продукті або опис якоїсь його особливості. Дефект заноситься в систему користувачем, що виявив його. У кожного дефекту є певний набір станів, за якими керівництво може як відстежити історію проекту, так і оцінити поточне становище справ.

Основні можливості ClearQuest:

управління змінами, що виникають в ході процесу розробки ПЗ;

оптимізація шляху проходження запитів на зміни, а також пов'язаних з ним форм і процедур;

підтримка через World Wide Web зв'язку всередині команд, розподілених територіально;

впровадження надійного та перевіреного процесу CRM, або зміна вже існуючого процесу для задоволення специфічних вимог;

візуальний аналіз прогресу проекту за допомогою можливостей графічного представлення інформації і звітів;

інтеграція з засобами конфігураційного управління, такими, як Rational ClearCase, що дозволяє створювати зв'язки між запитами на зміну та розвитком коду;

підтримка основних СУБД від виробників Sybase, Oracle, Microsoft;

тісна інтеграція з усіма засобами тестування Rational, такими, як TeamTest, VisualTest, Purify, PureCoverage, Quantify і Robot;

побудова професійних звітів на базі Crystal Reports (що входить до складу постачання в конфігурації Professional);

інтеграція через COM з MS Word і MS Excel;

система тригерів, що настроюються, для розширення можливостей.

Як позитивні риси даного продукту можна відзначити його адаптивність (тобто продукт можна на місці доопрацьовувати з урахуванням конкретної функціональності), масштабованість, простоту в адмініструванні й легкість у використанні.

У роботі з програмою кожен учасник проекту може заходити в базу і визначати власні запити.

Станів у дефектів всього п'ять. "Поданий" (Submitted) – опис дефекту тільки що внесений; "Призначений" (Assigned) – опис переданий певному працівникові. Початок роботи над запитом переводить його у "Відкритий" стан (Open), і вся команда може бачити, що хтось обробляє запит. Нарешті, коли запит перевірений і закритий, він проходить відповідно стадії "Перевірка" (Verify) і "Закритий" (Resolved).

Стани здатні вказати тільки на статус того або іншого дефекту. А його опис заноситься в спеціалізовані поля. Будь-який дефект містить певну кількість описових полів, причому якщо використовувати ClearQuest спільно із засобами тестування, то при виявленні помилки дані продукти самі заповнюють всі необхідні поля.

Отже, дефект одержує ідентифікаційний номер і набір описових даних. Опис до дефекту включає:

State – поточний статус дефекту (закритий, відкритий, в роботі...).

ID – ідентифікаційний код дефекту. Впливати на це значення користувач не може, оскільки система привласнює номер автоматично.

Headline – коментар до дефекту. В разі ручного введення дається користувачем самостійно. При автоматичній інтеграції заповнюється автоматично програмою, що виявила дефект.

RA – асоціація з проектом (репозиторієм). Необхідна для асоціації дефекту з вимогою до системи.

Priority – пріоритет виправлення дефекту. Цей параметр можна змінювати по ходу проекту.

Severity – критичність дефекту. Тобто на першому етапі дефект можна визначити як некритичний і виправити в останню чергу або в наступній версії. Цей параметр також можна змінювати по ходу проекту.

Owner – власник дефекту. ClearQuest має два контрольні поля – Submitter і Owner. Перше поле містить ім'я користувача, який активізував помилку, а друге – ім'я користувача, який повинен цю помилку виправити.

Keyword – набір ключових слів для даного дефекту. Спеціальний механізм, що дозволяє конкретній компанії вводити в ужиток свій набір помилок і приводити їх в стан асоціативного зв'язку з помилкою в ClearQuest.

Symptoms – ознака дефекту (Cosmetic Flaw, Data Corruption, Data Loss, Slow Performance... і т. д.). Заздалегідь зумовлені типи.

Description – опис проблеми, коментар.

Resolution – спосіб вирішення проблеми (fixed/nofixed).

Attachment – сюди можна приєднати будь-який документ (скажімо, код програми).

History – історія внесення змін у дефекті.

TestData – певний набір супровідних документів. Заповнюється або вручну, або автоматично засобами тестування.

Environment – середовище супроводу. Тут можна визначити тип операційної системи, тип процесора, за яких виявляється дефект.

Requirements – тут можна задати зв'язок дефекту з вимогою з RequisitePro.

ClearCase – зв'язок дефекту з конкретною версією файлу. Даний модуль з'являється тільки при правильному налаштуванні ClearQuest і ClearCase.

## 3.9. Засоби тестування

### 3.9.1. Quality Works

Один із найбільш розвинених засобів тестування QA (нова назва – Quality Works) є інтегрованим, багатоплатформним середовищем для розробки автоматизованих тестів будь-якого рівня, включаючи тести регресії (регресійне тестування – це тестування, що проводиться після удосконалення функцій програми або внесення до неї змін) для додатків з графічним інтерфейсом користувача [10; 17].

QA дозволяє починати тестування на будь-якій фазі ЖЦ, планувати й управляти процесом тестування, відображати зміни в додатку і повторно використовувати тести більше ніж 25 різних платформ.

Основними компонентами QA є:

QA Partner – середовище для розробки, компіляції і виконання тестів;

QA Planner – модуль для розробки планів тестування і обробки результатів. Для створення і виконання тестів у процесі роботи QA Planner викликає QA Partner;

Agent – модуль, що підтримує роботу в мережі.

Процес тестування складається з наступних етапів:

створення плану тестування;

зв'язування плану з тестами;

виконання тестів;

отримання звітів про тестування та управління результатами.

Створення тестового плану в QA Planner включає складання схеми тестових вимог і виділення рівнів деталізації. Для цього необхідно визначити все, що має бути протестоване, підготувати функціональну декомпозицію додатка, оцінити, скільки тестів необхідно для кожної функції і характеристики, визначити, скільки з них буде реалізовано залежно від доступних ресурсів і часу. Ця інформація використовується для створення схеми тестових вимог.

Для зв'язування плану з тестами необхідно створити управляючі пропозиції (скрипти) на спеціальній мові 4Test і тести, які виконують вимоги плану, і зв'язати компоненти будь-яким способом. Для уникнення перевантаженості тестів використовують управління тестовими даними.



При виконанні плану результати записуються у форматі, схожому на план. Усі результати пов'язані з планом. Є можливість проглянути або приховати загальну інформацію про виконання, злити файли результатів, розмітити невдалі тести, порівняти результати попереднього виконання тестів, виконати або відмінити звіт.

Одним із атрибутів тесту є ім'я його розробника, що дозволяє за необхідності виконувати тести, створені конкретним розробником.

### **3.9.2. AllFusion Data Model Validator**

AllFusion Data Model Validator (раніше – Erwin Examiner) – це інструмент для перевірки правильності бази даних, який забезпечує несуперечність інформації і дозволяє удосконалити інформаційну архітектуру всього підприємства. Засоби діагностики та перевірки, засновані на правилах реляційного моделювання, допоможуть переконатися в структурній цілісності моделей даних AllFusion ERwin Data Modeler або коди SQL/DDI. Всі невідповідності в проекті будуть виявлені негайно, після чого програма запропонує рекомендації щодо усунення несправностей і автоматично згенерує сценарії для реалізації вибраних виправлень [18].

Функція діагностики і створення звітів дозволяє проводити аналіз структури бази даних (зокрема організовувати стосунки і застосовувати правила реляційного моделювання) для виявлення невідповідностей в проекті і проблем з продуктивністю. Детальні діагностичні звіти дозволяють виявити в дизайні бази даних будь-які суперечності або відхилення від норми. Тепер ви завжди будете упевнені в тому, що модель даних сконструйована правильно і працює так, як передбачалося.

Функція "Show me" (локалізація суперечностей в проекті) відображає певні елементи структури бази даних, які вимагають модифікації. Розробники зможуть також формувати підмножини моделей і працювати з ними. Вказана функція допоможе підвищити продуктивність персоналу за рахунок можливості швидшого перегляду. Докладні звіти, "діагнози", що надаються засобами, передбачають можливість відокремлення конкретних несправностей в складних моделях або структур бази даних, що вимагають модифікації.

Функція "Teach Me" використовує максимально доступну термінологію для ідентифікації проблеми, а також пропонує приклади для кращого розуміння проблеми і знаходження оптимальних шляхів для її розв'язання. Тепер розробники моделей зможуть навчатися і створювати бази даних ще вищої якості. Ефекти, викликані прийнятими рішеннями, відбиваються в документації, що полегшує розробку якісніших проектів надалі.

Функція інтеграції з продуктом AllFusion ERwin Data Modeler призначена для перевірки цілісності та якості моделей даних, створених за допомогою цієї програми або шляхом реконструкції існуючих баз даних. Зручний інтерфейс допомагає виявити проблеми в процесі моделювання на ранніх стадіях, що сприяє створенню високоякісних реляційних моделей в найкоротші терміни і без зайвих витрат.

Власні "стандарти якості" можуть встановлюватися для кожного окремого проекту або відділу, а також на рівні всієї організації. При визначенні стандарту вибирається діагностична процедура і встановлюється рівень серйозності помилки, пов'язаної з кожною з процедур. Ця функціональна можливість дозволяє переконатися в тому, що кожна з моделей у вказаному діапазоні (на рівні компанії, відділу, проекту і ін.) відповідає вимогам, що висуваються.

## **3.10. Засоби документування**

### **3.10.1. SODA**

Для створення документації в процесі розробки ІС використовуються різноманітні засоби формування звітів, а також компоненти видавничих систем. Зазвичай засоби документування вбудовані в конкретні CASE-засоби. Виключенням є деякі пакети, що надають додатковий сервіс при документуванні. З них найактивніше використовується SODA (Software Document Automation).

Продукт SODA призначений [9] для автоматизації розробки проектної документації на всіх фазах ЖЦ ПЗ. Він дозволяє автоматично витягувати різноманітну інформацію, що отримується на різних стадіях розробки проекту, і включати її у вихідні документи. При цьому контролюється відповідність документації проекту, взаємозв'язок документів, забезпечується їх своєчасне оновлення. Результатна

документація автоматично формується із безлічі джерел, число яких не обмежене.

SODA не залежить від інструментальних засобів, що використовуються. Зв'язок з додатками здійснюється через стандартний програмний інтерфейс API. Перехід на нові інструментальні засоби не спричиняє за собою додаткових витрат із документування проекту.

Пакет включає графічний редактор для підготовки шаблонів документів. Він дозволяє задавати необхідний стиль, фон, шрифт, визначати розташування заголовків, резервувати місця, де розміщуватиметься інформація, котра витягується з різноманітних джерел. Зміни автоматично вносяться лише до тих частин документації, на яку вони вплинули в програмі. Це скорочує час підготовки документації за рахунок відмови від регенерації всієї документації.

SODA реалізована на базі видавничої системи FrameBuilder і надає повний набір засобів щодо редагування та верстки документації, що випускається. Різні версії документації можуть бути для наочності відмічені своїми відмітними ознаками. У системі створюються таблиці вимог до проекту, за яким можна прослідити, як реалізуються ці вимоги. Різні види документації, що супроводжують різні етапи ЖЦ, пов'язані між собою, і можна прослідкувати стан проекту від первинних вимог до аналізу, проектування, кодування і тестування програмного продукту.

По суті, SODA є макросом для MS Word. Система викликів і меню інтегрована в Word і дозволяє генерувати шаблони на базі наявних файлів. SODA допускає використання як стандартних шаблонів, так і створених користувачем за допомогою спеціального Wizard, також вбудованого в систему меню Word.

Набір звітів, що підтримуються SODA, такий: Rational Rose, Rose RealTime, Requisite PRO, ClearCase, TeamTest.

### **3.10.2. Crystal Reports**

Crystal Reports – найбільш популярний генераторів звітів, що дозволяє створювати звіти на основі даних практично з будь-якого джерела, включаючи реляційні бази даних, ERP і CRM-системи, OLAP, XML/.NET-, Java- і COM-джерела даних. Crystal Reports має інтуїтивно зрозумілий інтерфейс для швидкого створення гнучких, інформативних і зручних звітів будь-якого рівня складності і високої якості. Створені звіти

можуть бути вбудовані в Web-додатки, портали і в клієнтські додатки, опубліковані як Web-сервіси, а також можуть розповсюджуватися в організації через систему корпоративної звітності, побудовану на базі Crystal Reports Server. Генератор звітів Crystal Reports інтегрований з провідними системами розробки додатків для платформ .NET, Java і COM (IBM WebSphere Studio Application Developer, IBM Rational Application Developer, Borland JBuilder, BEA WebLogic Workshop; Microsoft Visual Studio .NET, Borland Delphi 2005).

Генератор звітів Crystal Reports XI доступний у чотирьох редакціях – Crystal Reports Server, Crystal Reports Developer, Crystal Reports Professional і Crystal Reports Standart [21]:

Crystal Reports Server – система корпоративної звітності, яка забезпечує повний набір можливостей зі створення, публікації, доставки звітів і управління звітами. Crystal Reports Server покращує підтримку прийняття рішень і бізнес-ефективність, надаючи користувачам швидкий і безпечний доступ до даних. У комплект постачання Crystal Reports Server входить Crystal Reports Developer;

Crystal Reports Developer призначений для розробки звітів з використанням широкого спектру персональних і корпоративних джерел даних, а також ADO .NET, JavaBeans і COM-джерел. Ця редакція містить необхідні інструменти (Software Development Kit, SDK) для інтеграції звітів у власні розробки: клієнтські і Web-додатки, Web-сервіси і корпоративні портали;

Crystal Reports Professional призначений для розробки звітів будь-якої складності з використанням будь-якого типу доступу до даних, від локальних джерел даних до корпоративних серверних баз даних;

Crystal Reports Standard призначений тільки для розробки звітів із використанням файлових (персональних) баз даних, таких, як Access, Paradox, dBase і ін., без можливості роботи з реляційними СУБД (Oracle, MS SQLServer, SQLBase, Interbase і ін.).

Можливості генератора звітів Crystal Reports XI [21]:

використання динамічних списків вибору значень;

засіб перевірки залежностей (Dependency checker)

використовується для швидкого знаходження неробочих посилань, помилок у формулах і інших потенційних проблем із залежностями перед тим, як звіти будуть опубліковані та поширені;

зображення та малюнки можуть бути вставлені у звіти з графічних файлів за допомогою посилань, що зберігаються в базі даних. Можна також запрограмувати шлях і ім'я графічних файлів. Графічні дані безпосередньо в базі даних не зберігаються;

розробники можуть створювати ієрархічні звіти, а не лише документи послідовного представлення даних, що властиве звичайним реляційним звітам. Ієрархічна форма звіту найбільше підходить для створення зведених звітів і звітів, що містять організаційні схеми;

візуальний дизайнер звітів забезпечує швидку розробку інтерактивних звітів з використанням інтуїтивного зрозумілого інтерфейсу, функціональності drag-and-drop і об'єктно-орієнтованих провідників;

численні експерти і майстри середовища дизайну звітів допомагають виконати і значно спростити такі стандартні завдання, як підключення до бази даних, створення вибірки, групування, сортування, обчислення значень і ін.;

генератор звітів Crystal Reports забезпечує створення практично будь-якого необхідного типу звіту, включаючи звіти з крос-таблицями, звіти, що параметризуються, звіти з різними угрупованнями даних і спеціальними сортуваннями груп, аналітичні звіти з деталізацією і підсумками, звіти в стилі форм, багатоклонкові звіти (наприклад, для друку товарних наклейок), OLAP-звіти, звіти з підзвітними сумами і ін.;

користувачеві надаються широкі можливості для графічного представлення даних у звітах завдяки використанню різних типів графіків і діаграм, таких, як, гістограми, у тому числі і 3D-гістограми, кругові, секторні і кільцеві діаграми, лінійні діаграми, діаграми Ганта, діаграми у вигляді циферблатів, воронок, координатні, бульбашкові, точкові діаграми і багато інших;

додавання діаграм і крос-таблиць простим перетяганням забезпечує інтелектуальні графічні можливості для автоматичного визначення типу і стилю діаграми/графіка на підставі аналізу структури даних звіту. Діаграми/графіки автоматично оновлюються при додаванні нових змінних;

використання центрального сховища (репозиторію), в якому зберігаються ключові об'єкти звітів, такі, як текстові об'єкти, запити SQL, графічні зображення, призначені для користувача функції (формули) і бізнес-уявлення. Всі ці об'єкти доступні для багаторазового і одночасного

використання в декількох звітах, а управління цими об'єктами відбувається простим і одноманітним способом;

використання шаблонів звітів, що настроюються, дозволяє скоротити час на форматування індивідуальних звітів. Розробка і застосування шаблонів, що настроюються, містять стандартне форматування та логіку, включаючи шаблони для операцій доступу до даних, гарантує одноманітність дизайну звітів. Як шаблони для нових звітів можна використовувати вже існуючі звіти;

використання розширеної мови для написання формул, що має більше 160 вбудованих функцій і що дозволяє створювати призначені для користувача процедури та функції. Ці можливості можна використовувати для реалізації ефективного управління форматуванням звітів, виконання складної логіки і для роботи з даними. Використання стека викликів спрощує відладку і виявлення помилок. В наявності єдине середовище роботи з формулами з різноманітними майстрами та експертами;

використання власних функцій, що зберігаються в репозиторії, усуває дублююче та надмірне створення формул, що реалізують бізнес-логіку, для кожного нового звіту;

можливість налаштування зображення OLAP-даних в таблиці. За допомогою функціональності асиметричних звітів можна приховати деякі вимірювання в OLAP-таблиці, що дозволяє користувачеві переглядати лише необхідні і найбільш важливі дані;

використання майстрів для генерації звітів Crystal Reports безпосередньо в Microsoft Excel і Access;

можливість здійснювати операції над групами об'єктів, такі, як перегляд, створення і публікація;

можливість збереження конфігурації експорту безпосередньо у звіті, що усуває необхідність постійного настроювання параметрів експорту;

можливість зручного редагування експортованого звіту при точному збереженні його форматування, а також використання звітів в системах обробки форм; можливість редагування і доповнення звітів у стандартних додатках для роботи з документами (формат rtf);

параметризований контроль для звітів виду "N value for top N" дає можливість використовувати один звіт, який задовольнить вимоги різних

користувачів. Така функціональність зменшить загальну кількість звітів, які мають бути розроблені і які згодом доведеться підтримувати;

можливість експортувати звіти у звичному і зручному для них форматі (Excel, PDF, XML, HTML, RTF і ін.);

можливість вибору варіантів проглядання звітів з використанням різних серверних і клієнтських компонентів, включаючи такі редактори: DHTML-сторінок (WebForms), .NET WinForms, Java, ACTIVEEX і Report Part для мобільних пристроїв. Ці переглядачі реалізують представлення звітів у різних форматах без додаткового кодування. У переглядачів включені різні опції, що забезпечують інтерактивну взаємодію зі звітами і дозволяють налаштовувати проглядання звіту відповідно до вимог користувачів. Серед таких опцій – можливість перегортання сторінок вперед і назад, інструменти наскрізного перегляду даних (Drill Down/Up), експорт і друк;

використання параметрів дає можливість проглядати одні й ті ж дані різними способами без необхідності створення додаткових звітів. Вибираючи зумовлені параметри, можна отримати різне представлення даних для одного звіту;

отримання повідомлень про настання певних умов у звітах. Ця функціональність ефективно використовується при публікації звітів у Crystal Reports Server/BusinessObjects Enterprise, щоб попередження могли пересилатися кінцевому користувачеві з прямими посиланнями на оригінал звіту;

можливість налаштування керованої навігації між об'єктами в межах одного звіту і для доступу до об'єктів інших звітів;

використання різноманітних типів програмованих гіперпосилань перетворює звіти на інтерактивні Web-документи, які забезпечують доступ до пов'язаної із звітом інформації, включаючи Web-сайти і інші звіти;

функціональність деталізації у звітах (Drill Down) дозволяє без додаткового кодування створювати звіти для перегляду оперативних (реляційних) або OLAP-даних з подальшою деталізацією для виявлення інформації, яка інакше могла б залишитися непоміченою;

забезпечення механізму для представлення звітної інформації на мобільних пристроях і в корпоративних порталах; надання доступу до графіків, таблиць, специфічних записів і іншої ключової інформації звіту

через портали, мобільні телефони з підтримкою WML, комунікатори RIM Blackberry, портативні пристрої Compaq iPAQ і ін;

забезпечення доступу до будь-якого джерела даних за допомогою власних драйверів прямого доступу (Native API), стандартних механізмів ODBC, OLE DB і JDBC до реляційних баз даних, до OLAP і XML, успадкованим і корпоративним джерелам даних, включаючи такі СУБД як Oracle, IBM DB2, Sybase, Microsoft SQL Server і Informix;

забезпечення доступу до даних додатків, визначених користувачем (в оперативній пам'яті) за допомогою використання JavaBeans, ADO.NET, і COM-провайдерів даних;

технологія Crystal Reports забезпечує можливість доступу до одного джерела даних або комбіноване одночасне використання декількох різнорідних джерел в одному звіті;

забезпечення повної інтеграції з додатками для партнерів (Business-to-Business, B2B) і/або замовників (Business-to-Consumers, B2C) за допомогою доступу до даних у форматі XML або експорту даних в цьому форматі;

реалізація всіх звітів Crystal Reports в Unicode, що гарантує коректність відображення текстових даних на будь-якій мові, а також відображення в одному звіті текстових даних на різних мовах, що використовують різноманітні кодові таблиці;

використання набору різних SDK (Software Development Kit, набір інструментальних засобів розробки) як інтерфейс для взаємодії генератора звітів Crystal Reports з додатками J2EE .NET і COM дозволяє забезпечити гнучку роботу користувача зі звітами;

можливість вбудовування звітів у додатки за допомогою могутніх компонентів Java .NET, і COM для ефективною роботи зі звітами і їх перегляду;

бібліотека тегів, що настроюються, дозволяє скоротити кодування, необхідне для того, щоб вбудувати шаблони звітів в JSP-сторінки. Завдяки використанню тегів JSP, розробники можуть легко додавати можливості проглядання звітів у свої Web-додатки, істотно зменшуючи кількість необхідного коду;

спеціальні редакції Crystal Reports для Java і .NET дозволяють інтегрувати інструменти для створення звітів у додатки, які розробляються в провідних системах розробки на платформах Java і .NET, що дає можливість вести розробку звітів з використанням звичного



і знайомого середовища. В даний час реалізована інтеграція Crystal Reports з наступними продуктами: Microsoft Visual Studio .NET, Borland Delphi 2005, IBM WebSphere Studio Application Developer, IBM Rational Application Developer, Borland JBuilder і BEA WebLogic Workshop та ін.;

можливість інтеграції звітів з додатками, що розробляються на Borland Delphi 7 і Borland C++ Builder 6.

## **Розділ 4**

### **Практичне використання CASE-засобів при проектуванні окремих підсистем інформаційної системи**

#### **4.1. Використання VPwin для опису предметної області "Страховання нерухомості та майна фізичних осіб"**

**Постановка завдання та характеристика предметної області процесу страхування нерухомості та майна фізичних осіб.**

Для автоматизації процедур бізнесу, пов'язаних з укладенням, виконанням, обліком і аналізом договірних зобов'язань страхування з фізичними особами, в АІС компанії призначений модуль "Страховання нерухомості та майна фізичних осіб".

Типовий бізнес-процес страхування нерухомості та майна фізичних осіб включає наступні роботи:

1. Від фізичних осіб надходять замовлення (заявки) на страхування нерухомості та майна.
2. На тривалий період на основі заявок, виписки з техпаспорту БТІ (бюро технічної інвентаризації) та виписки з техпаспорту на будівлю складається договір, в якому містяться юридичні та розрахункові реквізити сторін; загальні умови (права, обов'язки сторін); умови надання послуг страхування і порядок оплати; визначення виду страхування, групи та терміну дії полісу; № договору, термін дії, географічні межі.
3. На підставі договору формується квитанція на оплату страхової суми.
4. Проводиться видача копії договору фізичній особі.
5. На підставі договору та квитанції з банку про оплату страхової

суми формується поліс фізичній особі.

6. У разі настання страхового випадку фізичною особою або її представником надається Акт про страховий випадок.

7. Проводиться огляд місця настання страхового випадку.

8. На підставі рішення щодо достовірності Акту про страховий випадок розраховується сума страхової компенсації.

9. На підставі розрахованої суми страхової компенсації формується квитанція на виплату страхової компенсації.

10. Проводиться виплата страхової компенсації.

11. Проводиться облік і контроль виконання договірних зобов'язань і оплати за надані страхові послуги.

Необхідна функціональність модуля (котрий повинен бути автоматизований):

автоматизоване формування і друк заявок, заявок на страхування, договорів страхування, полісів (ключовий реквізит – № договору), запрошення на консультацію;

автоматизоване ведення реєстрів договорів і відповідних їм полісів з ідентифікацією їх статусу (стану): підписаний, такий, що виконується, виконаний, закритий;

автоматизоване планування страхової діяльності: формування прогнозу розвитку страхових операцій на період, визначення умов страхування;

автоматизований облік і контроль: страхових випадків, виконання договірних зобов'язань;

автоматизований багатовимірний аналіз виконання договірних зобов'язань: за номерами договорів, за регіонами, за клієнтами, за період, за видами страхування;

автоматизований облік оплати послуг страхування та виплати страхових компенсацій (повнота оплати в строк).

Інформаційна база модуля "Страхування нерухомості та майна фізичних осіб" включає:

1) довідник клієнтів, в якому накопичена інформація про реальних і потенційних клієнтів (покупцях) підприємства у процесі їх контактів, пропозицій, укладення договорів і інших взаємовідносин із спеціалістами служб маркетингу і збуту. Довідник містить наступні поля: код клієнта, повна назва компанії, ідентифікаційний код, сфера діяльності (галузь) клієнта, юридична адреса, адреса електронної пошти, телефон

контактної особи, ПІБ. контактної особи, посада контактної особи, номер поточного рахунку в банку, транспортні реквізити, що характеризують відвантаження продукції (станції відправлення і призначення), код регіону (країни), в якому розташований клієнт;

2) довідник страховиків містить дані про страхового інспектора, що уклав договір від компанії із даним клієнтом. Довідник включає поля: код страховика, ПІБ страховика, посада страховика;

3) довідник географічних меж містить інформацію про регіони відповідно до адміністративно-територіального поділу України та назви інших країн, на які розповсюджується дія договору. Довідник містить поля: код географічний, назва країни (регіону);

4) довідник категорій нерухомості містить поля: код категорії, найменування категорії, примітка;

5) довідник страхових випадків містить поля: код випадку, найменування випадку;

6) класифікатор видів страхування, в якому накопичена інформація про послуги страхування, що надаються. Класифікатор містить наступні поля: код страхування, вид страхування;

7) класифікатор страхових груп містить поля: номер групи, назва групи;

8) класифікатор страхових тарифів містить поля: код тарифу, тарифний коефіцієнт;

9) довідник нерухомості формується на підставі інформації документу "Виписка з техпаспорту БТІ" і містить наступні поля: кадастровий номер, найменування нерухомості, оцінювальна вартість, адреса нерухомості, кількість кімнат, загальна площа, житлова площа, матеріал стін, рік будівлі, код категорії;

10) довідник банків, що містить поля: МФО банку, найменування банку;

11) оперативний масив договорів містить поля: код договору, дата початку дії, дата закінчення дії, дата укладання, причина відмови, дата відмови, строк дії, особливі умови, страхова сума, строк оплати;

12) оперативний масив страхових внесків містить поля: код договору, дата початку дії, дата закінчення дії, дата укладання, причина відмови, дата відмови, строк дії, особливі умови, страхова сума, строк оплати;

13) оперативний масив карти збитків формується на основі документа "Акт про страховий випадок" та містить поля: код карти, доля збитку, заявлена сума, дата випадку, адреса випадку, примітка.

До складу модуля "Страхування нерухомості та майна фізичних осіб" включені наступні задачі:

0401 – "Облік договорів страхування";

0402 – "Облік сплати страхових платежів за страховими полісами";

0403 – "Облік страхових виплат";

0201 – "Формування заявки на страхування";

0202 – "Формування прогнозу розвитку страхових операцій на період";

0203 – "Формування квитанції на оплату страхової суми";

0204 – "Формування звіту обліку договорів";

0205 – "Формування полісу";

0206 – "Формування квитанції на виплату страхової компенсації";

0207 – "Формування звіту про страхові виплати";

0604 – "Аналіз акту про нещасний випадок";

0301 – "Облік та контроль страхових випадків";

0601 – "Ведення матеріалів сайту";

0602 – "Складання запрошення на консультацію";

0404 – "Облік документів для укладання договору";

0603 – "Визначення умов страхування".

Вимоги до задач приведені у додатку А.

Для початку роботи з ВРwin необхідно зайти в пункт меню **Пуск** та зі списку програм вибрати **Computer Associates ВРwin 4.0 / ВРwin 4.0** (рис. 4.1).

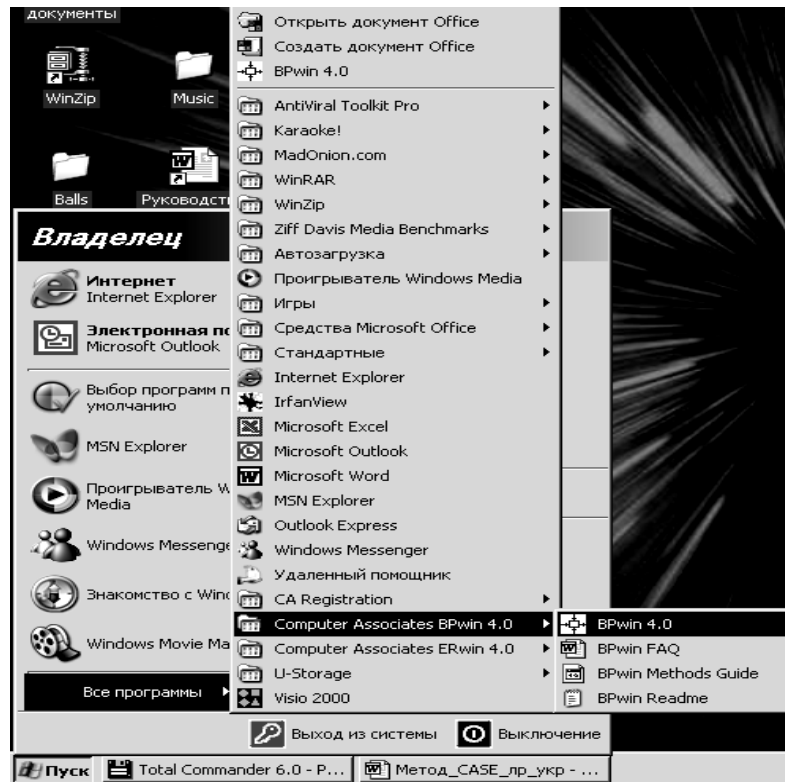


Рис. 4.1. Завантаження BPwin 4.0

Далі, відповідно до попередніх налаштувань, Вам буде потрібно або вибрати пункт меню **File / New**, або вікно створення/відкриття моделі буде автоматично відкрито при завантаженні BPwin (рис. 4.2).

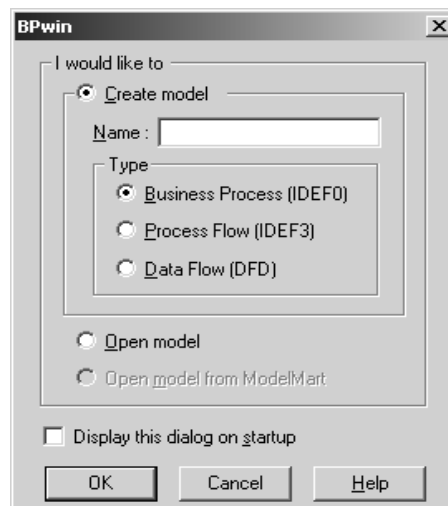


Рис. 4.2. Вікно створення / відкриття моделі

У цьому вікні можна вибрати наступні дії:

Create model – створення нової моделі; при цьому необхідно обов'язково вказати ім'я моделі (Name) та вибрати один з можливих стандартів – IDEF0, IDEF3, DFD;

Open model – відкрити існуючу модель;

Display this dialog on startup – починати роботу BPwin з цього вікна.

Нам необхідно створити нову модель з назвою „Страховання нерухомості та майна фізичних осіб" в стандарті IDEF0.

Після натискання кнопки **OK** вам буде запропоновано внести властивості для створеної моделі, а саме: **Author** – внести прізвище та ініціали автора моделі (розробника) – внесіть у це поле свої дані. Після цього натисніть на кнопку **OK**.

При цьому буде автоматично створена контекстна діаграма з єдиною роботою, що зображає систему в цілому. Контекстна діаграма представлена процесом (роботою) верхнього рівня. Робота зображається у вигляді прямокутника (рис. 4.3).

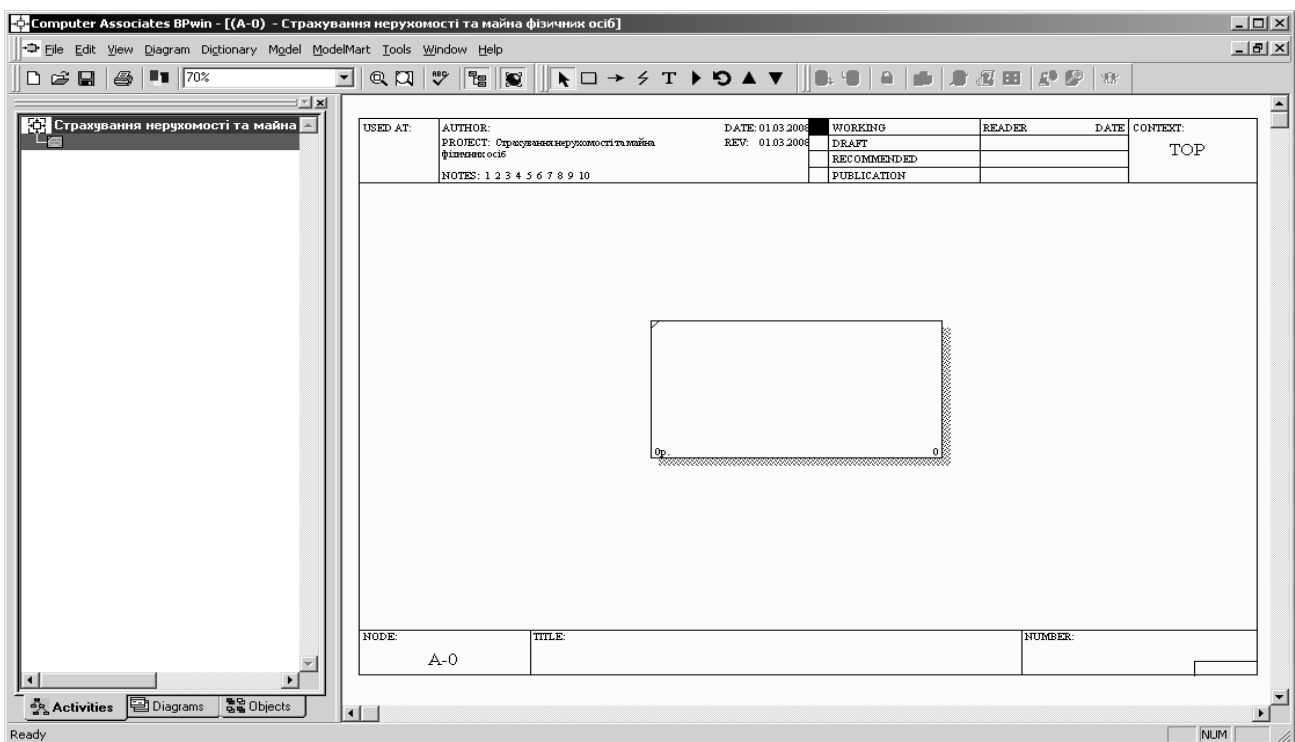


Рис. 4.3. Автоматично створена контекстна діаграма

**Примітка:** При побудові моделей дуже частим явищем є незрозуміле зображення даних, які введені кирилицею (російською, українською та іншими мовами). Для їх адекватного сприйняття необхідно зробити наступне: зайти в пункт меню **Model / Default Fonts** та зробити наступні настройки як мінімум в пунктах **Frame User Text** та **Frame System Text**: змінити шрифт у полі **Font** на **Times New Roman**.

Після цього треба зайти в пункт меню **View / Redraw Diagram** або натиснути F12.

Побудова контекстної діаграми включає: опис роботи (процесу верхнього рівня), проектування і опис інтерфейсних дуг (стрілок) і формування (опис) каркасу моделі (рамки).

При створенні та опису робіт слід зазначити, що вони повинні бути названі та визначені. Ім'я роботи повинно бути виражено віддієслівним іменником, що позначає дію (наприклад, "Планування постачань", "Моніторинг стану договорів" і т. д.). В нашому випадку процес буде мати ту ж саму назву, що й модель, тобто „Страховання нерухомості та майна фізичних осіб” (рис. 4.4). Для внесення цієї назви необхідно двічі клікнути по лівій кнопці миші або один раз клікнути по правій кнопці та в контекстному меню, що з'явилося, вибрати пункт **Name**. При цьому відкривається діалог **Activity Properties**. Замість ім'я „Untitled Object 0” необхідно внести нове ім'я, наприклад, „Страховання нерухомості та майна фізичних осіб”.

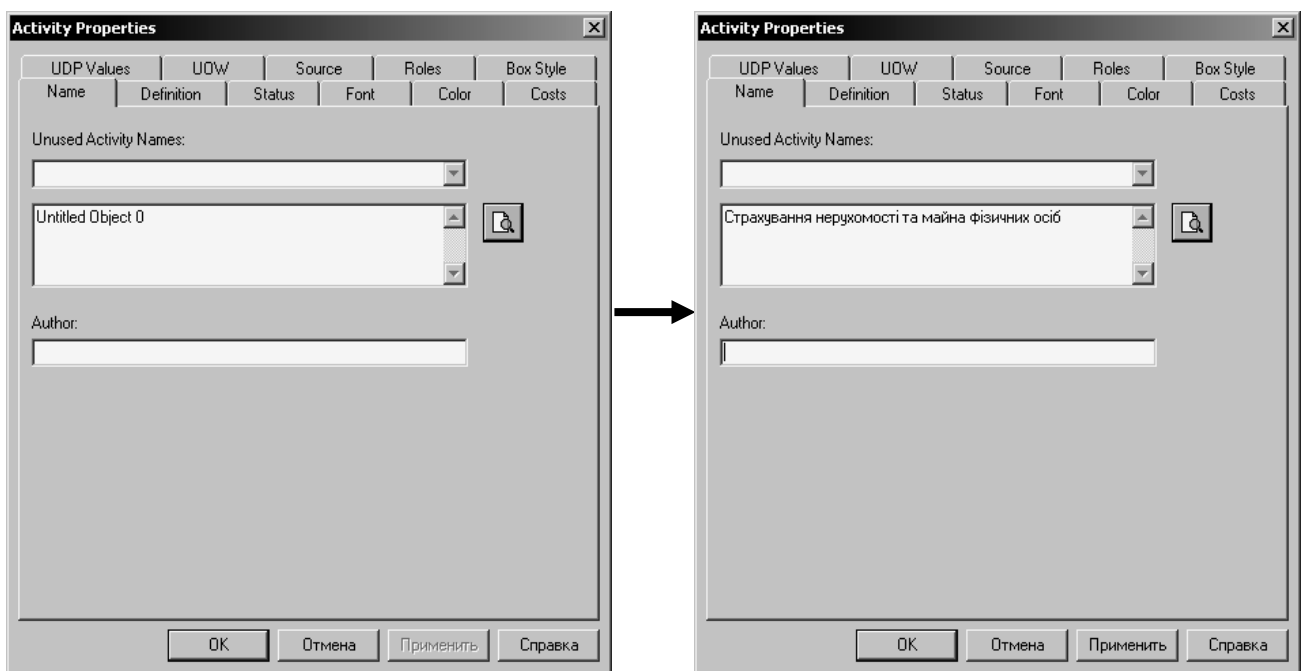


Рис. 4.4. Внесення назви бізнес-процесу

Основним роботам бізнес-процесу потрібно дати визначення – коротку характеристику. Так робота "Страховання нерухомості та майна фізичних осіб" може мати наступне визначення: "Робота відноситься до повного циклу проходження договору страхування від отримання заявки

на страхування та підписання договору для контролю його виконання та виплати страхової компенсації". Для того, щоб її внести, необхідно перейти на закладку **Definition** діалогу **Activity Properties** та у відповідне поле внести вищезазначене визначення (рис. 4.5).

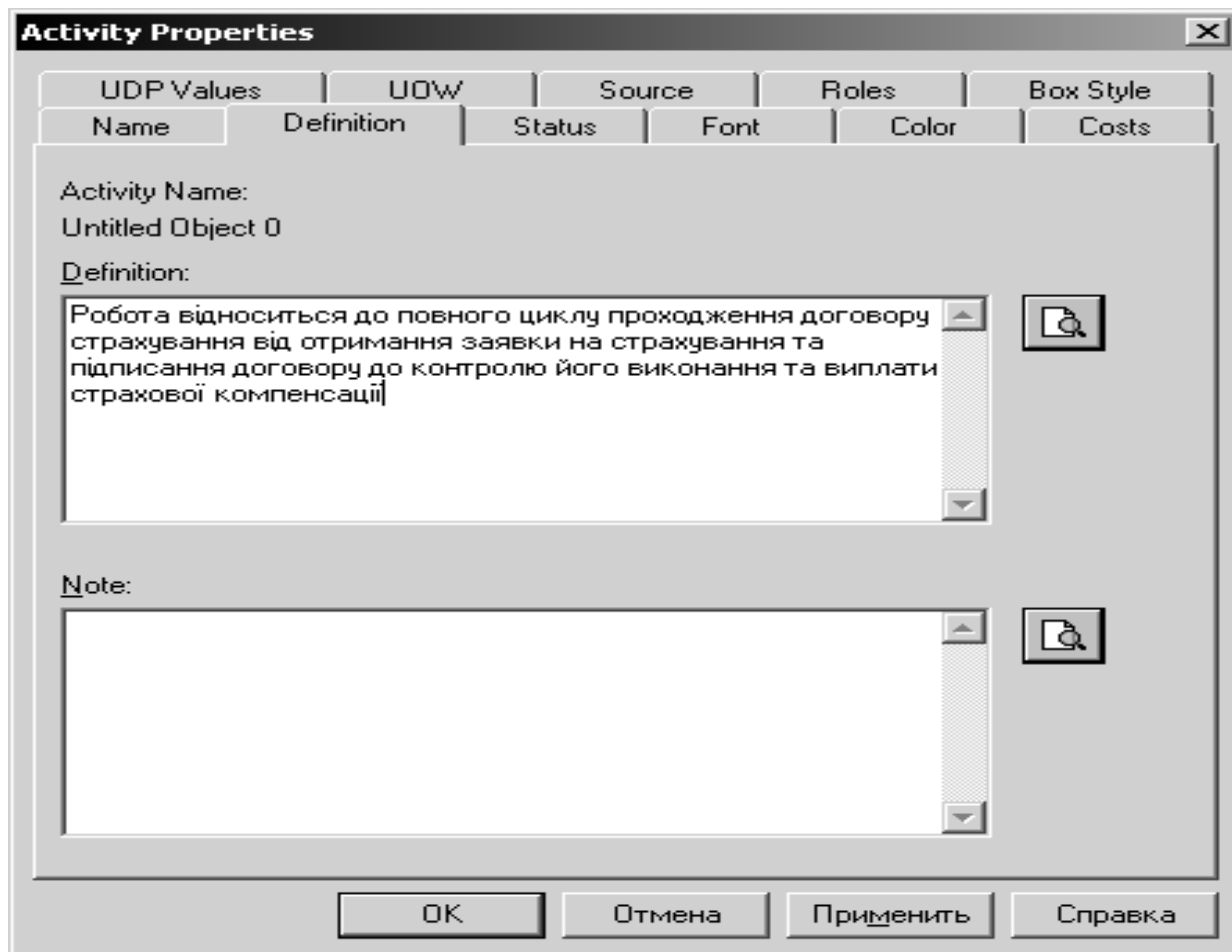


Рис. 4.5. Внесення визначення роботи

На закладці **Status** діалогу **Activity Properties** можна описати статус моделі: Working – робочий варіант, Draft – черновий, Recommended – для редагування експертами, Publication – кінцевий варіант.

На вкладці **Font** цього ж діалогу треба обов'язково змінити параметри Font – на Times New Roman Cyr або Arial Cyr, можна змінити стиль тексту (жирний, курсив), кегль шрифту. Після того, як був змінений шрифт, необхідно зробити наступні дії: відмітити пункти цієї закладки **All activities in this diagram**, **All activities in this model**, **Change all occurrences of font in model**. Це необхідно проробити, щоб для



наступних робіт, інтерфейсних дуг та інших елементів моделі текст, котрий виводиться кирилицею зображувався зрозуміло користувачеві (рис. 4.6).

На вкладці **Color** можна змінити колір об'єктів для їх більш наочного зображення.

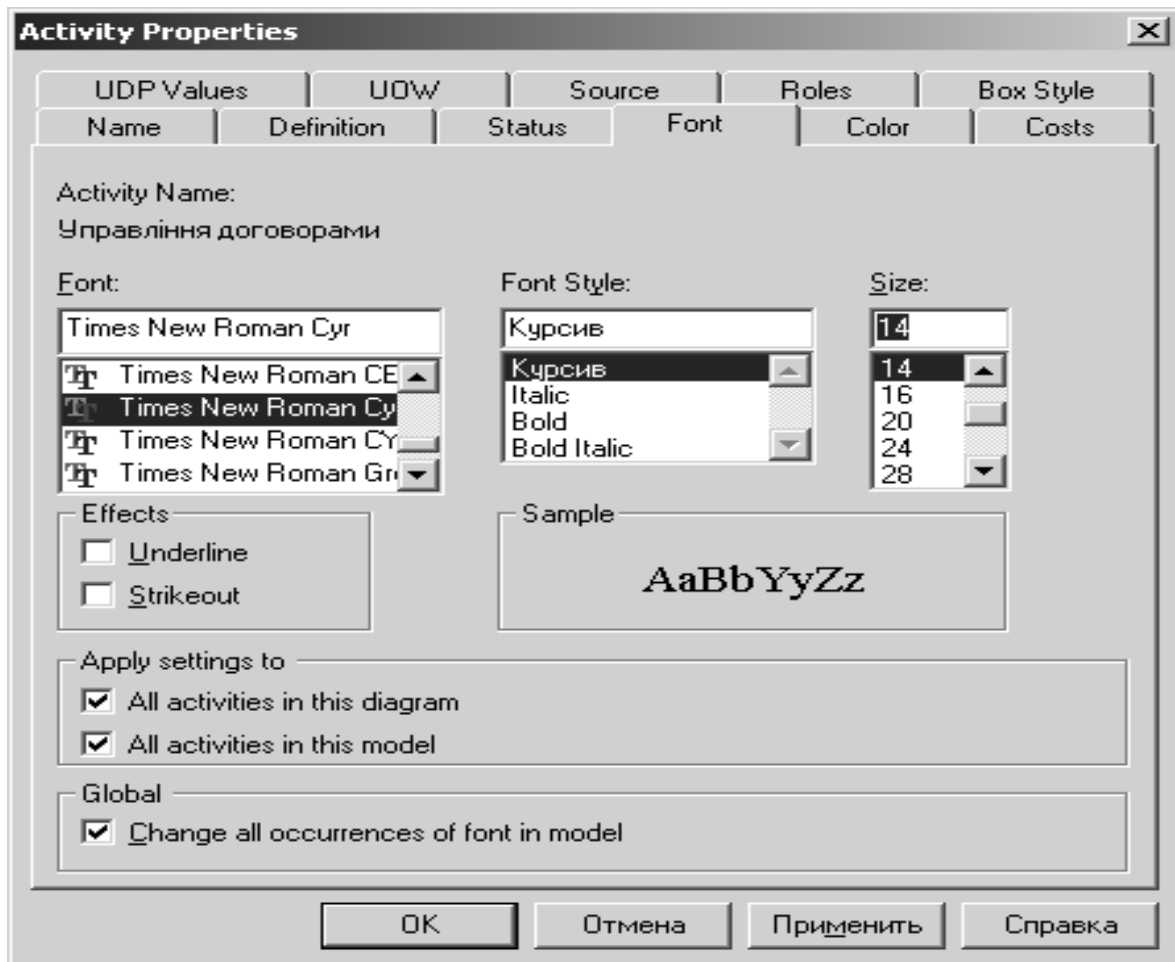


Рис. 4.6. Зміна параметрів шрифту для робіт

Для кожної моделі, як зазначалося вище, необхідно внести мету та точку зору. Для того, щоб їх внести, необхідно на робочій області моделі (будь-яка частина моделі, на якій немає об'єктів) натиснути на праву кнопку миші та з контекстного меню обрати пункт **Model Properties**, або обрати пункт меню **Model / Model Properties**. На вкладці **Purpose** необхідно внести мету (**Purpose**) та точку зору (**Viewpoint**). Після цього треба натиснути на кнопку **OK** (рис. 4.7).

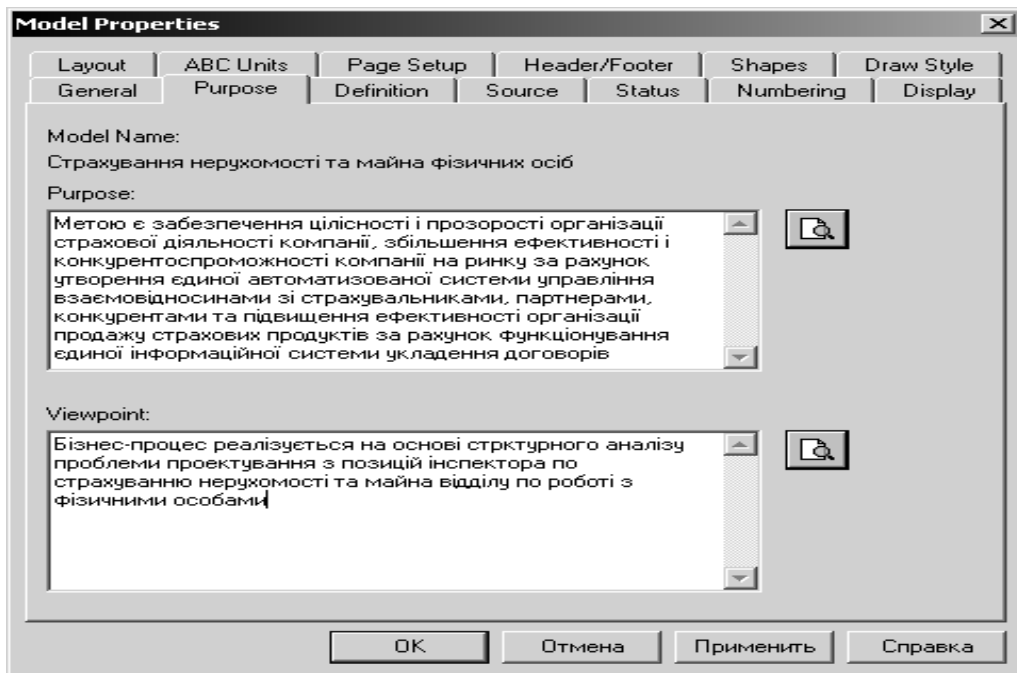


Рис. 4.7. Внесення мети та точки зору на модель що проектується

Для створення і опису об'єктів на діаграмах використовуються інструменти BPwin Toolbox, які відмінні для різних моделей і стандартів (рис. 4.8).











Рис. 4.8. Інструментарій BPwin Toolbox

У табл. 4.1 наведений опис призначення інструментів моделі IDEF0.

Таблица 4.1

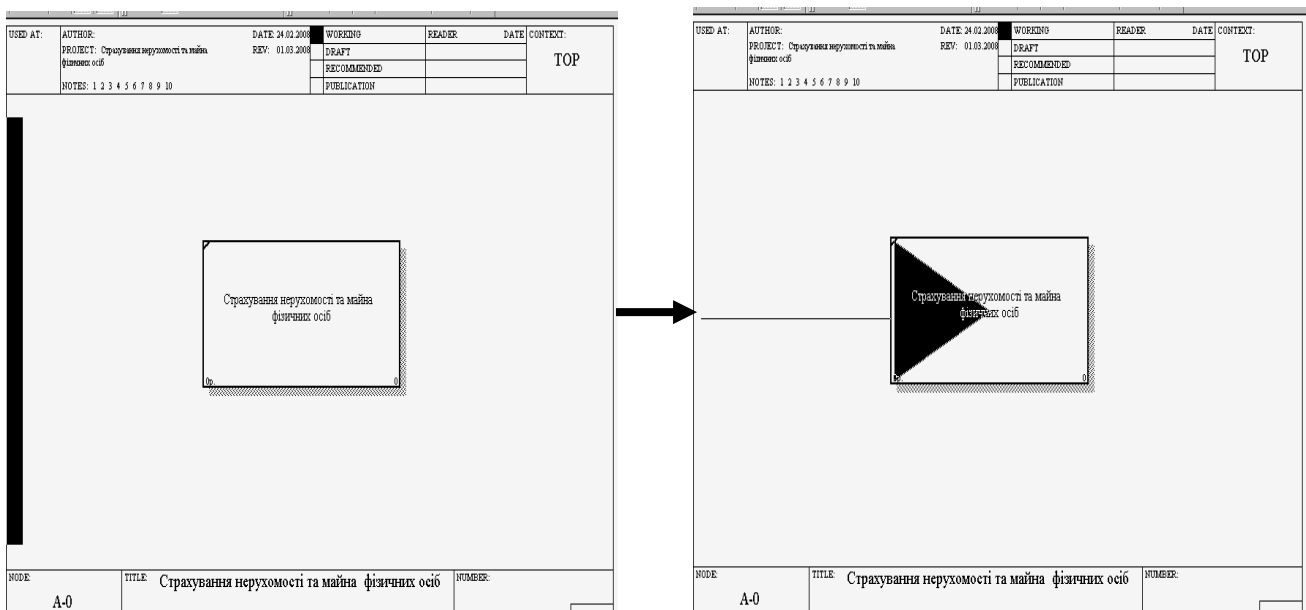
### Опис призначення інструментів моделі IDEF0

Інструмент	Найменування	Призначення
	Pointer Tool	Використовується для зміни положення вже існуючих об'єктів, зміни розмірів, або внесення даних у них
	Activity Box Tool	Блок відображає дію (процес, роботу) в діаграмі

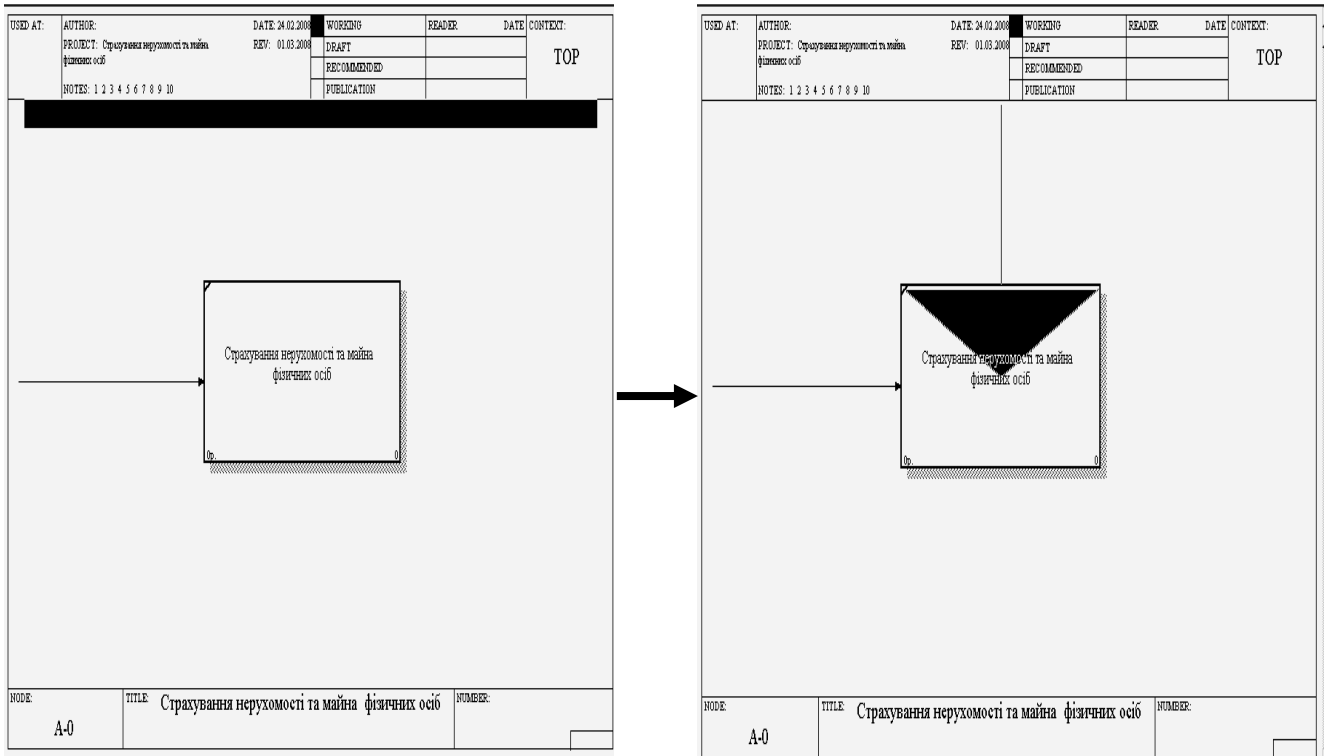
Інструмент	Найменування	Призначення
	Precedence Arrow Tool	Інструмент використовується для зображення стрілки
	Squiggle Tool	Інструмент використовується для створення „блискавки“, котра зв'язує стрілку з її ім'ям
	Text Tool	Інструмент використовується для створення текстових коментарів на діаграмах
	Diagram Dictionary Editor	Інструмент використовується для відкриття діалогового вікна „Редактор словника діаграм“
	Go to Parent Diagram	Використовується для переходу на батьківську діаграму
	Go to Child Diagram	Використовується для переходу на діаграму нижнього рівня або для декомпозиції блока процесу на діаграмі

Після того, як блок бізнес-процесу описаний, можна перейти до малювання інтерфейсних дуг. Для цього обираємо інструмент **Precedence Arrow Tool**. Для того щоб намалювати інтерфейсну дугу „вхід“ підводимо інструмент до лівої межі діаграми і коли з'явиться чорна бордюрна лінія, натискаємо один раз на ліву кнопку миші. Підводимо інструмент до лівої грані блока процесу до моменту, коли з'явиться чорний трикутник (рис. 4.9) та знову один раз натискаємо на ліву кнопку миші.

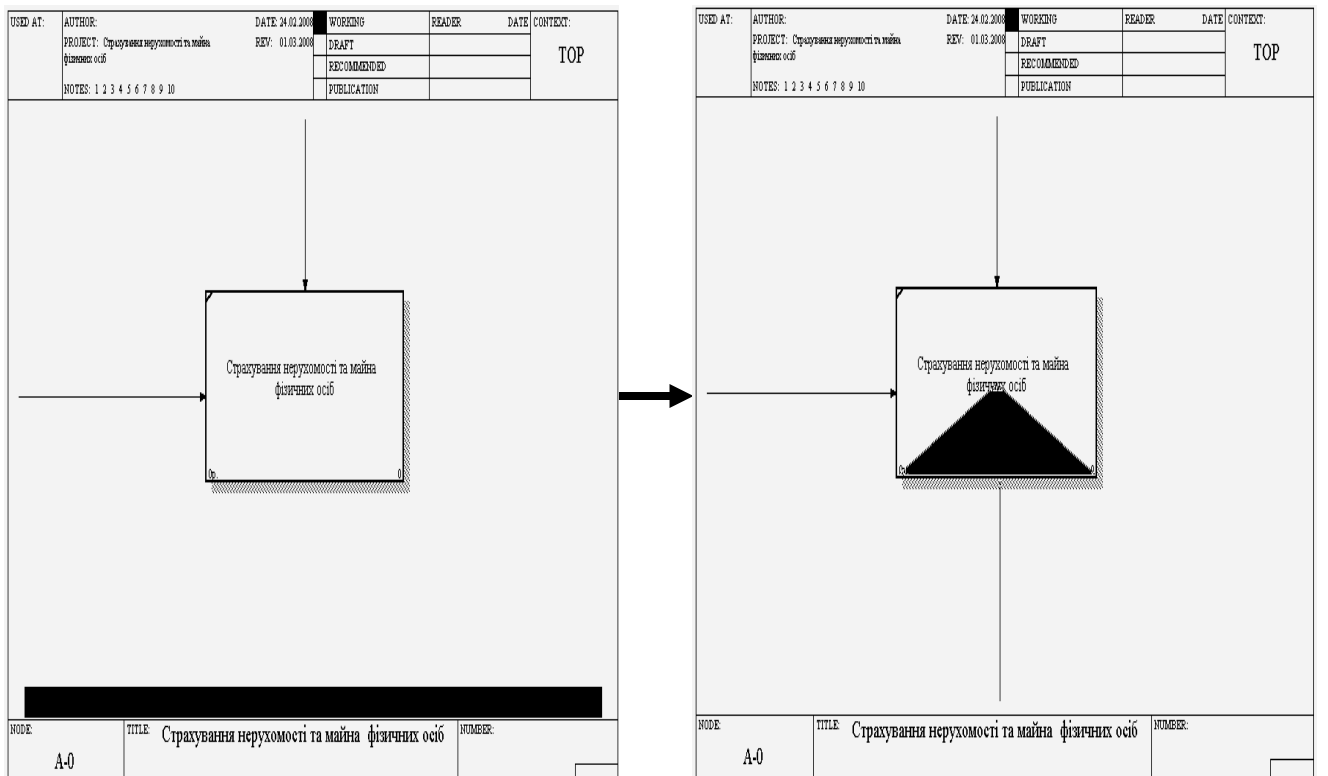
Аналогічним чином малюються інтерфейсні дуги управління (рис. 4.10) та механізму (рис. 4.11).



**Рис. 4.9. Внесення інтерфейсної дуги „вхід”**



**Рис. 4.10. Внесення інтерфейсної дуги „управління”**



**Рис. 4.11. Внесення інтерфейсної дуги „механізм”**

Інтерфейсна дуга „вихід" малюється навпаки. Підводимо інструмент до правої грані блока процесу до моменту, коли з'явиться чорний трикутник (рис. 4.12) та один раз натискаємо на ліву кнопку миші, після цього підводимо його до правої межі діаграми і коли з'явиться чорна бордюрна лінія, натискаємо ще один раз на ліву кнопку миші.

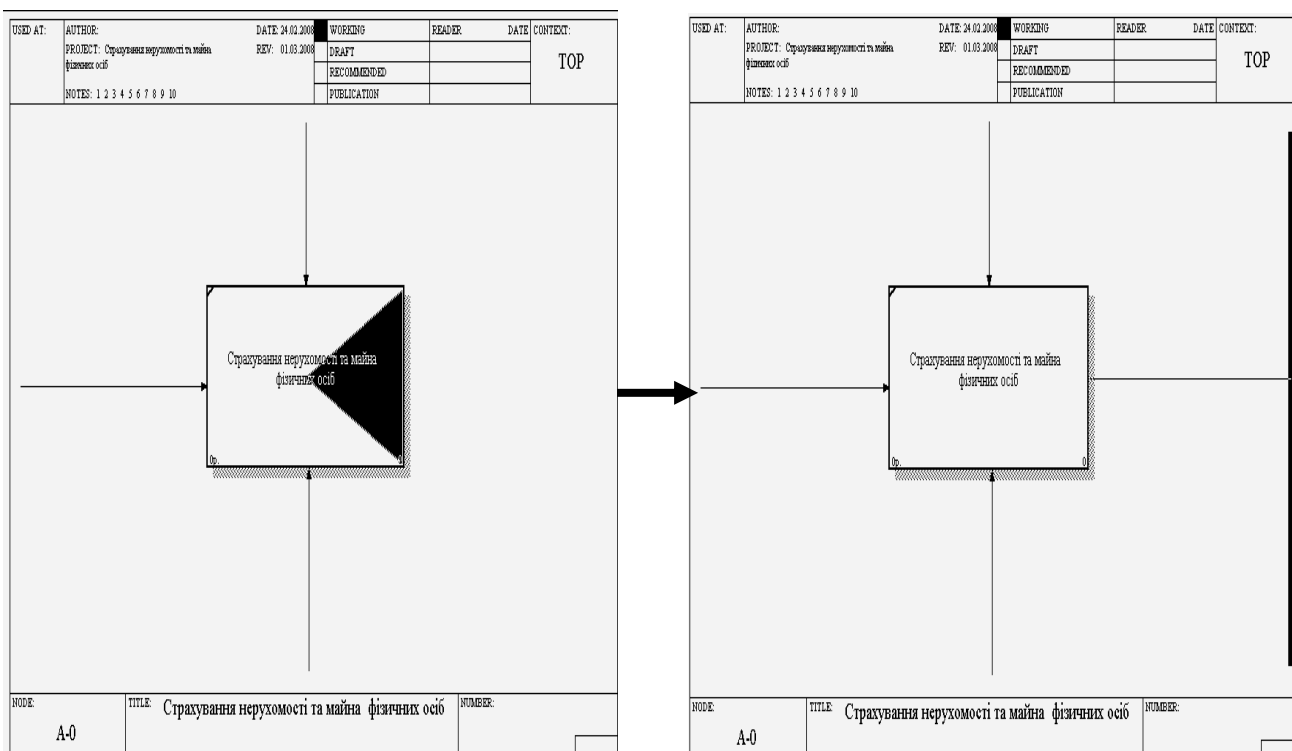


Рис. 4.12. Внесення інтерфейсної дуги „вихід"

Після внесення інтерфейсної дуги, необхідно дати їй назву. Наприклад, нам потрібно шість інтерфейсних дуг „вхід" з назвами: "Акт про страховий випадок", "Квитанція з банку", "Довідка про доходи фізичної особи", "Виписка з техпаспорту на будівлю", "Виписка з техпаспорту БТІ" та "Прогноз розвитку страхових операцій". Друга інтерфейсна дуга „вхід" малюється аналогічно першій. Для внесення назви необхідно два рази кликнути лівою кнопкою миші на інтерфейсній дузі, або один раз кликнути правою кнопкою миші та з контекстного меню обрати пункт **Name**. На закладці Name у полі **Arrow Name** треба внести назву інтерфейсної дуги та натиснути кнопку **OK** (рис. 4.13).

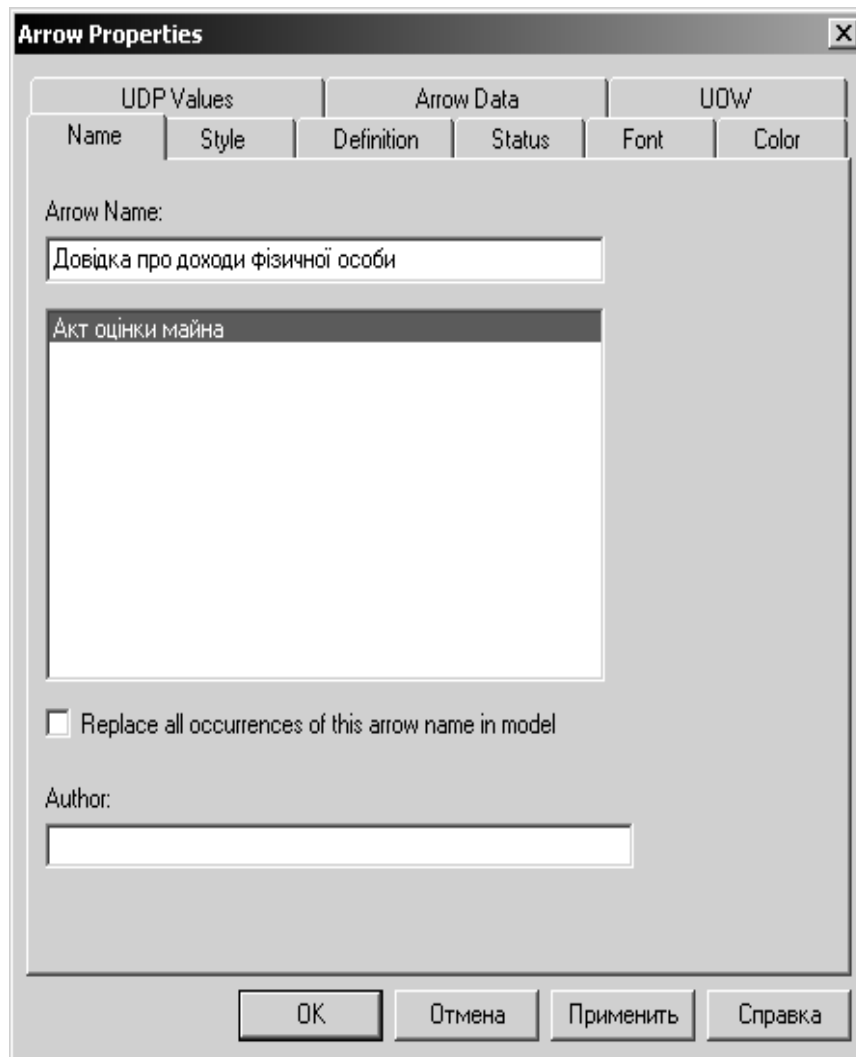
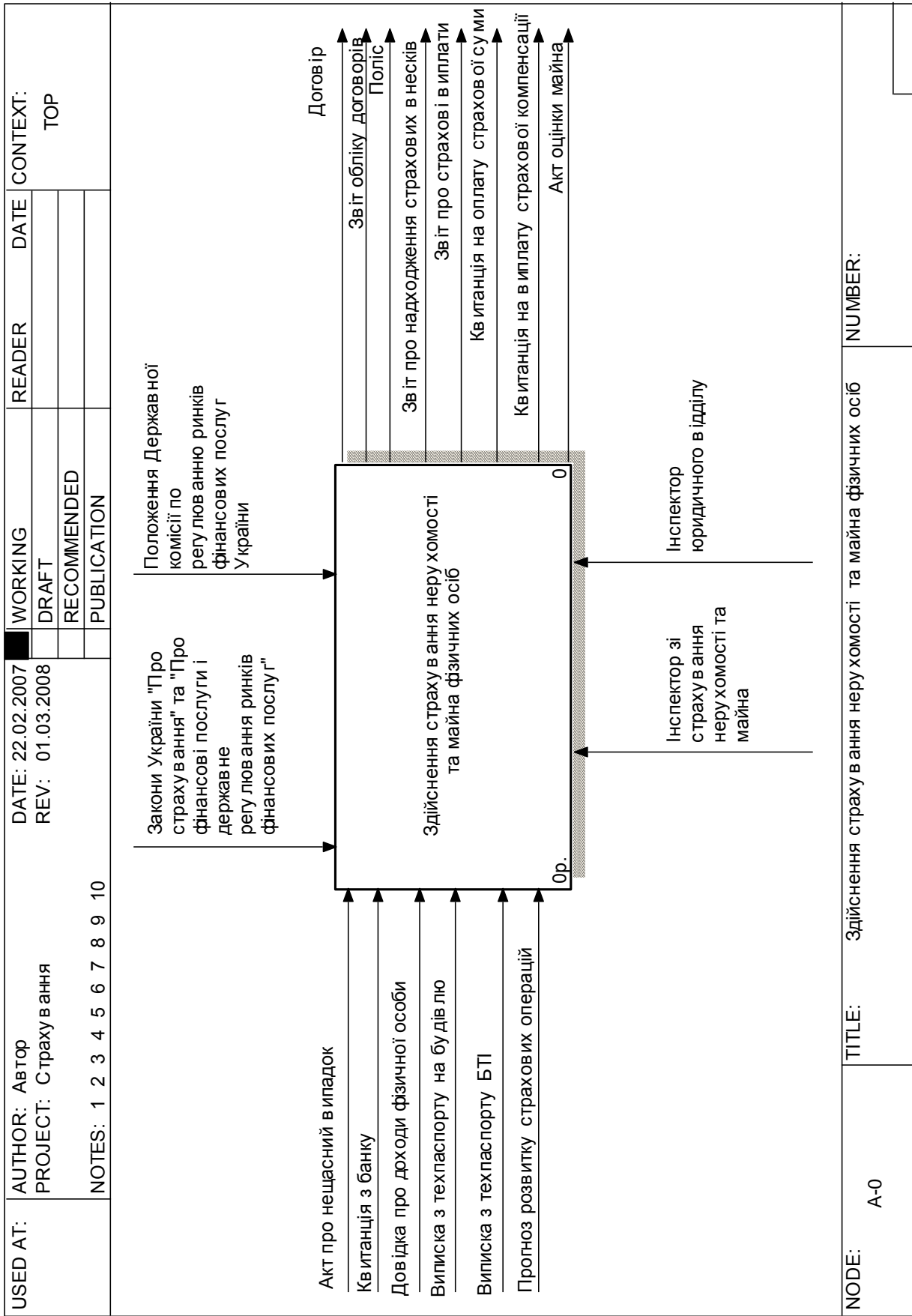


Рис. 4.13. Внесення назви інтерфейсної дуги

Після того, як внесені всі дані по контекстній діаграмі, ми отримаємо те, що наведено на рис. 4.14.

Для переходу на наступний рівень, потрібно провести декомпозицію контекстної діаграми. Для цього треба виділити необхідну роботу (у даному випадку вона одна) та вибрати інструмент **Go to Child Diagram**. У вікні, що відкрилося (рис. 4.15) треба обрати стандарт, котрий буде використовуватися на наступному рівні декомпозиції та кількість робіт на ньому.

У нашому випадку обираємо стандарт IDEF0 та кількість робіт – 3. Натискаємо на кнопку **OK**.



**Рис. 4.14. Контекстна діаграма моделі**

Після цього отримуємо діаграму декомпозиції, що зображена на рис. 4.16.

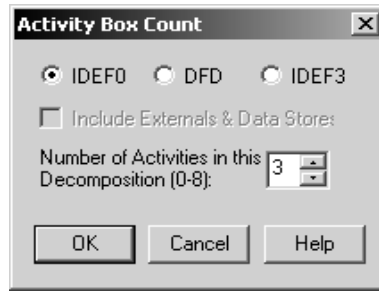


Рис. 4.15. Вікно вибору типу дочірньої діаграми та кількості робіт на ній

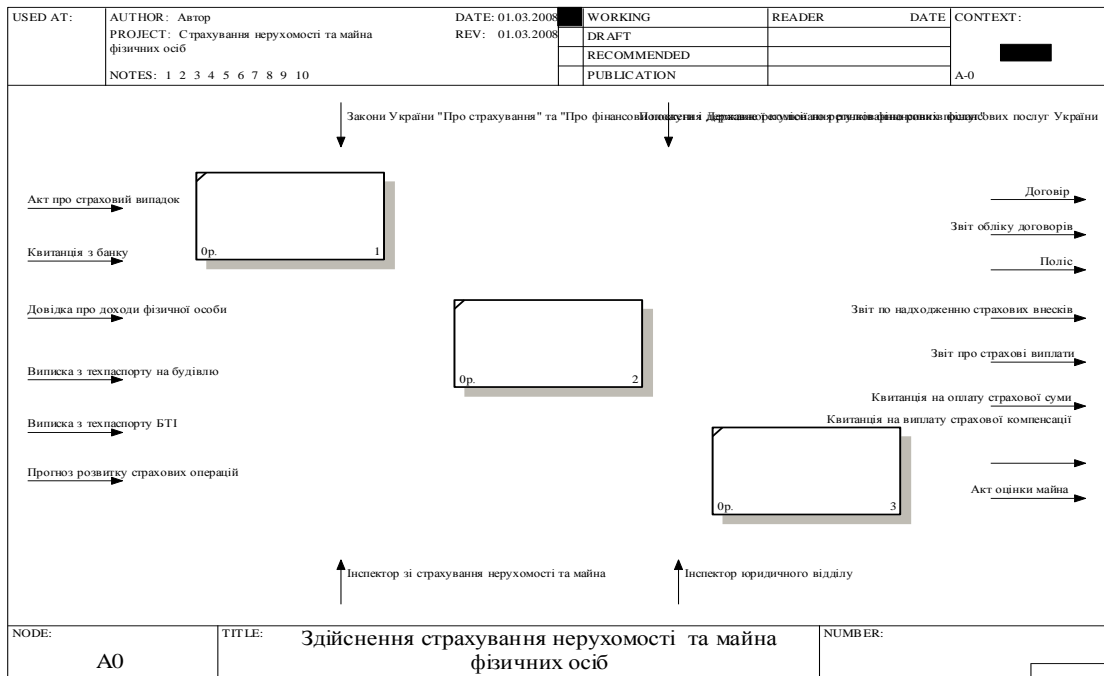


Рис. 4.16. Автоматична декомпозиція контекстної діаграми

Роботи на діаграмах декомпозиції розташовуються по діагоналі від лівого верхнього кута до правого нижнього. Даний порядок називається порядком домінування.

Кожна з робіт на діаграмі декомпозиції може бути, у свою чергу, декомповована. На діаграмі декомпозиції роботи нумеруються автоматично зліва направо. Номер роботи показується в правому нижньому кутку. Всі роботи моделі нумеруються. Номер складається з префікса та числа. Може бути використаний префікс будь-якої довжини, але зазвичай використовують префікс "A". Контексна (коренева) робота



дерева має номер A0. Роботи декомпозиції A0 мають номери A1, A2, A3 і т. д. Роботи декомпозиції нижнього рівня мають номер батьківської роботи і черговий порядковий номер, наприклад, роботи декомпозиції A3 матимуть номери A31, A32, A33, A34 і т. д. Роботи утворюють ієрархію, де кожна робота може мати одну батьківську і декілька дочірніх робіт, утворюючи дерево. Таке дерево називають деревом вузлів, а вищеописану нумерацію – нумерацією по вузлах. Є незначні варіанти нумерації, які можна налаштувати у вкладці Presentation діалогу Model Property (рис. 4.17). ВРwin автоматично підтримує нумерацію по вузлах, тобто при проведенні декомпозиції створюється нова діаграма і її автоматично привласнюється відповідний номер.

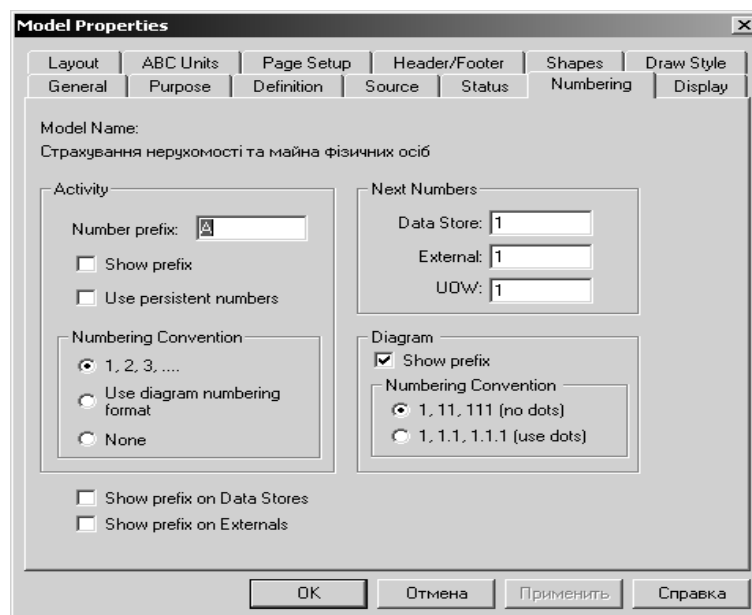


Рис. 4.17. Визначення стандарту ідентифікації блоків процесу

Як видно, на наступний рівень перенесені всі інтерфейсні дуги, котрі використовувалися на контекстній діаграмі. Необхідно дати назви всім роботам, котрі були авторами додані на діаграми (найменування робіт, починаючи з контекстної діаграми і завершуючи операціями на найнижчому рівні декомпозиції, **повинні бути унікальними**). Для того, щоб приєднати інтерфейсні дуги до необхідних робіт, можна зробити наступне:

перший варіант: обрати інструмент **Pointer Tool**, натиснути лівою кнопкою миші один раз на наконечнику стрілку (для інтерфейсних дуг „вхід”, „управління” чи механізм”) та приєднати її до блоку, коли з’явиться чорний трикутник, ніби вона тільки-но вами створена; або натиснути на

кінцівці стрілки (для інтерфейсної дуги „вихід“) та приєднати її до правої межі, коли з'явиться чорна бордюрна лінія.

другий варіант: обрати інструмент **Precedence Arrow Tool** та виконати ту ж послідовність дій, що і для першого варіанта.

Якщо потрібно зробити розщеплення стрілок, необхідно обрати інструмент **Precedence Arrow Tool**, нажати лівою кнопкою миші на необхідній інтерфейсній дузі, а потім приєднати її до необхідної роботи.

**Примітка:** інтерфейсні дуги входу та виходу однієї й тієї ж роботи не можуть бути ідентичними, бо інакше не має сенсу виконувати роботу, яка не дала жодного результату; кожна інтерфейсна дуга повинна мати назву.

Аналогічним чином будуються всі наступні рівні декомпозиції. Приклад опису задачі для модуля „Страхування нерухомості та майна фізичних осіб " наведений далі.

#### **Приклад опису задачі, що моделювалася**

У процесі аналізу предметної області, була складена контекстна діаграма (рис. 4.14), для якої були визначені наступні інтерфейсні дуги:

вхід: акт про страховий випадок, квитанція з банку, довідка про доходи фізичної особи, виписка з техпаспорту на будівлю, виписка з техпаспорту БТІ, прогноз розвитку страхових операцій на 2005 – 2010 роки;

вихід: договір, поліс, звіт обліку договорів, звіт щодо надходження страхових внесків, звіт про страхові виплати, квитанція на оплату страхової суми, квитанція на виплату страхової компенсації, акт оцінки майна;

управління: Закон України "Про страхування" та "Про фінансові послуги і державне регулювання ринків фінансових послуг", Положення Державної комісії про державне регулювання ринків фінансових послуг України;

механізм: інспектор зі страхування нерухомості та майна, інспектор юридичного відділу.

Декомпозиція контекстної діаграми реалізована на виділенні наступних робіт: облік договорів страхування, облік сплати страхових платежів за страховими полісами, облік страхових виплат, що призводить до наступної діаграми 1-го рівня декомпозиції (рис. 4.18).

Робота 1-го рівня декомпозиції "Облік договорів страхування" має такі інтерфейсні дуги:

вхід: довідка про доходи фізичної особи, виписка з техпаспорту на будівлю, виписка з техпаспорту БТІ, прогноз розвитку страхових операцій;  
вихід: договір, поліс, звіт обліку договорів, квитанція на оплату страхової суми, акт оцінки майна;

управління: Закон України "Про страхування" та "Про фінансові послуги і державне регулювання ринків фінансових послуг", Положення Державної комісії про державне регулювання ринків фінансових послуг України;

механізм: інспектор зі страхування нерухомості та майна, інспектор юридичного відділу.

Декомпозиція роботи 1-го рівня – "Облік договорів страхування" подана на рис. 4.19.

Робота 1-го рівня декомпозиції "Облік сплати страхових платежів за страховими полісами" має такі інтерфейсні дуги:

вхід: договір, квитанція з банку;

вихід: поліс, звіт щодо надходження страхових внесків;

управління: Закон України "Про страхування" та "Про фінансові послуги і державне регулювання ринків фінансових послуг", Положення Державної комісії про державне регулювання ринків фінансових послуг України;

механізм: інспектор зі страхування нерухомості та майна.

Декомпозиція роботи 1-го рівня – "Облік сплати страхових платежів за страховими полісами" подана на рис. 4.20.

Робота 1-го рівня декомпозиції "Облік страхових виплат" має такі інтерфейсні дуги:

вхід: договір, поліс, виписка з техпаспорту БТІ, акт про страховий випадок;

вихід: квитанція на виплату страхової компенсації, звіт про страхові виплати;

управління: Закон України "Про страхування" та "Про фінансові послуги і державне регулювання ринків фінансових послуг", Положення Державної комісії про державне регулювання ринків фінансових послуг України;

механізм: інспектор зі страхування нерухомості та майна, інспектор юридичного відділу.

Декомпозиція роботи 1-го рівня – "Облік страхових виплат" подана на рис. 4.21.

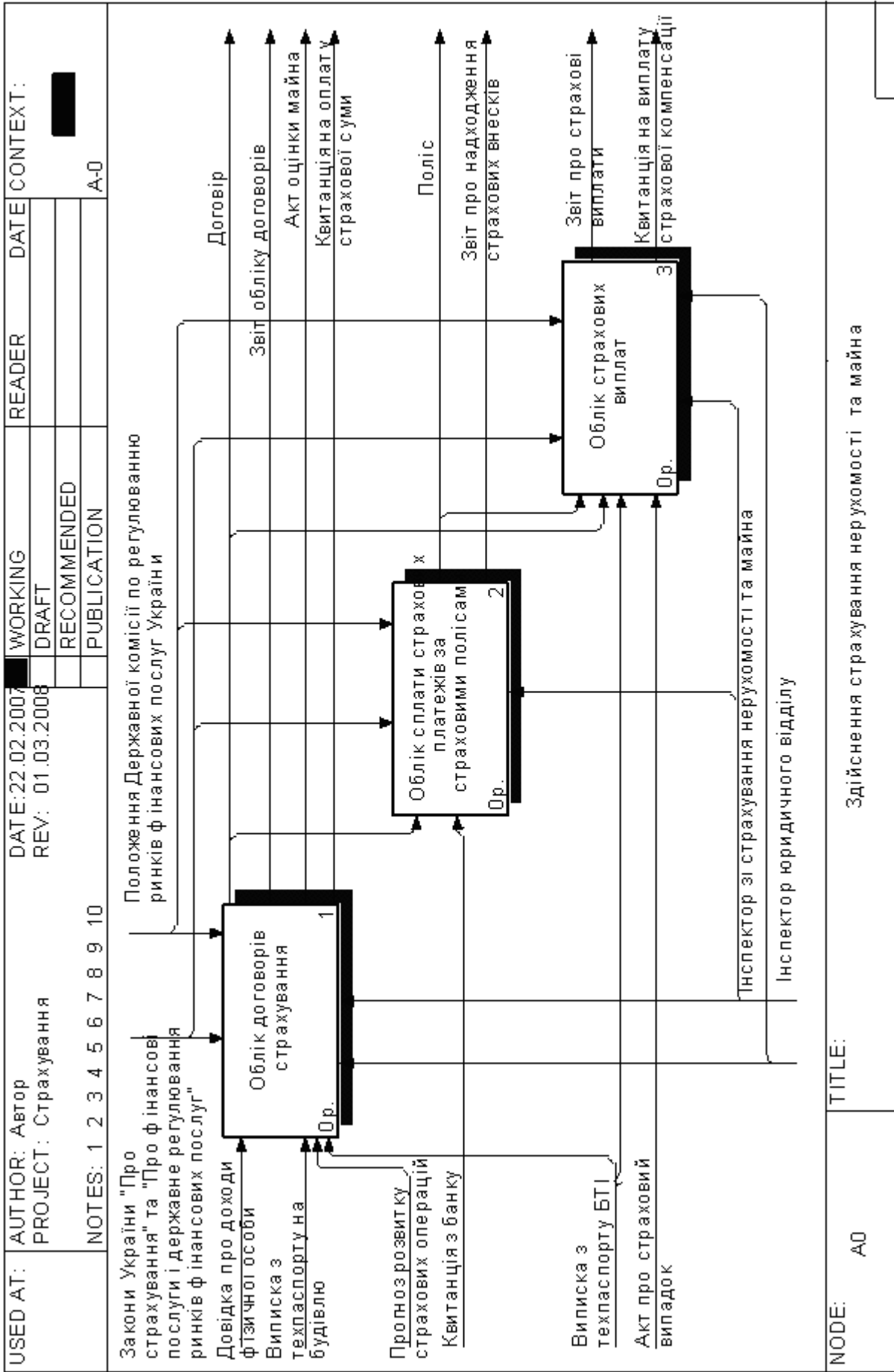
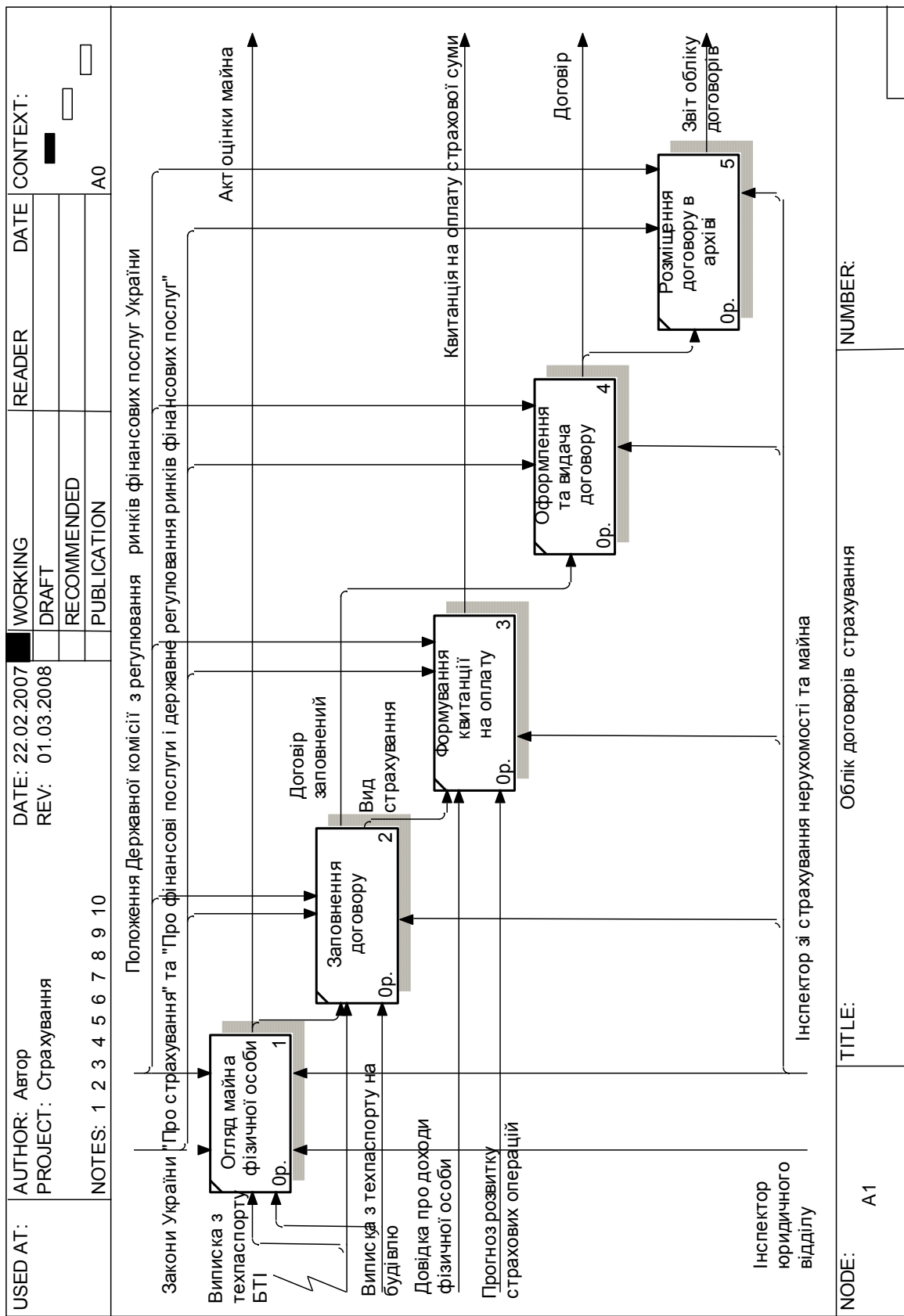


Рис. 4.18. Декомпозиція контексної діаграми



NODE: A1

TITLE: Облік договорів страхування

NUMBER:

Рис. 4.19. Декомпозиція роботи «Облік договорів страхування»

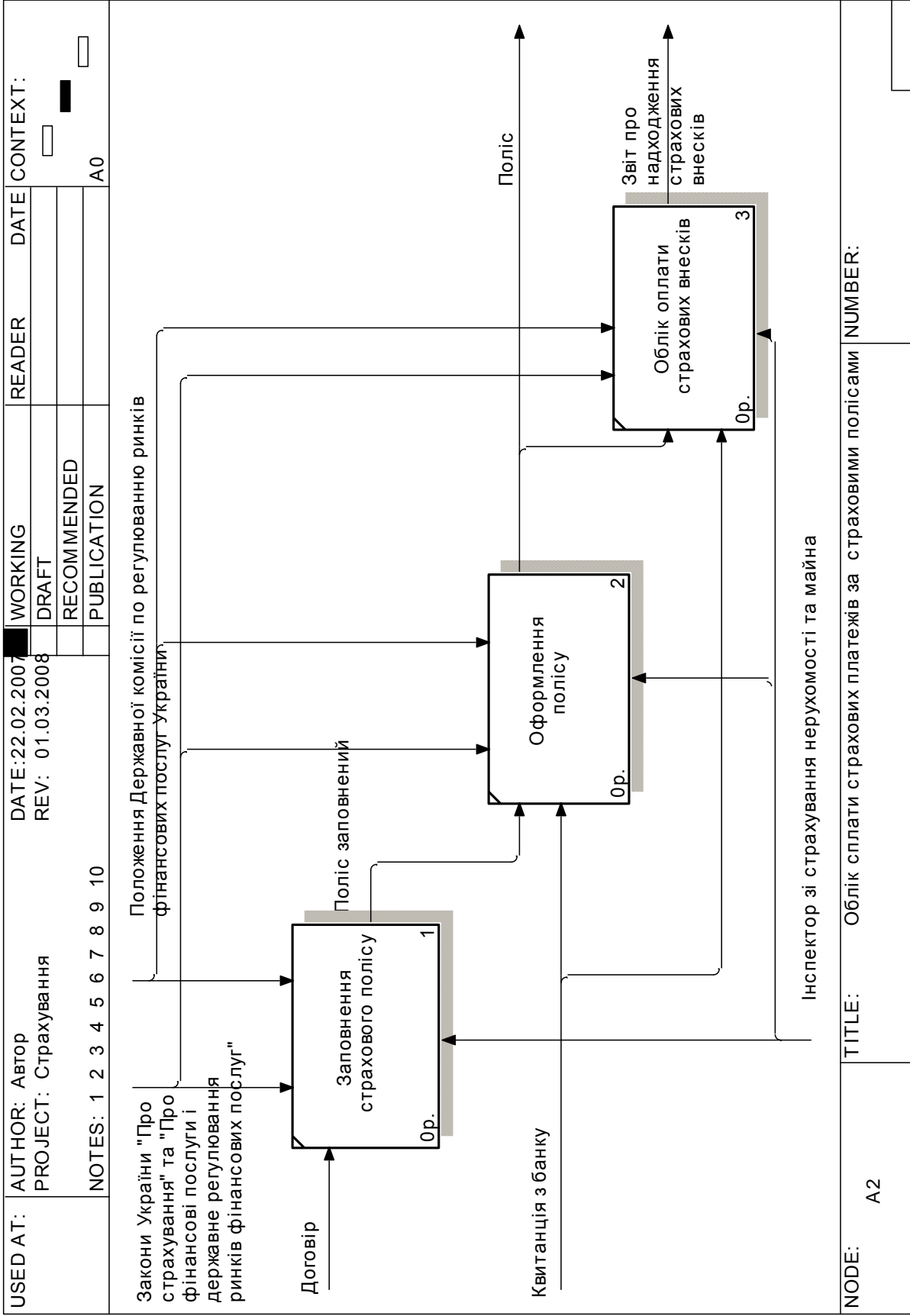


Рис. 4.20. Декомпозиція роботи «Облік сплати страхових платежів»

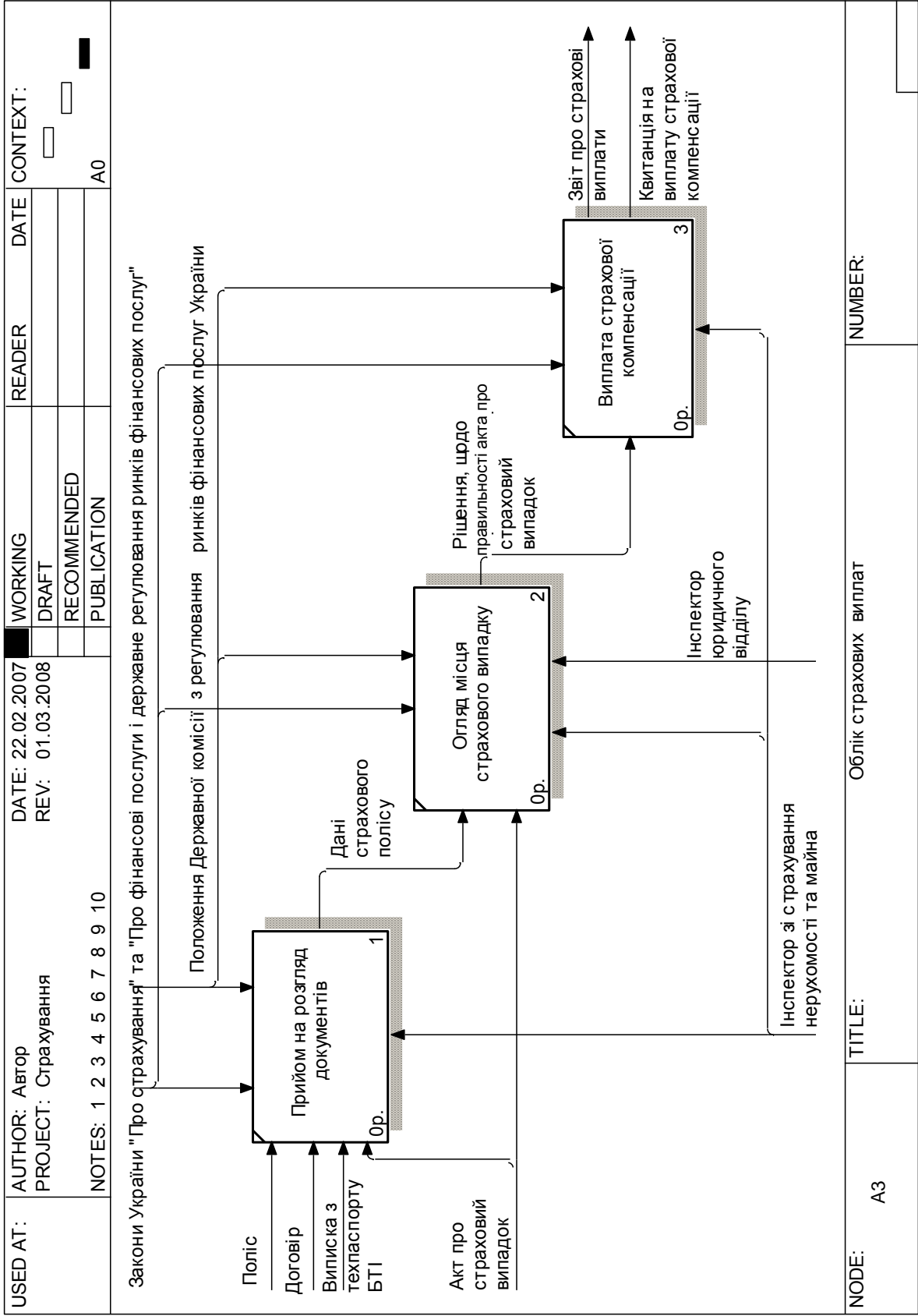


Рис. 4.21. Декомпозиція роботи «Облік страхових виплат»

#### **4.1.2. Використання діаграм потоків даних (DFD) для опису предметної області**

Контекстна діаграма модулю "Страхування нерухомості та майна фізичних осіб" й зовнішні об'єкти, з якими цей модуль взаємодіє (ці взаємодії позначені за допомогою вхідних і вихідних інформаційних потоків) наведена на рис. 4.22.

Зовнішня сутність Планово-економічний відділ моделює відділ страхової компанії, що формує документ "Прогноз розвитку страхових операцій на 2005 – 2010 роки" та отримує "Звіт обліку договорів", "Звіт щодо надходження страхових внесків", "Звіт про страхові виплати".

Зовнішня сутність Клієнт моделює будь-яких клієнтів, що обмінюється з проєктованим модулем наступною інформацією:

акт про страховий випадок, квитанція з банку, довідка про доходи фізичної особи, виписка з техпаспорту на будівлю, виписка з техпаспорту БТІ – від Клієнта до Компанії;

договір, поліс, квитанція на оплату страхової суми, квитанція на виплату страхової компенсації – від Компанії до Клієнта.

На рис. 4.23. наведена діаграма потоків даних, що деталізує модуль "Страхування нерухомості та майна фізичних осіб" за основними роботами, які виконуються при його вирішенні у страховій компанії. Функціонально модуль розбивається на наступні процеси: облік договорів страхування, облік страхових випадків, облік страхових виплат.

На даному рівні також уведені накопичувачі даних, які використовуються при вирішенні завдань модуля. Однак перед тим, як вставити сховища даних на діаграми, розробник повинен визначити загальний набір сутностей, а також основні їх атрибути, які використовуватимуться ним надалі.

Після цього необхідно зайти в пункт меню Dictionary / Entity і внести в Entity Dictionary всі сутності, яка використовуватиметься для сховищ даних (рис. 4.24).

Наступний крок проєктувальника – внесення для кожної з сутностей набору атрибутів. Для цього необхідно зайти в пункт Dictionary / Attribute (рис. 4.25).



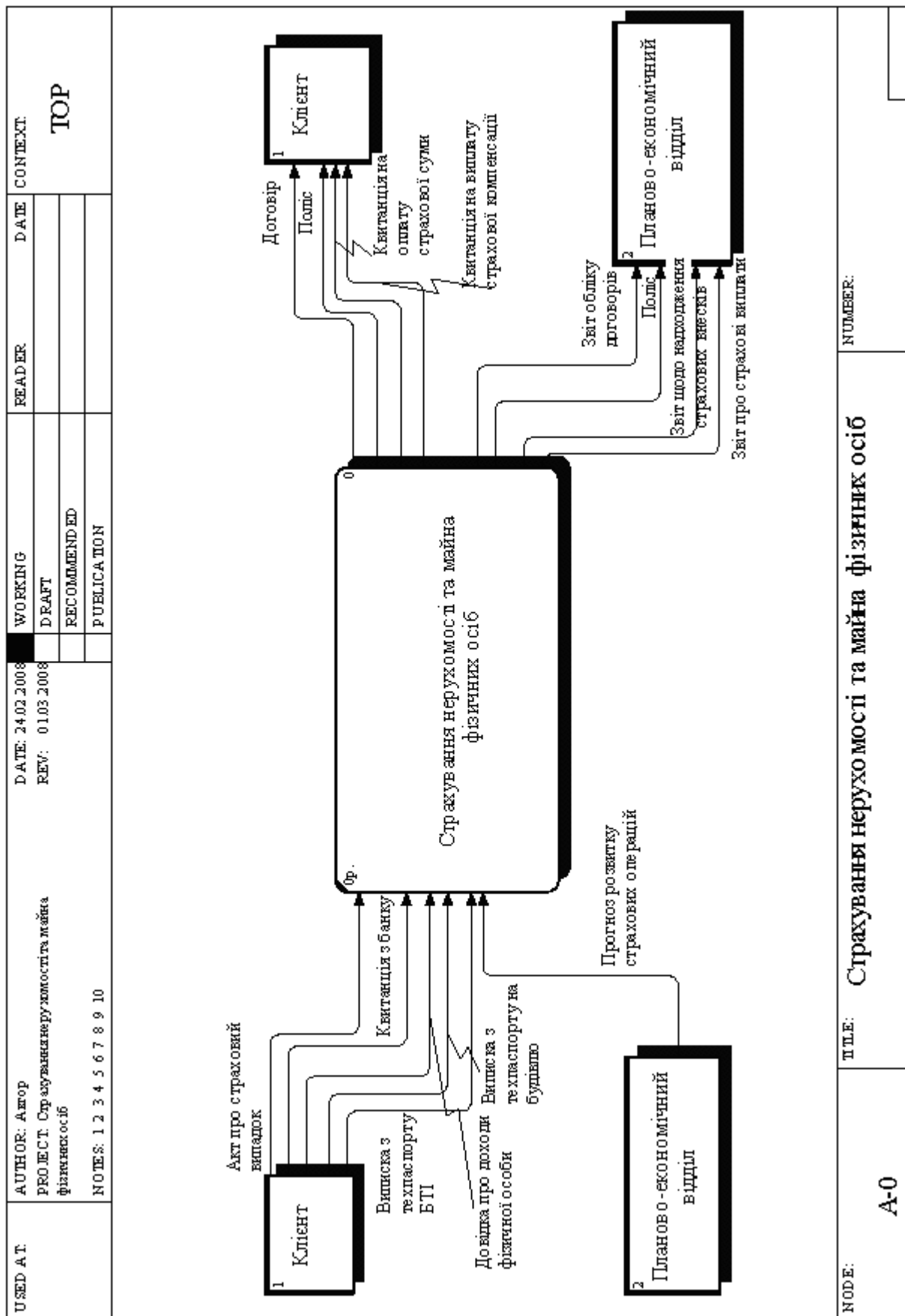


Рис. 4.22. Контекстна діаграма в стандарті DFD

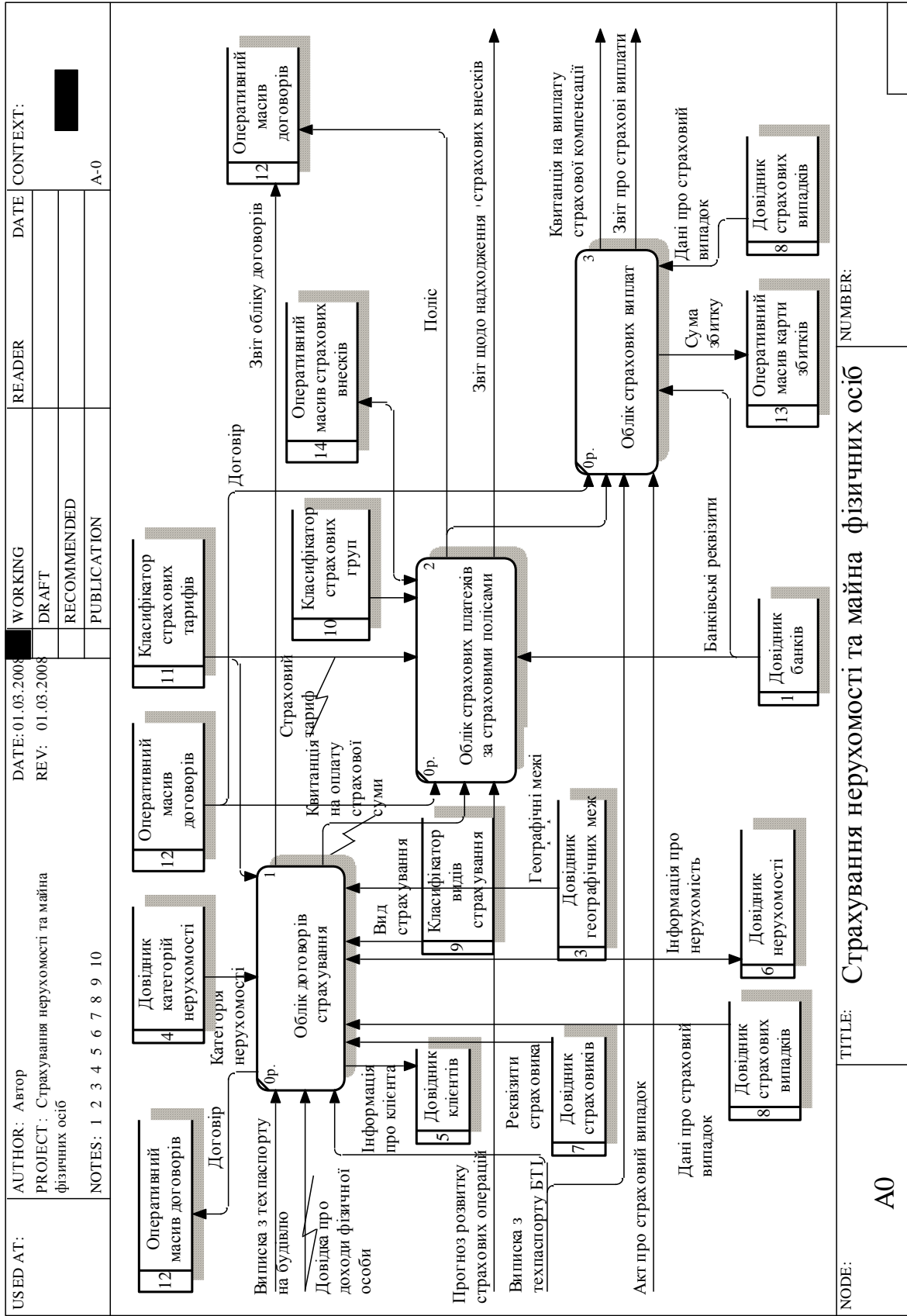


Рис. 4.23. Діаграма декомпозиції контекстної діаграми в стандарті DFD

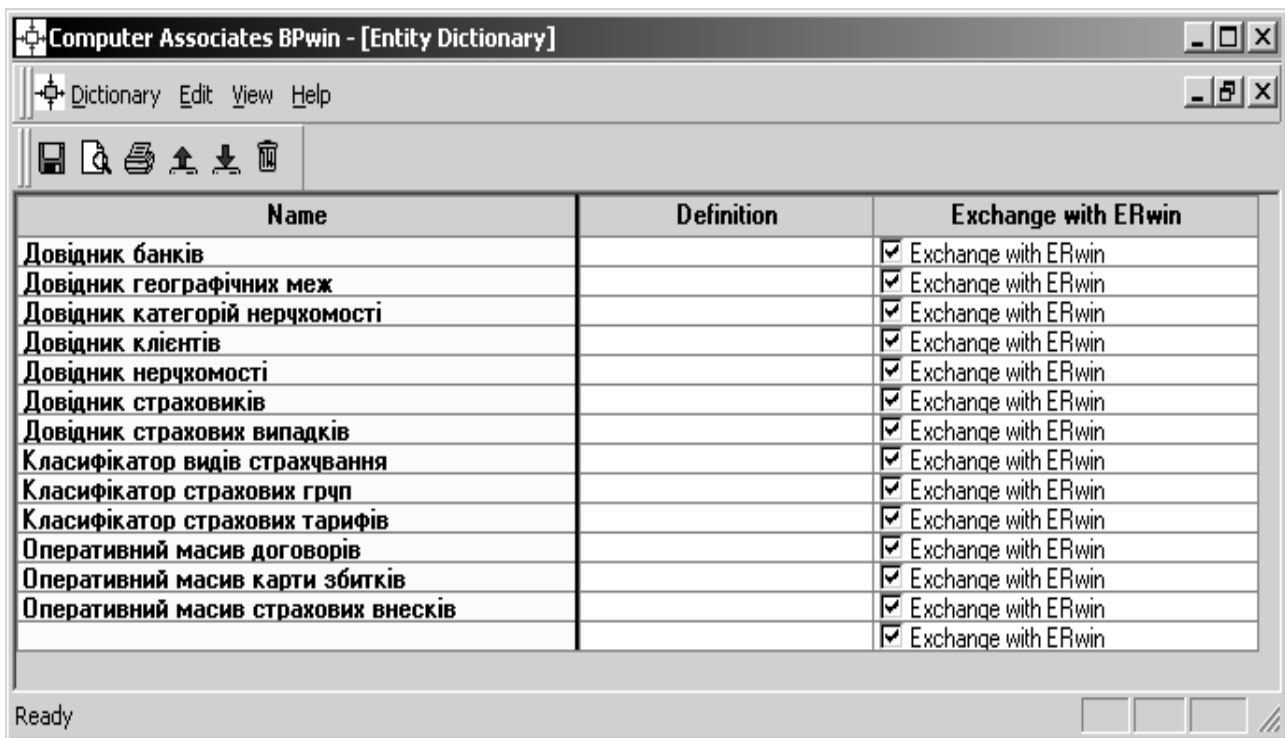


Рис. 4.24. Словник сутностей

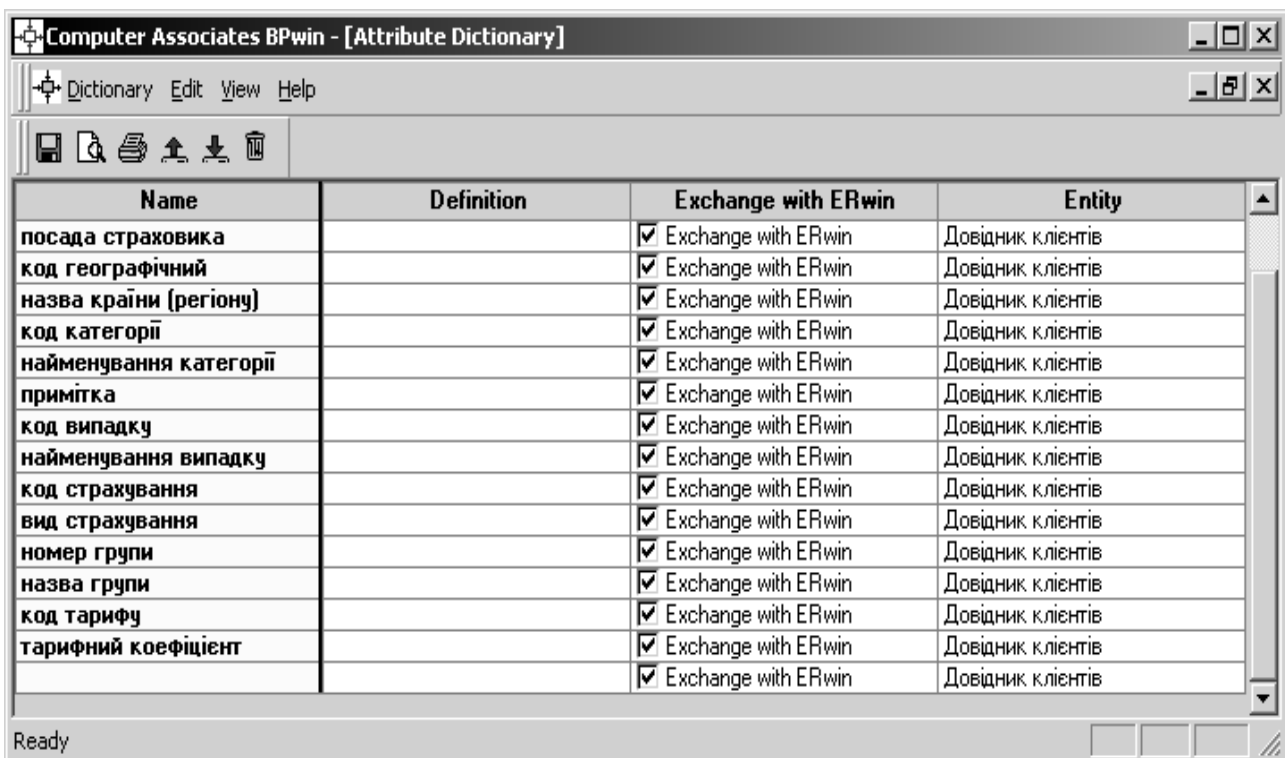


Рис. 4.25. Словник атрибутів

Тільки після вищезазначених дій можна вносити сховища даних у діаграму. При створенні сховища даних необхідно натиснути на кнопку


 на панелі інструментів, а потім на чистому полі робочої області. З'явиться вікно введення даних про сховище даних (рис. 4.26). Необхідно вибрати радіокнопку Entity і в зміненому вікні (рис. 4.27) вибрати найменування сутності зі спадного списку, які були внесені у словник сутностей заздалегідь. Після цього натиснути на кнопку ОК.



Рис. 4.26. Вікно створення сховища даних



Рис. 4.27. Вікно вибору сутності зі словника сутностей

Таким чином створюються всі сховища даних.

До них відносяться наступні:

довідник клієнтів, у якому накопичена інформація про реальних і потенційних клієнтів (покупців) підприємства у процесі їх контактів,

пропозицій, укладення договорів і інших взаємовідносин із спеціалістами служб маркетингу і збуту;

довідник страховиків, який містить дані про страхового інспектора, що уклав договір від компанії із даним клієнтом;

довідник географічних меж, що містить інформацію про регіони відповідно до адміністративно-територіального поділу України та назви інших країн, на які розповсюджується дія договору;

довідник категорій нерухомості, що містить дані про коди та найменування категорій нерухомості;

довідник страхових випадків містить перелік страхових випадків, за якими може бути укладено договір страхування;

класифікатор видів страхування, в якому накопичена інформація про послуги страхування, що надаються;

класифікатор страхових груп містить дані про групи страхування відповідно до Закону України "Про страхування";

класифікатор страхових тарифів містить дані про встановлені тарифні коефіцієнти відповідно до видів страхування і страхових груп;

довідник нерухомості формується на підставі інформації документу "Виписка з техпаспорту БТІ" і містить докладну інформацію про об'єкт страхування;

довідник банків, що містить поля: МФО банку, найменування банку;

оперативний масив договорів містить поля: код договору, дата початку дії, дата закінчення дії, дата укладання, причина відмови, дата відмови, строк дії, особливі умови, страхова сума, строк оплати;

оперативний масив страхових внесків містить поля: код договору, дата початку дії, дата закінчення дії, дата укладання, причина відмови, дата відмови, строк дії, особливі умови, страхова сума, строк оплати;

оперативний масив карти збитків формується на основі документа "Акт про страховий випадок" та містить поля: код карти, частка збитку, заявлена сума, дата випадку, адреса випадку, примітка.

Специфікації роботи "Облік договорів страхування" наведені нижче.

ВХІД: довідка про доходи фізичної особи, виписка з техпаспорту на будівлю, виписка з техпаспорту БТІ.

ВИХІД: договір, поліс, звіт обліку договорів, квитанція на оплату страхової суми.

АЛГОРИТМ: прийняти від клієнта заявку на страхування та оцінити нерухомість; оформити та виконати договір; провести контроль

виконання договорів; вибрати із відповідних довідників інформацію, необхідну для складання вихідних документів; передати квитанцію на оплату страхової суми клієнту.

Специфікації роботи "Облік страхових випадків" наведені нижче.

ВХІД: договір, квитанція з банку, прогноз розвитку страхових операцій.

ВИХІД: поліс, звіт щодо надходження страхових внесків.

АЛГОРИТМ: прийняти вхідні документи; вибрати із відповідних довідників інформацію, необхідну для складання вихідних документів; передати поліс клієнту.

Специфікації роботи "Облік страхових виплат" наведені нижче.

ВХІД: договір, поліс, виписка з техпаспорту БТІ, акт про страховий випадок.

ВИХІД: квитанція на виплату страхової компенсації, звіт про страхові виплати.

АЛГОРИТМ: прийняти вхідні документи; вибрати із відповідних довідників інформацію, необхідну для складання вихідних документів; вихідний документ передати на вхід до першої роботи та у юридичний відділ.

#### ***4.1.3. Використання діаграм, що описують логіку взаємодії робіт для опису предметної області із застосуванням стандарту IDEF3***

**Модель IDEF3** є однією з моделей SADT, яка реалізована в програмі VPwin. Модель IDEF3 використовується для опису технологічних процесів і логіки їх взаємозв'язку, тобто визначає характеристики робіт, послідовність і їх причинно-наслідкові зв'язки. При використанні стандарту IDEF3 у користувача-проектувальника з'являється можливість описати логіку взаємодії інформаційних потоків, сценарії дії співробітників організації з погляду проектувальника.

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

Робота в IDEF3 вимагає докладнішого опису, ніж робота в IDEF0, оскільки деталізує процес опису робіт до рівня операцій і логіки взаємодії

між ними. Кожна робота IDEF3 повинна мати асоційований документ, який включає текстовий опис компонентів роботи: об'єктів і фактів, пов'язаних з роботою, обмежень, що накладаються на роботу, і додаткові характеристики роботи. Ця інформація заноситься в діалозі **Activity Properties**. Ім'я UOW повинно бути представлено віддієслівним іменником, що позначає процес дії, одиночним або у складі словосполучення. Основний вихід (результат) роботи відображається іменником, найчастіше у складі того ж словосполучення, та залежить від віддієслівного іменника, що позначає UOW.

Тому після побудови контекстної діаграми і діаграми її декомпозиції (рис. 4.14, 4.15) в стандарті IDEF0 (не доцільно використовувати стандарт IDEF3 для контекстної діаграми і діаграми декомпозиції першого рівня, у зв'язку з тим, що тут неможливо описати інший порядок робіт і їх взаємодію, крім послідовного виконання), проведемо декомпозицію робіт (натиснувши кнопку ▼ "Go to Child Diagram" на панелі інструментів), та описавши послідовність надання послуг страхування нерухомості та майна фізичних осіб. З цього рівня декомпозиції вже доцільно використовувати стандарт IDEF3, котрий дозволить описати логіку взаємодії робіт, вказати осіб, які надають матеріали для початку робіт, та яким вони передаються після їх завершення. Тому в меню вибору типу діаграми треба встановити перемикач на діаграму IDEF3, та вказати кількість робіт – 5.

На рис. 4.28 представлена декомпозиція блоку "Облік договорів страхування" модулю "Страхування нерухомості та майна фізичних осіб" на основі діаграми потоків робіт, що відображає взаємодію між процесами обробки інформації і об'єктів, які є частиною цих процесів. При декомпозиції блок розбивається на наступні одиниці робіт (роботи): "Огляд майна фізичної особи", "Заповнення договору", "Формування квитанції на оплату", "Оформлення та видача договору", "Розміщення договору у архіві".


Найменування робіт формуються так само, як і в стандарті IDEF0.

Необхідно зазначити, що закінчення роботи "Огляд майна фізичної особи" служить сигналом для початку одразу двох робіт:

- 1) "Заповнення договору";
- 2) "Формування квитанції на оплату страхової суми".

І тільки за умови закінчення обох цих робіт може розпочатися робота "Оформлення та видача договору".

Для відображення логіки описаних процесів на діаграмі необхідно використати перехрестя "Асинхронне I", суть яких полягає в тому, що у випадку розгалуження стрілок **всі** наступні процеси повинні розпочатися (не обов'язково одночасно), а у випадку злиття стрілок **всі** попередні процеси повинні бути завершені.

Для того, щоб включити це перехрестя потрібно на панелі інструментів натиснути кнопку , потім натиснути на робочій області та вибрати відповідне перехрестя з меню "Select Junction Style" – в нашому випадку, це перехрестя „Асинхронного I". Аналогічним чином додається ще одне перехрестя. Перехрестя і роботи розміщуються раціонально, щоб було мінімум перетину стрілок; після чого можна з'єднувати роботи та перехрестя так, як показано на рис. 4.28.

**Примітка:** в стандарті IDEF3 використовуються тільки інтерфейсні дуги „вхід" та „вихід", а інтерфейсні дуги „управління" та „механізм" – відсутні.

Окрім перехресть на діаграмі потоків робіт необхідно зобразити об'єкти посилання, що виражають певну ідею, концепцію або дані, що неможливо зв'язати зі стрілкою чи перехрестям.

Робота "Огляд майна фізичної особи" повинна бути зв'язана з об'єктом посилань "Клієнт", отримуючи від нього випуску з техпаспорту на будівлю та випуску з техпаспорту БТІ. В даному випадку ці документи характеризують інформацію про об'єкт страхування (нерухомість).

З роботою "Формування квитанції на оплату" пов'язаний об'єкт посилання "Клієнт", який передає для неї довідку про доходи фізичної особи, а також отримує квитанцію на оплату страхової суми.

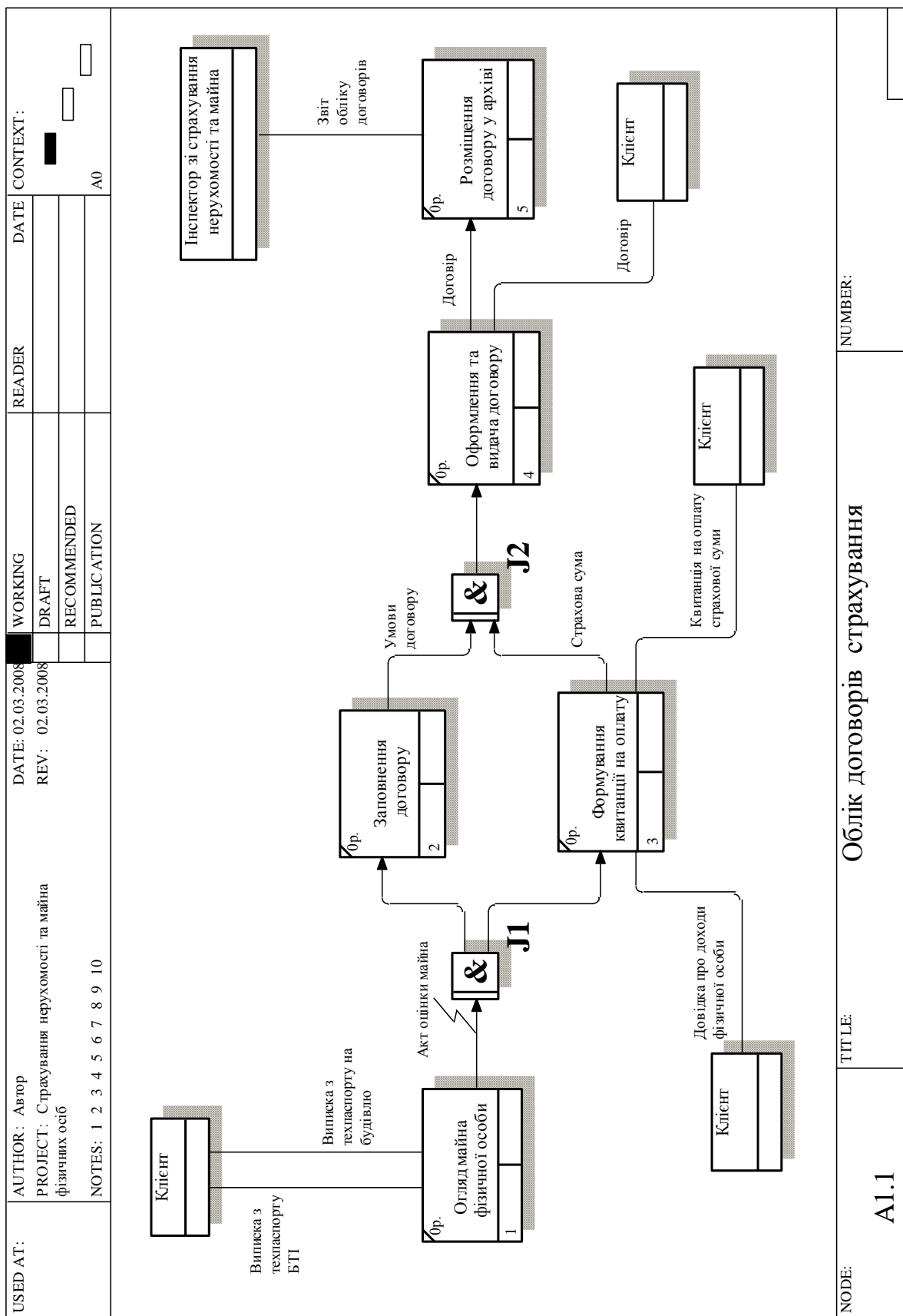
Результатом роботи "Оформлення та видача договору" є договір страхування, який передається клієнтові.

Робота "Розміщення договору у архіві" також пов'язана з певним об'єктом – "Інспектор зі страхування нерухомості та майна", котрий отримує копію договору страхування.

Для цього обирається кнопка "об'єкт посилання" та вказується відповідний об'єкт. Після цього вказується зв'язок у вигляді лінії без стрілок – на інтерфейсній дузі натиснути праву кнопку миші та на закладці Style обрати тип Referent.

При побудові діаграми проектувальник повинен сам визначити необхідний ступінь декомпозиції.





Облік договорів страхування

A1.1

Рис. 4.28. Декомпозиція роботи "Облік договорів страхування" в стандарті IDEF3

## 4.2. Застосування CASE-засобу Business Studio при вирішенні комплексу завдань "Керування договорами"

У даному підрозділі розглянуто практичне вирішення завдань у предметній області "Керування договорами" для проектування окремого модуля в пакеті Business Studio.

При проектуванні модулів комплексу "Керування договорами" потрібно виконати таку послідовність дій.

1. Запустити Business Studio (рис. 4.29).

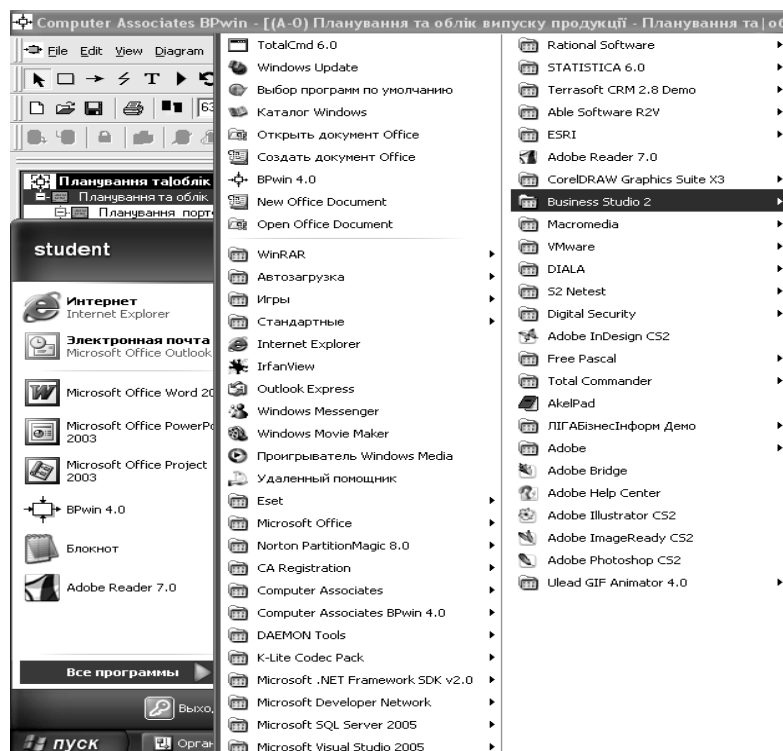


Рис. 4.29. Запуск Business Studio

Обираємо базу даних на сервері: у даному прикладі – це vcisserv (рис. 4. 30).

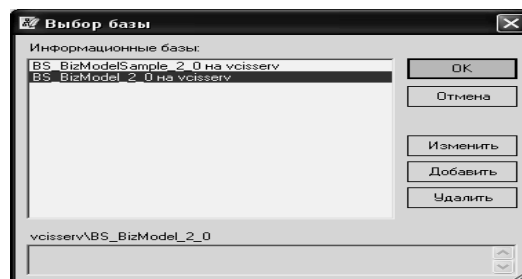


Рис. 4.30. Вікно вибору бази даних

Для відкриття БД вводимо ім'я користувача: для даного прикладу – bsuser – й пароль: bsuser (рис. 4.31).

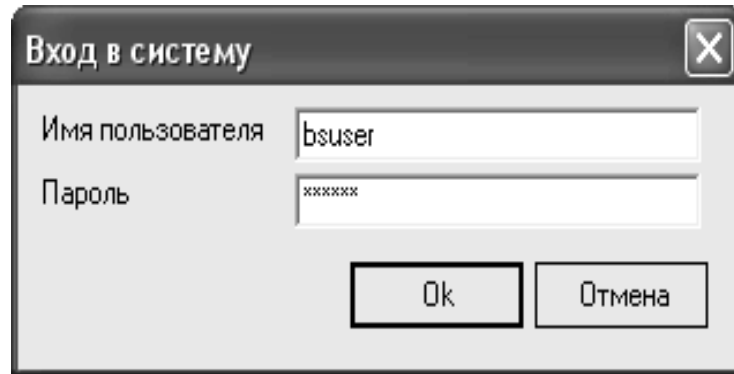


Рис. 4.31. Вікно входу в систему

Після цього відкривається головне вікно додатка Business Studio (рис. 4.32).

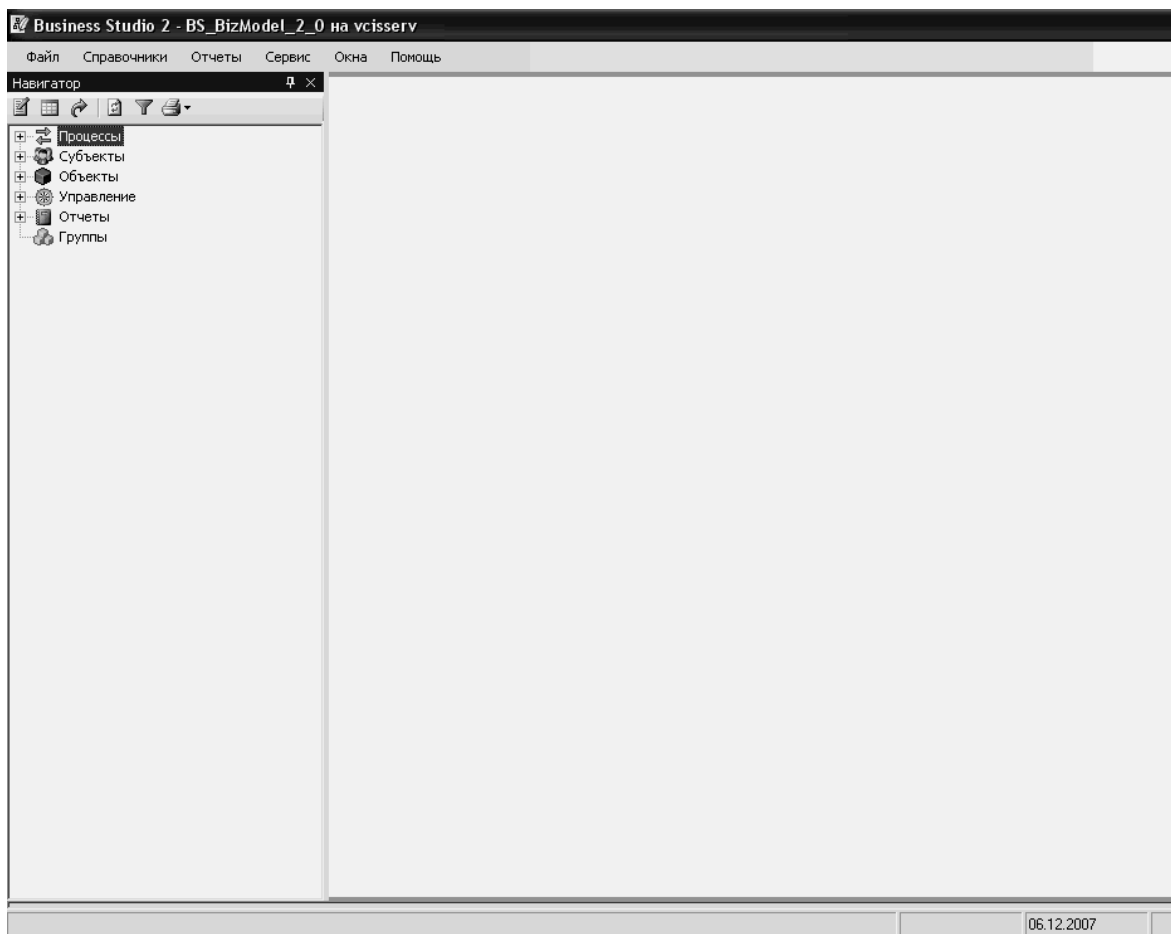


Рис. 4.32. Головне вікно Business Studio

Якщо Навигатор об'єктів у відкритому вікні відсутній, то викликаємо його через меню "Файл -> Навигатор об'єктів" (рис. 4.33).



Рис. 4.33. Відображення навігатора об'єктів

Для подальшої роботи скористаємось змістовним призначенням окремих стрілок для відображення об'єктів у різних розділах Business Studio (табл. 4.2).

Таблиця 4.2

### Відповідність типів стрілок типам об'єктів у різних розділах навігатора об'єктів Business Studio

Тип стрілки	Розділи навігатора об'єктів Business Studio	Типи об'єктів
Вход и Выход	Объекты -> ТМЦ	Предмети праці й продукція: сировина, матеріали, паливо, енергія, що комплектують вироби, запасні частини
	Объекты ->Информация	Різноманітні відомості, що надходять комунікаційними каналами або є вже наявні у виконавця роботи, зокрема книги, газети, журнали, графіки, діаграми, малюнки, мовні повідомлення, телетекст, відео-кадри й т. п.
	Объекты -> Прочее	Програмні продукти, нематеріальні активи, гроші й т. п.
	Объекты->Документы ->Бумажные документы	План постачань, виробнича програма, портфель замовлень і т. п.

	Объекты->Документы-> Электронные документы	Електронний лист, файли документів
Механизм	Субъекты	Посада
	Объекты -> Прочее	Програмний продукт
	Объекты -> ТМЦ	Інструмент, машина, пристрій
Управление	Объекты -> Информация	Сигнал, повідомлення
	Объекты -> Документы: Бумажный документ, электронный документ	Інструкції, правила, методики, керівництва, нормативно-правові акти, закони, постанови, кодекси й т.п.

2. Далі створюємо папку користувача.

Для цього вибираємо в навігаторі розділ "Процесс" і за допомогою контекстного меню, пункту "Добавить от текущего" вибираємо пункт "Папка" (рис. 4.34, 4.35).

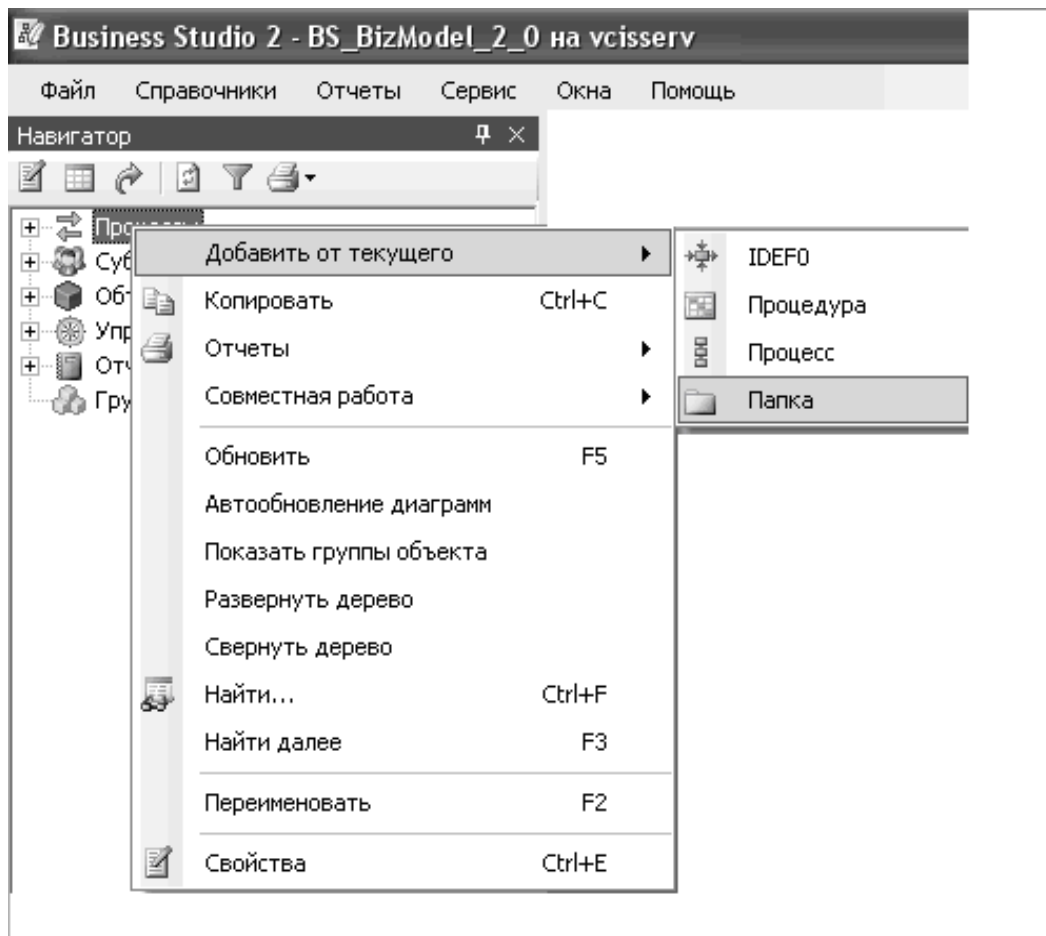


Рис. 4.34. Створення папки в розділі "Процесс"

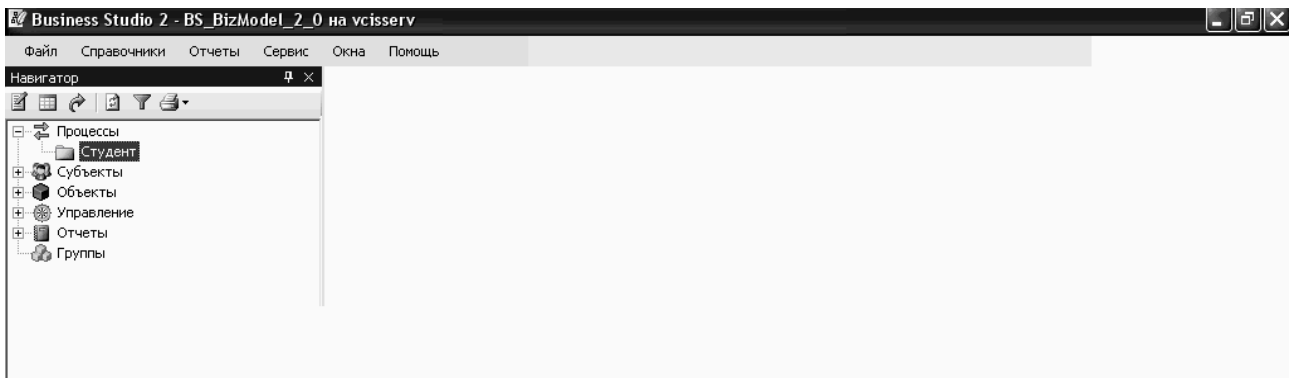


Рис. 4.35. Папки в розділі "Процессы"

3. На наступному етапі будуюмо контекстну діаграму.

Для цього вибираємо створену папку й за допомогою контекстного меню, пункту "Добавить от текущего" вибираємо пункт "IDEFO" (рис. 4.36).

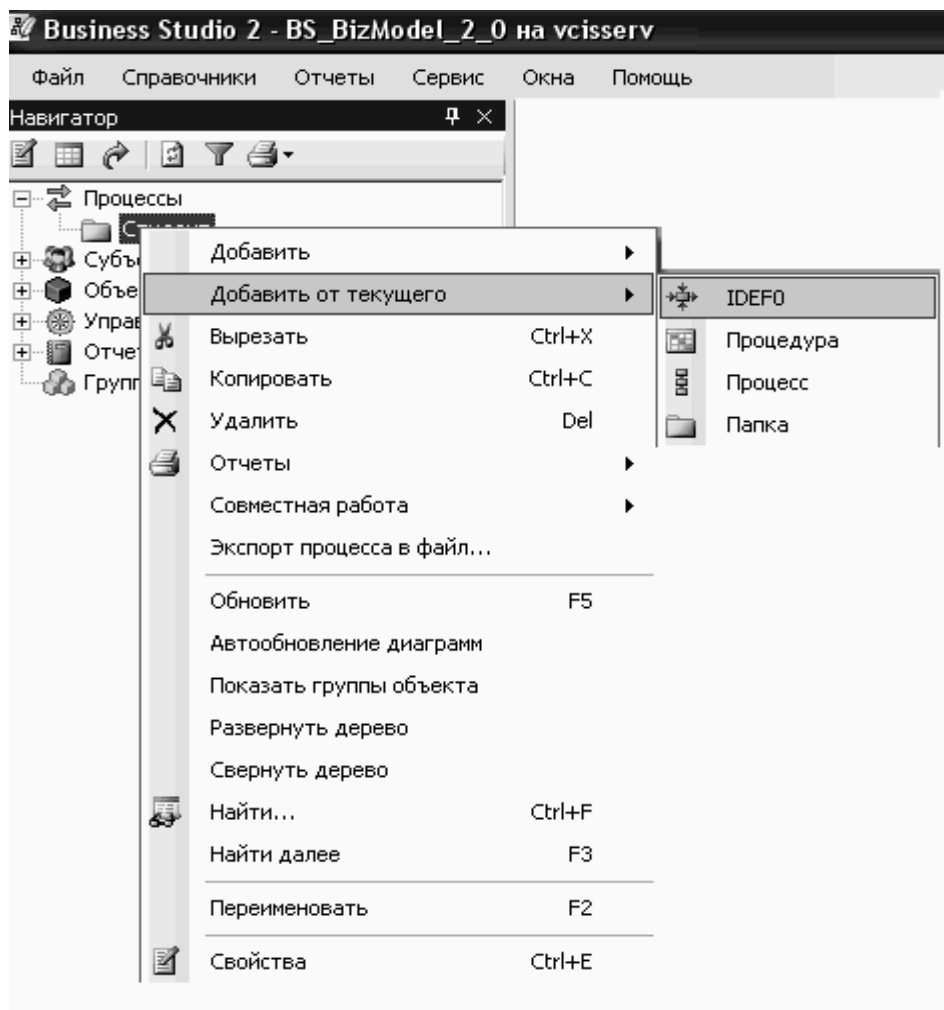


Рис. 4.36. Створення в процесі IDEFO у папці

4. Далі створюємо стрілки входів.

Для створення цих стрілок необхідно використовувати графічний редактор моделі.

Для відкриття графічного редактора необхідно натиснути на кнопку / у панелі інструментів або двічі клацнути лівою кнопкою миші по імені моделі (у нашому випадку – "Планування постачань"). У результаті в правій частині робочої області головного вікна відкриється графічний редактор (рис. 4.37).

Для малювання стрілок необхідно заповнити розділи навігатора "Субъекты" і "Объекты".

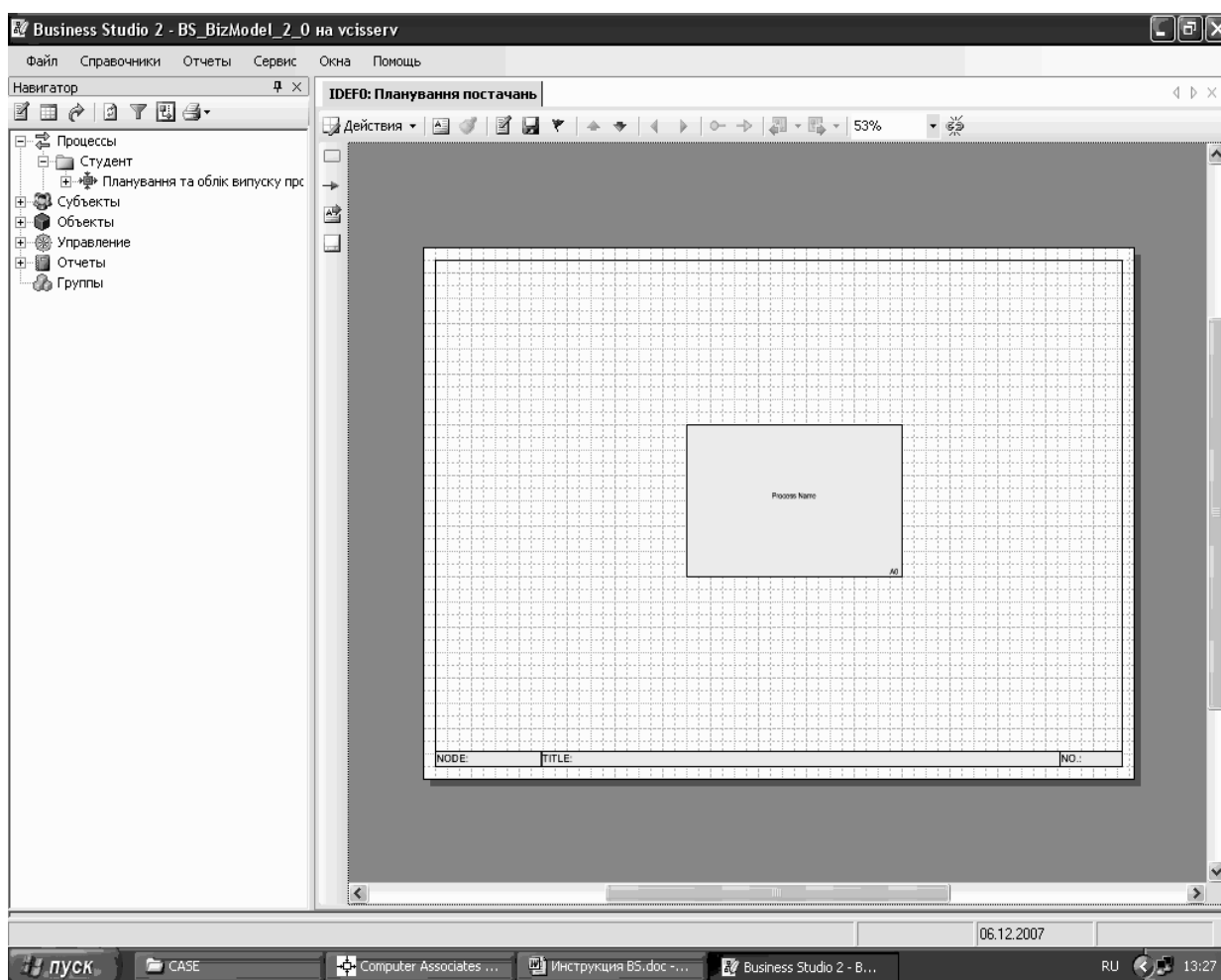


Рис. 4.37. Форма графічного редактора для контекстної діаграми

У розділі "Субъекты" спочатку треба створити папку, а потім – ієрархію підрозділів і посад відповідно до рис. 4.37а, та організаційну діаграму (рис. 4.37б, 4.37в).

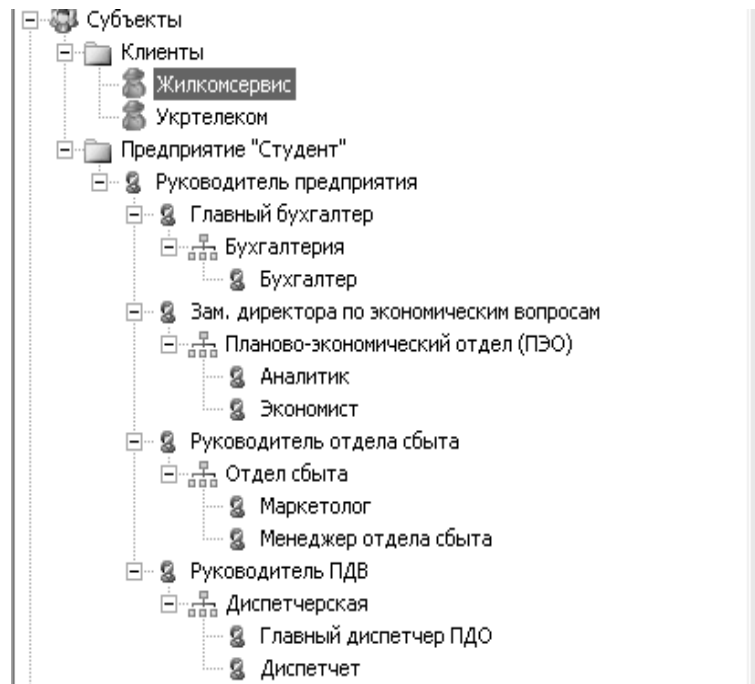


Рис. 4.37 а. Ієрархія підрозділів і посад

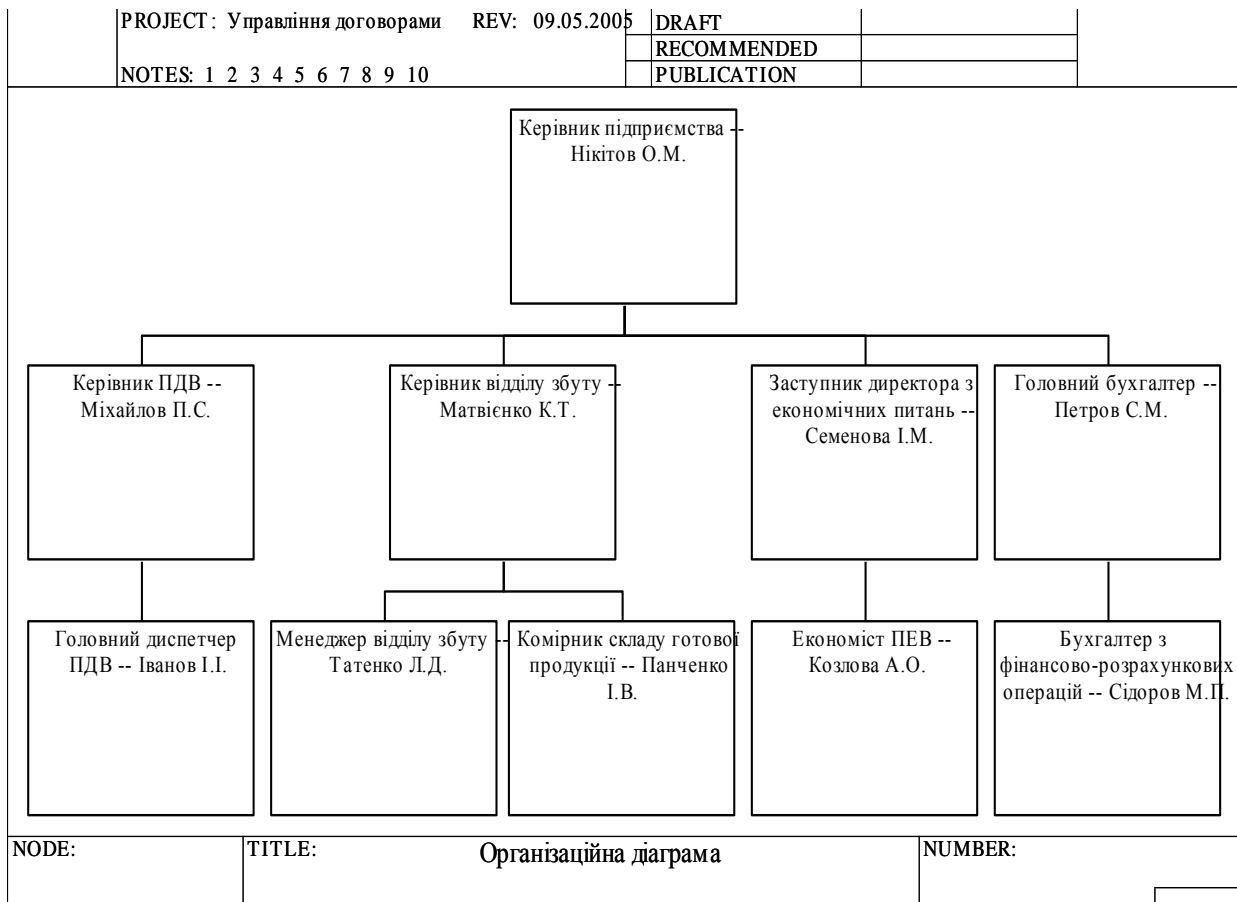


Рис. 4.37 б. Організаційна діаграма



Далі у розділі "Объекты" треба обрати підрозділ "Документы", у якому вибрати підрозділ "Бумажные документы" і створити папку. Потім, використовуючи контекстне меню папки, треба вибрати пункт "Добавить от текущего".

Таким же чином створити: вхідні документи: "План поставок" і "Інформація про залишки ГП на складі"; вихідні документи: "Накладна на передачу ГП на склад" і "Виробнича програма".

У розділі "Объекты" вибрати розділ "Прочее", у якому створити папку, а в ній папки: "Методики, справочники и т. п." і "Программные продукты".

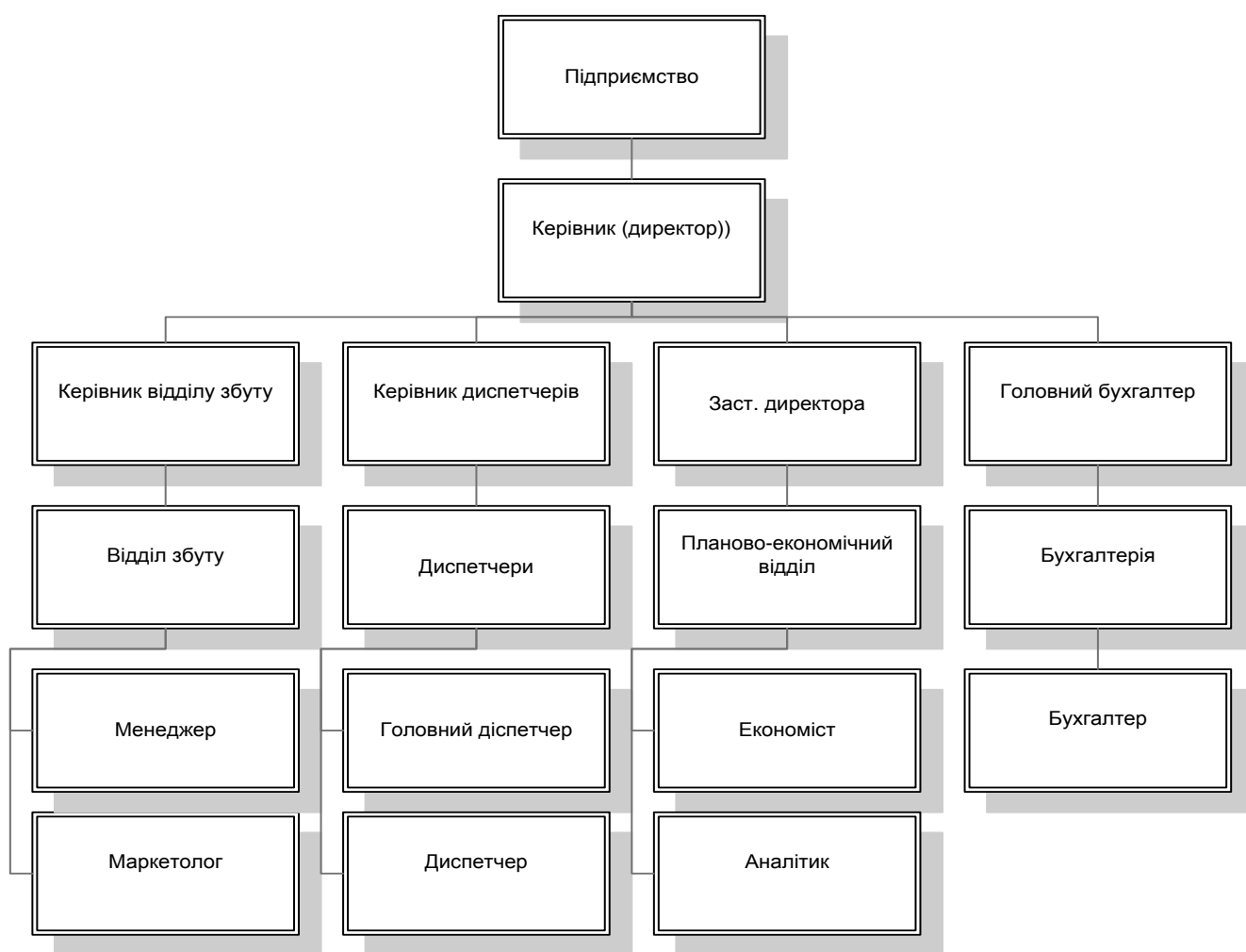


Рис. 4.37 в. **Організаційна діаграма**

Таким чином, вся необхідна для малювання стрілок інформація буде введена (рис. 4.38).

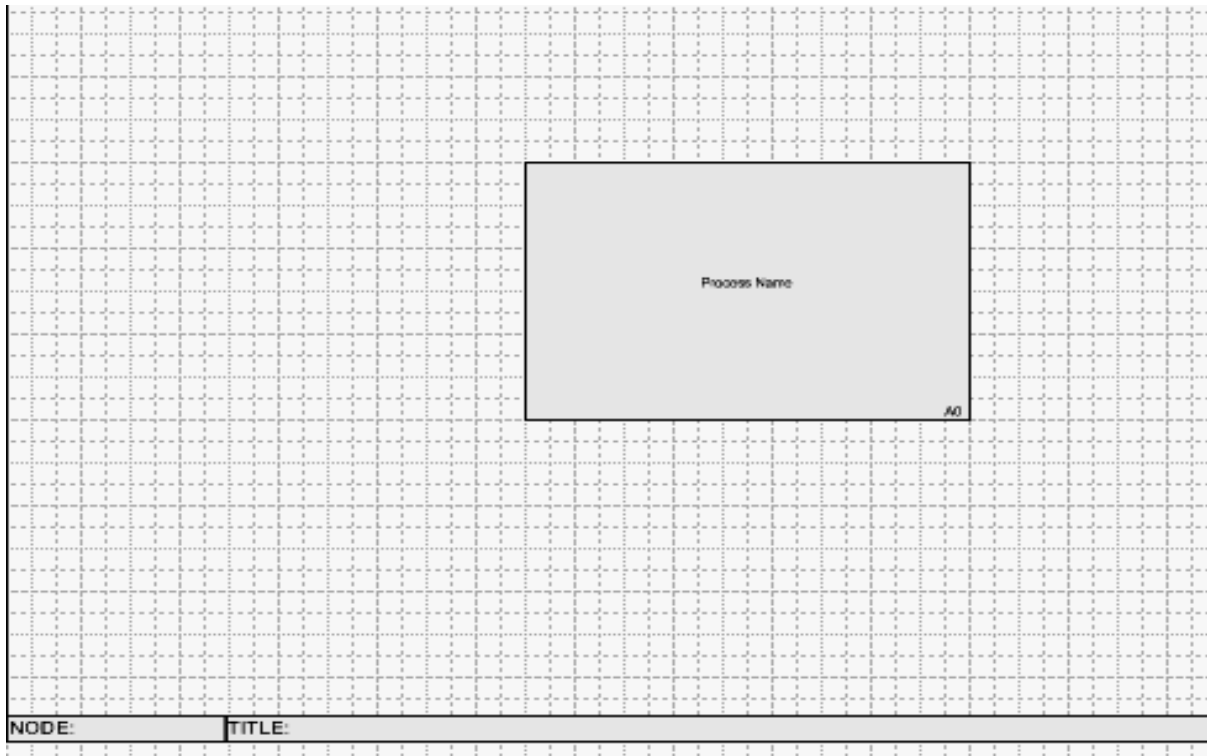


Рис. 4.38. Заповнені розділи в навігаторі об'єктів

Для створення стрілки необхідно: вибрати відповідний елемент у навігаторі, лівою клавшею миші перетягнути його в область діаграми, і лівий кінець автоматично створеної стрілки з'єднати з лівою гранню прямокутника роботи (рис. 4.39).

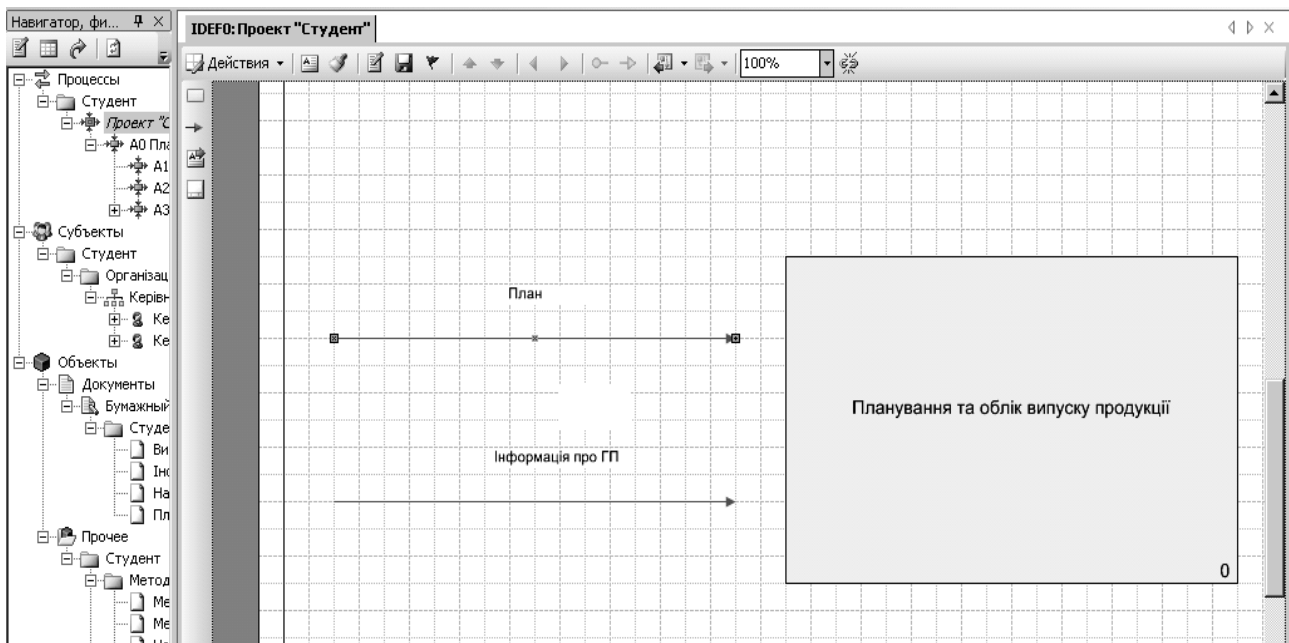


Рис. 4.39. Створення стрілки на діаграмі

Далі потрібно з'єднати правий кінець кожної стрілки з лівою межею прямокутника (рис. 4.40).

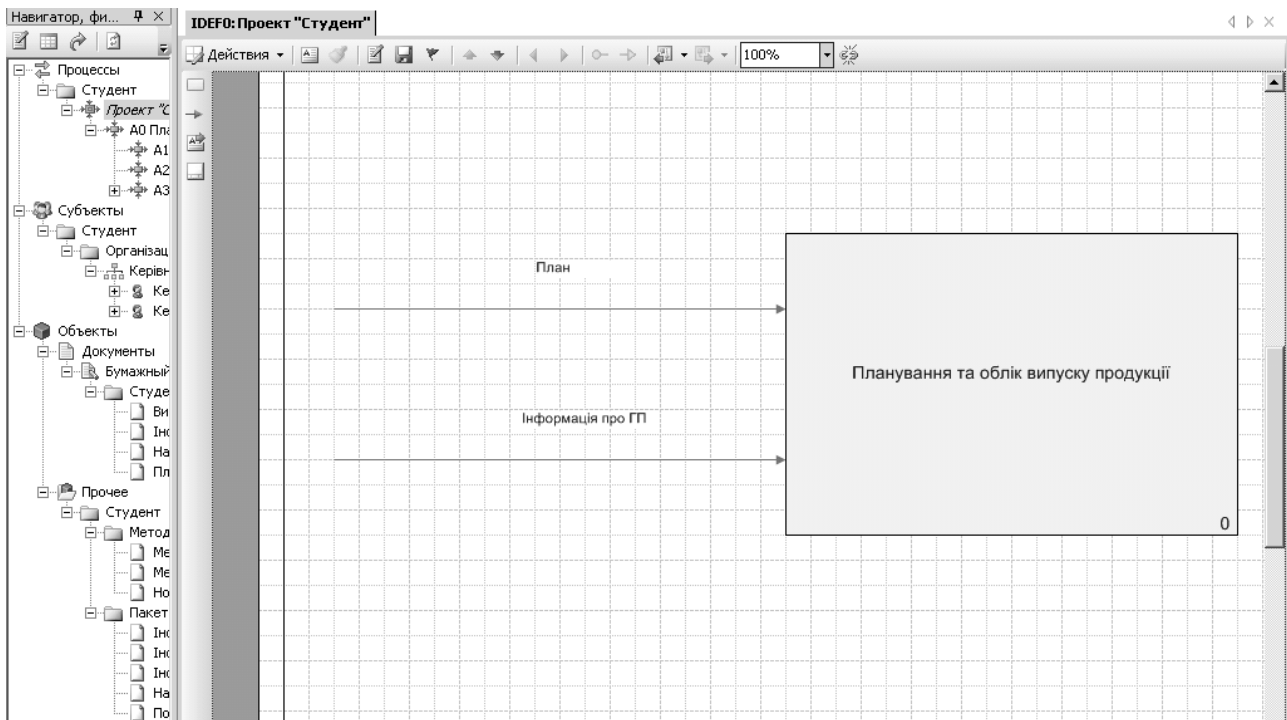



Рис. 4.40. Створення стрілок входів

5. Далі будемо стрілки керування.

Для створення стрілки керування необхідно: вибрати відповідний елемент у навігаторі, лівою клавішею миші перетягнути його в область діаграми, і нижній кінець автоматично створеної стрілки з'єднати з верхньою гранню прямокутника роботи (рис. 4.41).

Для редагування назви стрілки необхідно двічі клацнути по ній лівою клавішею миші і в поле, що з'явилося для редагування, ввести необхідну назву.

Для того, щоб змінити місце розташування назви стрілки на діаграмі необхідно включити режим "Работа с метками" за допомогою кнопки  на верхній частині панелі інструментів графічного редактора й одним клацанням лівою клавішею миші вибрати стрілку. Після того, як її мітка виділиться спеціальними маркерами, її можна переміщати, обертати, розтягувати (перетворюючи в однорядковий текстовий рядок), стискати й розширювати (перетворюючи текст мітки в багаторядковий текст) для економії вільного місця на діаграмі та її естетичного

компонування, через контекстне меню змінювати стиль, шрифт (меню Format->Text), колір та інші параметри візуального подання мітки.

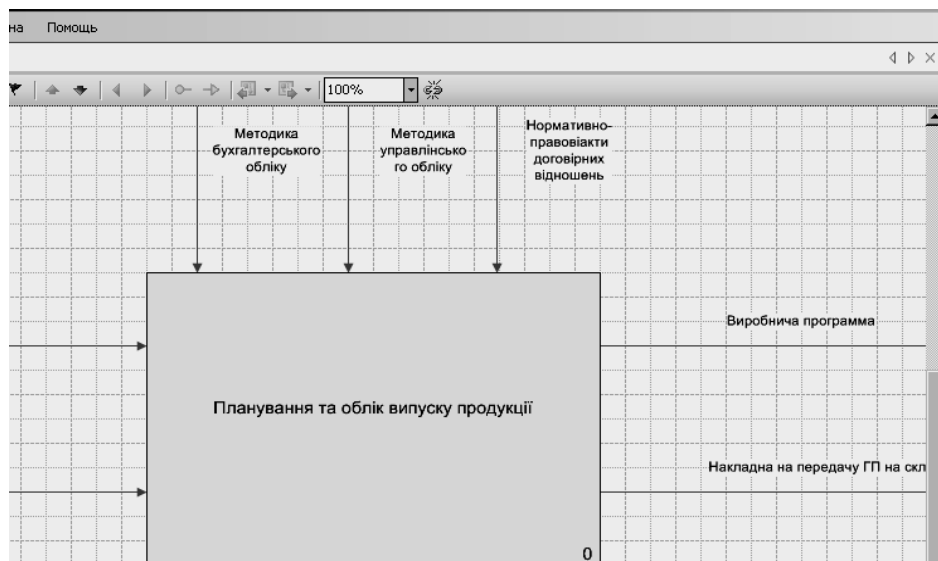


Рис. 4.41. Стрілки керування у верхній грані прямокутника

6. Далі створюємо стрілки механізмів.

Для створення стрілки механізму необхідно: вибрати відповідний елемент у навігаторі, лівою клавшею миші перетягнути його в область діаграми, і верхній кінець автоматично створеної стрілки з'єднати з нижньою гранню прямокутника роботи (рис. 4.42).

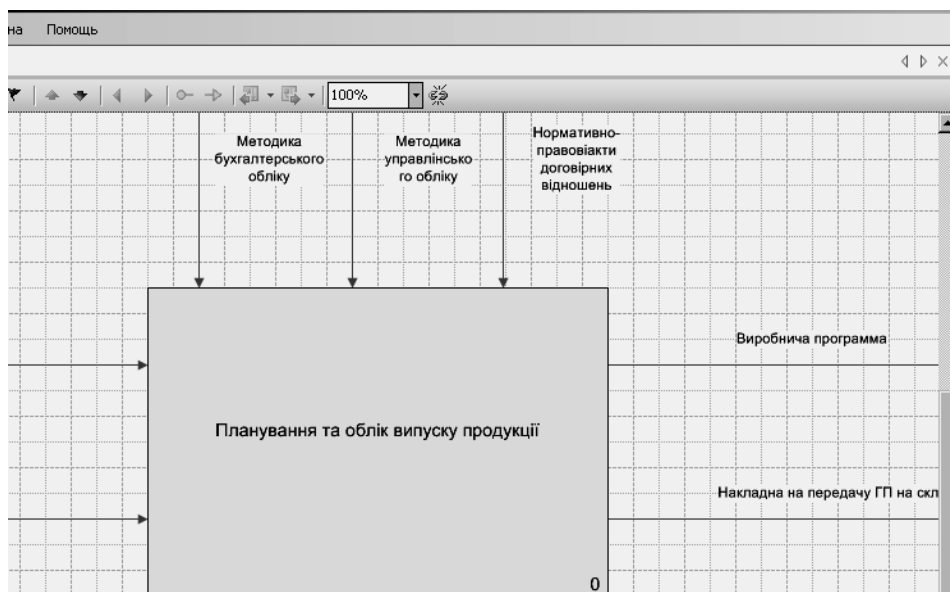


Рис. 4.42. Стрілки механізмів у нижній грані прямокутника

7. На наступному етапі створюємо стрілки виходів.

Для створення стрілки виходу необхідно: вибрати відповідний елемент / у навігаторі, лівою клавішею миші перетягнути його в область діаграми, і лівий кінець автоматично створеної стрілки з'єднати із правою гранню прямокутника роботи (рис. 4.43).

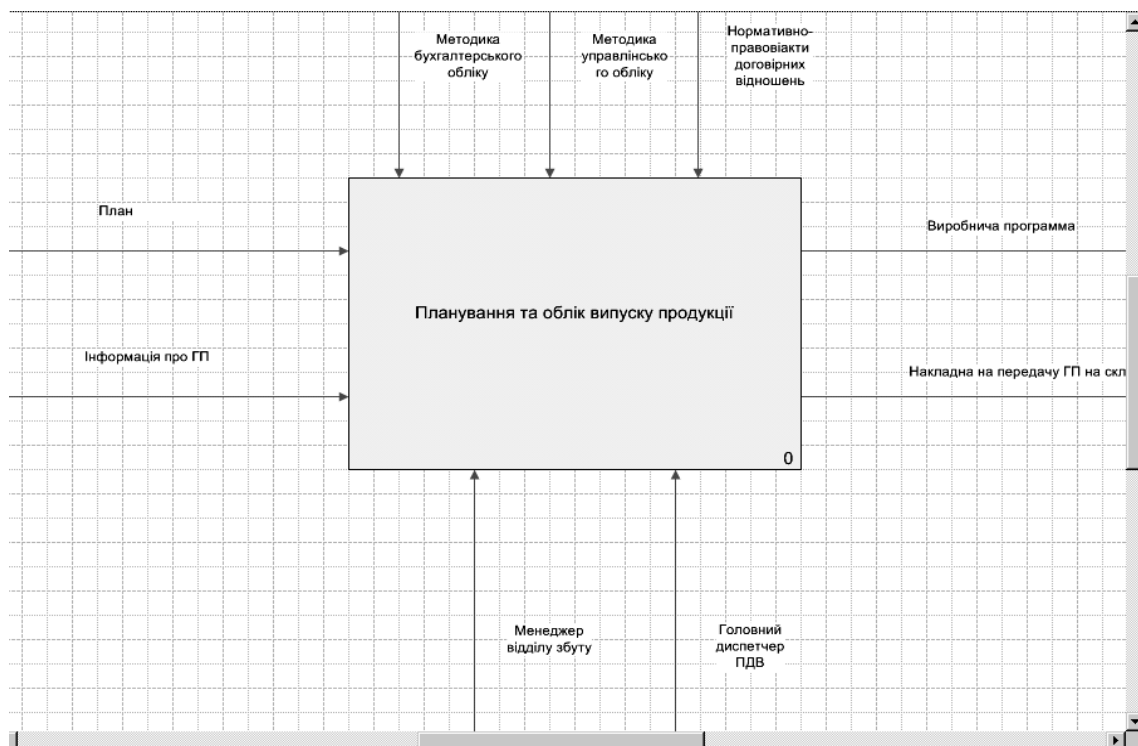


Рис. 4.43. Стрілки виходів у правій грані прямокутника

8. Відповідно до методології структурного підходу виконуємо декомпозицію контекстної діаграми.

Для здійснення декомпозиції необхідно двічі клацнути лівою клавішею миші на піктограмі у панелі інструментів графічного редактора, при цьому, при переході на іншу діаграму буде видаватися попередження (рис. 4.44):

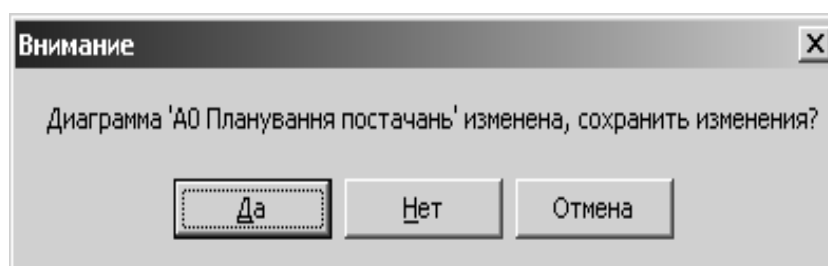


Рис. 4.44. Вікно діалогу

у якому вибрати кнопку "Да". Надалі перед переходом на іншу діаграму варто зберігати всі внесені в неї зміни за допомогою кнопки /.

Отже, одержуємо діаграму декомпозиції з перенесеними з батьківської діаграми стрілками (рис. 4.45).

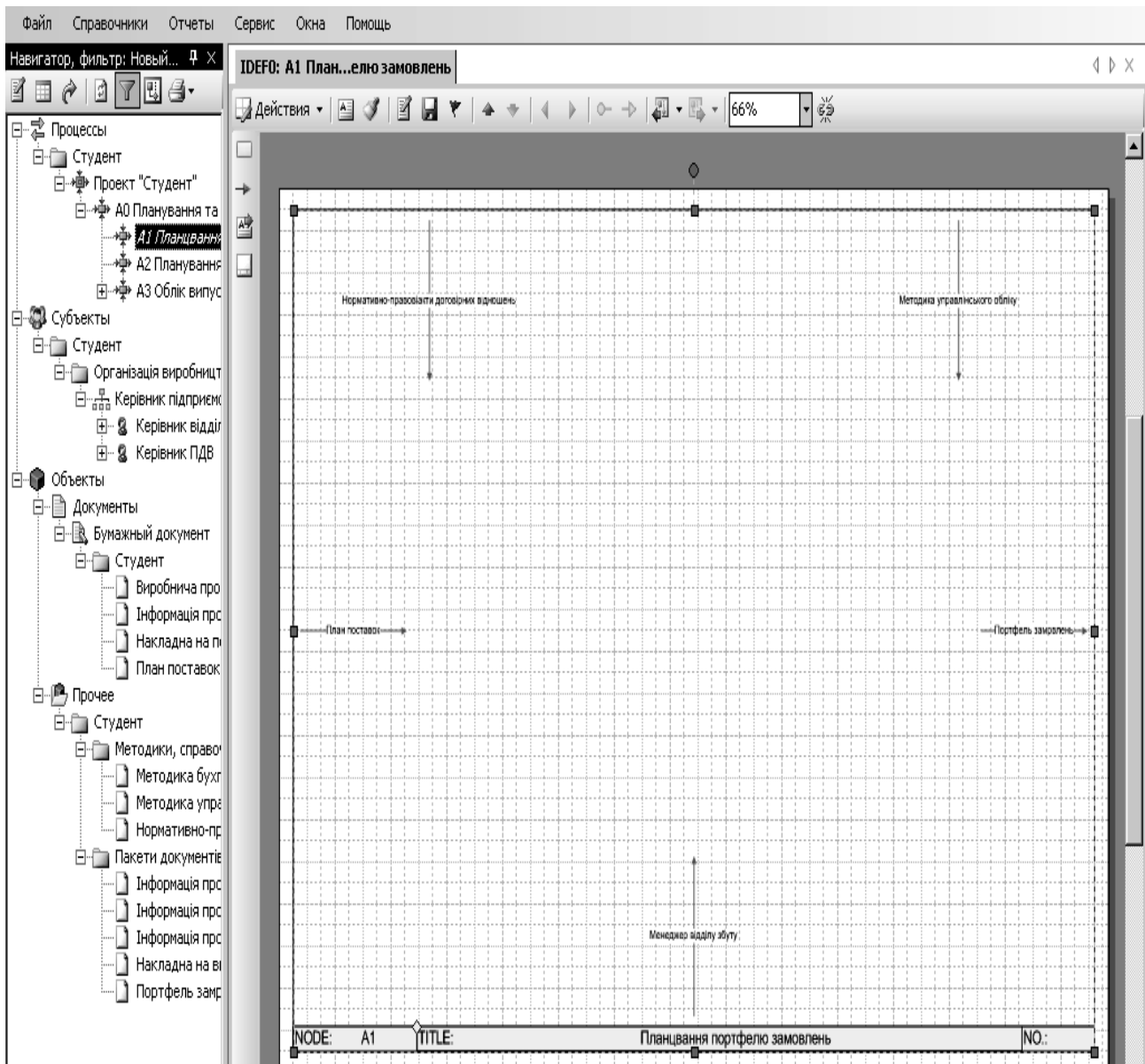


Рис. 4.45. Діаграма декомпозиції з межовими стрілками

## 9. Побудова діаграми 1-го рівня декомпозиції.

Для побудови цієї діаграми на неї необхідно внести об'єкти, що відповідають роботам даного рівня декомпозиції. Після збереження проекту на діаграму, що з'явилася, із лівої частини панелі інструментів перетягуємо необхідну кількість процесів за допомогою кнопки у лівій частині панелі інструментів графічного редактора (рис. 4.46).

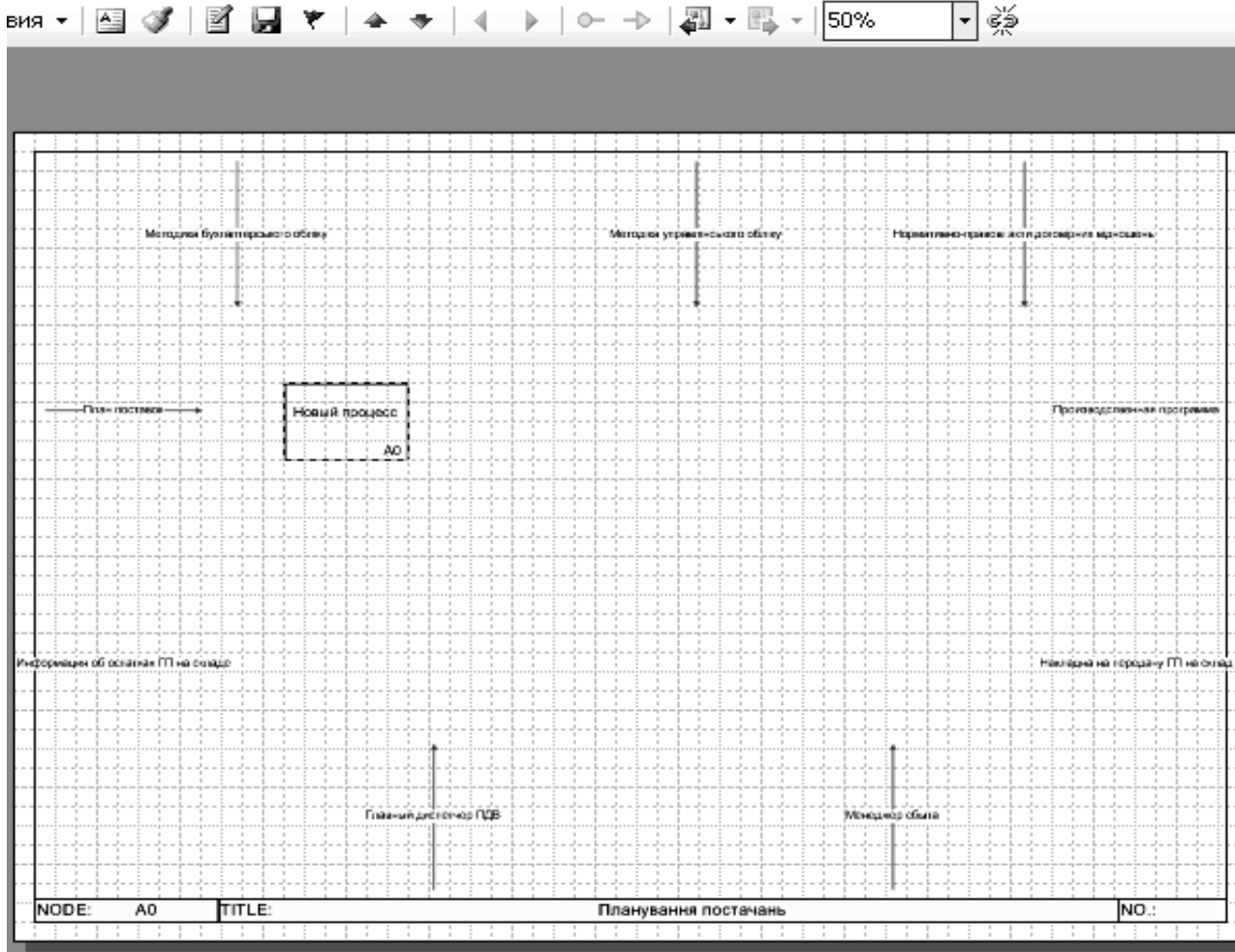


Рис. 4.46. Процес додавання символів "процес" на діаграму декомпозиції

Після додавання необхідної кількості процесів на діаграму, необхідно для кожного процесу задати їхні імена за допомогою подвійного клацання на прямокутнику процесу й введенням імені процесу в поле, що з'явилося, для редагування (рис. 4.47).

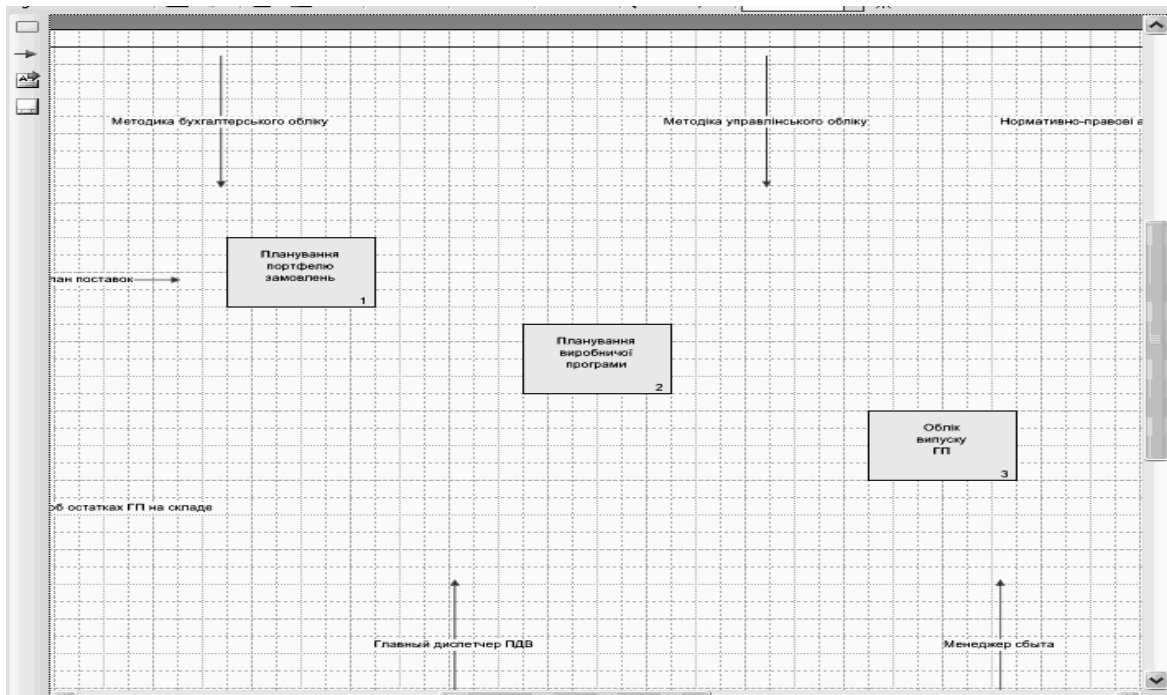


Рис. 4.47. Іменовані процеси на діаграмі декомпозиції

10. На подальшому етапі виконуємо з'єднання граничних стрілок із процесами.

Для цього використовуємо принцип переходу від доміантних блоків до менш доміантних, починаючи із кінця лівого блоку й переходячи послідовно до інших блоків на поточній діаграмі, та "обв'язуємо" усіма типами стрілок кожен із них. Приєднуємо стрілки входів до елементів діаграми (рис. 4.48).

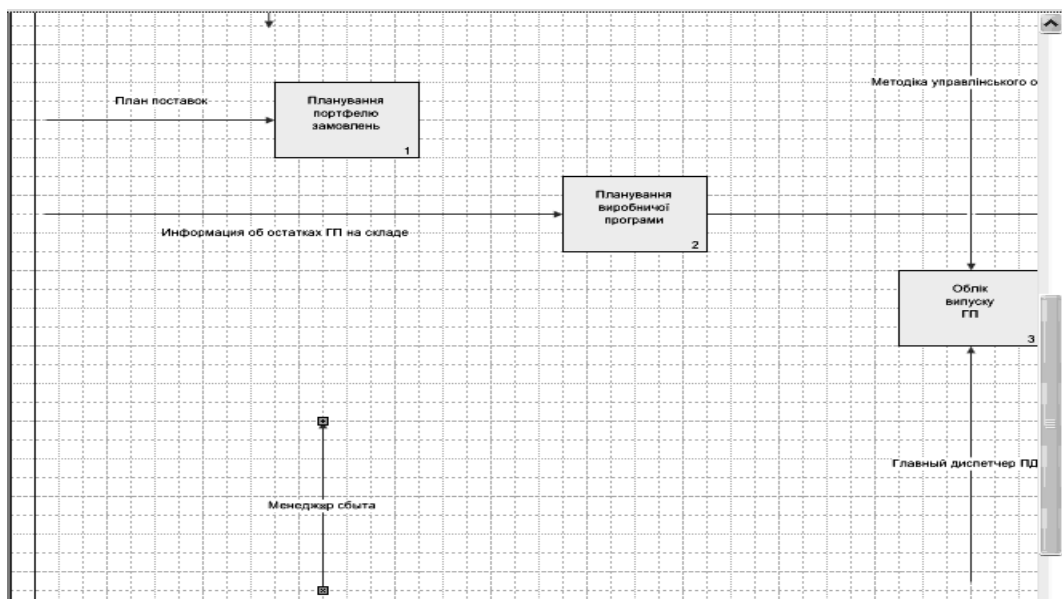


Рис. 4.48. Приєднання стрілок входів



Далі приєднуємо та називаємо стрілки механізмів (рис. 4.49 – 4.56).

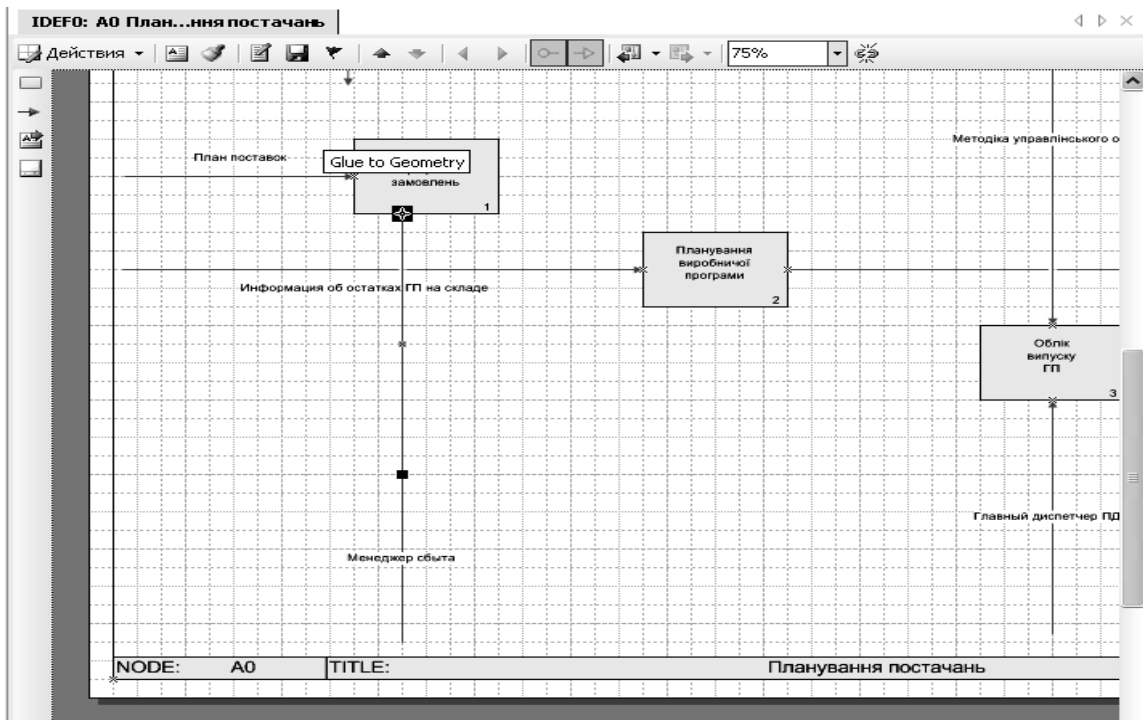


Рис. 4.49. Приєднання стрілки механізму "Менеджер збуту" до нижньої грані роботи

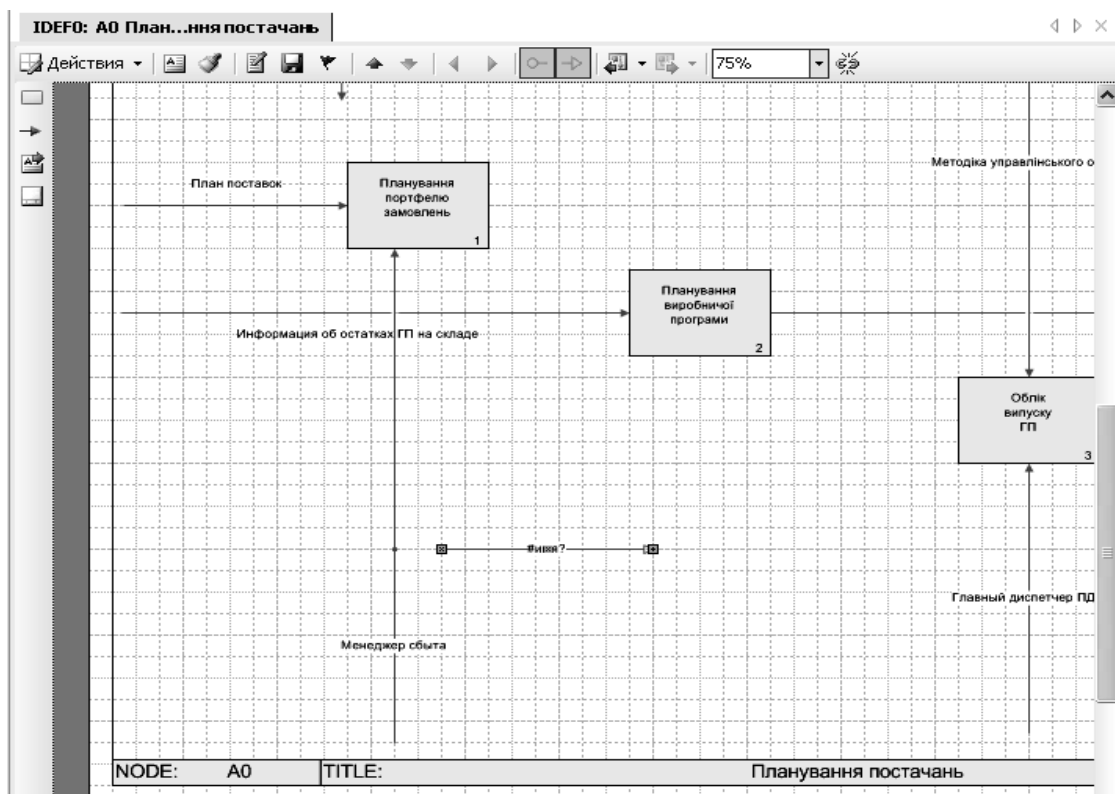


Рис. 4.50. Створення не названої стрілки на діаграмі за допомогою перетягання її з панелі інструментів

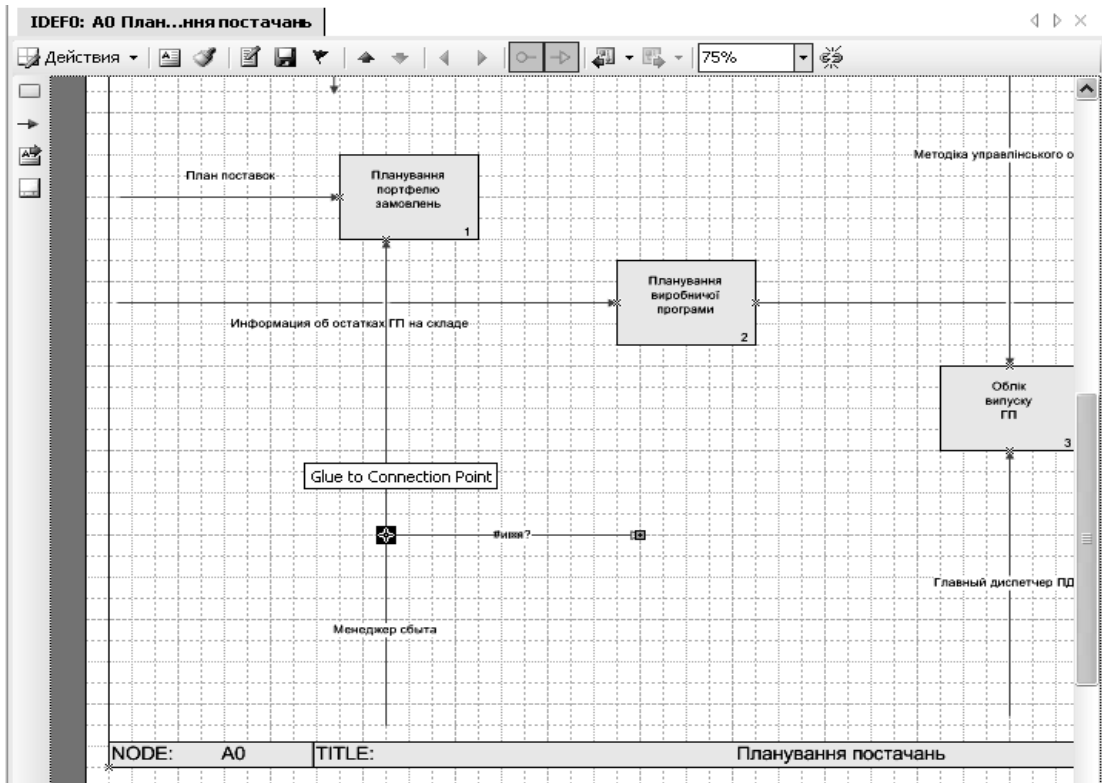


Рис. 4.51. З'єднання лівого кінця стрілки з лінією стрілки механізму "Менеджер збуту"

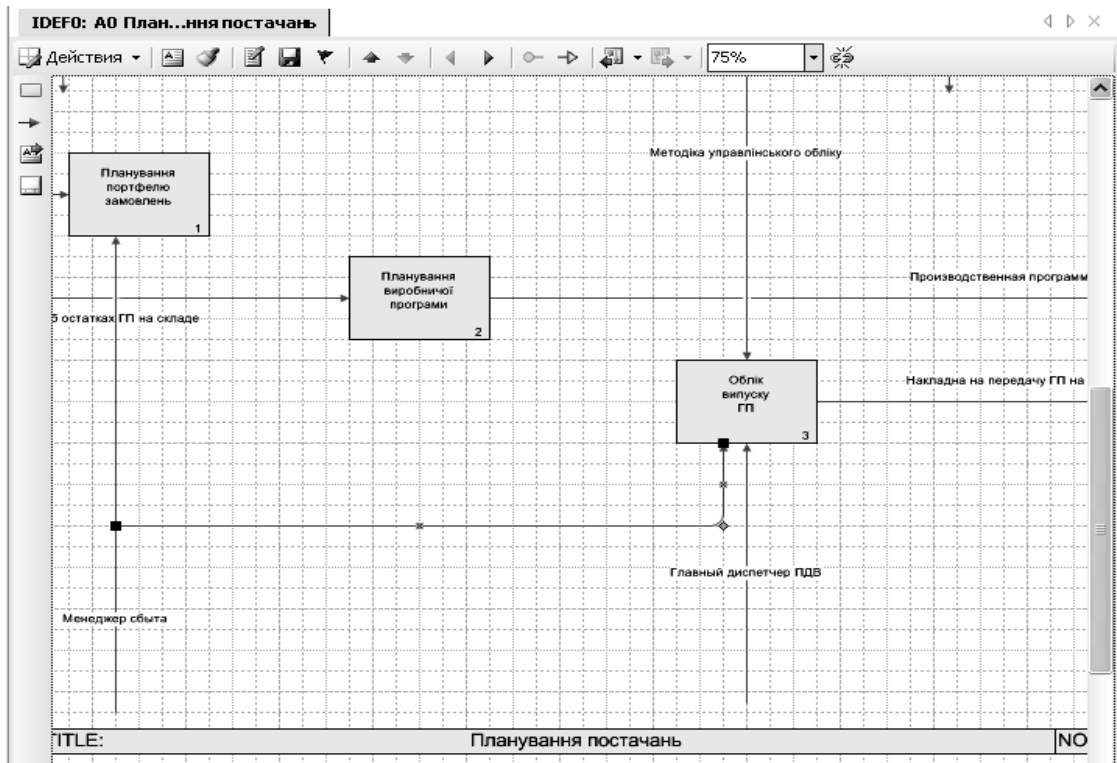


Рис. 4.52. З'єднання правого кінця стрілки з нижньою гранню іншої роботи

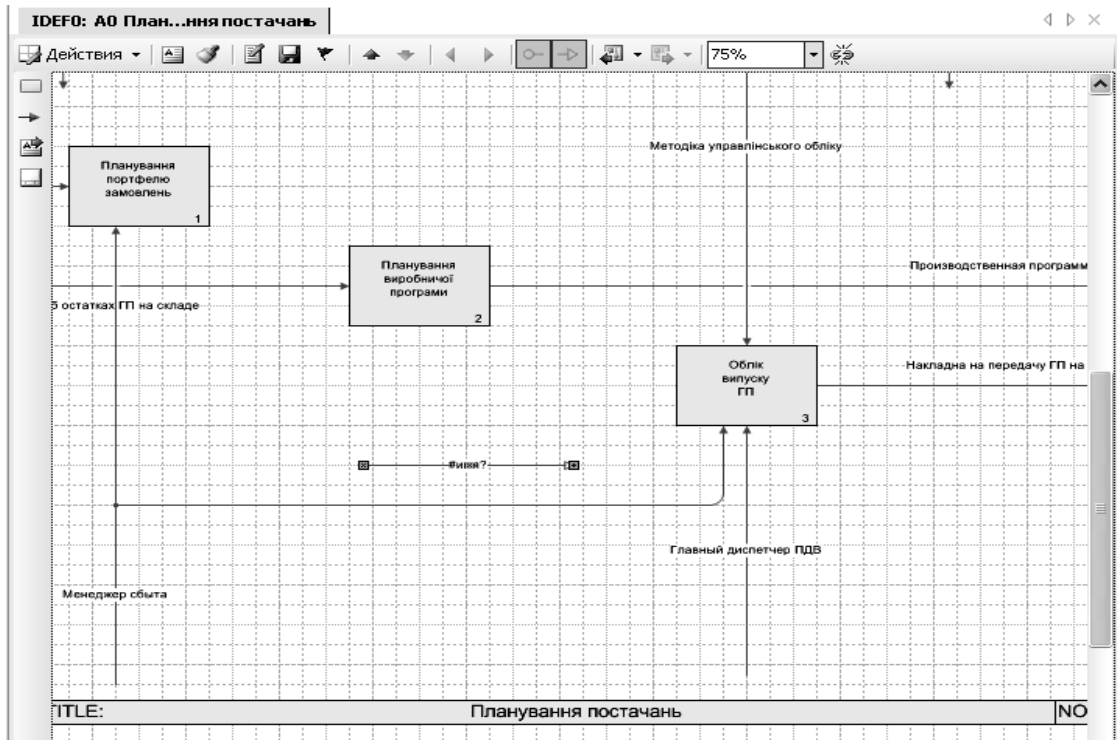


Рис. 4.53. Створення не названої стрілки на діаграмі перетяганням її з панелі інструментів

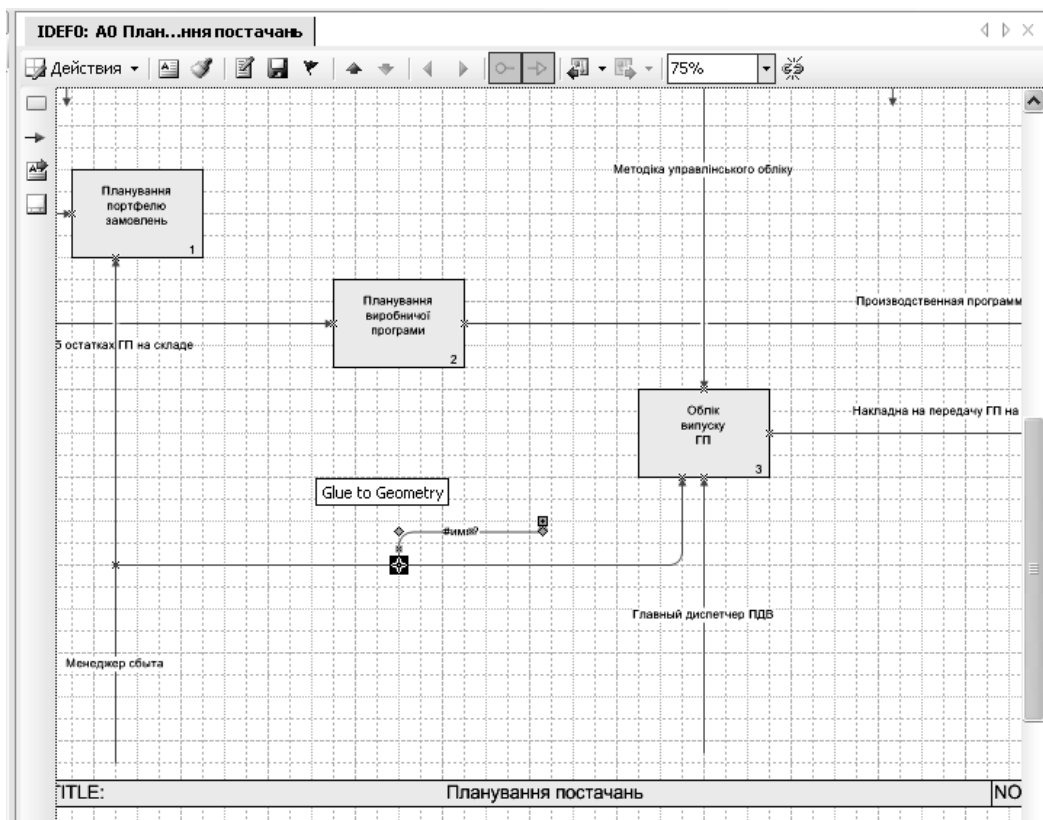


Рис. 4.54. З'єднання лівого кінця стрілки з лінією стрілки механізму "Менеджер збути"

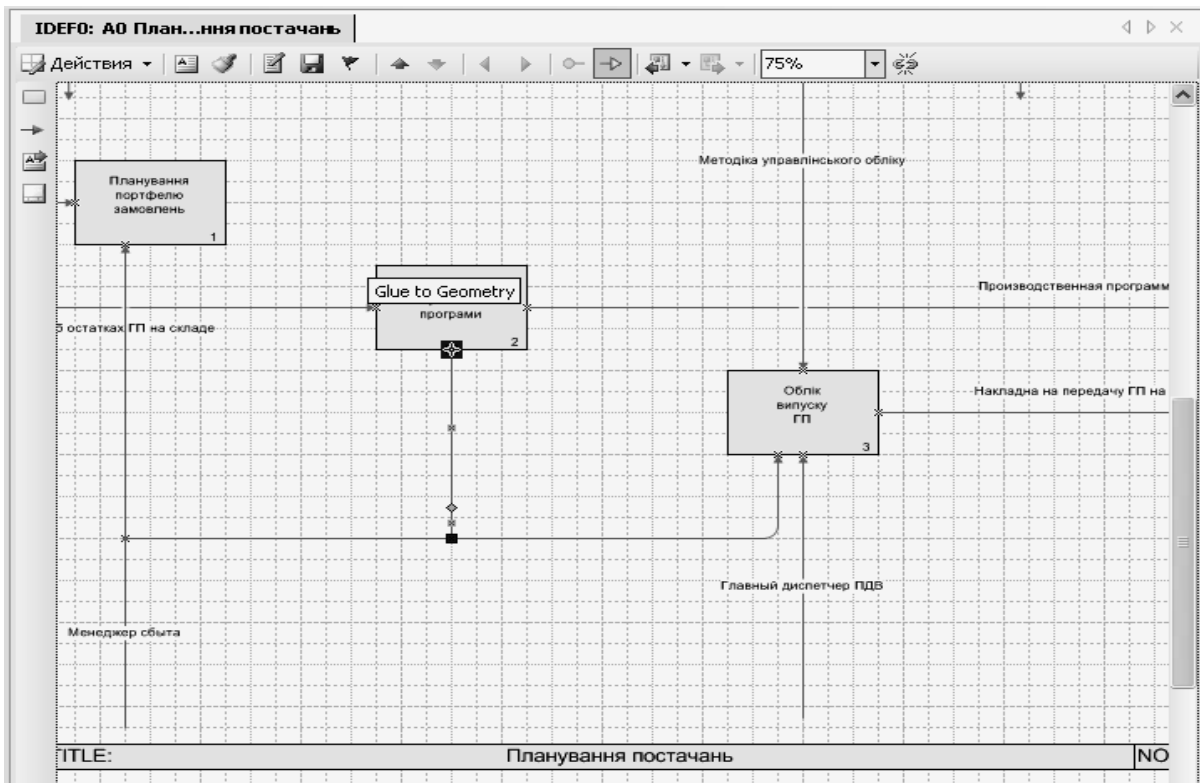


Рис. 4.55. З'єднання правого кінця стрілки з нижньою межею середньої роботи

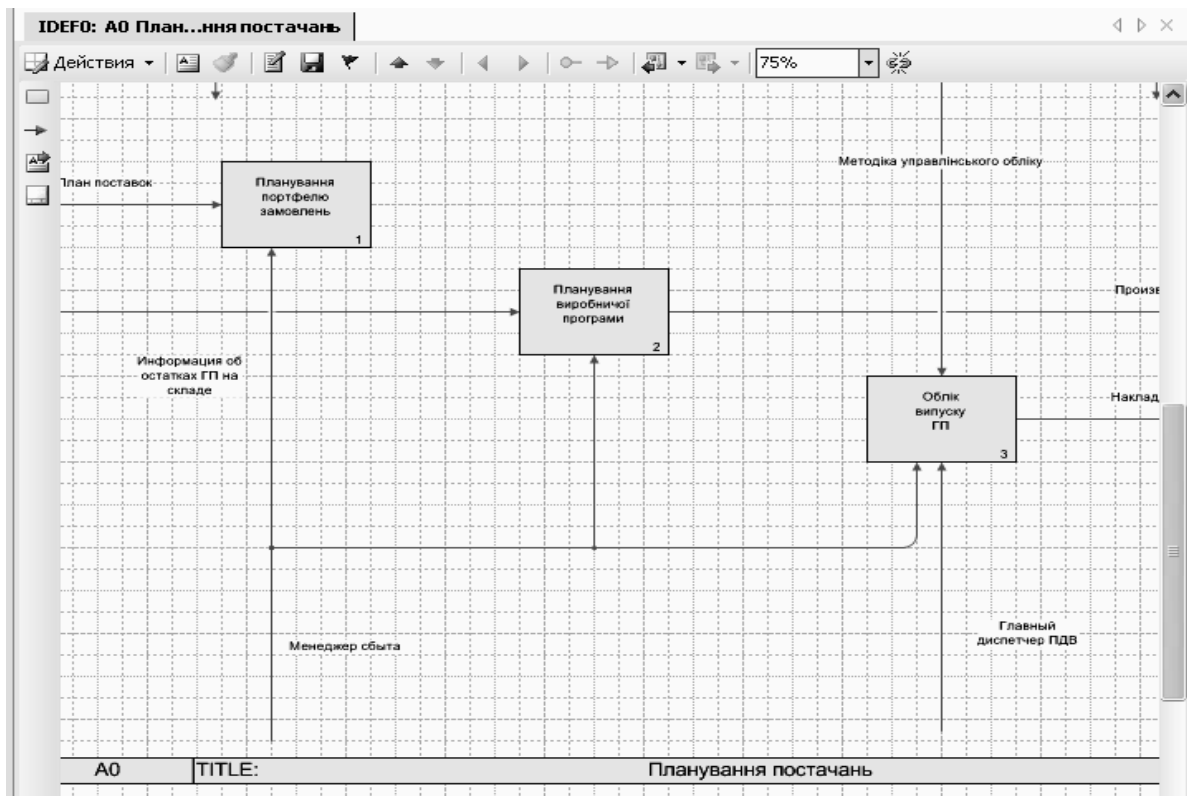


Рис. 4.56. Підключення стрілок механізмів до нижніх меж робіт

Аналогічно розгалужуємо стрілки керування (рис. 4.57).

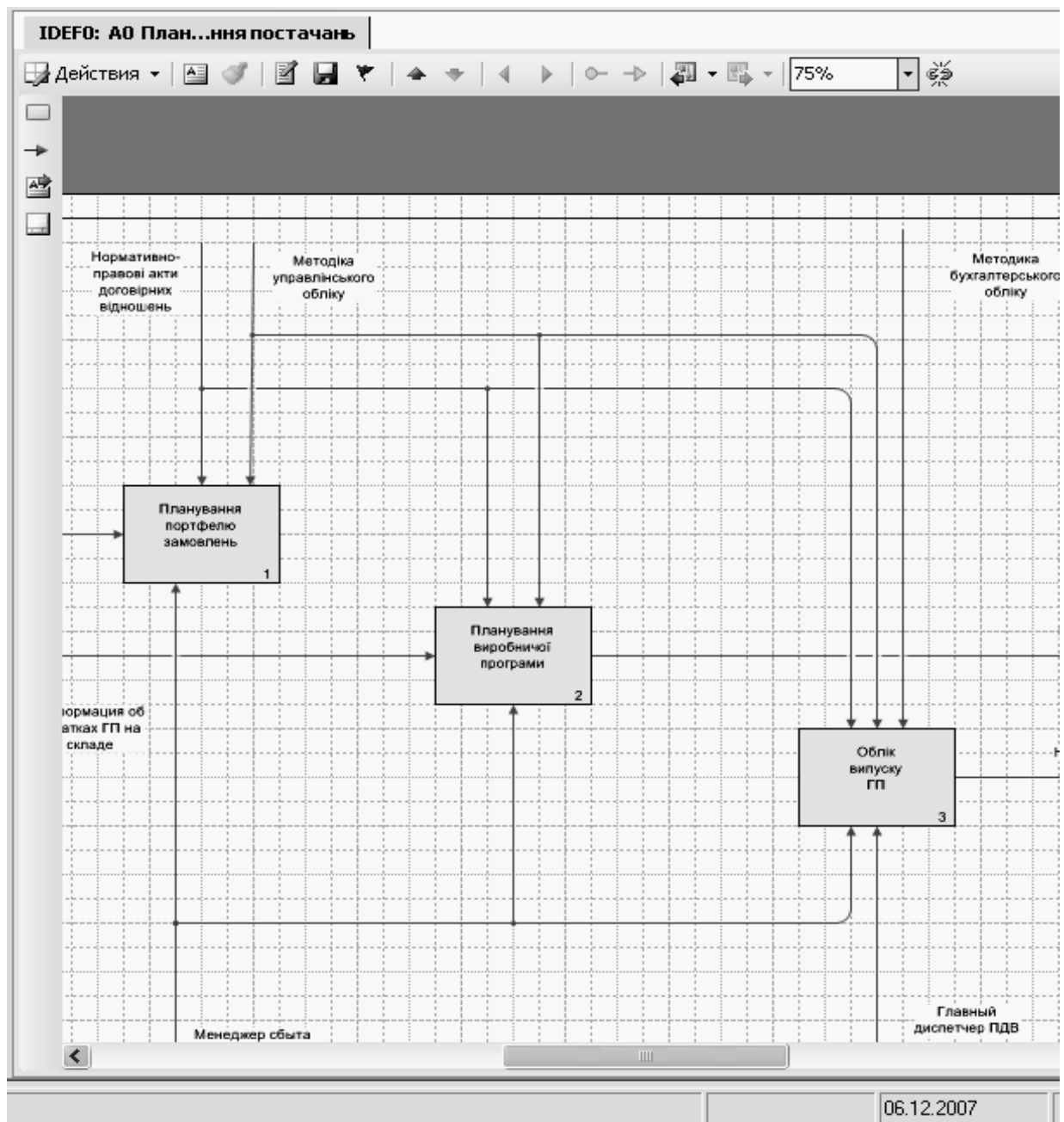
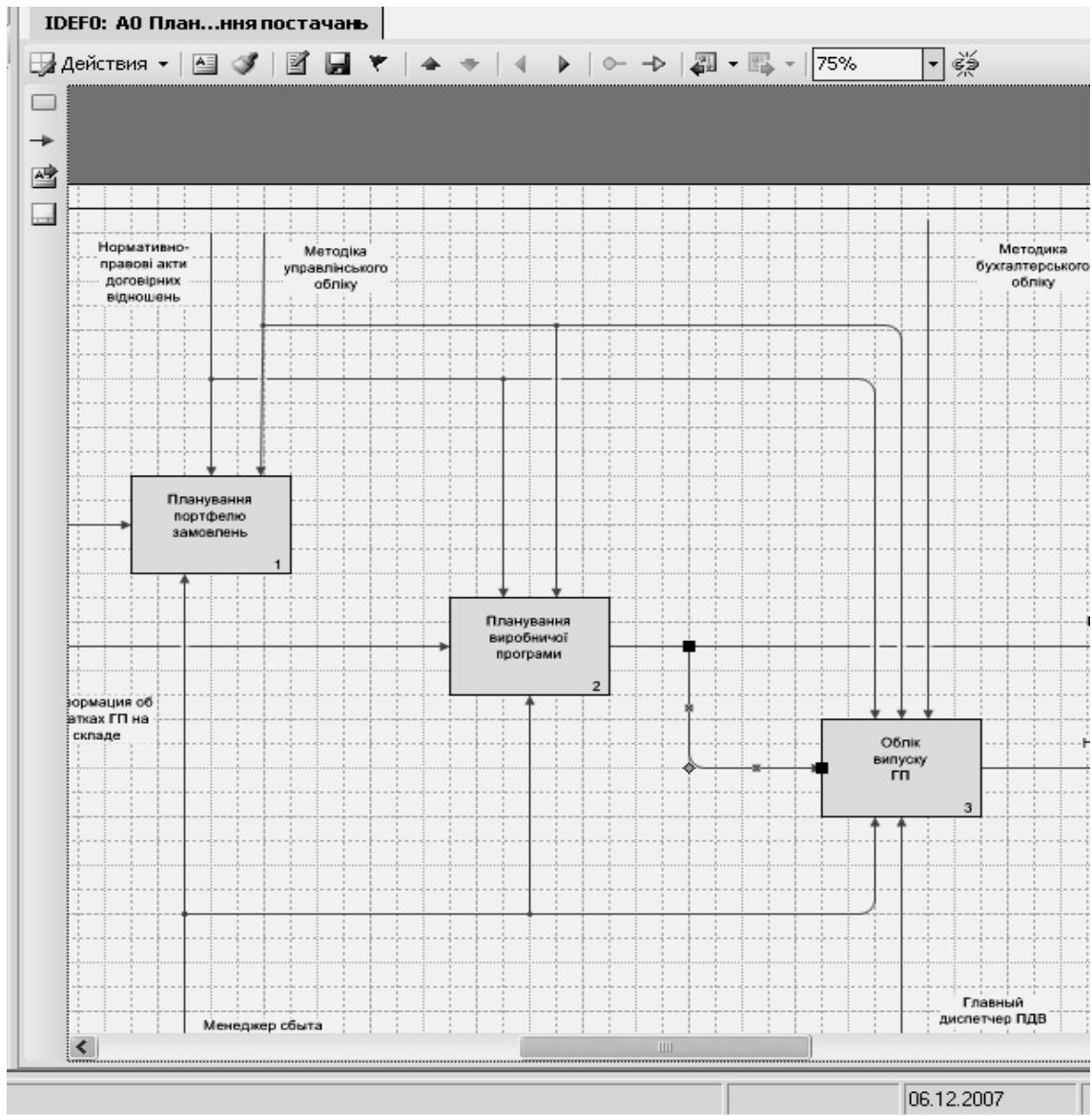


Рис. 4.57. Підключення стрілки керування до верхніх меж робіт

11. Далі створюємо внутрішні стрілки.

Створюємо неіменовану стрілку шляхом перетягання її з панелі інструментів на діаграму графічного редактора. Лівий кінець стрілки з'єднуємо з лінією стрілки виходу "Виробнича програма", правий кінець – із лівою гранню роботи 3 (рис. 4.58).



**Рис. 4.58. Створення внутрішньої стрілки за допомогою неназваної стрілки**

Після "обв'язування" створюємо міжблочні стрілки. Для цього в навігаторі, у розділі "Прочее" створюємо папку зі своїм прізвищем, і за допомогою пункту "Добавить от текущего" викликуваного контекстного меню папки створюємо папку "Пакеты документов", у якій створюємо елемент "Портфель заказов" і на його основі створюємо стрілку на діаграмі. З її допомогою з'єднуємо роботи 1 і 2. Роботи 2 і 3 з'єднуються іншим способом: у панелі інструментів графічного редактора ставимо курсор миші на значок і перетягаємо його на діаграму, після чого лівий кінець стрілки з'єднуємо з лінією стрілки "Виробнича програма", а правий – з лівою гранню наступної роботи (рис. 4.59 – 4.61).

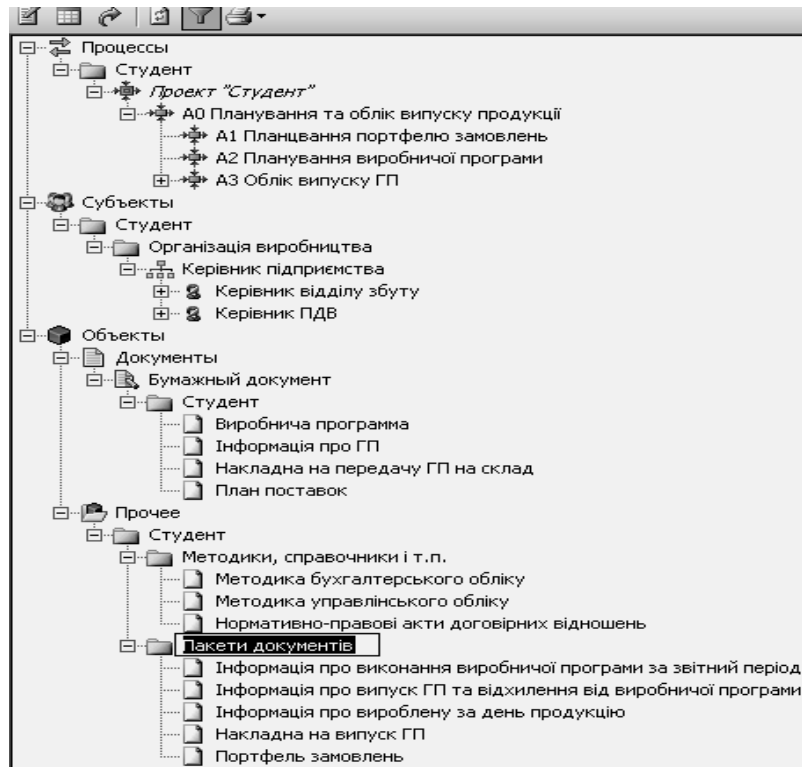


Рис. 4.59. Перейменування папки



Рис. 4.60. Елемент "Портфель заказов" створений у папці "Пакеты документов"

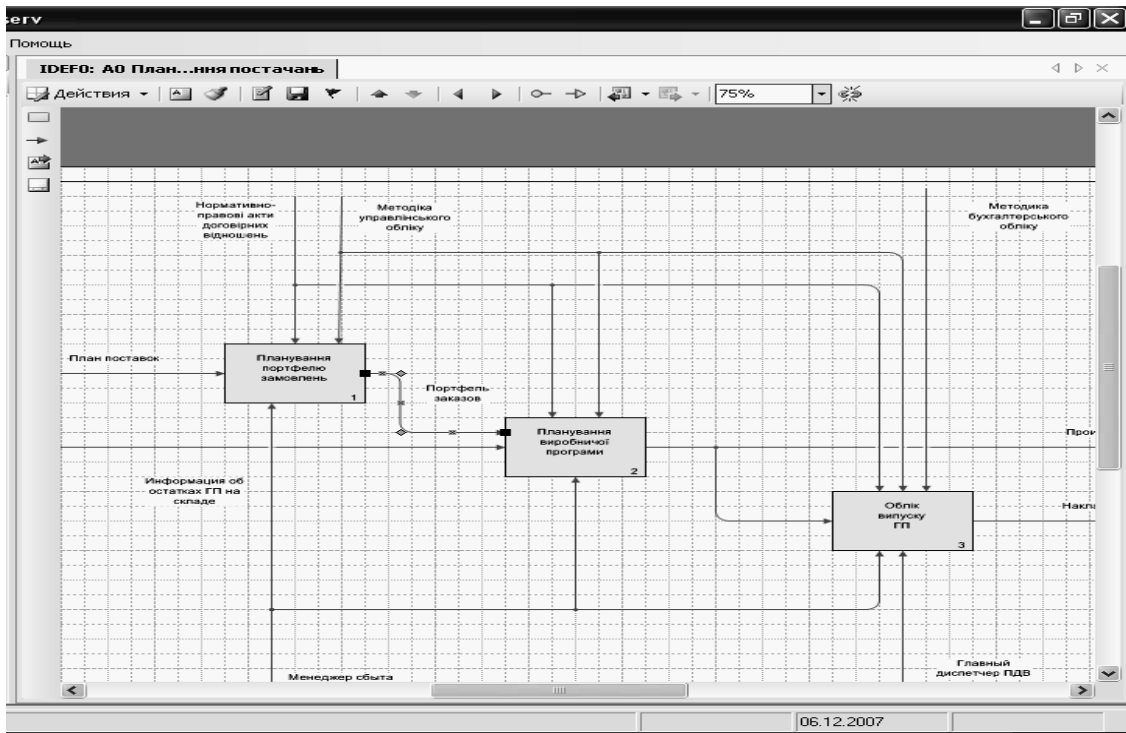


Рис. 4.61. Створення стрілки "Портфель заказов"

Для відображення зворотного зв'язку між блоками 3 і 2 створюємо стрілку зворотного зв'язку. Для цього в навігаторі, у розділі "Паперові документи" у своїй папці створюємо елемент "Інформація про випуск ГП та відхилення від виробничої програми" і перетягуємо його в область діаграми. Лівий кінець стрілки з'єднуємо із правою гранню роботи 3, правий – з лівою гранню блоку більш високої домінанти (роботою 2) (рис. 4.62).

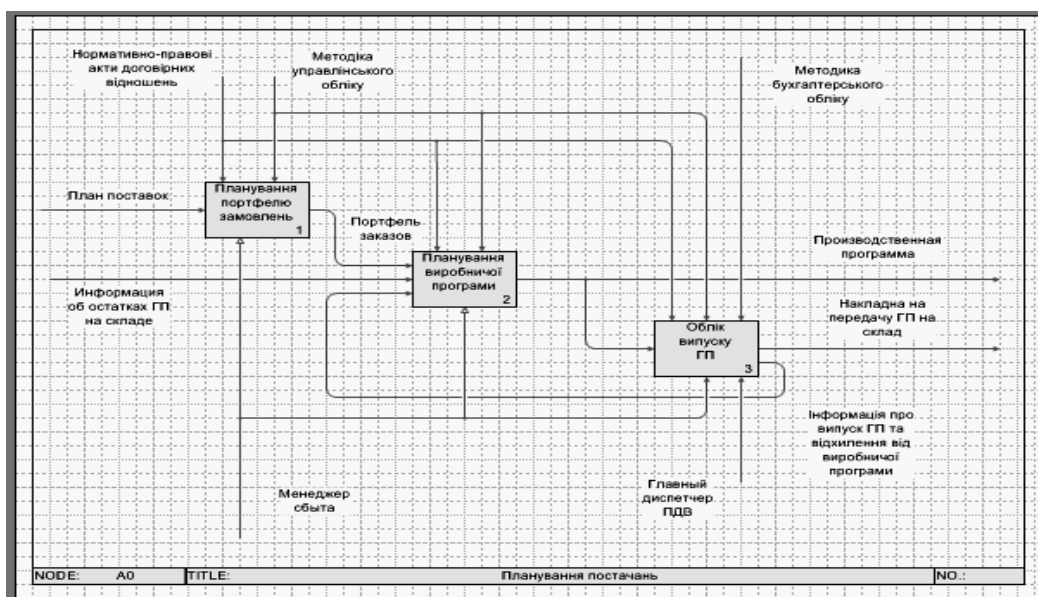




Рис. 4.62. Стрілка зворотного зв'язку між роботами 2 і 3  
12. Технологія тунелювання стрілок механізмів.

У тих випадках, коли відомо, що всі роботи на дочірній діаграмі виконуються тим самим виконавцем (механізмом), то доцільно стрілку цього механізму не відображати на дочірній діаграмі. Для цього її необхідно зробити тунельною за типом "не в дочерней диаграмме": виділити стрілку за допомогою лівою клавiшею миші на діаграмі, та за допомогою кнопки зі значком / виконати тунелювання (рис. 4.63).

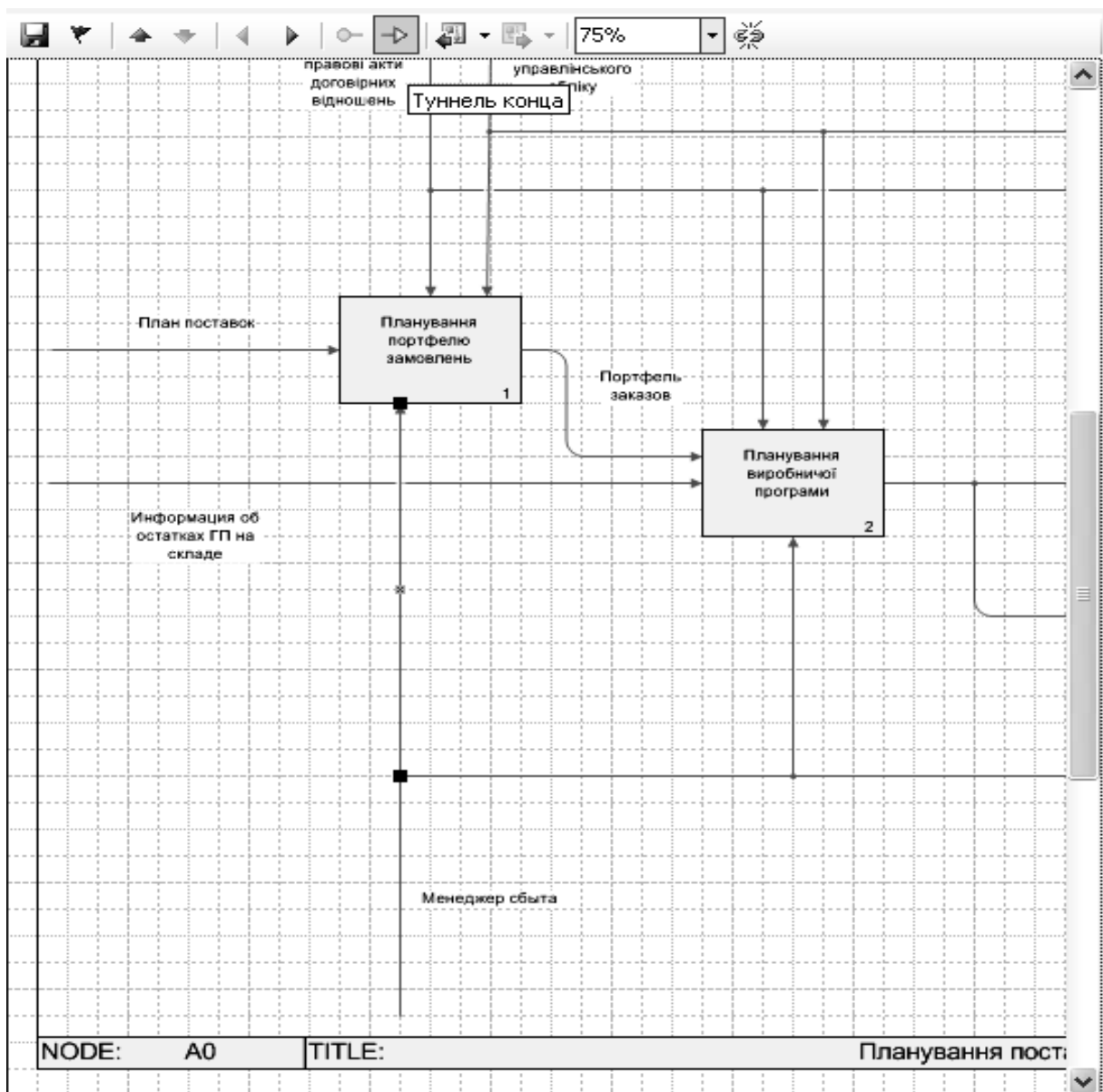


Рис. 4.63. Тонелювання стрілки механізму

Після цього вістря стрілки придбає форму світлого трикутника (рис. 4.64, 4.65).

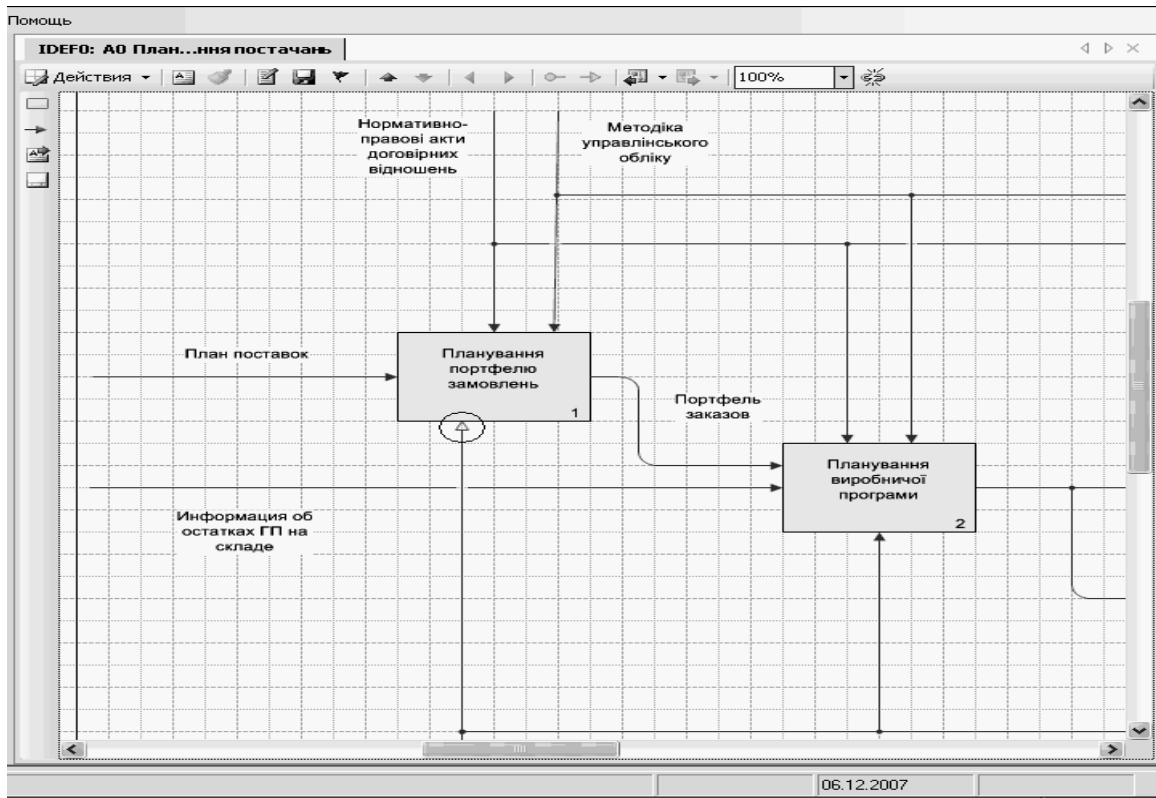
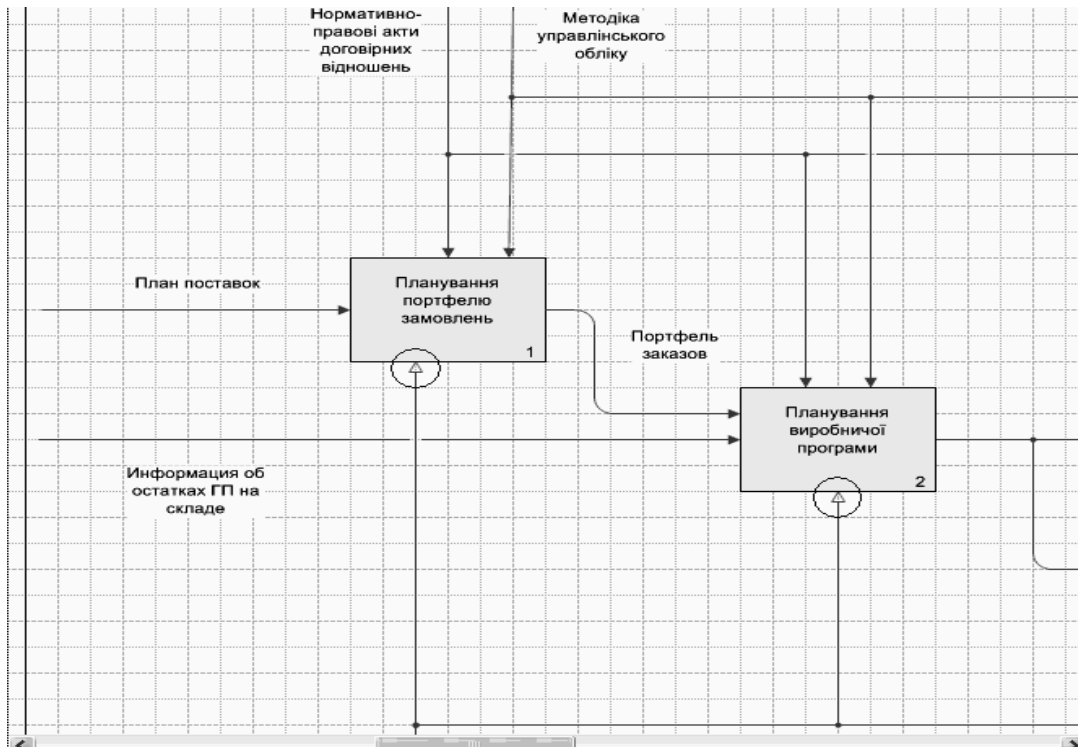


Рис. 4.64. Тонельований сегмент розгалуженої стрілки механізму для роботи 1



## Рис. 4.65. Тонельовані сегменти розгалуженої стрілки механізму для робіт 1, 2

Далі декомпозиємо роботу "Планування виробничої програми".

Відкорегуйте діаграму декомпозиції так, щоб одержати її зміст, аналогічний діаграмі у пакеті BPWin (рис. 4.66).

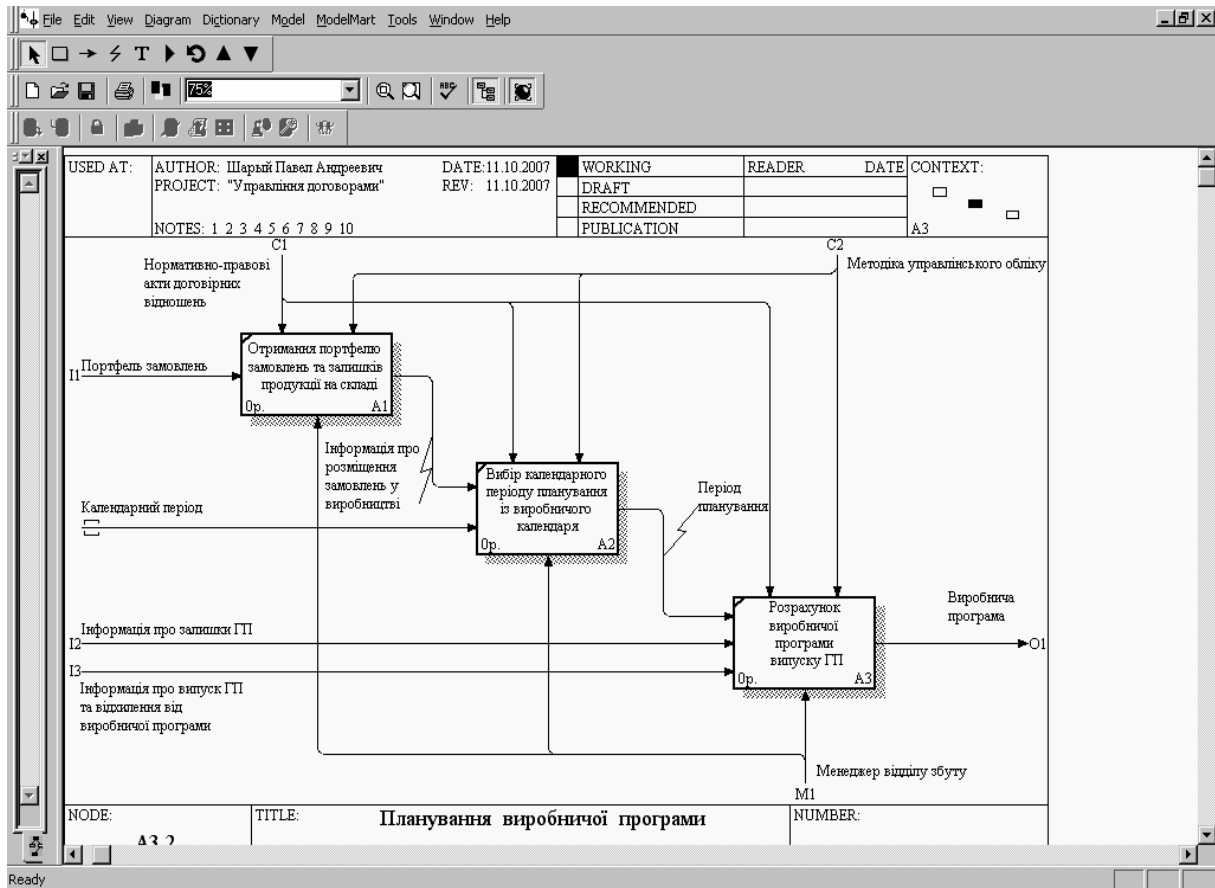
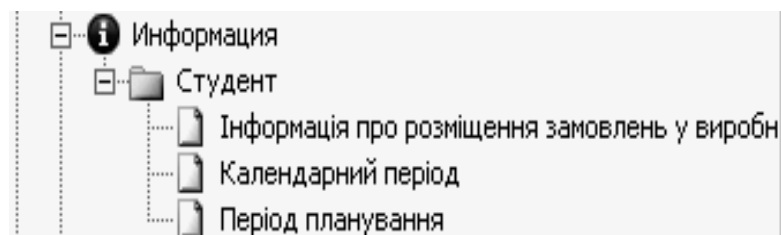


Рис. 4.66. Декомпозиція роботи "Планування виробничої програми" у пакеті BPWin

Спочатку в навігаторі об'єктів треба заповнити розділ "Інформація" (рис. 4.67). Для цього необхідно створити папку і за допомогою контекстного меню папки обрати пункт "Додати з поточного", та створити наступні елементи (рис. 4.68).



## Рис. 4.67. Зміст розділу "Інформація"

Далі декомпозиємо роботу "Планування виробничої програми" так, як показано на рис. 4.68.

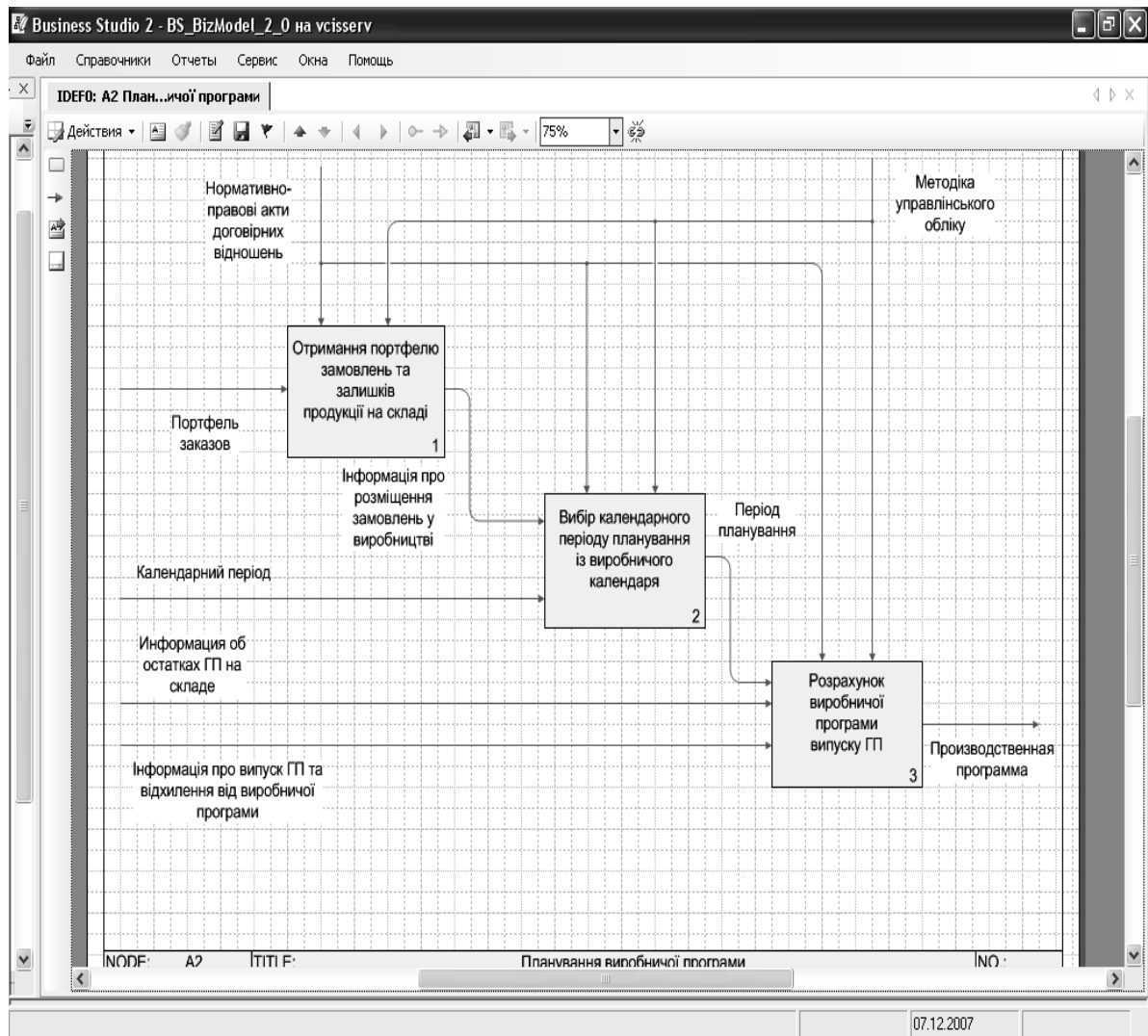


Рис. 4.68. Декомпозиція роботи "Планування виробничої програми" у пакеті Business Studio

Отримали, що на наведеній вище діаграмі стрілки механізмів, тунельовані на батьківській діаграмі, відсутні.

### 13. Метод тунелювання "не в батьківській діаграмі".

Для того, щоб не показувати певні стрілки, при їхньому введенні на дочірній діаграмі, також необхідно виконати операцію тунелювання. Для цього необхідно виділити стрілку за допомогою лівої клавіші миші на

діаграмі, й за допомогою кнопки зі значком / виконати тунелювання. Після цього вістря стрілки придбає форму світлого кружечка (рис. 4.69).

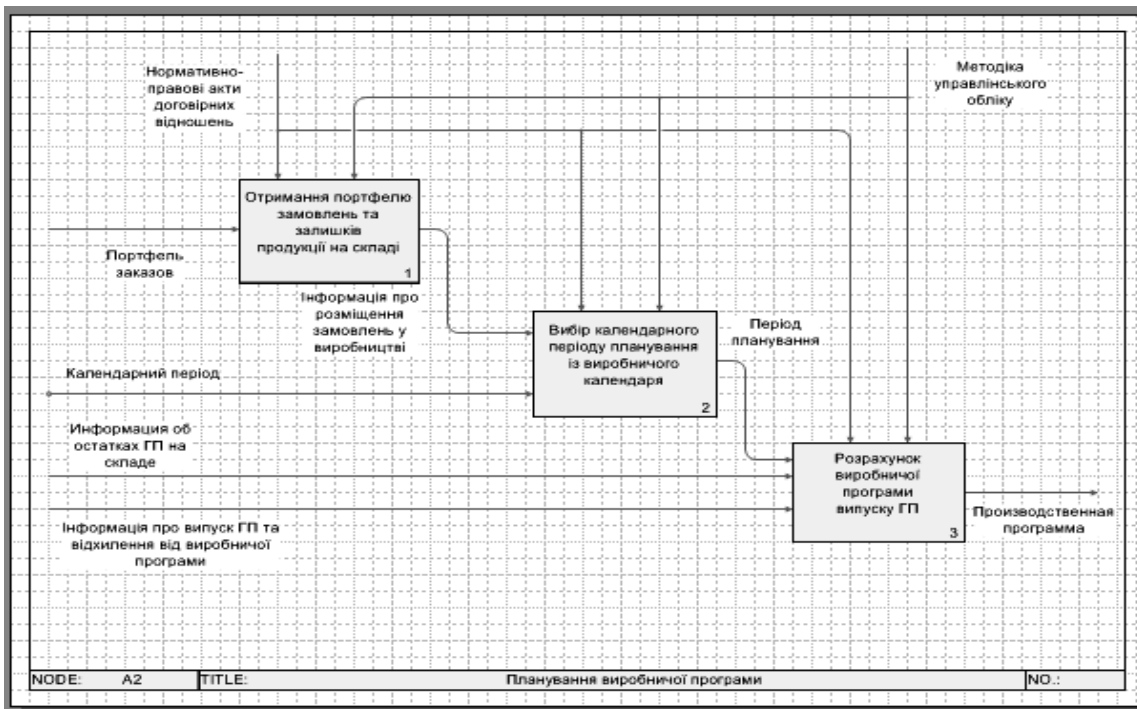
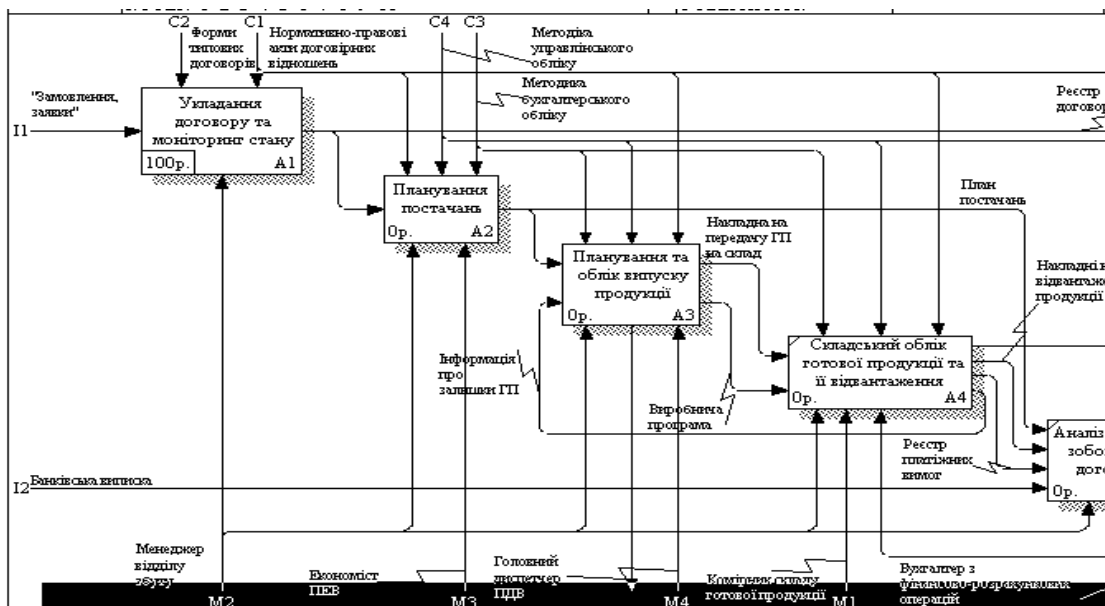


Рис. 4.69. Тонельована стрілка "календарний період"

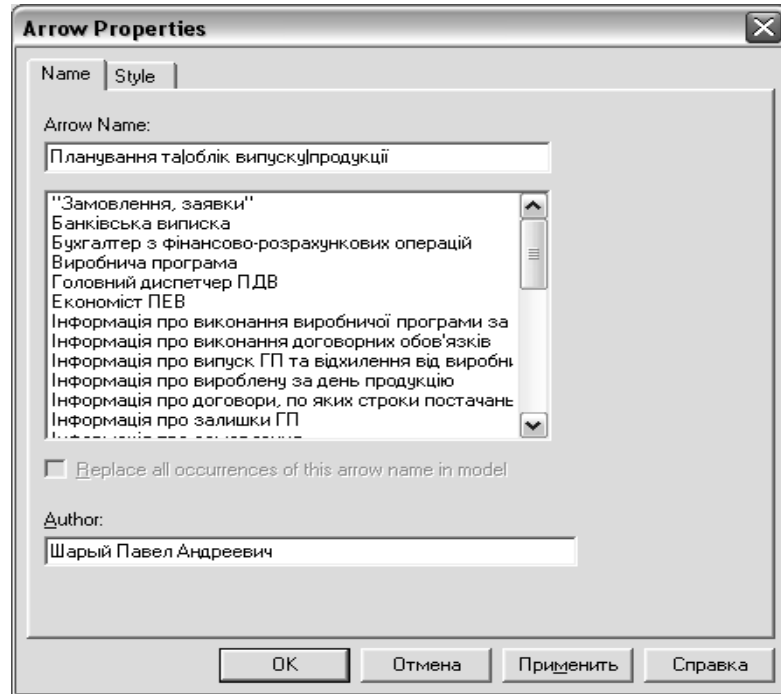
#### 14. Створення діаграми декомпозиції.

Для цього треба знайти певне завдання на діаграмі декомпозиції у пакеті BPWin бізнес-процесу "Керування договорами", знайти батьківську роботу та визначити її в навігаторі робіт. Потім треба перейти в область діаграми й створити стрілку зовнішнього виклику Call для цієї роботи (рис. 4.70).



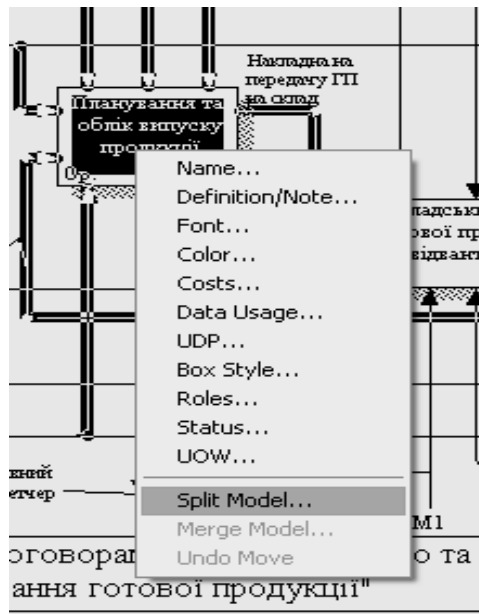
**Рис. 4.70. Створення стрілки виклику роботи  
"Планування та облік випуску продукції"**

Далі треба назвати цю стрілку аналогічно імені роботи (рис. 4.71).



**Рис. 4.71. Завдання імені стрілки виклику**

Далі треба викликати контекстне меню роботи на діаграмі та вибрати в ньому пункт Split Model (рис. 4.72).



**Рис. 4.72. Контекстне меню роботи "Планування та облік випуску продукції"**

У вікні діалогу, що з'явилося, треба натиснути на кнопку "OK" (рис. 4.73).

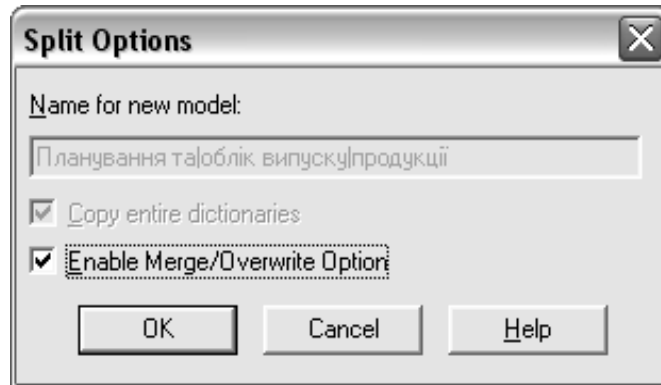


Рис. 4.73. Ім'я відщепленої моделі

У результаті розглянута робота разом з усіма вкладеними в ній роботами буде від'єднана від загального дерева робіт, і на її основі буде створена нова модель із ім'ям, аналогічним імені цієї роботи (рис. 4.74).

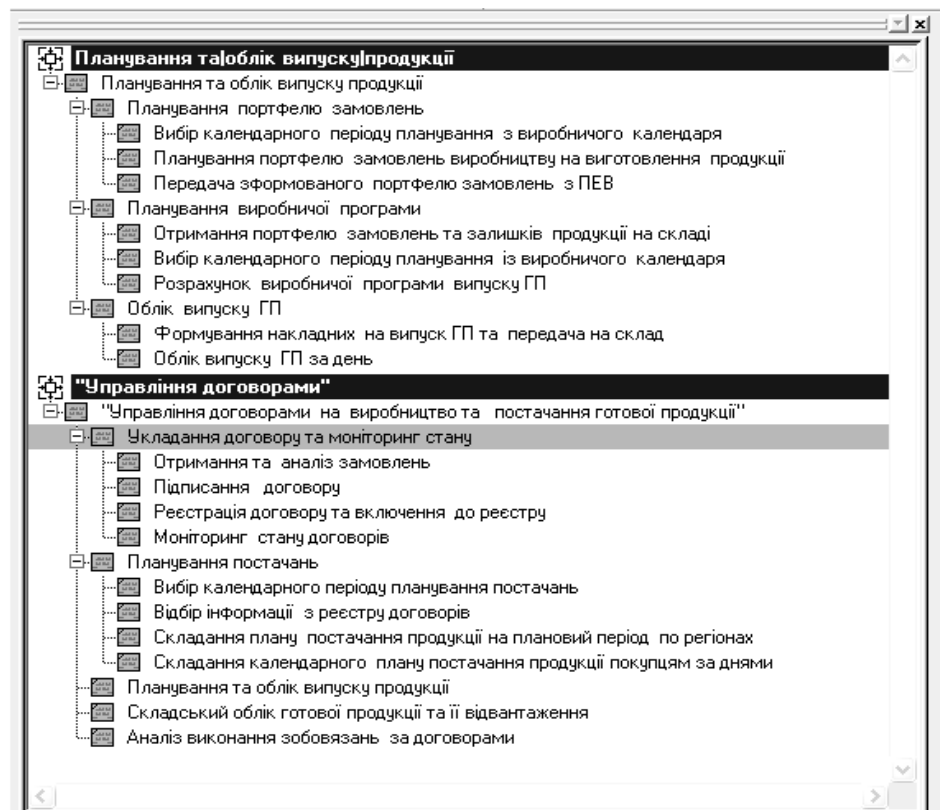


Рис. 4.74. Вихідна модель "Управління договорами" і відщеплена модель "Планування та облік випуску продукції" в Model Explorer

На основі відщепленої моделі "Планування та облік випуску продукції" треба побудувати ідентичну їй модель в Business Studio.

#### 15. Створення звітів.

Щоб створити звіт про організаційну структуру необхідно відзначити вашу папку в розділі "Субъекты" у навігаторі об'єктів, викликати її контекстне меню клацанням лівою клавшею миші, та вибрати меню "Отчёты", і в ньому виконати команду "Показать орг. структуру".

Щоб створити звіт "Регламент процесса IDEF0", необхідно відзначити модель у навігаторі об'єктів, викликати її контекстне меню клацанням лівою клавшею миші, вибрати меню "Звіти", і в ньому виконати команду "Регламент процесса IDEF0".

Щоб створити звіт "Процессы модели", треба обрати команду "Процессы модели".

Щоб створити звіт "Стрелки модели" треба обрати команду "Стрелки модели".

Щоб створити ряд файлів Microsoft Visio, що містять діаграми моделі, необхідно виконати команду "Печатать диаграмму и все вложенные". У результаті з'явиться вікно діалогу, в якому потрібно зробити установки так, як показано на рис. 4.75.

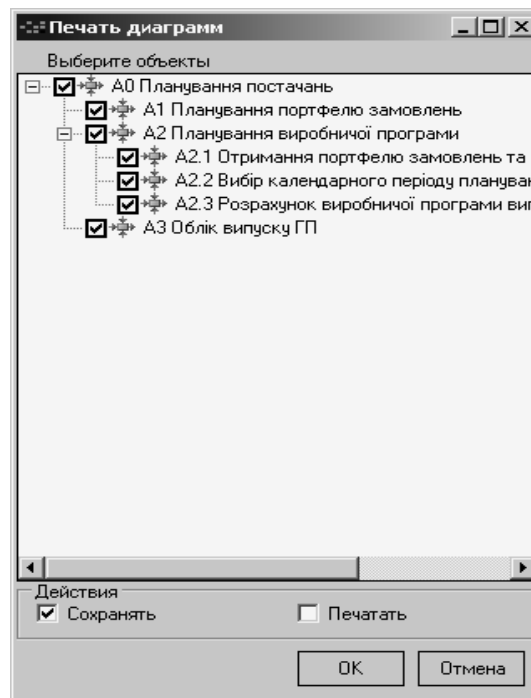


Рис. 4.75. Настроювання складу діаграм і їхнього збереження у файлах



У цьому вікні треба натиснути на кнопку "ОК", у результаті чого відкриється діалогове вікно "Обзор папок", у якому необхідно вибрати існуючу або створити нову папку, у якій будуть збережені файли Microsoft Visio (рис. 4.76).

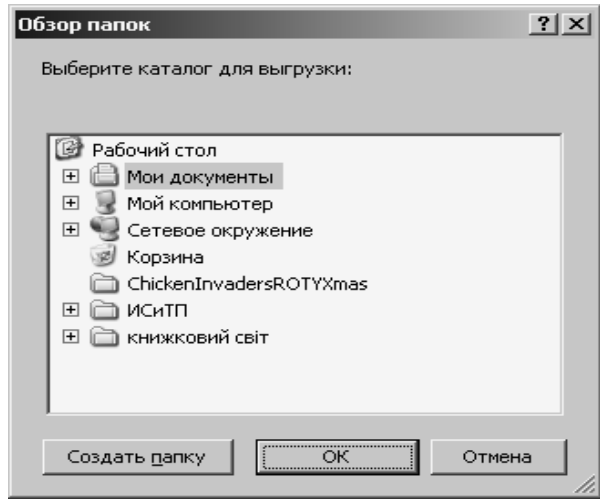


Рис. 4.76. Діалогове вікно "Выбор папки"

У результаті будуть створені файли Microsoft Visio (рис. 4.77).

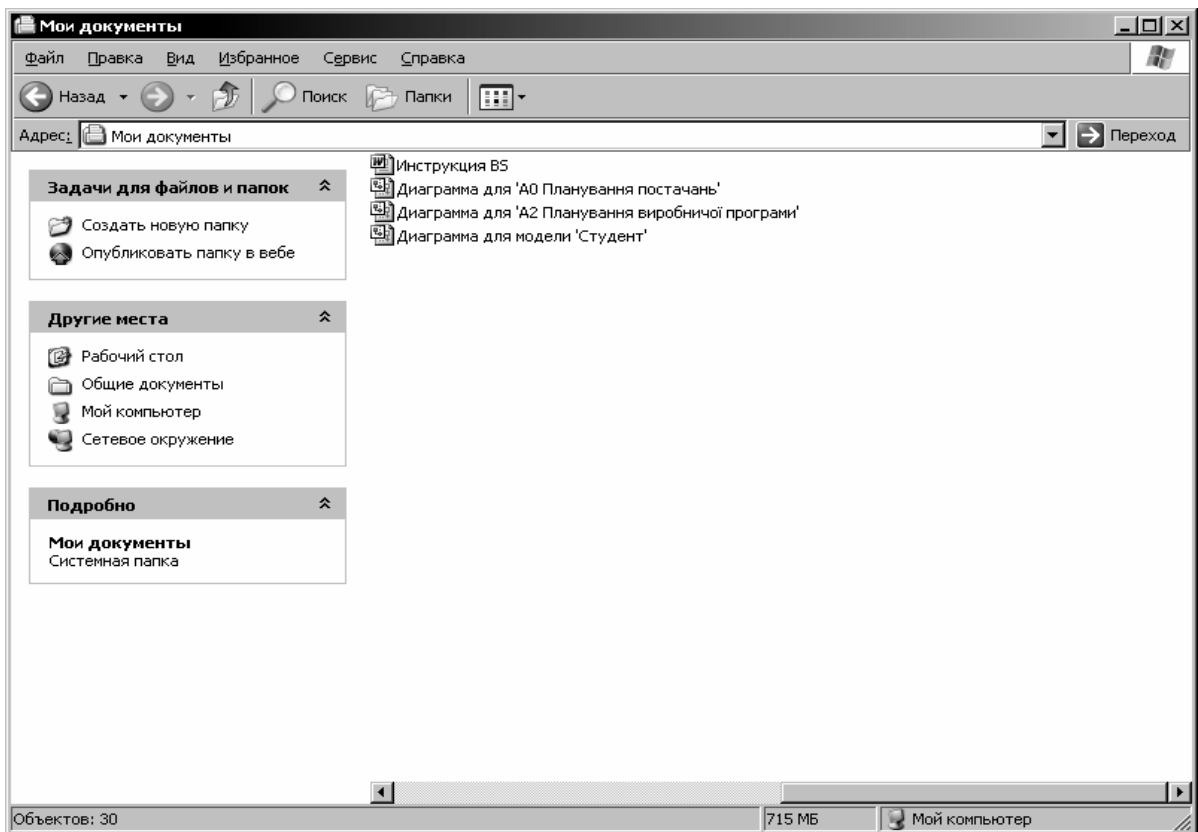


Рис. 4.77. Збереження файлів в обраній папці

Для більше зручної навігації й чіткості сприйняття можна використовувати інструментарій "Группы".

#### 16. Створення груп користувачів.

Для створення груп користувачів треба відкрити розділ "Группы". Далі у контекстному меню "Группы" обрати пункт "Добавить папку от текущего", та назвати її. Далі у контекстному меню цієї папки треба вибрати пункт "Добавить от текущего", щоб додати в папку новий елемент (групу). Дати цій групі назву "Все процессы, субъекты, объекты".

Далі у контекстному меню даної групи вибрати пункт "Свойства" та у вікні властивостей, що з'явилось, заповнити вкладку "Состав" елементами групи шляхом перетаскування лівою клавішею миші створених вами папок і окремих об'єктів у розділах "Процессы", "Субъекты", "Объекты" на область вкладки "Состав" (рис. 4.78). Після наповнення групи елементами треба натиснути на кнопку "ОК". При цьому можна користуватися таким принципом: після створення об'єкта або папки в навігаторі відразу включати їх у свою групу, при цьому об'єкти, що вміщуються в папках, уже включених у групу, не повинні включатися в групу.

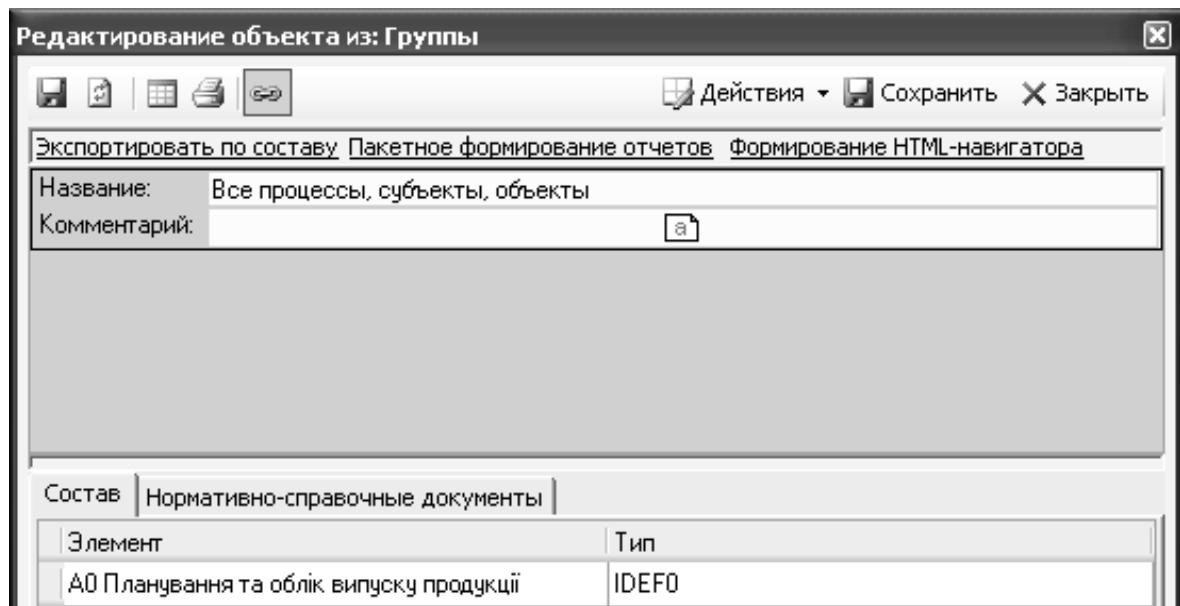


Рис. 4.78. Склад групи

Для того, щоб відобразити в навігаторі об'єктів тільки дані певного користувача, треба вибрати у розділі "Группы" створену групу "Все

процессы, субъекты, объекты", викликати контекстне меню групи й вибрати у ньому пункт "Фильтровать по группе" (рис. 4.79).

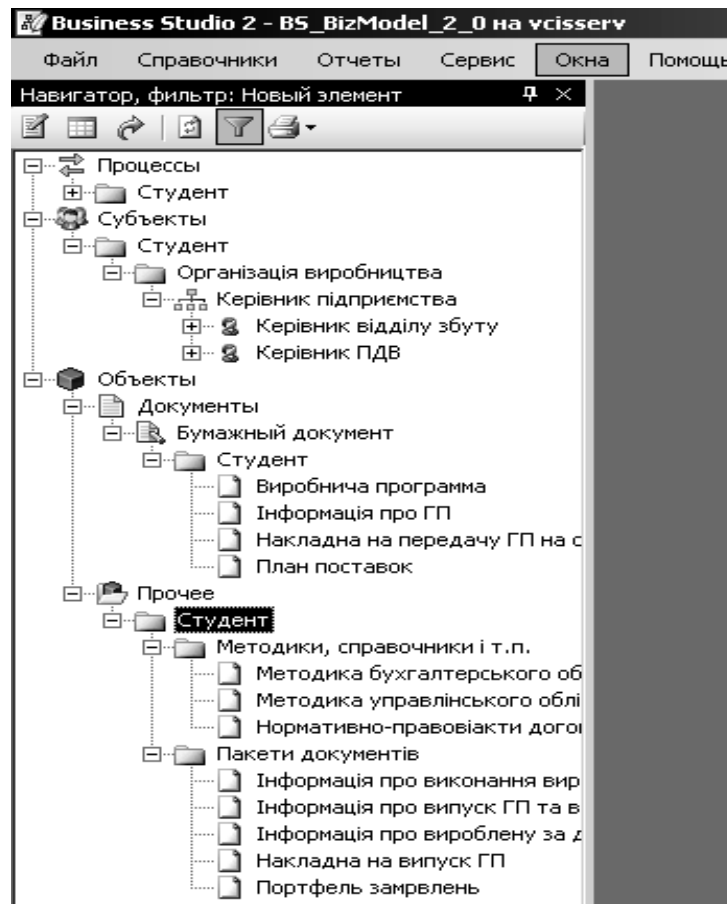


Рис. 4.79. Вікно Навігатора об'єктів зі фільтрованою групою

Щоб зняти встановлений фільтр, треба натиснути на кнопку / на панелі інструментів навігатора об'єктів.

## Контрольні питання

1. Як виконати розщеплення й злиття моделей в BPWin ?
2. Призначення й спосіб застосування груп у навігаторі об'єктів Business Studio.
3. Розкрити відповідність типів стрілок на діаграмі IDEF0 розділам навігатора об'єктів
4. Розкрити особливості формування, що розгалужуються й зливаються, стрілок на діаграмах IDEF0 при побудові моделей процесу в Business Studio.

5. Пояснити призначення й спосіб виконання тонелювання виду "не в батьківській діаграмі".

6. Пояснити призначення й спосіб виконання тонелювання виду "не в дочірній діаграмі".

7. Розкрити призначення й зміст звітів про процеси моделі в Business Studio.

## Глосарій

**Батьківська діаграма** – діаграма, яка містить батьківський блок.

**Батьківський блок** – блок, який детально описується дочірньою діаграмою.

**Блок** – прямокутник, що містить ім'я і номер і використовується для опису функції.

**Вихідна стрілка** – клас стрілок, які відображають вихід IDEF0-блоку, тобто дані або матеріальні об'єкти, отримані на основі виконання функції. Вихідні стрілки зв'язуються з правою стороною блоку IDEF0.

**Внутрішня стрілка** – вхідна стрілка, кінці якої зв'язують джерело і споживача, що є блоками однієї діаграм. Відрізняється від граничної стрілки.

**Вузлове посилання** – код, привласнений діаграмі для її ідентифікації і визначення положення в ієрархії моделі; формується з скороченого імені моделі та вузлового номера діаграми з додатковими розширеннями.

**Вузловий номер діаграми** – частина вузлового посилання діаграми, яка відповідає номеру батьківського блоку.

**Вузловий номер** – код, привласнений блоку, що визначає його положення в ієрархії моделі.

**Вузол** – блок, що породжує дочірні блоки; батьківський блок.

**Вхідна стрілка** – клас стрілок, які відображають вхід IDEF0-блоку, тобто дані або матеріальні об'єкти, які будуть перетворюватися функцією у вихід. Вхідні стрілки зв'язуються з лівою стороною блоку IDEF0.

**Галуження** – розділення стрілки на два або більше чисел сегментів. Може означати "розв'язування пучка".

**Глосарій** – список визначень для ключових слів, фраз і аббревіатур, пов'язаних з вузлами, блоками, стрілками або з моделлю IDEF0 в цілому.

**Гранична стрілка** – стрілка, один із кінців якої пов'язаний з джерелом або споживачем, а інший не приєднаний ні до якого блоку на діаграмі. Відображає зв'язок діаграми з іншими блоками системи і відрізняється від внутрішньої стрілки.

**Декомпозиція** – розділення модельованої функції на функції - компоненти.

**Дерево вузлів** – представлення відносин між батьківськими і дочірніми вузлами моделі IDEF0 у формі деревоподібного графа. Має те ж значення і зміст, що й перелік вузлів.

**Діаграма А-0** – спеціальний вид (контекстної) діаграми IDEF0, що складається з одного блоку, який описує функцію верхнього рівня, її входи, виходи, управління, і механізми разом з формулюваннями мети моделі і точки зору, з якою будується модель.

**Діаграма** – частина моделі, що описує декомпозицію блоку.

**Діаграма-ілюстрація (FEO)** – графічний опис, який використовується для повідомлення специфічних фактів про діаграму IDEF0. При побудові діаграм FEO можна не дотримуватися правила IDEF0.

**Дочірній блок** – блок на дочірній (породженої) діаграмі.

**Дочірня діаграма** – діаграма, що деталізує батьківський блок.

**Злиття** – об'єднання два або більшого числа сегментів стрілок в один сегмент. Може означати "розв'язування пучка".

**Ім'я блоку** – дієслово або дієслівний зворот, що поміщений усередині блоку і описує модельовану функцію.

**Інтерфейс** – роздільна межа, через яку проходять дані або матеріальні об'єкти; з'єднання між двома або великим числом компонентів моделі, що передає дані або матеріальні об'єкти від одного компоненту до іншого.

**Код ICOM** – аббревіатура ( Input – вхід, Control – управління, Output вихід, Mechanism – механізм), код, що забезпечує відповідність граничних стрілок дочірньої діаграми із стрілками батьківського блоку; використовується для посилань.

**Контекст** – навколишнє середовище, в якому діє функція (або комплект функцій на діаграмі).

**Контекстна діаграма** – діаграма, що має вузловий номер А-п ( $n > 0$ ), яка представляє контекст моделі, діаграма А-0, що складається з одного блоку, є необхідною (обов'язковою) контекстною діаграмою; діаграми з вузловими номерами А-1, А-2... – додаткові контекстні діаграми.

**Мета** – коротке формулювання причини створення моделі.

**Мітка стрілки** – іменник або зворот іменника, пов'язані із стрілкою або сегментом стрілки і такі, що визначають їх значення.

**Модель IDEF0** – графічний опис системи, розроблений з певною метою і з вибраної точки зору. Комплект однієї або більше діаграми IDEF0, які зображають функції системи за допомогою графіки, тексту і глосарію.

Номер блоку, число (0 – 6), що поміщається в правому нижньому кутку блоку і однозначно ідентифікує блок на діаграмі.

**Перелік вузлів** – список, що часто ступінчастий, такий, що показують вузли моделі IDEF0 у впорядкованому вигляді. Має те ж значення і зміст, що і дерево вузлів.

**Примітка до моделі** – текстовий коментар, що є частиною діаграми IDEF0 і використовується для запису факту, що не знайшов графічного зображення.

Сегмент стрілки, сегмент лінії, який починається або закінчується на стороні блоку, в точці галуження або злиття, або на межі (незв'язаний кінець стрілки).

**Семантика** – значення синтаксичних компонентів мови.

**Синтаксис** – Структурні компоненти або характеристики мови і правила, які визначають відносини між ними.

Скріплення/розв'язування, об'єднання значень стрілок в складене значення (скріплення в "пучок"), або розділення значень стрілок (розв'язування "пучка"), виражене синтаксисом злиття або галуження стрілок.

**С-номер** – номер, що створюється в хронологічному порядку і використовується для ідентифікації діаграми і дослідження її історії; може бути використаний як посилальний вираз при визначенні конкретної версії діаграми.

**Стрілка виклику** – вид стрілки механізму, який позначає звернення з блоку даної моделі (або частини моделі) до блоку іншої моделі (або іншій частині тієї ж моделі) і забезпечує зв'язок між моделями або між різними частинами однієї моделі.

**Стрілка механізму** – клас стрілок, які відображають механізми IDEF0, тобто засоби, використовувані для виконання функції; включає спеціальний випадок стрілки виклику. Стрілки механізмів зв'язуються з нижньою стороною блоку IDEF0.

Стрілка, направлена лінія, що складається з одного або декількох сегментів, яка моделює відкритий канал або канал, що передає дані або матеріальні об'єкти від джерела (початкова точка стрілки) до споживача

(кінцева точка з "наконечником"). Є 4 класи стрілок – вхідна стрілка, вихідна стрілка, стрілка, що управляє, стрілка механізму (включає стрілку виклику).

**Стрілка, поміщена в тунель (тунельна стрілка)** – стрілка (із спеціальною нотацією), що не задовольняє звичайній вимозі, згідно з якою кожна стрілка на дочірній діаграмі повинна відповідати стрілкам на батьківській діаграмі.

Керівна стрілка, клас стрілок, які в IDEF0 відображають управління, тобто умови, при виконанні яких вихід блоку буде правильним. Дані або об'єкти, що моделюються як управління, можуть перетворюватися функцією, що створює відповідний вихід. Стрілки, що управляють, зв'язуються з верхньою стороною блоку IDEF0.

**Текст** – будь-який текстовий (не графічний) коментар до графічної діаграми IDEF0.

**Тильда** – ламана (хвиляста) лінія, яка використовується для з'єднання влучної з конкретним сегментом стрілки або примітки моделі з компонентом діаграми.

**Точка зору** – вказівка на посадову особу або підрозділ організації, з позиції якого розробляється модель.

**Функція** – діяльність, процес або перетворення (модельовані блоком IDEF0), що ідентифікується дієсловом або дієслівною описовою формою, що повинно бути виконано.



## Використана література

1. Вендров А. М. CASE-технологии. Современные методы и средства проектирования информационных систем. – М.: Лори, 2002. – 432 с.
2. Вспомогательные средства поддержки жизненного цикла ПО  
// <http://program.rin.ru/razdel/html/523-7.html>
3. Вьюкова Н. NewEra – новая линия инструментальных средств компании Informix // <http://citforum.fast.net.ua/database/kbd96/62.shtml>
4. Гейн К., Сарсон Т. Структурный системный анализ: средства и методы. – М.: Эйтекс, 1993. – 286 с.
5. Гнатуш А. CASE-технологии: что, когда, как?  
// <http://www.thalion.kiev.ua/idx.php/69/082/article/>
6. Горин С. В. Применение CASE-средства Erwin 2.5 для информационного моделирования в системах обработки данных / С. В. Горин, А. Ю. Тандоев // <http://citforum.fast.net.ua/database/kbd96/65.shtml>
7. Горчинская О. Designer/2000 – новое поколение CASE-продуктов фирмы Oracle  
// <http://citforum.fast.net.ua/database/kbd96/63.shtml>
8. Дубейковский В. И. Практика функционального моделирования с AllFusion Process Modeler 4.1. Где? Зачем? Как?. – М.: ДИАЛОГ- МИФИ, 2004. – 258 с.
9. Дубейковский В. И. Функциональное моделирование с использованием продукта AllFusion Process Modeler 4.1.4.  
// <http://www.ca.com/ru/about/content.aspx?cid=142677>.
10. Дубейковский В. И. Эффективное моделирование с AllFusion Process Modeler 4.1.4 и AllFusion PM. – М.: ДИАЛОГ-МИФИ, 2007. – 248 с.
11. Закис А. В., Технология быстрой разработки приложений на основе CASE-средств фирмы CADRE / А. В. Закис, М. И. Макаров, Н. И. Приезжий // <http://program.rin.ru/razdel/html/523-2.html>
12. Калашян А. Н. Структурные модели бизнеса: DFD-технологии. / А. Н. Калашян, Г. Н. Калянов. — М.: Финансы и статистика, 2003. – 256 с.
13. Калянов А. Н. Консалтинг при автоматизации предприятий (подходы, методы, средства). — М.: СИНТЕГ, 1997. – 356 с.

14. Калянов А. Н. CASE: структурный системный анализ (автоматизация и применение). — М.: ЛОПИ, 1996. — 288 с.
15. Марка Д. А. SADT. Методология структурного анализа и проектирования / Д. А. Марка, Л. М. Клемент, — М.: МетаТехнология, 1993. — 356 с.
16. Маклаков С. В. BPWin и ERWin. CASE-средства разработки информационных систем. — М.: Диалог-МИФИ, 2000. — 256 с.
17. Маклаков С. В. Создание информационных систем с AllFusion Modeling Suite. — М.: Диалог-МИФИ, 2003. — 432 с.
18. Мінухін С. В. CASE-технології: конспект лекцій. — Харків: Вид. ХНЕУ, 2005. — 156 с.
19. Науменко А. Современное CASE-средство S-Designor фирмы PowerSoft // <http://citforum.fast.net.ua/database/kbd96/69.shtml>
20. Новичков А. Система генерации проектной документации Rational SoDA // <http://citforum.fast.net.ua/programming/digest/soda.shtml>
21. Новичков А. Управление изменениями, тестированием и документированием с использованием технологий Rational // [http://citforum.fast.net.ua/programming/digest/ucm\\_rational.shtml](http://citforum.fast.net.ua/programming/digest/ucm_rational.shtml)
22. Новичков А. Управление тестированием, разработкой и конфигурацией на основе Rational Change Request Management // <http://citforum.fast.net.ua/programming/rational/rcrm.shtml>
23. Обзор некоторых CASE-систем // [http://www.mstu.edu.ru/education/materials/zelenkov/ch\\_5\\_7.html](http://www.mstu.edu.ru/education/materials/zelenkov/ch_5_7.html)
24. Орлик С. Открытая архитектура Delphi // <http://citforum.fast.net.ua/database/kbd96/64.shtml>
25. Панащук С. А. Разработка информационных систем с использованием CASE-системы Silverrun // Системы управления базами данных. — 1995. — №3. — С. 41 — 47.
26. Петров Ю. JAM7 — инструмент разработки переносимых приложений архитектуры "клиент-сервер" // <http://citforum.fast.net.ua/database/kbd96/612.shtml>
27. Сахаров П. Rational Rose, BPwin и другие — аспект анализа бизнес-процессов // <http://www.masters.donntu.edu.ua/2002/fvti/myskov/diss/article1.htm>
28. Средства документирования и тестирования // <http://program.rin.ru/razdel/html/523-8.html>

29. Средства моделирования (CASE) и поддержки всех стадий разработки ПО // [http://infoplus.kiev.ua/rus/index2.php?sales/text\\_1\\_9](http://infoplus.kiev.ua/rus/index2.php?sales/text_1_9)
30. Черемных С. В. Структурный анализ систем: IDEF-технологии. / С. В. Черемных, И. О. Семенов, В. С. Ручкин. – М.: Финансы и статистика, 2001. – 208 с.
31. Черемных С. В. Моделирование и анализ систем. IDEF-технологии: практикум. – М.: Финансы и статистика, 2002. – 358 с.
32. Чибисов А. PowerBuilder – среда разработки приложений масштаба предприятия // <http://citforum.fast.net.ua/database/kbd96/67.shtml>
33. Федотова Д. З. CASE-технологии: Практикум / Д. З. Федотова, Ю. Д. Семенов, К. Н. Чижик. — М.: Горячая линия-Телеком, 2003. – 160 с.
34. Методология функционального моделирования IDEF0. Руководящий документ. – М.: Госстандарт России. ИПК Издательство стандартов, 2000. – 76 с.
35. Business Objects XI  
// [http://infoplus.kiev.ua/rus/index2.php?sales/text\\_1\\_7\\_9](http://infoplus.kiev.ua/rus/index2.php?sales/text_1_7_9)
36. Crystall Reports XI // [http://infoplus.kiev.ua/rus/index2.php?sales/text\\_1\\_7\\_2](http://infoplus.kiev.ua/rus/index2.php?sales/text_1_7_2)
37. <http://www.idef.com/pdf/idef0.pdf>
38. <http://www.idef.com/pdf/IDEF1MR-part1.pdf> 39.
39. <http://www.idef.com/pdf/IDEF1MR-part2.pdf>
40. <http://www.idef.com/pdf/ldef1x.pdf>
41. [http://www.idef.com/pdf/ldef3\\_fn.pdf](http://www.idef.com/pdf/ldef3_fn.pdf)
42. <http://www.idef.com/pdf/ldef5.pdf>
43. <http://www.idef.com/pdf/ldef9.pdf>

## Зміст

Вступ	3
Розділ 1. Принципи структурно-функціонального проектування	9
1.1. Загальні принципи структурних методів та структурного аналізу	9
1.2. Засади структурно-функціональної методології	13
1.3. Вивчення предметної області	19
1.4. Концепція розробки іс підприємства на основі структурного підходу	43
Розділ 2. Стандарти структурного проектування систем сучасними CASE-засобами	54
2.1. Типи CASE-засобів	54
2.2. Альтернативна методологія опису предметної області проектування – об'єктно-орієнтоване проектування	134
Розділ 3. Інструментальні засоби структурного проектування	137
3.1. Класифікація CASE-засобів	137
3.2. Засоби аналізу (Upper CASE)	140
3.3. Засоби аналізу і проектування (Middle CASE)	142
3.4. Засоби проектування баз даних	160
3.5. Засоби розробки додатків	165
3.6. Засоби реінжинірингу	173
3.7. Засоби планування і управління проектом	174
3.8. Засоби конфігураційного управління	178
3.9. Засоби тестування	186
3.10. Засоби документування	189
Розділ 4. Практичне використання CASE-засобів при проектуванні окремих підсистем інформаційної системи	195
4.1. Використання VPwin для опису предметної області "Страхування нерухомості та майна фізичних осіб"	195
4.2. Застосування CASE-засобу Business Studio при вирішенні комплексу завдань "Керування договорами"	228
Глосарій	263
Використана література	267

НАВЧАЛЬНЕ ВИДАННЯ

**Мінухін Сергій Володимирович**  
**Беседовський Олексій Миколайович**  
**Знахур Сергій Вікторович**

## **МЕТОДИ І МОДЕЛІ ПРОЕКТУВАННЯ НА ОСНОВІ СУЧАСНИХ CASE-ЗАСОБІВ**

**Навчальний посібник**

Відповідальний за випуск **Пономаренко В. С.**

Відповідальний редактор **Сєдова Л. М.**

Редактор **Нещеретна О. М.**

Коректор **Бриль В. О.**

План 2008 р. Поз. №42-П.

Підп. до друку

Формат 60 × 90 1/16. Папір MultiCopy. Друк Riso.

Ум.-друк. арк. 17,0. Обл.-вид. арк. 21,25. Тираж

прим. Зам. №

---

Видавець і виготівник — видавництво ХНЕУ, 61001, м. Харків, пр. Леніна, 9а

*Свідоцтво про внесення до Державного реєстру суб'єктів видавничої справи  
Дк №481 від 13.06.2001 р.*

*Мінухін С. В.*  
*Беседовський О. М.*  
*Знахур С. В.*

**МЕТОДИ І МОДЕЛІ ПРОЕКТУВАННЯ НА ОСНОВІ СУЧАСНИХ  
CASE-ЗАСОБІВ**

**Навчальний посібник**