МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

РЕМЕНЯК ЛЕСЯ ВАСИЛІВНА

# ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Конспект лекцій

Одеса
Одеський державний екологічний університет
2016

**УДК** 681.518.2
Р37

Рекомендовано методичною радою Одеського державного екологічного університету Міністерства освіти і науки України як конспект лекцій (протокол №10 від 04.07. 2016 р.)

**Ременяк Л.В.**
Проектування інформаційних систем: конспект лекцій. Одеса, Одеський державний екологічний університет, 2016. 152с.

Конспект лекцій з навчальної дисципліни " Проектування інформаційних систем " для студентів III курсу денної форми навчання напрямку – комп'ютерні науки спеціальності – інформаційні управляючі системи та технології. Дисципліна Проектування інформаційних систем є нормативною дисципліною у напряму бакалаврської підготовки за напрямком "Комп'ютерні науки" та належить до циклу професійної та практичної підготовки.

Метою дисципліни Проектування інформаційних систем є формування у студентів теоретичних знань, та практичних навичок у галузі проектування інформаційних систем з використанням сучасних методів та засобів створення інформаційних систем (ІС) під час розробки, налагодження та експлуатації ІС.

**ISBN 978-966-186-095-6**

ЗМІСТ

# ВСТУП

Дисципліна ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ є нормативною дисципліною у напряму бакалаврської підготовки за напрямком "Комп’ютерні науки" та належить до циклу професійної та практичної підготовки.

МЕТОЮ дисципліни ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ – є формування у студентів теоретичних знань, та практичних навичок у галузі проектування інформаційних систем з використанням сучасних методів та засобів створення інформаційних систем (ІС) під час розробки, налагодження та експлуатації ІС.

Вивчення навчальної дисципліни дозволяє студентам оволодіти знаннями в галузі проектування ІС: підходами, принципами, технологіями, інструментальними засобами, шаблонами та стандартами проектування. Дозволяє студентам вирішувати задачі аналізу і проектування ІС, їх модернізацію, вирішувати задачі аналізу і реінжинирінгу бізнес-процесів.

Після вивчення дисципліни студенти повинні бути здатними до проектної діяльності в професійній сфері, вміти будувати і використовувати моделі для опису об’єктів та процесів, здійснювати їх якісний аналіз, застосовувати їх під час розробки та інтеграції систем, продуктів і сервісів інформації. Володіти навиками проектування ІС з використанням сучасних інструментальних засобів.

За результатами вивчення дисципліни студент повинен ЗНАТИ:

- задачі, функції, види та класифікацію ІС;
- стандарти проектування ІС та оформлення проектної документації;
- системний підхід до проектування ІС, топології та архітектури інформаційних систем;
- методи аналізу, вимоги до ІС, формування вимог до ІС;
- методології і технології проектування ІС;
- основи структурного і об’єктно-орієнтованого підходу до аналізу і проектування ІС;
- основні етапи проектування ІС, визначення і сферу застосування CASE засобів і технологій ІС;
- аналіз даних з застосуванням діаграм «сутність-зв’язок», методи реляційного аналізу даних
- інтерфейси інформаційних систем;
- інструментальні засоби технологій проектування ІС:

  - RAD-методологію,
  - CASE-технологію створення й супроводу ІС,
  - технологію RUP,
  - технологію ARIS,
- паттерн-технології;
- реінжиніринг ІС.

# 1 ПРИЗНАЧЕННЯ, ЗАДАЧІ І ФУНКЦІЇ ІС

## 1.1 Система і її властивості

Існує безліч визначень поняття системи, кожне з них визначається по різному і має безліч смислових значень.

Слово *«система»* походить від грецького systema, що означає ціле, складене з частин або безлічі елементів.

***Система*** – це сукупність взаємопов'язаних елементів, які функціонують для досягнення певної мети.

***Система*** – це сукупність елементів (об'єктів), які знаходяться між собою в певних відносинах і зв'язках, і які утворюють певну цілісність, єдність будь-якого явища або предмета дослідження.

***Системою*** називається впорядкована сукупність взаємодіючих елементів, об'єднаних певними зв'язками, призначена для досягнення заданої мети і досягає її найкращим (по можливості) чином.

*Під системою будемо розуміти* будь-який об'єкт, який одночасно розглядається і як єдине ціле, і як об'єднана в інтересах досягнення поставлених цілей сукупність різнорідних елементів.

У найзагальнішому випадку поняття «система» характеризується:

- наявністю безлічі елементів;
- наявністю зв'язків між ними;
- цілісним характером даного пристрою або процесу.

***Елементом*** назвемо деякий об'єкт (матеріальний, енергетичний, інформаційний), що володіє рядом важливих для нас властивостей, але внутрішня будова (зміст) якого не є в даному випадку метою розгляду.

***Зв'язком*** назвемо важливий для цілей розгляду обмін між елементами речовиною, енергією, інформацією. Окремим випадком зв'язку виступає вплив.

Системи відрізняються між собою як по складу так і по головній меті.

Ознаками системи є безліч складових її елементів, єдність головної мети для всіх елементів, наявність зв'язків між ними, цілісність і єдність елементів, наявність структури та ієрархічності, відносна самостійність і наявність управління цими елементами.

У табл..1.1 наведено приклади кількох систем, що складаються з різних елементів і спрямованих на реалізацію різних цілей.

Таблиця 1.1 - Приклади систем

| **Система** | **Елементи системи** | **Головна мета системи** |
|---|---|---|
| Фірма | Люди, обладнання, матеріали, будівлі та ін | Виробництво товарів |
| Комп'ютер | Електронні та електромеханічні елементи, лінії зв'язку і ін. | Обробка даних |
| Телекомунікаційна система | Комп'ютери, модеми, кабелі, мережеве програмне забезпечення та ін. | Передача інформації |
| Інформаційна система | Комп'ютери, комп'ютерні мережі, люди, інформаційне і програмне забезпечення | Виробництво професійної інформації |

Система має свої властивості.

*Властивості системи* − це якості елементів, що дають можливість кількісного опису системи, та вираження її в певних величинах.

***Цілісність системи***. Елементи системи функціонують в часі та взаємопов'язані як єдине ціле. Кожен з них працює заради досягнення єдиної мети, що стоїть перед всією системою. Система не повинна розглядатися як проста сума елементів. Потрібно враховувати ефект взаємодії елементів, завдяки якому деякі властивості накопичуються, посилюються і в сукупності може з'явитися нова властивість, властива всій системі.

***Емерджентність системи*** − здатність складної системи проявляти загальносистемні властивості і породжувати системний ефект, не притаманний окремих елементах системи.

***Цільове призначення системи***. Для якої мети функціонує система, які перед нею ставляться завдання.

*Великий системою* назвемо систему, що включає значне число однотипних елементів і однотипних зв'язків.

*Складною системою* назвемо систему, що складається з елементів різних типів і володіє різнорідними зв'язками між ними.

В даний час важливим класом складних систем виступають так звані автоматизовані системи. Слово «автоматизований» вказує на участь людини, використання його активності всередині системи при збереженні значної ролі технічних засобів.

Отже, автоматизованою системою називається складна система з визначальною роллю елементів двох типів у вигляді:

- технічних засобів;
- дій людини.

Коли ми доповнюємо поняття "система" словом "інформаційна", тоді ми хочемо підкреслити мету її створення і функціонування.

***Інформаційна система (ІС)*** – це система, яка реалізує інформаційну модель предметної області, найчастіше будь-якої сфери людської діяльності. ІС повинна забезпечувати: отримання (введення або збір), зберігання, пошук, передачу та обробку даних, інформації.

***Інформаційна система*** – це організаційно-упорядкована взаємопов'язана сукупність засобів, і методів інформаційних технологій (ІТ), використовуваних для зберігання, обробки та видачі інформації в інтересах досягнення поставленої мети. Таке розуміння інформаційної системи припускає використання в якості основного технічного засобу переробки інформації ЕОМ і засобів зв'язку, що реалізують інформаційні процеси і видачу інформації, необхідної в процесі прийняття рішень задач з будь-якої області.

### 1.1.1 Структура і ієрархія систем

***Структурою системи*** називається представлення системи у вигляді груп елементів із зазначенням зв'язків між ними, незмінними на весь час розгляду і дає уявлення про систему в цілому. Зазначене подання може мати матеріальну, функціональну, алгоритмічну та інші основи.

Групи елементів в структурі зазвичай виділяються за принципом простих або відносно більш слабких зв'язків між елементами різних груп. Структуру системи зручно зображати у вигляді графічної схеми, що складається з осередків (груп) і з'єднують їх ліній (зв'язків). Такі схеми називаються структурними.

Структура системи може бути охарактеризована за наявними в ній (або переважаючим) типам зв'язків. Найпростішими з них є послідовне, паралельне з'єднання елементів і зворотний зв'язок.

Пояснимо поняття зворотного зв'язку. Воно означає, що результат функціонування елемента впливає на вчідний до нього вплив. Як правило, зворотний зв'язок виступає важливим регулятором в системі. Вкрай рідко зустрічається система без того чи іншого виду зворотного зв'язку.

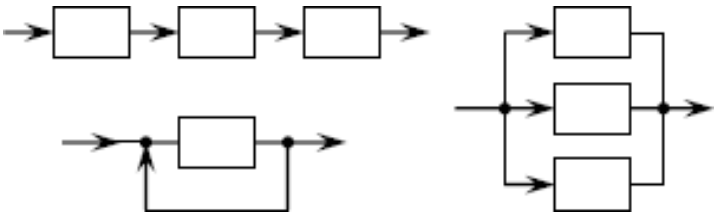


Рисунок 1.1 − Найпростіші типи зв'язків

Близьким до поняття структури є термін декомпозиція.

***Декомпозицією*** називається розподіл системи на частини, яке буде зручним для будь-яких операцій з цією системою.

Найважливішим стимулом і суттю декомпозиції є спрощення системи, занадто складною для розгляду цілком.

***Ієрархією*** назвемо структуру з наявністю підпорядкованості, тобто нерівноправних зв'язків між елементами, коли впливу в одному з напрямків роблять набагато більший вплив на елемент, ніж в іншому.

***Деревоподібна структура*** найбільш проста для аналізу і реалізації. У ній завжди зручно виділяти ієрархічні рівні − групи елементів, що знаходяться на однаковій відстані від верхнього (головного) елемента.

***Ромбовидна структура*** веде до подвійної (іноді і більше) підпорядкованості, звітності, приналежності нижнього елемента. У техніці − це участь даного елемента в роботі більш ніж одного вузла, блоку, використання одних і тих же даних або результатів вимірювань в різних завданнях.

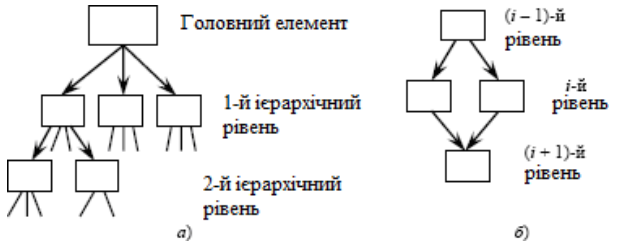



Рисунок 1.1 – Приклади ієрархічних структур:
а) деревовидної; б) ромбовидної.

Будь-яка ієрархія звужує можливості і гнучкість системи. Елементи нижнього рівня сковуються домінуванням зверху, вони здатні впливати на це домінування лише частково і з затримкою. Введення ієрархії різко спрощує створення і функціонування системи, і тому її можна вважати вимушеним, але необхідним прийомом розгляду складних систем.

Негативні наслідки введення ієрархії багато в чому можуть бути подолані наданням окремим елементам можливості реагувати на частину впливів без жорсткої регламентації зверху.

### 1.1.2 Модульна будова системи

До цих пір ми називали зв'язком вплив одного елемента (або групи елементів) на інший елемент (групу). Ніщо не заважає поширити поняття зв'язку на взаємодію системи з елементами, що знаходяться за межами розгляду, які зазвичай називають зовнішнім середовищем.

Наступний крок у дослідженні зв'язків в системі полягає у виділенні:

- всіх тих впливів, які даний елемент відчуває з боку інших елементів і з боку зовнішнього середовища;
- впливів, які він надає на інші елементи і на зовнішнє середовище.

Першу групу впливів прийнято називати ***входами*** (вплив на елемент), а другу − ***виходами*** (впливу від елемента).

Виходи елемента визначаються входами і його внутрішньою будовою. Кажуть, що вихід є функція від входу і самого елемента.

Входи і виходи переноситься на довільну сукупність елементів, включаючи і всю систему цілком. І тут можна говорити про всі вхідні і вихідні впливи. Це не просто зручний, але вельми плідний підхід до розгляду системи, оскільки, характеризуючи групу елементів тільки входами і виходами, можна отримати можливість оперувати цією частиною системи, не вникаючи, як пов'язані і взаємодіють між собою її елементи, тобто піти від деталізації в описі при збереженні основних особливостей системи.

***Модулем*** називається сама система, група елементів системи або окремий елемент системи, які описані тільки своїми входами і виходами і володіють певною цілісністю.

Система може представлятися набором модулів і сама розглядатися як модуль. Модульна побудова системи, як правило, визначає її декомпозицію. Нерідко воно визначає і структуру. Розподіл системи на модулі − це зручний

і найбільш поширений прийом роботи з штучними системами, включаючи їх створення (проектування), перевірку, настройку, удосконалення.

### 1.1.3 Цілеспрямовані системи та управління

Обговоримо тепер поняття, пов'язані з постановкою перед системою деякої сформульованої мети. Системи при цьому називають ***цілеспрямованими***, такими будуть штучні системи.

Поняття мети системи визначимо як задачу отримання бажаного вихідного впливу або досягнення бажаного стану системи.

Вибір глобальної мети для системи спричиняє необхідність:

- формулювання локальних цілей, що стоять перед елементами системи і групами елементів;
- цілеспрямованого втручання у функціонування системи.

Обидві операції тісно пов'язані, хоча з точки зору практики спочатку розбивають глобальну мету на ряд локальних, а потім шукають шляхи досягнення локальних цілей.

Набір локальних цілей, як правило, сам має ієрархічну багаторівневеу будову і в тій чи іншій мірі відповідає загальній ієрархії в системі. У цьому випадку поняття «*локальні цілі*» є збірним терміном для цілей всіх ієрархічних рівнів; для будь-якої з них можна вказати, в яку мету більш високого рівня вона входить і (крім цілей самого нижчого рівня) на що вона дробиться сама.

При модульній будові системи локальні цілі виступають як вимоги до виходів модулів. Саме продумані вимоги на виходи узгоджують модулі так, що система яка з них складається виконує глобальну мету.

## 1.2 Поняття інформаційної системи

В цілому інформація в системі виступає як збірний термін для позначення всіх потрібних відомостей про об'єкти і явища навколишнього середовища, їх параметри, властивості і стан, які використовуються для зменшення наявної про них ступеня невизначеності, неповноти знань.

Поняття інформації має високий ступінь універсальності. У широкому сенсі функціонування системи можна трактувати як перетворення вхідної інформації у вихідну. Така точка зору особливо корисна в системному аналізі.

Часто під терміном інформація мають на увазі дані.

Інформація виходить з даних, якщо над ними проведена деяка обробка, що підвищує їх цінність. Дані можуть розглядатися як ознаки або записані спостереження, які з якихось причин не використовуються, а тільки зберігаються. У тому випадку, якщо з'являється можливість використовувати ці дані для зменшення невизначеності про що-небудь, дані перетворюються в інформацію. Тому інформацією є використання даних.

Властивості інформації:

- важливість;
- достовірність;
- своєчасність;
- доступність.

Крім того, слід розрізняти такі поняття, як інформація і знання. Наведемо порівняльний аналіз зазначених понять, виділивши їх основні відмінні ознаки.

Інформація:

- завжди пов'язана з якими-небудь даними;
- широко поширена, знаходиться в усіх напрямках;
- може генеруватися людьми, комп'ютерами, іншими машинами;
- легко сприймається і легко передається;
- як правило, статична;
- взаємопов'язана з іншою інформацією;
- володіє вартістю, необхідної на створення і підтримку;
- може використовуватися будь-ким і коли завгодно.

Знання:

- мають відношення до даних та інформації, але не завжди з ними пов'язані;
- дефіцитні, їх непросто добувати;
- генеруються тільки людьми;
- важкі для сприйняття;
- динамічні; мають швидкість передачі і сприйняття;
- для успішного сприйняття вимагають чітких меж їх розуміння;
- можуть бути дуже дорогі, ціна при цьому не фіксована;
- володіють терміном і метою використання.

Додавання до поняття система слова інформаційна відображає мету її створення і функціонування.

***Інформаційна система*** – це взаємозв'язана сукупність засобів, методів і персоналу, використовуваних для зберігання, обробки та видачі інформації в інтересах досягнення поставленої мети.

Сучасне розуміння інформаційної системи припускає використання в якості основного технічного засобу переробки інформації персонального комп'ютера. У великих організаціях інформаційні системи можуть базуватися на локальних або глобальних мережах. Крім того, технічне втілення інформаційної системи нічого не означатиме, якщо не врахована роль людини, для якого призначена інформація і без якої неможливе її одержання і представлення.

З огляду на те, що в основі лежить середовища зберігання і доступу до даних, інформаційні системи являють собою прикладні програмні комплекси, призначені для збору, зберігання і обробки даних, тобто отримання необхідної інформації в інтересах досягнення поставленої мети.

Інформаційні системи орієнтуються на кінцевого користувача, що не володіє високою кваліфікацією в області застосування обчислювальної техніки. Тому клієнтські програми інформаційної системи повинні володіти простим, зручним інтерфейсом.

Таким чином, при розробці інформаційної системи доводиться вирішувати три основні завдання:

1) розробка програмного комплексу, призначеного для зберігання інформації;
2) завдання розробки графічного інтерфейсу;
3) облік конкретної середовища або технології: топології мережі, конфігурації апаратних засобів, використовуваної архітектури.

## 1.3 Класифікація ІС

Різноманітність задач, що вирішуються за допомогою інформаційних систем, привело до появи безлічі різнотипних систем, що відрізняються принципами побудови і закладеними в них правилами обробки інформації.

Інформаційні системи можна класифікувати по цілому ряду різних ознак. В основу даної класифікації покладено найбільш істотні ознаки, що визначають функціональні можливості і особливості побудови сучасних систем.

Залежно від обсягу вирішуваних задач, використання технічних засобів, організації функціонування, інформаційні системи діляться на ряд груп (класів).

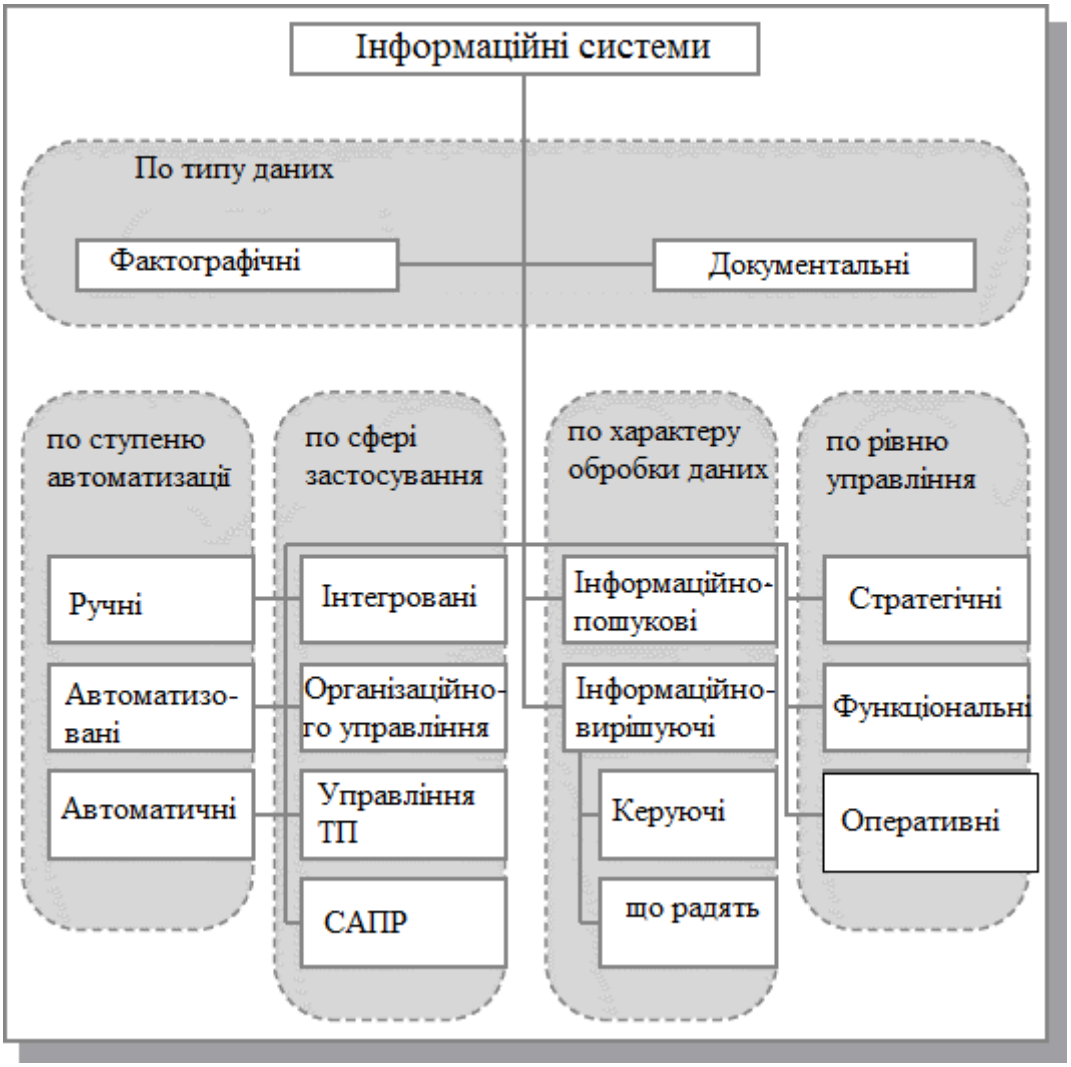



Рисунок 1.2 −Класифікація інформаційних систем

***Класифікація за типом даних, що зберігаються.***

За типом даних, що зберігаються інформаційні системи діляться на групи:

- фактографічні;
- документальні.

Фактографічні системи призначені для зберігання і обробки структурованих даних у вигляді чисел і текстів. Над такими даними можна виконувати різні операції.

У документальних системах інформація представлена у вигляді документів, що складаються з найменувань, описів, рефератів і текстів. Пошук по неструктурованим даним здійснюється з використанням семантичних ознак. Відібрані документи надаються користувачеві, а обробка даних в таких системах практично не проводиться.

***Класифікація за ступенем автоматизації інформаційних процесів.***

За ступенем автоматизації інформаційних процесів в системі управління інформаційні системи діляться на групи:

- ручні;
- автоматичні;
- автоматизовані.

*Ручні інформаційні системи* характеризуються відсутністю сучасних технічних засобів переробки інформації та виконанням всіх операцій людиною.

*В автоматичних інформаційних системах* всі операції з переробки інформації виконуються без участі людини.

*Автоматизовані інформаційні системи* передбачають участь в процесі обробки інформації і людини, і технічних засобів, причому головна роль у виконанні рутинних операцій обробки даних відводиться комп'ютера. Саме цей клас систем відповідає сучасному уявленню інформаційної системи.

***Класифікація по сфері застосування.***

За сферою застосування інформаційні системи діляться на:

- системи організаційного управління;
- системи управління технологічними процесами;
- системи автоматизованого проектування;
- нтегровані (корпоративні) інформаційні системи.

*Інформаційні системи організаційного управління* призначені для автоматизації функцій управлінського персоналу як промислових підприємств, так і непромислових об'єктів (готелів, банків, магазинів та ін.).

Основними функціями подібних систем є: оперативний контроль і регулювання, оперативний облік і аналіз, перспективне і оперативне планування, бухгалтерський облік, управління збутом, постачанням і інші економічні і організаційні завдання.

*Інформаційні системи управління технологічними процесами* служать для автоматизації функцій виробничого персоналу з контролю та управління виробничими операціями. У таких системах зазвичай передбачається наявність розвинених засобів вимірювання параметрів технологічних

процесів (температури, тиску, хімічного складу і т.д.), процедур контролю допустимості значень параметрів і регулювання технологічних процесів.

*Інформаційні системи автоматизованого проектування* призначені для автоматизації функцій інженерів-проектувальників, конструкторів, архітекторів, дизайнерів при створенні нової техніки або технології. Основними функціями подібних систем є: інженерні розрахунки, створення графічної документації, створення проектної документації, моделювання проектованих об'єктів.

*Інтегровані (корпоративні) інформаційні системи* використовуються для автоматизації всіх функцій фірми і охоплюють весь цикл робіт від планування діяльності до збуту продукції. Вони включають в себе ряд модулів (підсистем), що працюють в єдиному інформаційному просторі і виконують функції підтримки відповідних напрямів діяльності.

***Класифікація за характером обробки даних.***

За характером обробки даних інформаційні системи діляться на групи:

- інформаційно-пошукові;
- інформаційно-вирішальні.

*Інформаційно-пошукові системи* роблять введення, систематизацію, зберігання, видачу інформації за запитом користувача без складних перетворень даних (інформаційна система бібліотечного обслуговування, резервування і продажу квитків на транспорті, бронювання місць в готелях).

*Інформаційно-вирішальні системи* здійснюють, крім того, операції переробки інформації за певним алгоритмом.

***За характером використання вихідної інформації*** такі системи прийнято ділити на:

- керуючі;
- що радять.

Результуюча інформація *керуючих інформаційних сис*тем безпосередньо трансформується в прийняті людиною рішення. Для цих систем характерні задачі розрахункового характеру і обробка великих обсягів даних (інформаційна система планування виробництва і замовлень, бухгалтерського обліку).

*Інформаційні системи яка радить* виробляє інформацію, яка приймається людиною до відома і враховується при формуванні управлінських рішень, а не ініціює конкретні дії. Ці системи імітують інтелектуальні процеси обробки знань, а не даних (експертні системи).

***Класифікація за рівнем управління.***

Залежно від рівня управління бувають:

- системи оперативного рівня;
- функціональні системи;
- стратегічні системи.

*Інформаційна система оперативного рівня* підтримує виконавців, обробляючи дані про угоди і події (рахунки, накладні, зарплата, кредити, потік сировини і матеріалів). Інформаційна система оперативного рівня є сполучною ланкою між фірмою і зовнішнім середовищем. Задачі, цілі, джерела інформації та алгоритми обробки на оперативному рівні заздалегідь визначені і у високому ступені структуровані.

*Інформаційні системи фахівців* підтримують роботу з даними і знаннями, підвищують продуктивність і продуктивність роботи інженерів і проектувальників. Задачі подібних інформаційних систем − інтеграція нових відомостей в організацію і допомогу в обробці паперових документів.

*Інформаційні системи рівня менеджменту* − використовуються працівниками середньої управлінської ланки для моніторингу, контролю, прийняття рішень і адміністрування.

*Стратегічна інформаційна система* − це комп'ютерна інформаційна система, що забезпечує підтримку прийняття рішень по реалізації перспективних стратегічних планів розвитку організації.

Інформаційні системи стратегічного рівня допомагають вищій ланці управлінців вирішувати неструктуровані задачі, здійснювати довгострокове планування. Основна задача - порівняння змін, які відбуваються в зовнішньому оточенні до існуючого потенціалом фірми. Вони покликані створити загальне середовище комп'ютерної телекомунікаційної підтримки рішень в несподівано виникаючих ситуаціях. Використовуючи найдосконаліші програми, ці системи здатні в будь-який момент надати інформацію з багатьох джерел. Деякі стратегічні системи мають обмежені аналітичними можливостями.

***Класифікація за способом організації.***

За способом організації інформаційні системи діляться на:

- системи на основі архітектури файл-сервер;
- системи на основі архітектури клієнт-сервер;
- системи на основі багаторівневої архітектури;
- системи на основі Інтернет-технологій.

*Архітектура файл-сервер* не має мережевого поділу компонентів діалогу і використовує комп'ютер для функцій відображення, що полегшує

побудову графічного інтерфейсу. Файл-сервер тільки отримує дані з файлів, так що додаткові користувачі і додатки додають лише незначну навантаження на центральний процесор. Кожен новий клієнт додає обчислювальну потужність до мережі.

*Архітектура клієнт-сервер* призначена для поділу компонентів програми та розміщення їх там, де вони будуть функціонувати найбільш ефективно. Особливістю архітектури клієнт-сервер є використання виділених серверів баз даних. Архітектура клієнт-сервер є дворівневою: додаток працює у клієнта, СУБД − на сервері.

*Багаторівнева архітектура* складається з трьох рівнів:

1) нижній рівень являє собою додатки клієнтів, що мають програмний інтерфейс для виклику програми на середньому рівні;
2) середній рівень − сервер додатків, на якому виконується прикладна логіка і з якого логіка обробки даних викликає операції з базою даних;
3) верхній рівень являє собою віддалений спеціалізований сервер бази даних, виділений для послуг обробки даних і файлових операцій.

*Інтернет-технології* об'єднуються з багаторівневою архітектурою. При цьому структура інформаційного додатку набуває вигляду: браузер-сервер додатків - сервер баз даних - сервер динамічних сторінок - web-сервер.

## 1.3 Життєвий цикл ІС

Кожна система має свій життєвий цикл.

Життєві цикли розрізняються за властивостями, цілям, використання системи, а також по переважаючих умов. Проте, не дивлячись на очевидне безліч відмінностей в життєвих циклах систем, існує базовий набір стадій життєвого циклу, що становлять повний життєвий цикл будь-якої системи. Кожна стадія має певну мету і внесок в повний життєвий цикл і розглядається при плануванні і виконанні життєвого циклу системи.

*Життєвий цикл інформаційної системи* характеризується періодом часу від ідеї створення інформаційної системи і закінчуючи моментом виведення її з експлуатації та включає в себе наступні стадії:

1. Передпроектне обстеження.
2. Проектування.
3. Створення інформаційної системи.
4. Введення в експлуатацію.

5. Експлуатація інформаційної системи.

6. Виведення з експлуатації

Під *стадією* розуміється певний етап процесу розробки інформаційної системи.

Стадії являють собою основні періоди життєвого циклу, пов'язані з системою і пов'язані з станом опису системи або безпосередньо до системи. Стадії відображають значний прогрес і досягнення запланованих етапів розвитку системи протягом усього життєвого циклу і дають початок найважливіших рішень щодо своїх входів і виходів. Ці рішення використовуються організаціями для обліку невизначеностей і ризиків, безпосередньо пов'язаних з витратами, термінами і функціональністю при створенні або застосуванні системи.

Таким чином, стадії забезпечують організації структурою робіт, в рамках яких управління підприємством володіє високою здатністю для огляду і контролю проекту і технічних процесів.

### 1.3.1 Моделі життєвого циклу інформаційної системи

Життєвий цикл може бути описаний з використанням абстрактної функціональної моделі, що представляє концептуалізацію потреби в системі, її реалізації, застосування, розвитку і ліквідації.

Моделлю життєвого циклу інформаційної системи будемо називати деяку структуру, визначальну послідовність здійснення процесів, дій і задач, що виконуються протягом життєвого циклу інформаційної системи, а також взаємозв'язку між цими процесами, діями та задачами.

Модель життєвого циклу інформаційної системи може включати:

1) Стадії.
2) Основні результати виконання робіт на кожній стадії.
3) Ключові події.

Найбільшого поширення набули такі моделі життєвого циклу інформаційних систем: *каскадна* (класична або водоспадна), *ітераційна* і *спіральна*.

***Каскадна (класична, водоспадна) модель життєвого циклу інформаційної системи.***

Модель була запропонована в 1970 році Уинстоном Ройсом.

Каскадна модель − передбачає послідовне виконання всіх етапів проекту в строго фіксованому порядку.

Перехід на наступний етап здійснюється після повного закінчення робіт за попереднім етапу, при цьому оформляється повний комплект робочої документації.

На першому етапі проводиться дослідження проблеми, яка повинна бути вирішена, чітко формулюються всі вимоги замовників. Результатом, даного етапу, є технічне завдання, узгоджене з усіма зацікавленими сторонами.

На другому етапі розробляються проектні рішення, що задовольняють всім вимогам, сформульованим у технічному завданні. Результатом даного етапу є комплект проектної документації, що містить всі необхідні дані для реалізації проекту.

Третій етап − реалізація проекту. Тут здійснюється розробка програмного забезпечення відповідно до проектних рішень, отриманими на попередньому етапі. Результатом виконання даного етапу є готовий програмний продукт.

На четвертому етапі проводиться перевірка отриманого програмного забезпечення на предмет відповідності вимогам, заявленим в технічному завданні. Тестування дозволяє виявити приховані недоліки, які проявляються в реальних умовах роботи інформаційної системи.

Останній етап − здача готового проекту і його супровід.

*Основні переваги каскадної моделі*

Каскадний підхід добре зарекомендував себе при побудові інформаційних систем, для яких на самому початку розробки можна досить точно і повно сформулювати всі вимоги, з тим, щоб надати розробникам свободу реалізувати їх якнайкраще з технічної точки зору. У цю категорію потрапляють складні розрахункові системи, системи реального часу і інші подібні завдання.

*Позитивні сторони застосування каскадного підходу.*

На кожному етапі формується закінчений набір проектної документації, який відповідає критеріям повноти і узгодженості.

Виконувані в логічній послідовності етапи робіт дозволяють планувати терміни завершення всіх робіт і відповідні витрати.

*Недоліки каскадної моделі*

У процесі використання каскадної моделі виявився ряд її недоліків, викликаних, перш за все, тим, що реальний процес створення інформаційної системи ніколи повністю не вкладався в таку жорстку схему. У процесі

створення інформаційної системи постійно виникала потреба в поверненні до попередніх етапах і уточнення або перегляд раніше прийнятих рішень.

Ця модель подобаєть замовникам, і розробникам через жорстку дисципліну етапів тільки після їх пред'явлення. Але повністю відсутня гнучкість в роботі над створенням ІС. Каскадна модель представлена на рис.1.3


Рисунок 1.3 – Каскадна (водоспадна, класична) модель ЖЦ ІС

На практиці, все ж доводиться повертатися до попередніх етапів і в цьому випадку, останнім часом найбільш затребуваною стала итерационная модель ЖЦ ІС.

***Ітераційна модель життєвого циклу ІС (поетапна модель з проміжним контролем)***

Розробка ІС ведеться ітераціями з циклами зворотного зв'язку між етапами. Міжетапні коригування дозволяють враховувати реально існуюче взаємовплив результатів розробки на різних етапах; час життя кожного з етапів розтягується на весь період розробки.

При цьому трудомісткість робіт і тимчасові витрати істотно скорочуються в порівнянні з Водоспадної моделлю життєвого циклу. Ітераційна модель ЖЦ інформаційної системи представлена на рис.1.4.

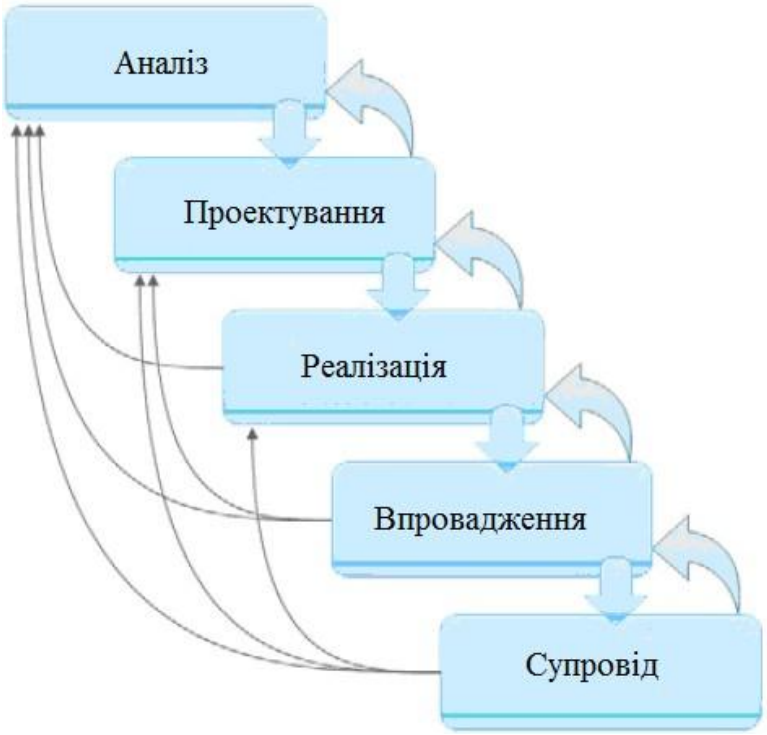



Рисунок 1.4 − Ітераційна модель ЖЦ ІС

Однак і ця схема не дозволяє оперативно враховувати виникаючі зміни і уточнення вимог до системи. Узгодження результатів розробки з користувачами проводиться тільки в точках, планованих після завершення кожного етапу робіт, а загальні вимоги до ІС зафіксовані у вигляді технічного завдання на весь час її створення. Таким чином, користувачі часто отримують систему, не задовольняє їх реальним потребам.

***Спіральна модель життєвого циклу інформаційної системи***

Спіральна модель була запропонована для подолання основних проблем каскадної моделі.

Спіральна модель запропонована Баррі Боем в 1988 році і визначає, в основному стартові етапи життєвого циклу інформаційної системи. При цьому обґрунтовується і перевіряється можливість реалізації спроектованих технічних рішень.

На кожному витку створюється прототип проектованої інформаційної системи, який на наступних витках спіралі ЖЦ ІС вдосконалюється, доповнюється і доводиться до повного впровадження. При цьому необов'язково чекати закінчення кожного етапу, дана модель дозволяє переходити на наступні витки спіралі і вирішувати проблеми або недоліки на

наступному рівні, що робить роботу над проектом більш ефективної, гнучкої і завершити в більш стислі терміни.

Новий виток спіралі відповідає поетапній моделі створення фрагмента інформаційної системи.

При використанні спіральної моделі ЖЦ:

- відбувається орієнтація на модернізацію інформаційної системи;
- здійснюється акумулювання всіх рішень в процесі проектування і створення моделей і прототипів інформаційної системи;
- проводиться аналіз витрат і всіх ризиків в процесі проектування ІС.

Спіральний процес складається з наступної періодичної послідовності:

1. Визначення вимог.
2. Аналіз.
3. Проектування.
4. Реалізація та тестування.
5. Інтеграція.
6. Впровадження.

*Переваги спіральної моделі*

Спіральний підхід дозволяє подолати більшість недоліків каскадної моделі і забезпечує ряд додаткових можливостей, роблячи процес створення інформаційної системи більш гнучким.

До основних переваг спіральної моделі можна віднести:

- спрощення внесення змін до проекту при зміні вимог замовника;
- поступова інтеграція окремих елементів інформаційної системи в єдине ціле;
- зменшення рівня ризиків, які зазвичай виявляються під час інтеграції інформаційної системи;
- гнучкість в управлінні проектом, дається можливість внесення тактичних змін в розробку моделі;
- спрощення повторного використання окремих компонентів системи;
- отримання більш надійної і стійкої системи, так як помилки і слабкі місця виявляються і виправляються на кожній ітерації;
- вдосконалення процесу розробки від ітерації до ітерації.

Основна проблема спірального циклу − визначення моменту переходу на наступний етап. Для її вирішення необхідно ввести тимчасові обмеження на кожен з етапів життєвого циклу. Перехід здійснюється відповідно до плану, навіть якщо не вся запланована робота закінчена. План складається на

основі статистичних даних, отриманих в попередніх проектах, і особистого досвіду розробників.

Кожна з цих стадій створення систем передбачає виконання певного обсягу робіт, які подаються у вигляді процесів ЖЦ. Процес визначає як сукупність взаємопов'язаних дій, що перетворюють вхідні дані у вихідні. Опис кожного процесу включає в себе перелік вирішуваних завдань, вихідних даних і результатів.

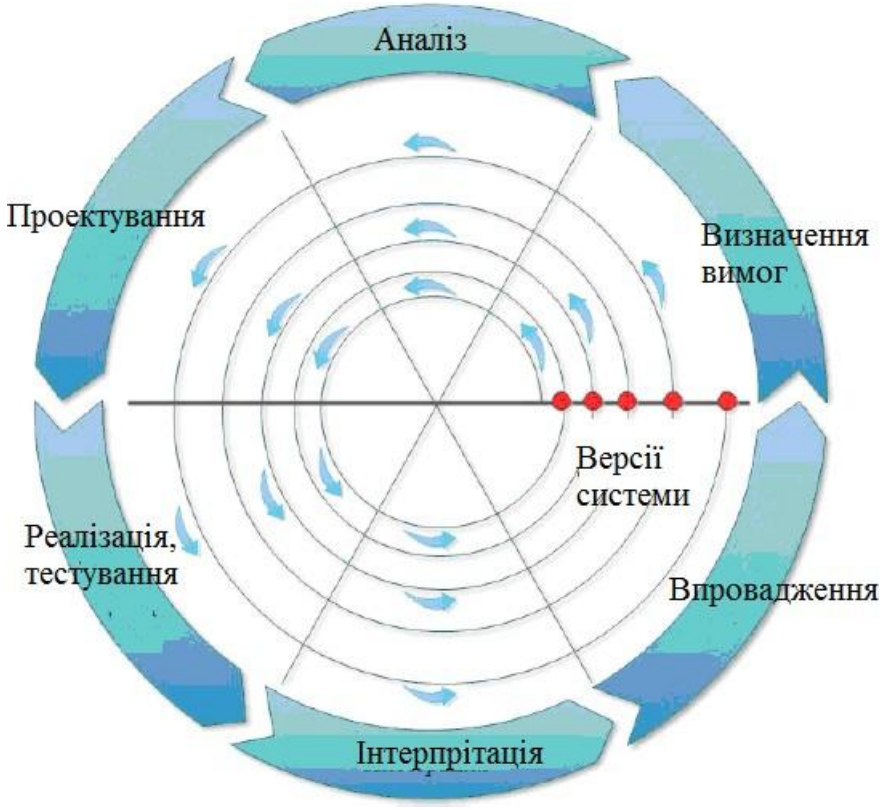



Рисунок 1.5 – Спіральна модель ЖЦ ІС

# 2 ФУНКЦІЇ ТА ВИМОГИ ДО ІС

## 2.1 Вимоги до ІС

При створенні і розвитку системи має бути забезпечено виконання основних і спеціальних вимог, а також вимог до стадартізаціі і уніцфікаціі.

***Основні вимоги, що пред'являються до системи***

*Системність*, яка полягає в раціональній декомпозиції системи, в тому числі на компоненти і підсистеми системи, що надає можливість автономної розробки та впровадження складових частин системи на основі єдиної технічної політики, що забезпечує цілісність системи при її взаємодії зі змінною зовнішнім середовищем.

*Відкритість*, складається в здатності системи до розширення складу послуг, які надаються і технологій. Та збільшення числа джерел інформації і користувачів без порушення її внутрішнього функціонування і погіршення експлуатаційних характеристик.

*Стандартизація* (уніфікація), складається в раціональному застосуванні типових, уніфікованих або стандартизованих проектних рішень і технологій, внутрішніх та зовнішніх інтерфейсів і протоколів, що закладає фундамент для блочного, модульної побудови компонентів і підсистем системи в цілому.

*Здійснення узгоджених між собою процесів* проектування і поетапної модернізації структурних складових системи, що забезпечують її постійну адаптацію до мінливих вимог користувачів.

***Спеціальні вимог, що пред'являються до системи***

*Повнота інформації,* забезпечує ефективну інформаційно-аналітичну підтримку органів державної влади суб'єктів та органів місцевого самоврядування при вирішенні повного комплексу функціональних задач в рамках управління реалізацією пріоритетних національних проектів і соціально-економічним розвитком суб'єктів.

*Ієрархічність* − подання в систему інформації незалежно від організаційного та територіального розподілу користувачів системи.

*Семантична єдність*, складається в здійсненні комплексу заходів, покликаних забезпечити формування єдиного інформаційного простору при створенні і розвитку системи і її підсистем (термінологічна система показників, формати представлення даних, регламенти звітності).

*Переносність елементів системи*, складається в забезпеченні можливості функціонування розроблених компонентів системи на будь-яких однотипних елементах інформаційно-телекомунікаційної інфраструктури.

*Комплексна безпека*, яка полягає в здійсненні комплексу заходів, покликаних забезпечити захист системи від випадкових або навмисних впливів природного або штучного характеру, пов'язаних з можливістю нанесення збитку системі і її користувачам.

Інформаційна система повинна відповідати *вимогам гнучкості*, *надійності*, *ефективності* та *безпеки*.

***Гнучкість***

Гнучкість, здатність до адаптації і подальшого розвитку має на увазі можливість пристосування інформаційної системи до нових умов, новим потребам підприємства. Виконання цих умов можливо, якщо на етапі розробки інформаційної системи використовувалися загальноприйняті засоби і методи документування, так що після певного часу збережеться можливість розібратися в структурі системи і внести в неї відповідні зміни, навіть якщо всі розробники або їх частину з яких-небудь причин не зможуть продовжити роботу.

Будь-яка інформаційна система рано чи пізно морально застаріє, і постане питання про її модернізації або повної заміни. Розробники інформаційних систем, як правило, не є фахівцями в прикладній області, для якої розробляється система. Участь в модернізації або створення нової системи тієї ж групи проектувальників істотно скоротить терміни модернізації. Разом з тим виникає ризик застосування застарілих рішень при модернізації системи. Рекомендація в такому випадку одна - уважніше ставитися до підбору розробників інформаційних систем.

***Надійність***

Надійність інформаційної системи має на увазі її функціонування без спотворення інформації, втрати даних з «технічних причин». Вимога надійності забезпечується створенням резервних копій інформації, що зберігається, виконання операцій протоколювання, підтриманням якості каналів зв'язку і фізичних носіїв інформації, використанням сучасних програмних і апаратних засобів. Сюди ж слід віднести захист від випадкових втрат інформації в силу недостатньої кваліфікації персоналу.

***Ефективність***

Система є ефективною, якщо з урахуванням виділених їй ресурсів вона дозволяє вирішувати покладені на неї завдання у мінімальні терміни. У будь-

якому випадку оцінка ефективності буде проводитися замовником, виходячи з вкладених в розробку засобів і відповідності представленої інформаційної системи його очікуванням.

Негативної оцінки ефективності інформаційної системи з боку замовника можна уникнути, якщо представники замовника будуть залучатися до проектування системи на всіх його стадіях. Такий підхід дозволяє багатьом кінцевим користувачам вже на етапі проектування адаптуватися до змін умов роботи, які інакше були б прийняті вороже.

Активна співпраця з замовником з ранніх етапів проектування дозволяє уточнити потреби замовника. Часто зустрічається ситуація, коли замовник чогось хоче, але сам не знає чого саме. Чим раніше будуть враховані доповнення замовника, то з меншими витратами і в коротші терміни система буде створена.

Крім того, замовник, не будучи фахівцем в області розробки інформаційних систем, може не знати про нових інформаційних технологіях. Контакти з замовником під час розробки для нього інформаційної системи можуть підштовхнути замовника до модернізації його апаратних засобів, застосування нових методів ведення бізнесу, що відповідає потребам, як замовника, так і проектувальника. Замовник отримує зростання ефективності свого підприємства, проектувальник - розширення можливостей, що застосовуються при проектуванні інформаційної системи.

Ефективність системи забезпечується оптимізацією даних і методів їх обробки, застосуванням оригінальних розробок, ідей, методів проектування.

Не слід забувати і про те, що працювати з системою доведеться звичайним людям, що є фахівцями у своїй предметній області, але часто мають вельми середніми навичками в роботі з комп'ютерами. Інтерфейс інформаційних систем повинен бути їм інтуїтивно зрозумілий. У свою чергу, розробник-програміст повинен розуміти характер виконуваних кінцевим користувачем операцій. Рекомендаціями в цьому випадку можуть служити підвищення ефективності управління розробкою інформаційних систем, поліпшення інформованості розробників про предметної області.

***Безпека***

Під безпекою, перш за все, мається на увазі властивість системи, в силу якого сторонні особи не мають доступу до інформаційних ресурсів організації, крім тих, які для них призначені. Захист інформації від стороннього доступу забезпечується управлінням доступу до ресурсів системи, використанням сучасних програмних засобів захисту інформації. У

великих організаціях доцільно створювати підрозділи, основним напрямком діяльності яких було б забезпечення інформаційної безпеки, в менш великих організаціях призначати співробітника, відповідального за дану ділянку роботи.

Крім злого умислу, при забезпеченні безпеки інформаційних систем доводиться стикатися ще з кількома факторами. Зокрема, сучасні інформаційні системи є досить складними програмними продуктами. При їх проектуванні з високою ймовірністю можуть виникати неточності, викликані великим обсягом програмного коду, недосконалістю компіляторів, людським фактором, несумісністю з використовуваними програмами сторонніх розробників у разі модифікації цих програм і т.п. Тому за фазою розробки інформаційної системи неминуче слідує фаза її супроводу в процесі експлуатації, в якій відбувається виявлення прихованих помилок і їх виправлення.

Вимога безпеки забезпечується сучасними засобами розробки інформаційних систем, сучасною апаратурою, методами захисту інформації, застосуванням паролів і протоколированием, постійним моніторингом стану безпеки операційних систем і засобів їх захисту.

***Вимоги, що пред'являються до інформаційних систем***

Обгрунтоване і ретельне формування вимог до інформаційної системи − необхідна умова успішного виконання робіт по створенню системи.

Початок формування вимог пов'язано вже з першої (передпроектної) стадією створення системи, коли проводиться обгрунтування доцільності розробки.

Чим повніше, більш обгрунтованими будуть сформульовані вимоги на початковому етапі (на стадії ТЗ), тим успішніше (швидше, дешевше) може виявитися процес створення системи.

Вимоги до ІС ділять на три групи:

**1. Вимоги до системи в цілому**

Включають в себе

*1.1. Вимоги до структурних характеристиках і режимам функціонування системи:*

- склад основних функцій (склад функціональних підсистем);
- об'єктна структура системи (число рівнів ієрархії, основні об'єктні підсистеми на кожному рівні);

- вимоги до засобів і способів обміну інформацією між об'єктними підсистемами в разі їх територіальної роз'єднаності;
- вимоги до інтегрованості (сумісності) з суміжними системами або вже реалізованими елементами створюваної системи, з якими повинна бути забезпечена можливість взаємодії;
- вимоги до режимів функціонування системи (пакетний, інтерактивний).

*1.2. Вимоги до показників призначення*, т.ї. до найважливіших характеристик системи, які визначають ступінь відповідності системи з її основним призначенням.

*1.3. Вимоги до надійності:*

- перелік відмов системи або її частин, за якими слід висувати вимоги щодо надійності;
- склад і кількісні значення (норми) показників надійності за типами відмов для системи або її елементів;
- вимоги до методів оцінки і контролю надійності на різних етапах створення системи (життєвого циклу системи).

*1.4. Вимоги до якості даних:*

- показники достовірності даних (що вводяться, зберігаються, видаються системою) і їх кількісні значення;
- ситуації (події), при яких повинна бути забезпечена схоронність даних;
- можливі способи несанкціонованого доступу до даних, від яких система повинні бути захищена;

*1.5. Вимоги по стандартизації та уніфікації*: використані стандарти при створенні системи документообігу, які використовуються класифікатори, вимоги щодо застосування типових програмних і технічних засобів при створенні системи.

*1.6. Вимоги до розвитку системи*: можливості модифікації, включення нових функцій, відкритості (можливості взаємодії з іншими системами), масштабованості (збільшення числа користувачів, числа підключаються терміналів та ін.)

**2. Вимоги до функцій (завдань), що виконуються системою**

Включають в себе:

- переліки задач по кожній функціональній підсистемі (комплексу інформаційних технологій) з їх розподілом за рівнями системи;

- вимоги до якості реалізації кожної функції (задачі, комплексу задач);
- форми подання вхідної та вихідної інформації;
- -тимчасової регламент (вимоги до часових параметрів); вимоги до якості результатів (вірогідності видаваної інформації, точності розрахунків і т. д.).

**3. Вимоги до видів забезпечення (інформаційного, технічного, програмного)**

Склад вимог до видів забезпечення залежить від типу і призначення системи.

Вимоги до *інформаційного забезпечення* можуть включати в себе вимоги до якості даних, складу і способу організації даних, їх сумісності із суміжними системами, використання класифікаторів та уніфікованих документів, методам контролю, зберігання, обновленія і відновлення даних.

До складу вимог до *програмного забезпечення* можуть входити вимоги до якості програмних засобів, до інтерфейсів, що використовуються мов програмування, операційній системі.

До складу вимог до *технічного забезпечення* можуть входити вимоги до функціональних, конструктивних, експлуатаційних характеристик окремих видів апаратних засобів, наприклад, до швидкодії засобів передачі даних, продуктивності засобів обчислень, обсягами запам'ятовуючих пристроїв, надійності окремих пристроїв або комплексів.

Перераховані вище вимоги можуть бути представлені у вигляді:

- списку необхідних елементів (список задач; перелік способів несанкціонованого доступу до даних, проти яких система повинна бути захищена);
- переліку можливих елементів, наприклад, вказівка на те, що в якості лінії зв'язку можуть бути використані оптоволоконні лінії або мідні дроти (вита пара) ;
- вимог якісного типу, наприклад, вимога відкритості, масштабованості та ін;
- кількісних показників (норм значень відповідних показників) − вимоги до надійності, достовірності інформації, тимчасовим характеристикам.

Класифікація інформаційних систем ділиться на ряд ознак: призначення, структура апаратних засобів, режим роботи, вид діяльності.

Розробка переліку вимог до інформаційної системи описується нормативними документи, які визначають такі стадії створення інформаційних систем:

- формування вимог до системи;
- розробка концепції;
- технічне завдання;
- ескізний проект, технічний проект;
- робоча документація;
- введення в дію,
- супровід.

Основний зміст стадії формування вимог − збір даних і аналіз об'єкта, для підтримки функціонування якого передбачається створення ІС, аналіз існуючої інформаційної системи (включаючи вивчення інформаційних потоків, виявлення недоліків) і обгрунтування доцільності створення ІС. На цій стадії проводяться наступні підготовчі операції для формування вимог:

- вибір показників;
- виявлення чинників, що обумовлюють доцільність створення;
- виявлення вимог користувачів.

На стадії розроблення концепції ІС здійснюється пошук шляхів задоволення вимог користувача на рівні концепції створюваної системи (структура, функції, програмно-технічна платформа, режими).

На стадії технічне завдання розробляється технічне завдання (ТЗ) на ІС. Склад і зміст технічного завдання визначені в нормативних документах. Основою ТЗ є вимоги до створюваної системи.

На стадії ескізний проект проводиться опрацювання попередніх проектних рішень по системі і її частинам. Ця стадія може бути об'єднана зі стадією технічний проект.

На стадії технічний проект здійснюється розробка основних проектних рішень по системі і її частинам: визначення функціональної структури; вибір комплексу технічних засобів; вибір системи управління базами даних (СКБД) і проектування баз даних, вхідних і вихідних форм; розробка технології обробки інформації, що забезпечує виконання вимог, що пред'являються до даних; розробка алгоритму і обробка даних при виконанні різних функцій. На цій стадії здійснюється розробка проектної документації на систему і її частини, необхідної для створення системи.

На стадії робоча документація проводиться розробка програмних засобів системи, здійснюється адаптація придбаних програмних засобів,

готується робоча документація, яка містить відомості, необхідні і достатні для набрання нею чинності і експлуатації ІС.

Стадія введення в дію включає в себе виконання будівельно-монтажних робіт, організаційну підготовку до введення ІС в дію, навчання персоналу, пуско-налагоджувальні роботи, дослідну експлуатацію (з необхідною доопрацюванням ІС за її результатами), приймальні випробування.

На стадії супроводження ІС здійснюється гарантійне та післягарантійне обслуговування АС, проводиться аналіз функціонування АС, виявляються відхилення експлуатаційних характеристик і встановлюються їх причини, вносяться необхідні зміни до документації

## 2.2 Технологія проектування ІС

Сучасні інформаційні технології пропонують широкий набір способів реалізації ІС, вибір яких здійснюється на основі вимог з боку майбутніх користувачів. Потреба у створенні ІС може обумовлюватися необхідністю автоматизації чи модернізації існуючих інформаційних процесів, або необхідністю корінної реорганізації в діяльності організації (проведенні бізнес-реінжинірингу). Потреба у створенні ІС вказує на те, по-перше, для досягнення якої саме мети необхідно розробити систему; по-друге, до якого моменту часу доцільно здійснити розробку; по-третє, які витрати необхідно здійснити для проектування системи.

*Мета проектування ІС* – створити проект інформаційної системи, який становить технічна документація з докладним описом усіх проектних рішень щодо створення й експлуатації ІС.

Для теорії ухвалення рішень процес проектування ІС – це процес ухвалення проектно-конструкторських рішень, спрямованих на одержання опису системи (проекту ІС), що задовольняє вимоги замовника.

Під проектом ІС слід розуміти проектно-конструкторську і технологічну документацію, в якій наведений опис проектних рішень щодо створення й експлуатації ІС у конкретному програмно-технічному середовищі.

*Під проектуванням ІС* мається на увазі процес перетворення вхідної інформації про об’єкт проектування, про методи проектування і про досвід проектування об’єктів аналогічного призначення в проект ІС відповідно до державних стандартів (рис.2.1). З цього погляду проектування ІС зводиться

до послідовної формалізації проектних рішень на різних стадіях життєвого циклу ІС: планування й аналізу вимог, технічного та робочого проектування, впровадження й експлуатації ІС.

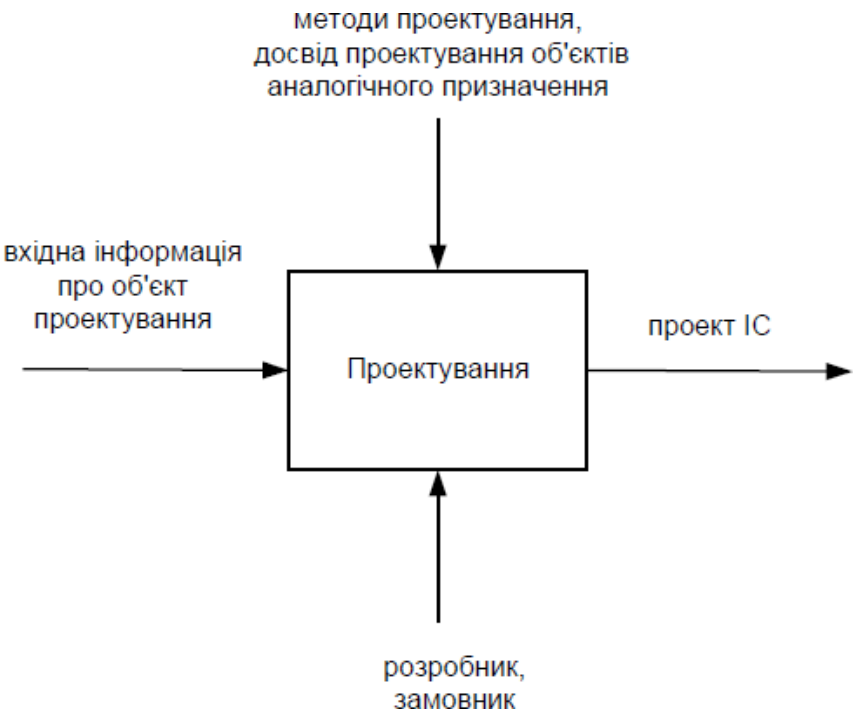



Рисунок 2.1 – Проектування ІС

Об’єктами проектування ІС є окремі елементи або комплекси їх функціональних і забезпечувальних частин. Так, функціональними елементами відповідно до традиційної декомпозиції є завдання, комплекси завдань і функції управління. У складі забезпечувальних частин ІС об’єктами проектування слугують елементи і комплекси інформаційного, програмного та технічного забезпечення системи.

Предметною областю проектування ІС є система організаційно-економічного управління діяльністю підприємства.

Суб’єктами проектування ІС є колективи фахівців, що здійснюють проектну діяльність, а саме:

– організація-розробник, як правило, це спеціалізована (проектна) організація;
– організація-замовник, для якої необхідно розробити ІС.

Масштаби розроблюваних систем визначають склад і кількість учасників процесу проектування. За великого обсягу й жорстких строків виконання проектних робіт у розробці системи може брати участь декілька проектних колективів (організацій-розробників). У цьому випадку

виділяється головна організація, що координує діяльність всіх організацій-співвиконавців.

Форма участі співвиконавців у розробці проекту системи може бути різною. Найбільш розповсюдженою є форма, за якої кожен співвиконавець виконує проектні роботи від початку до кінця для якої-небудь частини розроблюваної системи. Зазвичай це буває функціональна підсистема або взаємозалежний комплекс завдань управління. Рідше зустрічається форма участі співвиконавців, за якої окремі співвиконавці виконують роботи на окремих етапах процесу проектування. Можливий варіант, за якого функції замовника і розробника поєднуються. У такому випадку ІС проектується власними силами.

Організація проектування передбачає визначення методів взаємодії проектувальників між собою і з замовником у процесі створення проекту ІС, що можуть також підтримуватися набором специфічних засобів.

Проектування ІС передбачає використання проектувальниками певної технології проектування, що відповідає масштабу й особливостям розроблюваного проекту.

*Технологія проектування ІС* – це сукупність методів і засобів проектування ІС, організації проектування (рис. 2.2).

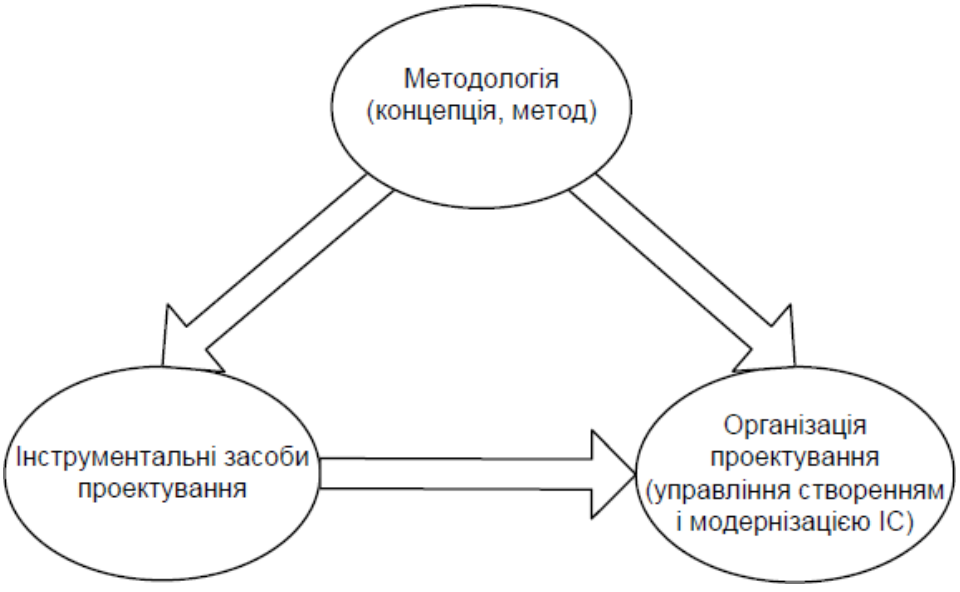



Рис. 2.2 – Компоненти технології проектування

Технологію проектування можна розглядати як сукупність наступних складових:

– покрокової процедури, яка визначає послідовність технологічних опе- рацій проектування;
– критеріїв і правил, що використовуються для оцінки результатів виконання технологічних операцій;
– графічних і текстових засобів (нотацій), які використовуються для опису проектованої системи.

В основі технології проектування лежить технологічний процес, що визначає дії, їхню послідовність, склад виконавців, засоби і ресурси, необхідні для виконання цих дій. Технологічний процес проектування ІС у цілому поділяється на сукупність послідовно-паралельних, зв'язаних і супідрядних ланцюжків дій, кожен із яких може мати свій предмет. Дії, що виконуються в процесі проектування ІС, можуть бути визначені як неподільні технологічні операції (рис.2.3) або як підпроцеси технологічних операцій.

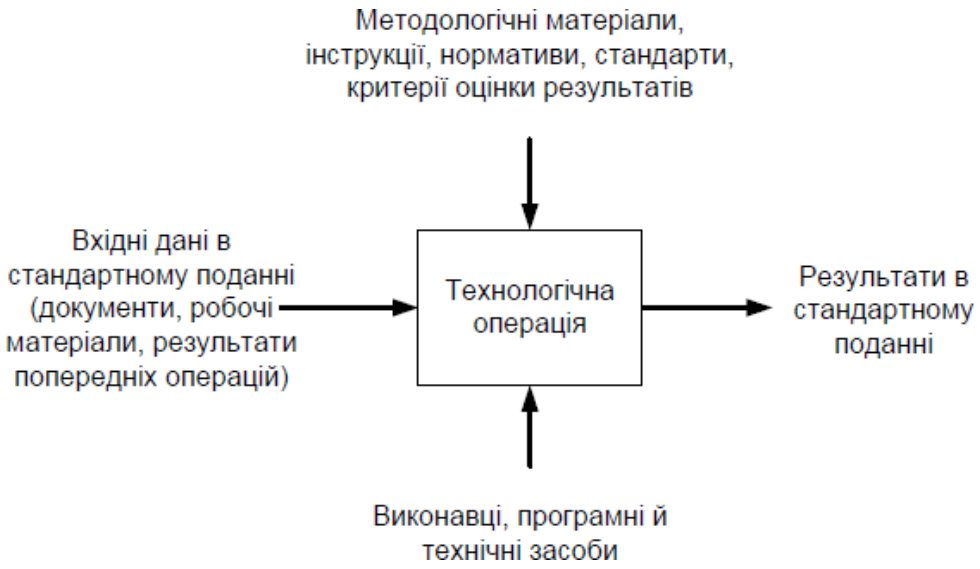



Рисунок 2.3 –Технологічна операція проектування

Усі технологічні операції можна розподілити на два основних види:
– проектувальні, які формують або модифікують результати проектування;
– оцінні, які виробляють за встановленими критеріями оцінки результатів проектування.

Таким чином, технологія проектування задається регламентованою послідовністю технологічних операцій, виконуваних у процесі створення проекту на основі того чи іншого методу, в результаті чого стало б

зрозумілим не тільки те, що повинно бути зроблене для створення проекту, але й те, ким і в якій послідовності це повинно бути зроблене.

Технологічні інструкції є основною складовою технології проектування. Вони складаються з опису послідовності технологічних операцій проектування, умов, залежно від яких виконується та або інша технологіч-на операція, та опису самих операцій.

Предметом будь-якої обраної технології проектування повинне слугувати відображення взаємозалежних процесів проектування на всіх стадіях життєвого циклу ІС.

До основних вимог, які висуваються до обраної технології проектування, відносяться наступні:

- створений за допомогою цієї технології проект повинен відповідати вимогам замовника;

До основних вимог, які висуваються до обраної технології проектування, відносяться наступні:

- створений за допомогою цієї технології проект повинен відповідати вимогам замовника;
- обрана технологія повинна максимально відбивати всі етапи циклу життя проекту;
- обрана технологія повинна забезпечувати мінімальні трудові та вартісні витрати на проектування й супровід проекту;
- технологія повинна бути основою зв’язку між проектуванням і супроводом проекту;
- технологія повинна сприяти зростанню продуктивності праці проектувальника;
- технологія повинна забезпечувати надійність процесу проектування й експлуатації проекту;
- технологія повинна сприяти простому веденню проектної документації; технологія повинна забезпечувати надійність процесу проектування й експлуатації проекту;
- технологія повинна сприяти простому веденню проектної документації;
- технологія повинна забезпечити можливість виконання великих проектів у вигляді підсистем, тобто можливість декомпозиції проекту на складові частини, що розробляються групами виконавців обмеженої чисельності з наступною інтеграцією складових частин;

- технологія повинна забезпечити можливість ведення робіт з проектування окремих підсистем невеликими групами (3 – 7 осіб), що обумовлено принципами управління колективом і підвищенням продуктивності за рахунок мінімізації зовнішніх зв’язків;
- технологія повинна забезпечити мінімальний час отримання працездатної ІС за рахунок реалізації за більш короткі строки окремих підсистем меншою кількістю розробників;
- технологія повинна передбачати можливість управління конфігурацією проекту, ведення версій проекту і його складових, можливість автоматичного випуску проектної документації і синхронізацію її версій з версіями проекту;
- технологія повинна забезпечувати незалежність виконуваних проектних рішень від засобів реалізації ІС – СУБД, ОС, мов і систем програмування;
- технологія повинна бути підтримана комплексом так званих CASE-засобів, що забезпечують автоматизацію процесів, виконуваних на всіх стадіях процесу створення ІС.

Застосування будь-якої технології проектування конкретного проекту для конкретної організації неможливе без вироблення ряду стандартів, правил і угод, яких повинні дотримуватися усі учасники проекту. До них відносяться наступні стандарти:

- проектування; проектування;
- оформлення проектної документації;
- призначеного для користувача інтерфейсу.

Стандарт проектування повинен встановлювати:

- набір необхідних діаграм (моделей) на кожній стадії проектування і ступінь їх деталізації;
- правила фіксації проектних рішень на моделях, зокрема правила присвоєння імен об’єктам (включаючи угоди з термінології), набір атрибутів для всіх об’єктів і правила їх заповнення на кожній стадії, правила оформлення діаграм, включаючи вимоги до форми та розмірів об’єктів і т. д.;
- вимоги до конфігурації робочих місць розробників, включаючи настройки операційної системи, настройки CASE-засобів, спільні настройки проекту і т. д.;
- механізм забезпечення спільної роботи над проектом, зокрема: правила інтеграції підсистем проекту, правила підтримки проекту в

однаковому для всіх розробників проекту стані (регламент обміну проектною інформацією, механізм фіксації спільних об’єктів і т. д.), правила перевірки проектних рішень на несуперечливість і т. д. Стандарт оформлення проектної документації повинен встановлювати: комплектність, склад і структуру документації на кожній стадії проектування;

- вимоги до оформлення документації, включаючи вимоги до змісту розділів, підрозділів, пунктів, таблиць і т. д.;
- правила підготовки, розгляду, узгодження і затвердження документації із зазначенням граничних термінів для кожної стадії;
- вимоги до настройки видавничої системи, використовуваної як вбудований засіб підготовки документації;
- вимоги до настройки CASE-засобів для забезпечення підготовки документації відповідно до встановлених вимог.

Стандарт інтерфейсу користувача повинен встановлювати:

- правила оформлення екранів (шрифти і колірну палітру), склад і розташування вікон та елементів управління;
- правила використання клавіатури й мишки;
- правила оформлення текстів допомоги;
- перелік стандартних повідомлень;
- правила обробки реакції користувача.

## 2.2 Методологія проектування ІС

Методологія проектування складає основу проекту будь-якої ІС. Методологія проектування реалізується за допомогою конкретних технологій, стандартів, що їх підтримують, методів та інструментальних засобів.

Основу технології проектування ІС складає методологія проектування, яка визначає сутність, основні відмінні технологічні особливості проектування. Методологія проектування передбачає наявність певної концепції, принципів проектування реалізованих набором методів проектування, які, у свої чергу, повинні підтримуватися певними засобами проектування.

Методи проектування – це різні способи створення ІС, що підтримуються відповідними засобами проектування. Для організації

процесу проектування й автоматизації виконання проектних робіт застосовуються різні засоби проектування.

Методи проектування ІС можна класифікувати за ступенем використання засобів автоматизації, типових проектних рішень, адаптивності до передбачуваних змін (рис.2.4).

За ступенем автоматизації методи проектування розподіляються на методи:

- ручного проектування, за якого проектування компонентів ІС здійснюється без використання спеціальних інструментальних програмних засобів, а програмування – алгоритмічними мовами;
- комп'ютерного проектування, за якого генерація або конфігурація (настроювання) проектних рішень виконується на основі використання спеціальних інструментальних програмних засобів.

За ступенем використання типових проектних рішень розрізняють наступні методи проектування:

- оригінального (індивідуального) проектування, коли проектні рішення розробляються "з нуля" відповідно до вимог до ІС;
- типового проектування, що передбачає конфігурацію ІС з готових типових проектних рішень (програмних модулів).

Оригінальне проектування ІС характеризується тим, що всі види проектних робіт орієнтовані на створення індивідуальних для кожного об'єкта проектів, що максимально відображає всі його особливості. При цьому можуть створюватися не тільки індивідуальні проекти, але й відповідні методики проведення проектних робіт. Тому технологію оригінального проектування використовують у тому випадку, якщо хочуть, щоб одержуваний у результаті проектування індивідуальний проект повною мірою відображав усі особливості відповідного об'єкта управління за умов невисокої вартості розробки, зрозумілості й доступності одержуваного рішення замовнику.

До числа обмежень з використання оригінального проектування можна віднести низький ступінь автоматизації проектних робіт, тривалі терміни розробки, низьку якість документування, відсутність наступності в проектних рішеннях.

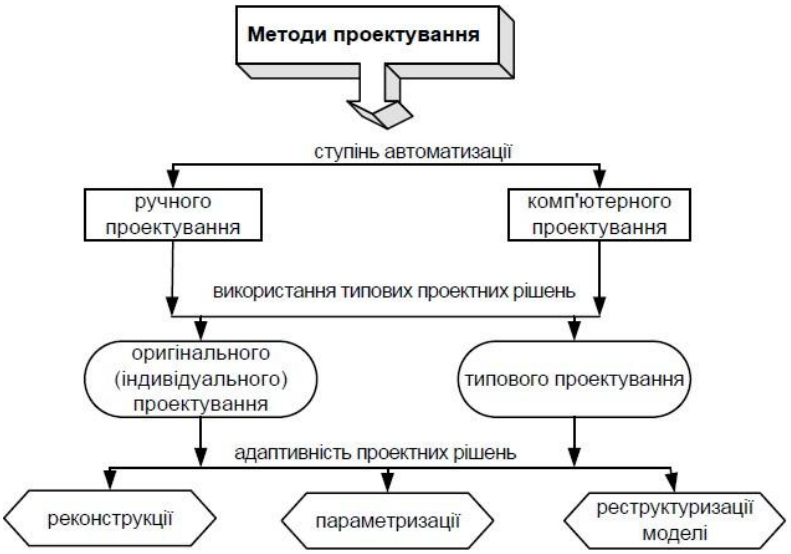



Рисунок 2.4 – Методи проектування

Типове проектування виконується на основі досвіду, отриманого під час розробки індивідуальних проектів. Типові проекти як узагальнення досвіду для деяких груп організаційно-економічних систем чи видів робіт у кожному конкретному випадку пов’язані з безліччю специфічних особливостей і розрізняються за ступенем охоплення функцій управління, виконуваними роботами і розроблюваною проектною документацією.

За ступенем адаптивності проектних рішень методи проектування поділяються на методи:

- реконструкції, коли адаптація проектних рішень виконується шляхом переробки відповідних компонентів (перепрограмування програмних модулів);
- параметризації, коли проектні рішення настроюються (перегенеруються) відповідно до змінюваних параметрів;
- реструктуризації моделі, коли змінюється модель проблемної області, на основі якої автоматично перегенеруються проектні рішення.

## 2.3 Методи і засоби проектування

Поєднання різних ознак класифікації методів проектування обумовлює характер використовуваної технології проектування ІС, серед яких виділяються два основних види технологій: канонічна й індустріальна (табл. 1). Індустріальна технологія проектування, у свою чергу, розбивається на

автоматизоване (використання CASE-технологій) і типове (параметрично-орієнтоване чи модельно-орієнтоване) проектування. Використання індустріальних технологій проектування не виключає використання в окремих випадках канонічної технології.

Характеристика технологій проектування.

Для конкретних видів технологій проектування властиве застосування певних засобів розробки ІС, що підтримують виконання як окремих проектних робіт, етапів, так і їхніх сукупностей. Тому перед розробниками ІС, як правило, постає завдання вибору засобів проектування, що за своїми характеристиками найбільшою мірою відповідають вимогам конкретного підприємства Засоби проектування повинні бути:

- інваріантними до об’єкта проектування;
- охоплювати в сукупності всі етапи життєвого циклу ІС:
- технічно, програмно й інформаційно сумісними;
- простими в освоєнні та застосуванні;
- економічно доцільними.

Засоби проектування ІС можна розподілити на ручні й комп’ютерні (рис. 6).

Ручні засоби проектування застосовуються на всіх стадіях і етапах проектування ІС. Як правило, це засоби організаційно-методичного забезпечення операцій проектування й, у першу чергу, різні стандарти, що регламентують процес проектування систем. Сюди ж відноситься єдина система класифікації і кодування інформації, уніфікована система документації, моделі опису й аналізу потоків інформації і т. д.

Комп’ютерні засоби проектування можуть застосовуватися як на окремих, так і на всіх стадіях та етапах процесу проектування ІС і відповідно підтримують розробку елементів проекту системи, розділів проекту системи, проекту системи в цілому. Усю множину комп’ютерних засобів проектування залежно від їх призначення поділяють таким чином:

- засоби проектування операцій обробки інформації;
- засоби загальносистемного призначення;
- функціональні засоби проектування;
- засоби автоматизації проектування.

До засобів проектування операцій обробки інформації належать алгоритмічні мови, бібліотеки стандартних підпрограм і класів об’єктів, макрогенератори, генератори програм типових операцій обробки даних тощо, а також засоби розширення функцій операційних систем (утиліти).

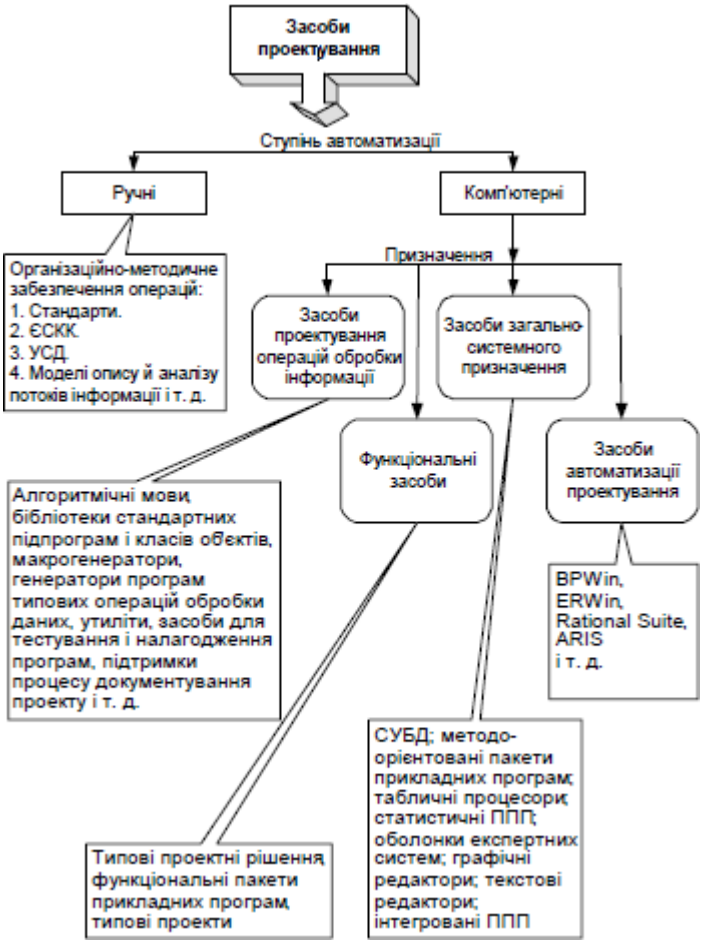



Рисунок 2.5 – Засоби проектування

Сюди включаються також такі найпростіші інструментальні засоби проектування,як засоби для тестування і налагодження програм, підтримки процесу документування проекту і т. д. Особливість останніх програм полягає в тому, що за їх допомогою підвищується продуктивність праці проектувальників, але не розробляється закінчене проектне рішення. Таким чином, засоби даного підкласу підтримують окремі операції проектування ІС і можуть застосовуватися незалежно один від одного.

До засобів загальносистемного призначення відносяться:

- системи управління базами даних;
- методо-орієнтовані пакети прикладних програм (вирішення завдань дискретного програмування, математичної статистики і т. д.);
- табличні процесори;
- статистичні ППП;
- оболонки експертних систем;

- графічні редактори;
- текстові редактори; табличні процесори;
- статистичні ППП;
- оболонки експертних систем;
- графічні редактори;
- текстові редактори;
- інтегровані ППП (інтерактивне середовище із вбудованими діалоговими можливостями, що дозволяє інтегрувати перераховані вище програмні засоби).

Для перерахованих засобів проектування характерне їх використання для розробки технологічних підсистем ІС: введення інформації, організації збереження й доступу до даних, обчислень, аналізу і відображення даних, ухвалення рішень.

Функціональні засоби проектування спрямовані на розробку автоматизованих систем, які реалізують функції, комплекси завдань і завдання управління. Різноманітність предметних областей породжує різноманітність даних засобів, орієнтованих на:

- тип організаційної системи (промислова, непромислова сфери);
- рівень управління (наприклад, підприємство, цех, відділ, ділянка, робоче місце);
- функцію управління (планування, облік і т. д.).

До функціональних засобів проектування систем обробки інформації відносяться:

- типові проектні рішення;
- функціональні пакети прикладних програм;
- типові проекти.

Засоби автоматизації проектування (CASE-засоби) підтримують проектування на стадіях і етапах створення ІС. Термін CASE (Computer Aided System/Software Engineering) тлумачиться досить широко – від автоматизації розробки програмного забезпечення до розробки складних ІС в цілому. Сучасні CASE-засоби, у свою чергу, класифікуються в основ-ному за двома ознаками:

- за охоплюваними етапами процесу розробки ІС;
- за ступенем інтегрованості:
- окремі локальні засоби (tools),

- набір неінтегрованих засобів, що охоплюють більшість етапів розробки ІС (toolkit);
- цілком інтегровані засоби, пов’язані загальною базою проектних даних репозиторієм (workbench).

# 3 СТАНДАРТИ ПРОЕКТУВАННЯ ІС ТА ОФОРМЛЕННЯ ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

## 3.1 Поняття стандартів

В даний час існує декілька стандартів на проектування та розробку інформаційних систем, які можна згрупувати наступним чином:

– по предмету стандартизації;

До цієї групи можна віднести функціональні стандарти (стандарти на мови програмування, інтерфейси, протоколи) і стандарти на організацію життєвого циклу створення і використання інформаціонних систем і програмного забезпечення.

– по організації;

Тут можна виділити офіційні міжнародні, офіційні національні або національні відомчі стандар¬ти (наприклад, ГОСТи, ANSI, IDEFO / і), стандарти міжнародних консорціу¬мов і комітетів по стандартизації (наприклад, консорціуму OMG), стандарти «де-факто» – офіційно ніким не затверджені, але фактично діючі (наприклад, стандартом «де-факто» довгий час були мову взаємодії з базами даних SQL і мова програмування с), фірмові стандарти (наприклад, Microsoft ODBC).

– по методичному джерелу.

До цієї групи належать різного роду методичні матеріали провідних фірм-розробників програмного забезпечення, фірм-консультантів, наукових центрів, консорціумів по стандартизації.

Стандарт проектування повинен встановлювати:

– набір необхідних моделей (діаграм) на кожному етапі проектування та ступінь їх деталізації;
– правила фіксації проектних рішень на діаграмах, в тому числі: правила іменування об'єктів (включаючи угоди за термінологією), набір атрибутів для всіх об'єктів і правила їх заповнення на кожній стадії, правила оформлення діаграм, включаючи вимоги до форми і розмірів об'єктів, і т. д .;
– вимоги до конфігурації робочих місць розробників, включаючи налаштування операційної системи, налаштування CASE-засобів, загальні настройки проекту і т. Д .;
– механізм забезпечення спільної роботи над проектом, в тому числі: правила інтеграції підсистем проекту, правила підтримки проекту в

однаковому для всіх розробників стані (регламент обміну проектної інформацією, механізм фіксації загальних об'єктів і т.д.), правила перевірки проектних рішень на несуперечність і т. д.

Стандарт оформлення проектної документації повинен встановлювати:

- комплектність, склад і структуру документації на кожній стадії проектування;
- вимоги до її оформлення (включаючи вимоги до змісту розділів, підрозділів, пунктів, таблиць і т.д.),
- правила підготовки, розгляду, погодження та затвердження документації із зазначенням граничних строків для кожної стадії;
- вимоги до налаштування видавничої системи, використовуваної в якості вбудованого засобу підготовки документації;
- вимоги до налаштування CASE-засобів для забезпечення підготовки документації відповідно до встановлених вимог.

Стандарт інтерфейсу користувача повинен встановлювати:

- правила оформлення екранів (шрифти і колірна палітра), склад і розташування вікон і елементів управління;
- правила використання клавіатури і миші;
- правила оформлення текстів допомоги;
- перелік стандартних повідомлень;
- правила обробки реакції користувача.

### 3.1.1 Види стандартів

Стандарт – це ухвалений компетентним органом нормативно-технічний документ, що встановлює комплекс норм, правил відносно до предмета стандартизації, або типовий зразок, еталон, модель, що беруться за основу для зіставлення з ними інших предметів.

Існуючі стандарти прийнято ділити на наступні основні види: корпоративні, галузеві, державні й міжнародні.

Корпоративні стандарти розробляються великими фірмами (корпораціями) з метою підвищення якості своєї продукції. Такі стандарти роз-робляються на основі власного досвіду і з урахуванням вимог світових стандартів. Корпоративні стандарти не сертифікуються, але є обов’язковими для застосування всередині корпорації. В умовах ринкової конкуренції вони можуть мати закритий характер. У сфері IT-технологій відомі стандарти, розроблені фірмами IBM, Intel, Microsoft тощо.

Галузеві стандарти діють у межах організацій певної галузі (міністерства, наприклад, стандарти спеціальностей, яким навчають у навчаль-них закладах Міністерства освіти і науки України. Ці стандарти розробляються з урахуванням вимог світового досвіду і специфіки галузі, є, як правило, обов'язковими для галузі і підлягають сертифікації.

Державні стандарти (ГОСТ, ДСТУ) приймаються державними органами, мають силу закону. Розробляються з урахуванням світового досвіду або на основі галузевих стандартів. Можуть мати як рекомендаційний, так і обов'язковий характер, наприклад, стандарти пожежної безпеки. Для сертифікації створюються державні органи або органи ліцензування.

Міжнародні стандарти розробляються, як правило, спеціальними міжнародними організаціями на основі світового досвіду і кращих корпоративних стандартів. Мають суто рекомендаційний характер. Право сертифікації отримують організації (державні й приватні), що пройшли ліцензування в міжнародних організаціях. Основні організації-розробники міжнародних стандартів у сфері програмної інженерії наведені в табл.3.1.

Таблиця 3.1 –Організації-розробники міжнародних стандартів у сфері

| **Програмної інженерії Скорочена назва** | **Назва англійською мовою** | **Назва українською мовою** | **Види стандартів** |
|---|---|---|---|
| **1** | **2** | **3** | **4** |
| ISO | International Organiza-tion for Standardization | Міжнародна організація зі стандартизації | Міжнародні стандарти: якості – ISO 9000; процесів життєвого циклу ПЗ – ISO/IEC 12207 |
| SEI | Software Engineering Institute | Інститут програмної інженерії | Програмна інженерія: зрілості організацій, що розробляють ПЗ, – CMM™ |

| Програмної інженерії Скорочена назва | Назва англійською мовою | Назва українською мовою | Види стандартів |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| PMI® | Project Management Institute | Інститут управління проектами | Загальний менеджмент проектів: основи знань у галузі управління проектами – PMBOK |
| IEEE | Institute of Electrical and Electronics Engi-neers | Інститут інженерів електротехніки та електроніки | Зведення стандартів з програмної інженерії IEEE |

ISO – це неурядова організація з консультативним статусом ООН. Головна її мета – розвиток стандартизації, а також споріднених напрямів діяльності у всьому світі. Міжнародна організація зі стандартизації ISO є всесвітньою федерацією національних організацій зі стандартизації (комітетів-членів ISO). Розробка міжнародних стандартів здійснюється технічними комітетами ISO. Міжнародні урядові й неурядові організації, що мають зв’язки з ISO, також беруть участь у роботах. Стандарти ISO є рекомендаційними; водночас деякі міжнародні стандарти (наприклад, з проблем охорони здоров’я, безпеки, охорони навколишнього середовища) прийняті рядом країн як обов’язкові на території даної країни. Сертифікації на відповідність своїм стандартам і контролю за їх виконанням організація не здійснює. Цим зазвичай займаються спеціально призначені державні органи реєстрації, лабораторії або аудиторські інститути. Ці процедури, зокрема, можуть виконувати на комерційній основі приватні аудитори. Проекти міжнародних стандартів приймаються технічними комітетами і розсилаються комітетам-членам на голосування. Для публікації стандартів у якості міжнародних необхідне схвалення не менше 75% комітетів-членів від числа тих, що беруть участь у голосуванні. Використання логотипу ISO на яких-небудь продуктах або в підприємствах є також незаконним, оскільки передбачає негласну сертифікацію даного продукту.

SEI – організація, яка заснована при університеті Карнегі/Меллон. Дослідження й розробки в організації проводяться на основі федерального фінансування і спонсоруються Міністерством оборони США. Основне завдання SEI – створення методик для оцінки рівня розвитку внутрішніх процесів в організації. З 1984 року організація розвиває і пропагує методики для розробки високоякісного ПЗ.

PMI – це міжнародна громадська організація, яка об'єднує професіоналів у галузі проектного менеджменту, яка була заснована в 1969 р. у США. У PMI беруть участь члени зі 125 країн світу. Основною метою PMI є просування, пропаганда, розвиток проектного менеджменту в різних країнах. Організація розробляє стандарти в галузі проектного менеджменту, а також займається підвищенням кваліфікації кадрів. PMI є провідною професійною організацією з управління проектами в наступних галузях: авіакосмічна й автомобільна промисловість, управління комерційними підприємствами, машинобудування, фінансові операції, інформаційні технології, фармацевтика, телекомунікації та багато інших.

PMI надає всеосяж-не керівництво розробкою стандартів для проектного менеджменту – стандарт з управління проектами PMBOK. PMI став першою організацією у світі, що має програму сертифікації фахівців з управління проектами – Project Management Professional (PMP).

IEEE (вимовляється як ай-тріпл-і) – організація, що об'єднує близько 400 тис. технічних фахівців із більш ніж 150 країн. До IEEE входять ряд професійних товариств. Найбільшим серед них є IEEE Com-puter Society, куди входять більше 100 тис. осіб. IEEE Computer Society широко відоме своєю діяльністю зі стандартизації, яку на сьогодні в рамках товариства здійснюють близько 200 робочих груп. Комп'ютерне товариство IEEE щорічно спонсорує близько ста п'ятдесяти наукових конференцій і симпозіумів, публікує більше 20 періодичних видань.

Розробка і практичне застосування стандартів стосовно ЖЦ ПЗ стикалися з рядом проблем:

- вкладення значних коштів на застосування стандартів, що не завжди окупалося;
- не було ясності, які процеси треба виконувати і в якому обсязі;
- до різних типів програмних систем (ІС організаційного управління, управління технологічними процесами і т. д.) висуваються різні вимоги;

- динамічність ІТ-галузі приводила до швидкого "старіння" стандартів;
- термінологічна невідповідність різних корпоративних стандартів.

Тому світовими організаціями було розроблено ряд стандартів у галузі програмної інженерії, в яких вони намагалися вирішити ці проблеми. Серед найбільш відомих стандартів програмної інженерії можна виділити наступні:

ISO/IEC 12207 – Information Technology – Software Life Cycle Processes – Процеси життєвого циклу програмних засобів;

ISO/IEC 15504 – Software Process Assessment – Оцінка і атестація зрілості процесів створення і супроводу ПЗ;

SEI CMM – Capability Maturity Model (for Software) – Модель зрілості процесів розробки програмного забезпечення;

SEI CMMI (Capability Maturity Model Integration) – Інтеграція моделей зрілості процесів розробки програмного забезпечення;

PMBOK – Project Management Body of Knowledge – Зведення знань з управління проектами. Стандарт PMBOK розроблений і розвивається PMI. Він містить описи 9 розділів (галузей знань) управління проектами:

- управління інтеграцією – Project Integration Management;
- управління обмеженнями – Project Scope Management;
- управління часом – Project Time Management;
- управління витратами – Project Cost Management;
- управління ризиками – Project Risk Management;
- управління персоналом – Project Personnel Management;
- управління комунікаціями – Project Communication Management;
- управління закупівлями – Project Procurement Management;
- управління якістю – Project Quality Management.

IEEE SWEBOK – IEEE Computer Society Software Engineering Body of Knowledge Software Engineering Body of Knowledge – Зведення знань з програмної інженерії – проект IEEE Computer Society. Основна ідея проекту аналогічна до PMBOK і полягає у створенні певного базового набору загальноприйнятих знань, необхідних будь-якому професійному програмістові. Він містить описи наступних 10 розділів (галузей знань) програмної інженерії:

- Software Requirements – вимоги до ПЗ;
- Software Design – проектування ПЗ;
- Software Construction – конструювання ПЗ;
- Software Testing – тестування ПЗ;

- Software Maintenance – супровід ПЗ;
- Software Configuration Management – управління конфігураціями;
- Software Engineering Management – управління IT-проектом;
- Software Engineering Process – процес програмної інженерії;
- Software Engineering Tools and Methods – методи та інструменти;
- Software Quality – якість ПЗ.

Стандарти є сумою досвіду, накопиченого експертами в інженерії ПЗ на основі великої кількості проектів, що проводилися в рамках комерційних структур США і Європи, а також у рамках військових контрактів. Велика частина стандартів створювалася як набір критеріїв для відбору постачальників програмного забезпечення для Міністерства оборони США, і це завдання вони вирішують достатньо успішно.

### 3.1.1.Стандарти процесів ЖЦ

Значними кроками у вирішенні проблем стандартизації ЖЦ ПЗ стали розробка й ухвалення в 1995 р. стандарту ISO/IEC 12207 – Information Technology – Software Life Cycle Processes. У 2000 р. він був прийнятий в Україні як ДСТУ 3918-1999 "Інформаційні технології. Процеси життєвого циклу програмного забезпечення".

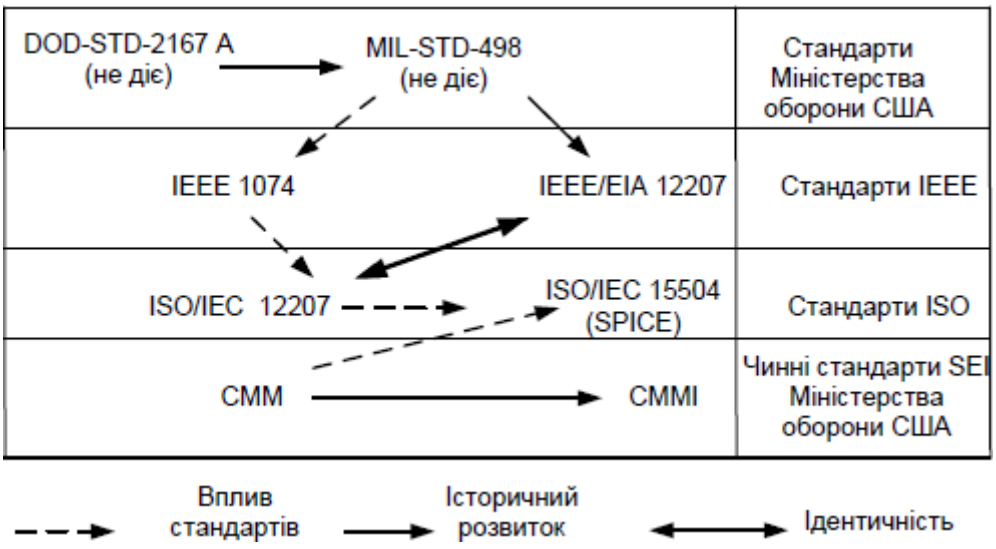



Рисунок 3.1 – Зв’язок між стандартами ЖЦ ПЗ

Стандарт ISO/IEC 12207 розроблявся з урахуванням кращого світового досвіду на основі наступних стандартів (рис. 10):

DOD-STD-2167 A "Розробка програмних засобів для систем військового призначення". Стандарт був прийнятий у 1985 р., а в 1988 р. – уточнений. Це перший формалізований і затверджений стандарт життєвого циклу для проектування програмних систем військового призначення на замовленнях Міністерства оборони США. Цим документом регламентовано 8 фаз (етапів) у процесі створення складних критичних програмних систем і близько 250 типових обов'язкових вимог до процесів і об'єктів проектування на цих етапах;

MIL-STD-498 "Розробка і документування програмного забезпечення". Стандарт прийнятий Міністерством оборони США в 1994 р. для заміни DOD-STD-2167A та ряду інших стандартів. Він призначений для застосування всіма організаціями і підприємствами, що одержують замовлення Міністерства оборони США. У 1996 р. затверджено докладне керівництво "Застосування і рекомендації до стандарту MIL-STD-498". Основну частину складають 75 підрозділів – рекомендацій щодо забезпечення й реалізації процесів ЖЦ складних критичних програмних систем високої якості та надійності, що функціонують у реальному часі;

IEEE 1074 "Процеси життєвого циклу для розвитку програмного забезпечення". Стандарт прийнятий у 1995 р. і охоплює повний життєвий цикл програмних систем, у якому виділяються шість великих базових процесів. Ці процеси деталізуються 16 приватними процесами. В останніх є ще дрібніша деталізація в сукупності на 65 процесів-робіт. Зміст кожного окремого процесу починається з опису загальних його функцій і завдань та переліку дій – робіт під час подальшої деталізації. Для кожного процесу в стандарті наведена вхідна і результівна інформація про його виконання і короткий опис суті процесу. У стандарті увага зосереджена переважно на безпосередньому створенні ПЗ і на процесах попереднього проектування.

Основними результатами стандарту ISO 12207 є:

- введення єдиної термінології з розробки й застосування ПЗ, призначеної не лише для розробників, але й для замовників, користувачів, усіх зацікавлених осіб;
- опис організації ЖЦ і його структури (процесів).

Стандарт поширюється на процеси замовлення систем, програмних продуктів та послуг, процеси постачання, розробки, функціонування та супроводу програмних продуктів, включно з програмним компонентом програмно-апаратних засобів, які виконуються як самою організацією, так і поза її межами. Причому процеси, що використовуються протягом життєвого

циклу програмного забезпечення, повинні бути сумісними з процесами, що використовуються протягом життєвого циклу системи.

У стандарті ISO /IEC 12207 даються визначення термінів стосовно розробки та застосування ПЗ, наведених у стандартах ISO 8402, ISO/IEC 2382-1 та ISO/IEC 2382-20 (додаток А).

Стандарт визначає організацію життєвого циклу програмного продукту як сукупність процесів, кожний із яких розбитий на дії, що складаються з окремих завдань, встановлює структуру (архітектуру) життєвого циклу програмного продукту у вигляді переліку процесів, дій і задач (рис.3.2).

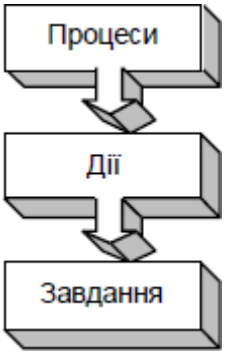



Рисунок 3.2 – Організація ЖЦ ПЗ згідно зі стандартом ДСТУ 3918-1999 (ISO/IEC 12207)

*Процес* – це множина взаємопов’язаних дій і ресурсів, які перетворюють входи на виходи.

*Дія* – це елемент процесу проекту. У дій звичайно наявна очікувана тривалість, потреба в ресурсах, вартість. Дії можуть далі розподілятися на завдання.

Відповідно до стандарту ISO/IEC 12207, процеси організації ЖЦ ПЗ діляться на три групи: основні процеси, процеси підтримки, організаційні процеси. Окремо описаний процес адаптації стандарту, що містить основні роботи, які повинні бути виконані під час адаптації даного стандарту до умов конкретного програмного проекту (табл. 3.2)

До основних процесів життєвого циклу відносяться наступні (рис.3.2).

*Замовлення* – визначає дії замовника, тобто організації, яка замов-ляє систему, програмний продукт або програмну послугу.

*Постачання* – визначає дії постачальника, тобто організації, яка надає систему, програмний продукт або програмну послугу замовнику.

*Розробка* – визначає дії розробника, тобто організації, яка визначає та проектує програмний продукт.

Таблиця 3.2 – Процеси життєвого циклу ПЗ (ISO/IEC 12207)

| Основні процеси | Процеси підтримки | Організаційні процеси | Процеси адаптації |
|---|---|---|---|
| Замовлення<br>Постачання<br>Розробка<br>Експлуатація<br>Супровід | Документування<br>Конфігурування<br>Забезпечення якості<br>Верифікація<br>Валідація<br>Спільний перегляд<br>Аудит<br>Вирішення проблем | Управління<br>Створення інфраструктури<br>Вдосконалення<br>навчання | Адаптація процесів до умов конкретного процесу |

*Експлуатація* – визначає дії оператора, тобто організації, що надає послуги із експлуатації комп'ютерної системи в її наявному середовищі для її користувачів.

*Супровід* – визначає дії супроводжувача, тобто організації, яка надає послуги із супроводу програмного продукту, тобто керує внесенням змін до програмного продукту з метою підтримання його в належному та працездатному стані. Даний процес охоплює перенесення і вилучення з експлуатації програмного продукту.

До процесів підтримки життєвого циклу відносяться.

*Документування* – визначає дії щодо реєстрації інформації, вироб-леної процесом життєвого циклу.

*Конфігурування* – визначає дії щодо управління конфігурацією.

*Забезпечення якості* – визначає дії щодо набуття об'єктивної впев-неності в тому, що програмні продукти та процеси відповідають заданим для них вимогам та відповідають встановленим для них планам. Як методи забезпечення якості можна використовувати процеси спільного перегляду, аудиту, верифікації та валідації.

*Верифікація* – визначає дії (замовника, постачальника або незалежного учасника) щодо проведення верифікації програмного продукту з різним ступенем глибини залежно від програмного проекту.

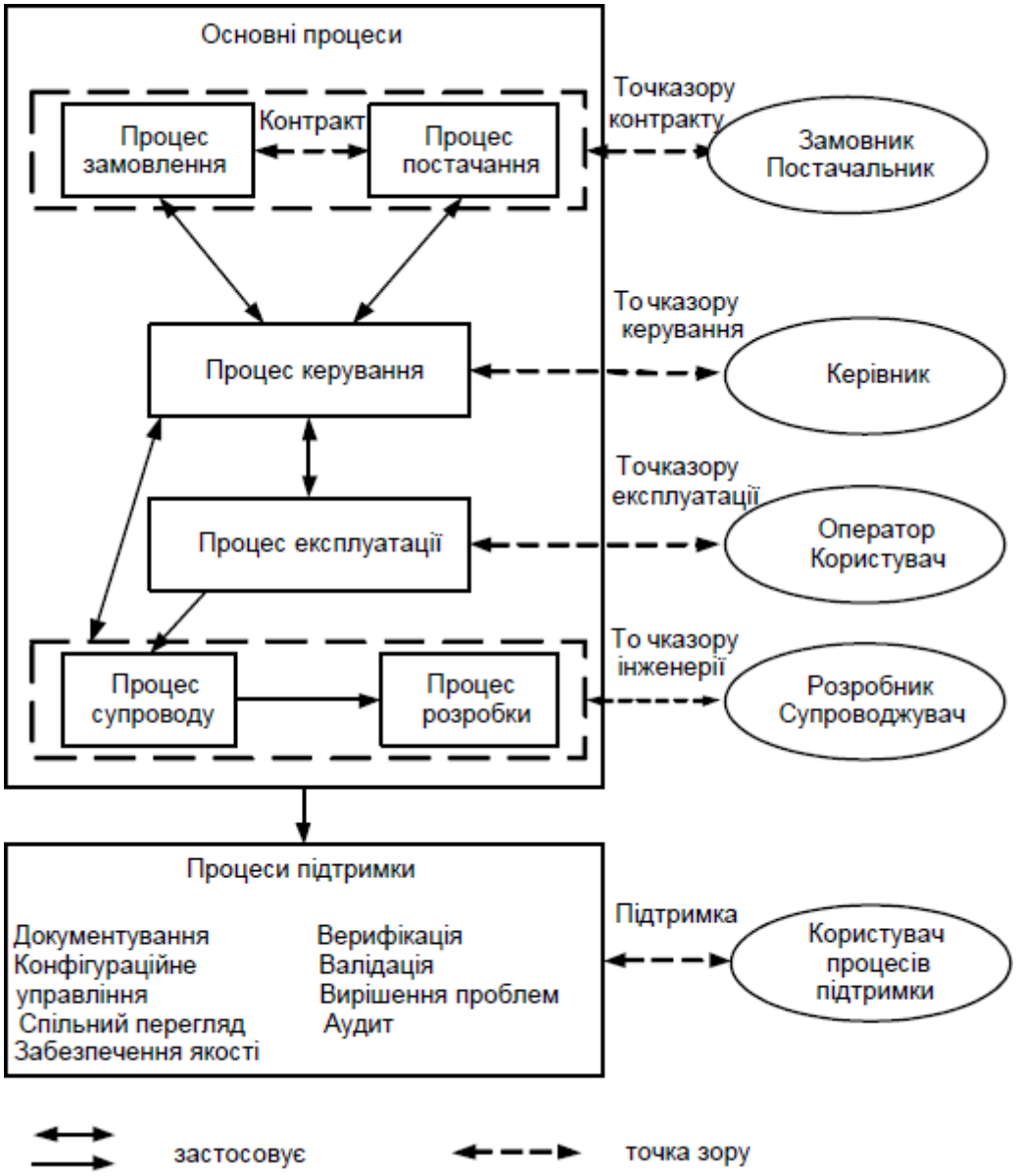



Рисунок 3.2 – Взаємозв’язки між процесами життєвого циклу

*Валідація* – визначає дії замовника, постачальника, або незалежного учасника щодо проведення валідації програмного продукту, одержаного в рамках програмного проекту.

*Спільний перегляд* – визначає дії щодо проведення оцінки стану та результатів певної дії. Цей процес може бути застосований будь-якими двома учасниками, один із яких (учасник, що здійснює перегляд) здій-снює перегляд дій іншого учасника (учасника, дії якого переглядаються в рамках спільного обговорення).

*Аудит* – визначає дії щодо визначення відповідності вимогам, планам та контракту. Цей процес може бути застосований будь-якими двома учас-

никами, один із яких (учасник, що перевіряє) проводить аудит програмного продукту або дій іншого учасника (учасника, якого перевіряють).

*Вирішення проблем* – визначає дії щодо аналізу і зняття проблем (включно з невідповідностями) незалежно від їхньої природи та причин, виявлених у процесі розробки, експлуатації, супроводу чи під час виконання інших процесів.

До організаційних процесів життєвого циклу відносяться.

*Керування* – визначає основні дії щодо управління, включно з керуванням проектом, протягом життєвого циклу.

*Створення інфраструктури* – визначає основні дії щодо створення основної структури процесів життєвого циклу.

*Удосконалення* – визначає базові дії, які організація (якою може бути замовник, постачальник, розробник, оператор, супроводжувач або керівник іншого процесу) виконує з метою створення, вимірювання, кон-тролю та вдосконалення процесів життєвого циклу, які вона проводить.

*Навчання* – визначає дії щодо забезпечення відповідного навчання персоналу.

## 3.2 Канонічне проектування

Організація канонічного проектування ІС орієнтована на використання головним чином каскадної моделі життєвого циклу ІС. Стадії і етапи роботи описані в стандарті ГОСТ 34.601-90.

Залежно від складності об'єкта автоматизації і набору задач, що потребують вирішення при створенні конкретної ІС, стадії і етапи робіт можуть мати різну трудомісткість. Допускається об'єднувати послідовні етапи і навіть виключати деякі з них на будь-якій стадії проекту. Допускається також починати виконання робіт наступної стадії до закінчення попередньої.

Стадії і етапи створення ІС, що виконуються організаціями-учасниками, прописуються в договорах і технічних завданнях на виконання робіт:

**Стадія 1.** Формування вимог до ІС.

На початковій стадії проектування виділяють наступні етапи робіт:

- обстеження об'єкта та обгрунтування необхідності створення ІС;
- формування вимог користувачів до ІС;

– оформлення звіту про виконану роботу і тактико-технічного завдання на розробку.

**Стадія 2.** Розробка концепції ІС.

– вивчення об'єкта автоматизації;
– проведення необхідних науково-дослідних робіт;
– розробка варіантів концепції ІС, які відповідають вимогам користувачів;
– оформлення звіту і затвердження концепції.

**Стадія 3**. Технічне завдання.

Розробка і затвердження технічного завдання на створення ІС.

**Стадія 4.** Ескізний проект.

– розробка попередніх проектних рішень по системі і її частинам;
– розробка ескізної документації на ІС та її частини.

**Стадія 5.** Технічний проект.

– розробка проектних рішень по системі і її частинам;
– розробка документації на ІС та її частини;
– розробка і оформлення документації на поставку комплектуючих виробів;
– розробка завдань на проектування в суміжних частинах проекту.

**Стадія 6.** Робоча документація.

– розробка робочої документації на ІС та її частини;
– розробка і адаптація програм.

**Стадія 7.** Введення в дію.

– підготовка об'єкта автоматизації;
– підготовка персоналу;
– комплектація ІС поставляються виробами (програмними і технічними засобами, програмно-технічними комплексами, інформаційними виробами);
– будівельно-монтажні роботи;
– пусконалагоджувальні роботи;
– проведення попередніх випробувань;
– проведення дослідної експлуатації;
– проведення приймальних випробувань.

**Стадія 8.** Супровід ІС.

– виконання робіт відповідно до гарантійних зобов'язань;
– післягарантійне обслуговування.

Обстеження – це вивчення і діагностичний аналіз організаційної структури підприємства, його діяльності та існуючої системи обробки інформації. Матеріали, отримані в результаті обстеження, використовуються для: обґрунтування розробки та поетапного впровадження систем; складання технічного завдання на розробку систем; розробки технічного і робочого проектів систем.

На етапі обстеження доцільно виділити дві складові: визначення стратегії впровадження ІС і детальний аналіз діяльності організації.

Основне завдання першого етапу обстеження – оцінка реального обсягу проекту, його цілей і завдань на основі виявлених функцій і інформаційних елементів об'єкта, що автоматизується. Ці завдання можуть бути реалізовані або замовником ІС самостійно, або із залученням консалтингових організацій. Етап передбачає тісну взаємодію з основними потенційними користувачами системи і бізнес-експертами. Основне завдання взаємодії – отримати повне і однозначне розуміння вимог замовника. Як правило, потрібна інформація може бути отримана в результаті інтерв'ю, бесід або семінарів з керівництвом, експертами і користувачами.

По завершенні цієї стадії обстеження з'являється можливість визначити ймовірні технічні підходи до створення системи та оцінити витрати на її реалізацію (витрати на апаратне забезпечення, що закуповується програмне забезпечення та розробку нового програмного забезпечення).

Результатом етапу визначення стратегії є документ (техніко-економічне обґрунтування проекту), де чітко сформульовано, що отримає замовник, якщо погодиться фінансувати проект, коли він отримає готовий продукт (графік виконання робіт) та скільки це буде коштувати (для великих проектів повинен бути складений графік фінансування на різних етапах робіт). У документі бажано відобразити не тільки витрати, але і вигоду проекту, наприклад час окупності проекту, очікуваний економічний ефект (якщо його вдається оцінити).

Орієнтовний зміст цього документа:

- обмеження, ризики, критичні чинники, які можуть вплинути на успішність проекту;
- сукупність умов, при яких передбачається експлуатувати майбутню систему: архітектура системи, апаратні і програмні ресурси, умови функціонування, обслуговуючий персонал і користувачі системи;
- терміни завершення окремих етапів, форма прийому / здачі робіт, які залучаються ресурси, заходи щодо захисту інформації;

- опис виконуваних системою функцій;
- можливості розвитку системи;
- інформаційні об'єкти системи;
- інтерфейси і розподіл функцій між людиною і системою;
- вимоги до програмних і інформаційних компонентів ПЗ, вимоги до СУБД;
- що не буде реалізоване в рамках проекту.

На етапі детального аналізу діяльності організації вивчаються завдання, що забезпечують реалізацію функцій управління, організаційна структура, штати і зміст робіт по управлінню підприємством, а також характер підпорядкованості вищим органам управління. На цьому етапі повинні бути виявлені:

- інструктивно-методичні та директивні матеріали, на підставі яких визначаються склад підсистем і перелік завдань;
- можливості застосування нових методів вирішення завдань.

Аналітики збирають і фіксують інформацію в двох взаємопов'язаних формах:

- функції – інформація про події та процеси, які відбуваються в бізнесі;
- сутності – інформація про речі, що мають значення для організації і про які щось відомо.

При вивченні кожної функціональної задачі управління визначаються:

- найменування задачі; терміни і періодичність її рішення;
- ступінь формалізуємості задачі;
- джерела інформації, необхідні для вирішення задачі;
- показники і їх кількісні характеристики;
- порядок коригування інформації;
- діючі алгоритми розрахунку показників і можливі методи контролю;
- діючі засоби збору, передачі і обробки інформації;
- ефективні засоби зв'язку;
- прийнята точність рішення задачі;
- трудомісткість рішення задачі;
- діючі форми представлення вихідних даних і результатів їх обробки у вигляді документів;
- споживачі результатної інформації по задачі.

Однією з найбільш трудомістких, хоча і добре формалізованих задач цього етапу є опис документообігу організації. При обстеженні документообігу складається схема маршруту руху документів, яка повинна відобразити:

- кількість документів;
- місце формування показників документа;
- взаємозв'язок документів при їх формуванні;
- маршрут і тривалість руху документа;
- місце використання і зберігання даного документа;
- внутрішні і зовнішні інформаційні зв'язки;
- обсяг документа в знаках.

За результатами обстеження встановлюється перелік задачі, вирішення яких доцільно автоматизувати, і черговість їх розробки.

На етапі обстеження слід класифікувати плановані функції системи за ступенем важливості. Один з можливих форматів представлення такої класифікації – MuSCoW.

Ця абревіатура розшифровується так: *Must have* – необхідні функції; *Should have* – бажані функції; *Could have* – можливі функції; *Will not have* – відсутні функції.

Функції першої категорії забезпечують критичні для успішної роботи системи можливості.

Реалізація функцій другої і третьої категорій обмежується часовими і фінансовими рамками: розробляється то, що необхідно, а також максимально можливе в порядку пріоритету число функцій другої і третьої категорій.

Остання категорія функцій особливо важлива, оскільки необхідно чітко уявляти межі проекту і набір функцій, які будуть відсутні в системі.

Моделі діяльності організації створюються в двох видах:

1) Модель "як є" ( "as-is") – відбиває існуючі в організації бізнес-процеси;
2) Модель "як повинно бути" ( "to-be") – відбиває необхідні зміни бізнес-процесів з урахуванням впровадження ІС.

На етапі аналізу необхідно залучати до роботи групи тестування для вирішення наступних задач:

- отримання порівняльних характеристик передбачуваних до використання апаратних платформ, операційних систем, СУБД, іншого оточення;

- розробки плану робіт по забезпеченню надійності інформаційної системи і її тестування.

Залучення тестувальників на ранніх етапах розробки є доцільним для будь-яких проектів. Якщо проектне рішення виявилося невдалим і це виявлено занадто пізно (на етапі розробки або, що ще гірше, на етапі впровадження в експлуатацію), то виправлення помилки проектування обходиться дуже дорого. Чим раніше групи тестування виявляють помилки в інформаційній системі, тим нижче вартість супроводу системи. Час на тестування системи і на виправлення виявлених помилок слід передбачати не тільки на етапі розробки, але і на етапі проектування.

Для автоматизації тестування слід використовувати системи стеження за вадами (bug tracking). Це дозволяє мати єдине сховище помилок, відстежувати їх повторній появі, контролювати швидкість і ефективність виправлення помилок, бачити найбільш нестабільні компоненти системи, а також підтримувати зв'язок між групою розробників і групою тестування (повідомлення про зміни по e-mail і т.п.). Чим більше проект, тим сильніше потреба в bug tracking.

Результати обстеження представляють об'єктивну основу для формування технічного завдання на інформаційну систему.

Технічне завдання – це документ, що визначає цілі, вимоги і основні вихідні дані, необхідні для розробки автоматизованої системи управління.

При розробці технічного завдання необхідно вирішити такі задачі:

- встановити спільну мету створення ІС, визначити склад підсистем і функціональних задач;
- розробити і обґрунтувати вимоги, що пред'являються до підсистем;
- розробити і обґрунтувати вимоги, що пред'являються до інформаційної бази, математичного та програмного забезпечення, комплексу технічних засобів (включаючи засоби зв'язку та передачі даних);
- встановити загальні вимоги до проектованої системи;
- визначити перелік задач створення системи і виконавців;
- визначити етапи створення системи і терміни їх виконання;
- провести попередній розрахунок витрат на створення системи і визначити рівень економічної ефективності її впровадження.

Типові вимоги до складу та змісту технічного завдання наведені в ГОСТ 34.

Ескізний проект передбачає розробку попередніх проектних рішень по системі і її частинам.

Виконання стадії ескізного проектування не є строго обов'язковою. Якщо основні проектні рішення визначені раніше або досить очевидні для конкретної ІС і об'єкта автоматизації, то ця стадія може бути виключена із загальної послідовності робіт.

Зміст ескізного проекту задається в ТЗ на систему. Як правило, на етапі ескізного проектування визначаються:

- функції ІС;
- функції підсистем, їх цілі та очікуваний ефект від впровадження;
- склад комплексів задач і окремих завдань;
- концепція інформаційної бази і її укрупненная структура;
- функції системи управління базою даних;
- склад обчислювальної системи і інших технічних засобів;
- функції і параметри основних програмних засобів.

За результатами виконаної роботи оформляється, узгоджується і затверджується документація в обсязі, необхідному для опису повної сукупності прийнятих проектних рішень і достатньому для подальшого виконання робіт зі створення системи.

На основі технічного завдання (і ескізного проекту) розробляється технічний проект ІС.

Технічний проект системи – це технічна документація, яка містить загальносистемні проектні рішення, алгоритми рішення задач, а також оцінку економічної ефективності автоматизованої системи управління і перелік заходів з підготовки об'єкта до впровадження.

На цьому етапі здійснюється комплекс науково-дослідних і експериментальних робіт для вибору основних проектних рішень та розрахунок економічної ефективності системи.

Склад і зміст технічного проекту наведені в табл. 3.1.

На завершення стадії технічного проектування проводиться розробка документації на поставку серійно випускаються виробів для комплектування ІС, а також визначаються технічні вимоги та складаються ТЗ на розробку виробів, які не виготовлялися серійно.

На стадії "робоча документація" здійснюється створення програмного продукту і розробка всієї супровідної документації. Документація повинна містити всі необхідні і достатні відомості для забезпечення виконання робіт по введенню ІС в дію і її експлуатації, а також для підтримки рівня

експлуатаційних характеристик (якості) системи. Розроблена документація повинна бути відповідним чином оформлена, узгоджена і затверджена.

Таблиця 3.2 – Зміст технічного проекту

| **№ п \п** | **Розділ** | **Зміст** |
|---|---|---|
| **1** | **2** | **3** |
| 1 | Пояснювальна записка | 1. підстави для розробки системи<br>2. перелік організацій розробників<br>3. коротка характеристика об'єкта із зазначенням основних техніко-економічних показників його функціонування і зв'язків з іншими об'єктами<br>4. короткі відомості про основні проектні рішення по функціональної і забезпечуючми частинам системи |
| 2 | Функціональна й організаційна структура системи | 1. обґрунтування з виділених підсистем, їх перелік та призначення<br>2. перелік задач, що вирішуються в кожній підсистемі, з короткою характеристикою їх змісту<br>3. схема інформаційних зв'язків між підсистемами і між задачами в рамках кожної підсистеми |
| 3 | Постановка задач і алгоритми вирішення | 1. організаційно-економічна сутність задачі (найменування, мета рішення, короткий зміст, метод, періодичність і час виконання задач, способи збору і передачі даних, зв'язок задачі з іншими задачами, характер використання результатів рішення, в яких вони використовуються)<br>2. економіко-математична модель задачі (структурна і розгорнута форма подання)<br>3. вхідна оперативна інформація (характеристика показників, діапазон зміни, форми подання)<br>4. нормативно-довідкова інформація (НДІ) (зміст і форми подання)<br>5. інформація, що зберігається для зв'язку з іншими завданнями<br>6. інформація, що накопичується для наступних |

| № п \п | Розділ | Зміст |
|---|---|---|
| 1 | 2 | 3 |
| | | варіантів розв'язання задачі<br>7. інформація щодо внесення змін (система внесення змін і перелік інформації, яка піддається змінам)<br>8. алгоритм вирішення задачі (послідовність етапів розрахунку, схема, розрахункові формули)<br>9. контрольний приклад (набір заповнених даними форм вхідних документів, умовні документи з накопичуваної і збереженої інформацією, форми вихідних документів, заповнені за результатами рішення економіко-технічної задачі і відповідно до розробленого алгоритму розрахунку) |
| 4 | Організація інформаційної бази | 1. джерела надходження інформації та способи її передачі<br>2. сукупність показників, що використовуються в системі<br>3. склад документів, терміни і періодичність їх надходження<br>4. основні проектні рішення по організації фонду НДІ<br>5. складу НДІ, включаючи перелік реквізитів, їх визначення, діапазон зміни і перелік документів НДІ<br>6. список масивів НДІ, їх обсяг, порядок і частота коригування інформації<br>7. структура фонду НДІ з описом зв'язку між його елементами; вимоги до технології створення і ведення фонду<br>8. методи зберігання, пошуку, внесення змін і контролю<br>9. визначення обсягів і потоків інформації НДІ<br>10. контрольний приклад по внесенню змін до НДІ<br>11. пропозиції щодо уніфікації документації |
| 5 | Альбом форм документів | |
| 6 | Система | 1. обґрунтування структури математичного |

| № п \п | Розділ | Зміст |
|---|---|---|
| 1 | 2 | 3 |
| | математичного забезпечення | забезпечення<br>2. обґрунтування вибору системи програмування<br>3. перелік стандартних програм |
| 7 | Принцип побудови комплексу технічних засобів | 1. опис і обґрунтування схеми технологічного процесу обробки даних<br>2. обґрунтування і вибір структури комплексу технічних засобів і його функціональних груп<br>3. обґрунтування вимог до розробки нестандартного обладнання<br>4. комплекс заходів щодо забезпечення надійності функціонування технічних засобів |
| 8 | Розрахунок економічної ефективності системи | 1. зведений кошторис витрат, пов'язаних з експлуатацією систем<br>2. розрахунок річної економічної ефективності, джерелами якої є оптимізація виробничої структури господарства (об'єднання), зниження собівартості продукції за рахунок раціонального використання виробничих ресурсів і зменшення втрат, поліпшення прийнятих управлінських рішень |
| 9 | Заходи з підготовки об'єкта до впровадження системи | 1. перелік організаційних заходів щодо вдосконалення бізнес-процесів<br>2. перелік робіт по впровадженню системи, які необхідно виконати на стадії робочого проектування, із зазначенням термінів і відповідальних осіб |
| 10 | Відомість документів | |

Для ІС, які є різновидом автоматизованих систем, встановлюють такі основні види випробувань: попередні, дослідна експлуатація та приймальні. При необхідності допускається додатково проведення інших видів випробувань системи та її частин.

Залежно від взаємозв'язків частин ІС і об'єкта автоматизації випробування можуть бути автономні або комплексні. Автономні випробування охоплюють частини системи. Їх проводять у міру готовності частин системи до здачі в дослідну експлуатацію. Комплексні випробування проводять для груп взаємопов'язаних частин або для системи в цілому.

Для планування проведення всіх видів випробувань розробляється документ "Програма і методика випробувань". Розробник документа встановлюється в договорі або ТЗ. Як додаток до документа можуть включатися тести або контрольні приклади.

Попередні випробування проводять для визначення працездатності системи і вирішення питання про можливість її приймання в дослідну експлуатацію. Попередні випробування слід виконувати після проведення розробником налагодження і тестування поставляються програмних і технічних засобів системи і подання ним відповідних документів про їх готовність до випробувань, а також після ознайомлення персоналу ІС з експлуатаційною документацією.

Дослідну експлуатацію системи проводять з метою визначення фактичних значень кількісних і якісних характеристик системи і готовності персоналу до роботи в умовах її функціонування, а також визначення фактичної ефективності і коригування, при необхідності, документації.

Приймальні випробування проводять для визначення відповідності системи технічним завданням, оцінки якості дослідної експлуатації і вирішення питання про можливість приймання системи в постійну експлуатацію.

## 3.3 Типове проектування ІС

Типове проектування ІС передбачає створення системи з готових типових елементів. Основоположним вимогою для застосування методів типового проектування є можливість декомпозиції проектованої ІС на безліч складових компонентів (підсистем, комплексів задач, програмних модулів і т.д.). Для реалізації виділених компонентів вибираються наявні на ринку типові проектні рішення, які налаштовуються на особливості конкретного підприємства.

Типове проектне рішення (ТПР) – це тиражується (придатне до багаторазового використання) проектне рішення.

Прийнята класифікація ТПР заснована на рівні декомпозиції системи. Виділяються наступні класи ТПР:

- елементні ТПР – типові рішення по завданню або за окремим видом забезпечення завдання (інформаційному, програмному, технічному, математичного, організаційного);
- підсистемні ТПР – як елементи типізації виступають окремі підсистеми, розроблені з урахуванням функціональної повноти і мінімізації зовнішніх інформаційних зв'язків;
- об'єктні ТПР – типові галузеві проекти, які включають повний набір функціональних і забезпечують підсистем ІС.

Кожне типове рішення передбачає наявність, крім власних функціональних елементів (програмних або апаратних), документації з детальним описом ТПР і процедур настройки відповідно до вимог, що розробляється.

Для реалізації типового проектування використовуються два підходи: *параметрически-орієнтоване* і *модельно-орієнтоване проектування*.

**1) Параметрично-орієнтоване проектування**

Включає наступні етапи: визначення критеріїв оцінки придатності пакетів прикладних програм (ППП) для вирішення поставлених завдань, аналіз і оцінка доступних ППП по сформульованим критеріям, вибір і закупівля найбільш підходящого пакета, настройка параметрів (доробка) закупленого ППП.

Критерії оцінки ППП поділяються на такі групи:

- призначення та можливості пакету;
- відмінні ознаки і властивості пакету;
- вимоги до технічних і програмних засобів;
- документація пакету;
- фактори фінансового порядку;
- особливості установки пакету;
- особливості експлуатації пакету;
- допомога постачальника по впровадженню і підтримці пакету;
- оцінка якості пакету і досвід його використання;
- перспективи розвитку пакету.

Усередині кожної групи критеріїв виділяється деяка підмножина приватних показників, що деталізують кожен з десяти виділених аспектів аналізу обираних ППП. Досить повний перелік показників можна знайти в літературі.

Числові значення показників для конкретних ППП встановлюються експертами з обраної шкалою оцінок (наприклад, 10-бальною). На їх основі формуються групові оцінки і комплексна оцінка пакета (шляхом обчислення середньозважених значень). Нормовані вагові коефіцієнти також виходять експертним шляхом.

**2) Модельно-орієнтоване проектування.**

Полягає в адаптації складу і характеристик типовий ІС відповідно до моделі об'єкта автоматизації.

Технологія проектування в цьому випадку повинна забезпечувати єдині засоби для роботи як з моделлю типовий ІС, так і з моделлю конкретного підприємства.

Типова ІС в спеціальній базі метаінформації – репозиторії – містить модель об'єкта автоматизації, на основі якої здійснюється конфігурація програмного забезпечення. Таким чином, модельно-орієнтоване проектування ІС передбачає, перш за все, побудова моделі об'єкта автоматизації з використанням спеціального програмного інструментарію (наприклад, SAP Business Engineering Workbench (BEW), BAAN Enterprise Modeler). Можливо також створення системи на базі типової моделі ІС зі сховищ, який поставляється разом з програмним продуктом і розширюється в міру накопичення досвіду проектування інформаційних систем для різних галузей і типів виробництва.

Репозиторій містить базову (кількість посилань) модель ІС, типові (референтні) моделі певних класів ІС, моделі конкретних ІС підприємств.

Базова модель ІС в репозиторії містить опис бізнес-функцій, бізнес-процесів, бізнес-об'єктів, бізнес-правил, організаційної структури, які підтримуються програмними модулями типовий ІС.

Типові моделі описують конфігурації інформаційної системи для певних галузей або типів виробництва.

Модель конкретного підприємства будується або шляхом вибору фрагментів основний або типової моделі у відповідності зі специфічними особливостями підприємства (BAAN Enterprise Modeler), або шляхом автоматизованої адаптації цих моделей в результаті експертного опитування (SAP Business Engineering Workbench).

Побудована модель підприємства у вигляді метаописания зберігається в репозиторії і при необхідності може бути відкоригована. На основі цієї моделі автоматично здійснюється конфігурація і настройка інформаційної системи.

Бізнес-правила визначають умови коректності спільного застосування різних компонентів ІС і використовуються для підтримки цілісності створюваної системи.

Модель бізнес-функцій є ієрархічною декомпозицію функціональної діяльності підприємства (докладний опис див. В розділі "Аналіз та моделювання функціональної області впровадження ІС").

Модель бізнес-процесів відображає виконання робіт для функцій самого нижнього рівня моделі бізнес-функцій (докладний опис див. В розділі "Технічна специфікація функціональних вимог до ІС"). Для відображення процесів використовується модель управління подіями (EPC – Event-driven Process Chain). Саме модель бізнес-процесів дозволяє виконати настройку програмних модулів - додатків інформаційної системи відповідно до характерними особливостями конкретного підприємства.

Моделі бізнес-об'єктів використовуються для інтеграції додатків, що підтримують виконання різних бізнес-процесів (докладний опис див. В розділі "Етапи проектування ІС із застосуванням UML").

Модель організаційної структури підприємства є традиційною ієрархічну структуру підпорядкування підрозділів і персоналу (докладний опис див. В розділі "Аналіз та моделювання функціональної області впровадження ІС").

Впровадження типової інформаційної системи починається з аналізу вимог до конкретної ІС, які виявляються на основі результатів передпроектного обстеження об'єкта автоматизації. Для оцінки відповідності цим вимогам програмних продуктів може використовуватися описана вище методика оцінки ППП. Після вибору програмного продукту на базі наявних в ньому референтних моделей будується попередня модель ІС, в якій відображаються всі особливості реалізації ІС для конкретного підприємства. Попередня модель є основою для вибору типової моделі системи і визначення переліку компонентів, які будуть реалізовані з використанням інших програмних засобів або зажадають розробки за допомогою наявних в складі типової ІС інструментальних засобів (наприклад, ABAP в SAP, Tools в BAAN).

Реалізація типового проекту передбачає виконання таких операцій:

- установку глобальних параметрів системи;
- задачі структури об'єкта автоматизації;
- визначення структури основних даних;
- завдання переліку реалізованих функцій і процесів;

- опис інтерфейсів;
- опис звітів;
- налаштування авторизації доступу;
- налаштування системи архівування.

# 4 СИСТЕМНИЙ ПІДХІД ДО ПРОЕКТУВАННЯ ІС

Наслідком недоліків класичних методів проектування (каскадної схеми проектування, спіральний процес розробки ІС) з'явився перехід до системного проектування.

Системний підхід оперує поруч категоріальних понять. Його фундаментальним поняттям є поняття системи, даючи яке необхідно переслідувати певну мету. Якщо метою є пізнання вже існуючої системи, то цілком придатним виявляється дескриптивне визначення системи, яке полягає в наступному:

Система – це сукупність об'єктів, властивості якої визначаються відношенням між цими об'єктами.

Об'єкти називають підсистемами або елементами системи. Кожен об'єкт при самостійному дослідженні може розглядатися як система. Функції об'єкта визначаються його внутрішньою будовою.

Таким чином, дескриптивне визначення системи відіграє пізнавальну роль для пояснення функцій, що реалізуються нею. Функції системи проявляються в процесі її взаємодії з зовнішнім середовищем. При цьому важливо визначити межу між зовнішнім середовищем і створюваної системою. Це можна здійснити на основі конструктивного визначення системи. Особливе значення конструктивний підхід має для технічних систем.

Системний підхід – це методологія спеціального наукового пізнання і соціальної практики, а також пояснювальний принцип, в основі якого лежить дослідження об'єктів як систем. Методологічна специфіка системного підходу визначається тим, що він орієнтує дослідження на: розкриття цілісності об'єкта і забезпечують його механізмів; виявлення різноманітних типів зв'язків складного об'єкта; зведення цих зв'язків в єдину теоретичну картину.

Системний підхід реалізує уявлення складного об'єкта у вигляді ієрархічної системи взаємопов'язаних моделей, що дозволяють фіксувати цілісні властивості об'єкта, його структуру і динаміку.

Системний аналіз сукупність методологічних засобів, використовуваних для підготовки і обгрунтування рішень щодо складних проблем соціального, технічного та економічного характеру. Він ґрунтується на системному підході, а також на ряді математичних дисциплін і сучасних методів управління. Основною процедурою системного аналізу є побудова

узагальненої моделі, адекватно відображає цікавлять дослідника властивості реальної системи і її взаємозв'язку. Основна концепція полягає в побудові за допомогою графічних методів системного аналізу сукупності моделей різних аспектів діяльності організації, які дають можливість управлінцям і аналітикам отримати ясну загальну картину бізнес-процесів.

***Принципи системного аналізу.***

Оптимальність – в результаті аналізу необхідно знайти оптимальне рішення задачі.

Емерджентність – чим більше система і чим більше розходження між частиною і цілим, тим вище ймовірність того, що властивості цілого можуть сильно відрізнятися від властивостей його частин. Принцип емерджментності дозволяє виявити розбіжність локальних оптимумів цілей системи з глобальним оптимумом системи. Цей фактор необхідно враховувати при проведенні системного аналізу діяльності організації, оскільки він грає важливу роль.

Системність – дослідження об'єкта, з одного боку, як єдиного цілого, а з іншого, як частини більшої системи, з якої об'єкт знаходиться в певних відносинах.

Ієрархічність – визначення в системі структурних відносин, що характеризуються впорядкованістю, організованістю взаємодій між окремими її рівнями по вертикалі. Більшість організацій є складними системами, і необхідність ієрархічної побудови цих систем обумовлена тим, що управління в них пов'язано з переробкою і використанням великих обсягів інформації.

Інтеграція – вивчення інтеграційних властивостей і закономірностей системи.

Формалізація – отримання комплексних кількісних характеристик.

Головним завданням системного аналізу є пошук шляхів по перетворенню складного в просте, з розкладання важкозрозумілі завдання на ряд завдань, що мають рішення. Єдиний ефективний підхід до вирішення цієї проблеми, який виробило людство за всю свою історію, полягає в побудові складної системи з невеликої кількості великих частин, кожна з яких, в свою чергу, будується з частин меншого розміру, і т.д., до тих пір , поки самі невеликі частини можна буде будувати з наявного матеріалу. Цей підхід відомий під різними назвами, серед них такі, як «розділяй і володарюй», ієрархічна декомпозиція та ін. По відношенню до проектування складної програмної системи це означає, що її необхідно розділити (декомпозировать)

на невеликі підсистеми, кожну з яких можна розробляти незалежно від інших. Це дозволяє при розробці підсистеми будь-якого рівня має справу тільки з нею, а не з усіма іншими частинами системи. Правильна декомпозиція є головним способом подолання складності розробки великих систем ПО. Поняття «правильна» по відношенню до декомпозиції означає наступне (слайд 3): кількість зв'язків між окремими підсистемами має бути мінімальним (принцип «слабкою пов'язаності»; зв'язність окремих частин всередині кожної підсистеми повинна бути максимальною (принцип «сильного зчеплення».

*Існують два основні підходи до декомпозиції систем.*

Перший підхід називають ***функціонально-модульним***, він є частиною більш загального структурного підходу. В його основу покладено принцип функціональної декомпозиції, при якій структура системи описується в термінах ієрархії її функцій і передачі інформації між окремими функціональними елементами.

Другий, ***об'єктно-орієнтований*** підхід, використовує об'єктну декомпозицію. При цьому структура системи описується в термінах об'єктів і зв'язків між ними, а поведінка системи описується в термінах обміну повідомленнями між об'єктами.

Практичне використання інформаційних технологій тісно пов'язане з питаннями маркетингу і менеджменту інформаційних ресурсів, технологій і послуг, методологією проектування інформаційних систем, управління якістю і стандартизації інформаційних технологій. В даний час в цілому сформувалася ідеологія і практика застосування інформаційних технологій. Однак необхідна організація інформаційних процесів і технологій як системи, для побудови якої доцільно застосувати системний підхід.

Найбільш повно системний підхід проявився при проектуванні інформаційних систем. Запропоновано методологію проектування інформаційних систем як колективного процесу. Проаналізовано основні етапи та завдання впровадження та супроводу інформаційних технологій на основі об'єктно-орієнтованої технології як основи створення відкритих, гнучких, багатофункціональних систем для різних предметних областей. Значну увагу приділено питанням формування моделі предметної області використання різних засобів для автоматизації процесу проектування, аналізу якості проектування.

## 4.1 Принципи системного підходу

Системним підходом, розуміється як частина методології системного аналізу, називається застосування ряду методологічних положень (принципів) загального характеру до дослідження систем. Відомо близько двох десятків таких принципів, пов'язаних з необхідністю вивчати систему комплексно, в її розумною повноті, зв'язності, організованості.

Що ж таке системність, що означає слово «системний», що застосовується разом з великою кількістю термінів і понять?

Пошуки відповіді на це питання приводять нас до переконання, що складний об'єкт треба розглядати і як ціле, і як що складається з окремих частин. Потрібно дослідити предмет з різних сторін і точок зору, вдаватися в його внутрішню будову і організацію ...

Все це призводить до формулювання положень, які прийнято називати принципами системного підходу, тобто до тверджень, узагальнюючим досвід роботи людини зі складними системами:

- принцип кінцевої мети: абсолютний пріоритет кінцевої (глобальної) мети;
- *принцип єдності*: спільне розгляд системи як цілого і як сукупності частин (елементів);
- *принцип зв'язності*: розгляд будь-якій частині спільно з її зв'язками з оточенням;
- *принцип модульної побудови*: корисно виділення модулів в системі і розгляд її як сукупності модулів;
- *принцип ієрархії*: корисно введення ієрархії частин (елементів) і (або) їх ранжування;
- *принцип функціональності*: спільний розгляд структури і функції з пріоритетом функції над структурою;
- *принцип розвитку*: облік змінності системи, її здатності до розвитку, розширення, заміни частин, накопичення інформації;
- *принцип децентралізації*: поєднання в прийнятих рішеннях і управлінні централізації і децентралізації;
- *принцип невизначеності*: облік невизначеностей і випадковостей у системі.

Відзначимо, що хоча всі перераховані принципи так чи інакше зачіпаються практично при будь-якому викладі системного підходу, їх формулювання поки не є загальноприйнятими.

Розглянемо введені принципи більш докладно.

*Принцип кінцевої мети*

Перший з принципів – принцип кінцевої мети – означає, що в цілеспрямованій системі все повинно бути підпорядковане глобальної мети. Будь-яка спроба зміни, вдосконалення та управління в такій системі повинна оцінюватися з точки зору того, допомагає чи заважає вона досягненню кінцевої мети. Це накладає особливу відповідальність на вибір мети і її чітке трактування.

Розпливчасті, в повному обсязі певні кінцеві цілі тягнуть за собою неясності в структурі і управлінні системою, і невірні дії в системі. Такі дії можуть бути і наслідком невіри в кінцеву мету або в можливість її досягнення.

В дещо зміненій трактуванні принцип кінцевої мети застосовують і до систем, які не є цілеспрямованими. У цьому випадку поняття кінцевої мети замінюють поняттями основної функції, основного призначення, властивості системи. При цьому принцип вказує, що вивчення і робота з системою повинні вестися на базі першочергового з'ясування цих понять.

*Принцип єдності і зв'язку*

Наступні два принципи: принцип єдності і принцип зв'язності, мають досить тісним взаємозв'язком і іноді навіть об'єднуються в один принцип єдності і зв'язку. Однак, існують причини, за якими їх корисно розглядати окремо.

По-перше, принцип єдності – це орієнтація на «погляд усередину» системи або її частини, а принцип зв'язку - на «погляд зсередини». У різні моменти дослідження корисна або та, або інша орієнтація.

По-друге, рекомендований в принципі єдності розчленовування системи зі збереженням цілісних уявлень про неї на практиці досить різко відрізняється від процедури виявлення всіляких зв'язків, рекомендованої в принципі зв'язності.

Нарешті, процедура виявлення зв'язків, застосована до всієї системи цілком, призводить до принципу врахування зовнішнього середовища, який згадують в літературі, але який, як випливає з вищесказаного, не вважають самостійним.

*Принцип модульного побудови*

Принцип модульного побудови вказує на можливість розгляду системи або її частини, як сукупності її вхідних і вихідних впливів. Він дозволяє

вберегти від зайвої деталізації системи при збереженні можливості правильного її опису.

*Принцип ієрархії*

Принцип ієрархії акцентує увагу на корисності визначення в системі ієрархічного (домінуючого) характеру зв'язків між елементами, модулями, цілями.

Пояснимо сенс слова «ранжування» в формулюванні принципу. Ієрархічні системи зазвичай досліджуються і створюються «зверху», починаючи з аналізу модулів першого ієрархічного рівня. У разі відсутності ієрархії дослідник повинен вирішити, в якому порядку він буде розглядати частини системи. Так, наприклад, конструктор при створенні нового зразка виділяє в ньому початковий елемент, до якого потім подумки або на кресленні підганяє другий, третій, наступні. Наладчик починає пошук несправності в системі з тестів, що визначають найбільш типові відмови. Таким чином, вони вводять порядок розгляду системи, який і називається ранжируванням. У поєднанні з ієрархією в системі, ранжування може бути застосовано і для введення черговості розгляду модулів одного і того ж рівня.

*Принцип функціональності*

Для розбору принципу функціональності нагадаємо, що ми визначали функцію системи як її деякий властивість. Функція для нас - це те, що система (модуль, елемент) «може робити» важливого для цілей розгляду.

Принцип функціональності стверджує, що будь-яка структура тісно пов'язана з функцією системи і її частин, і досліджувати або створювати структуру необхідно після з'ясування функцій в системі.

На практиці цей принцип означає, що в разі надання системі нових функцій корисно переглядати її структуру, а не намагатися втиснути нову функцію в стару схему. Так, перебудова виробництва, пов'язана з введенням автоматизації, веде як до виникнення нових підрозділів (обчислювальний центр, група системних програмістів, група створення і супроводу банку даних), так і до перебудови структури наявних. Ці зміни зачіпають, природно, і систему управління.

*Принцип розвитку*

Принцип розвитку досить добре пояснений в його формулюванні. Поняття розвитку, змінності при збереженні якісних особливостей виділяється майже в будь-якій природній системі, а в штучних системах можливість розвитку, вдосконалення, як правило, закладається в основу її створення. При модульному побудові такий розвиток зазвичай зводиться до

заміни і додавання модулів (частин). Так, можливості розширення функцій і модернізації закладаються в принципи побудови банків даних і знань, програмних комплексів, багатоцільових роботів і інших складних технічних систем.

Слід, однак, зауважити, що межі розширення функцій зазвичай визначені і досить обмежені. Навряд чи буде розумно створювати універсальне програмне засіб, здатний управляти верстатом і грати в шахи. Навряд чи кому-небудь знадобиться і робот, здатний працювати у плавильної печі і в квартирі. Але ось заміна частин, модернізація представляються нам безмежними. Практично безмежні і можливості запам'ятовування інформації, що ведуть до самонавчання, самоорганізації, штучного інтелекту. Таким чином, використання принципу розвитку лежить в основі розробки цих напрямків.

*Принцип децентралізації*

Принцип децентралізації рекомендує, щоб управляючі дії і прийняті рішення виходили не тільки з одного центру (головного елемента). Ситуація, коли всі управління виходять з одного місця, називається повною централізацією. Такий стан вважається виправданим лише при особливій відповідальності за все, що відбувається в системі, і при нездатності частин системи самостійно реагувати на зовнішні впливи. Система з повною централізацією буде негнучкою, непріспосабліваючейся, що не володіє «внутрішньої активністю». Досить імовірно, що в такій системі канали інформації, що ведуть до головного елементу, виявляться перевантаженими, а сам цей елемент, будучи не в змозі переробити таку кількість інформації, почне видавати неправильні управління.

Однак чим вище ступінь децентралізації рішень в системі, тим складніше вони узгоджуються з точки зору виконання глобальної мети. Досягнення спільної мети сильно децентралізованою системою може забезпечуватися лише якихось стійко працюючим механізмом регуляції, що не дозволяє сильно ухилятися від поведінки, що веде до виконання мети. Такий стан зустрічається досить рідко; у всіх цих випадках має місце ситуація з сильною зворотним зв'язком. Таке функціонування ринкової економіки; в області живої природи - взаємодія в системі, що складається з акули і маленьких рибок лоцманів, які наводять акулу на косяки риб і харчуються залишками її їжі.

В системах, де стійких механізмів регуляції немає, неминуче наявність тій чи іншій мірі централізації. При цьому виникає питання про оптимальний

поєднанні команд ззовні (зверху) і команд, що виробляються всередині даної групи елементів. Загальний принцип такого поєднання простий: ступінь централізації повинна бути мінімальною, що забезпечує виконання поставленої мети.

*Принцип невизначеності*

Принцип невизначеності стверджує, що ми можемо мати справу з системою, в якій нам не все відомо чи зрозуміло. Це може бути система з нез'ясованої структурою, з непередбачуваним ходом процесів, зі значною ймовірністю відмов у роботі елементів, з невідомими зовнішніми впливами і ін. Окремим випадком невизначеності виступає випадковість - ситуація, коли вид події відомий, але воно може або наступити, або не наступити . На основі цього визначення можна ввести повне поле подій - це таке їх безліч, про яке відомо, що одне з них настане.

# 5 ТОПОЛОГІЇ ІС ТА КЛІЄНТ-СЕРВІРНА АРХІТЕКТУРА ІС

## 5.1 Поняття і типи топології ІС

ІС має різну інформаційно-технологічну архітектуру, що залежить від використовуваних програмних і технічних засобів, типу мереж і організації БД. Архітектура ІС, в свою чергу, впливає на параметри ІС:

- час виконання одиночного запиту;
- продуктивність ІС (кількість транзакцій в одиницю часу);
- вартість створення, експлуатації та розвитку ІС.

Види архітектур ІС:

- централізована обробка даних;
- архітектура "файл - сервер";
- дворівневий "клієнт - сервер";
- багаторівневий "клієнт - сервер".

***Централізована обробка даних***

Централізована обробка даних на локальному комп'ютері (рис. 5.1) має такі особливості:

1) на одному комп'ютері функціонують:

- програмні засоби для користувача інтерфейсу, що забезпечують інтерактивний режим роботи користувача;
- програмні засоби додатків, що виконують змістовну обробку даних;
- БД.

2) розвиток ІС обмежений:

- технічними параметрами центрального комп'ютера: обсяг оперативної пам'яті, об'єм дискової пам'яті для БД, надійність роботи комп'ютера і програмного забезпечення;
- продуктивністю центрального комп'ютера, що впливає на своєчасність обробки всіх додатків.

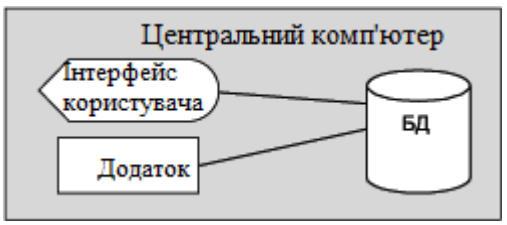



Рисунок 5.1 – ІС з архітектурою централізованої обробки даних

***Архітектура "файл - сервер"***

ІС з розподіленою обробкою даних типу "файл - сервер" (рис. 5.2) використовує комп'ютерні мережі, як правило, локального типу. Комп'ютери в мережі діляться на робочі станції і сервери. На робочій станції встановлені програмні засоби для користувача інтерфейсу, програмні засоби додатків, що виконують змістовну обробку даних. На файловому сервері знаходиться БД.

Гідність архітектури "файл-сервер" – забезпечення високого рівня захисту даних від несанкціонованого доступу.

Недоліки архітектури "файл-сервер":

- обмін на рівні файлів, доступ до яких в режимі коригування блокується для інших користувачів;
- перевантаження трафіку мережі;
- високі вимоги до технічного оснащення робочих станцій, на яких виконується змістовна обробка даних.

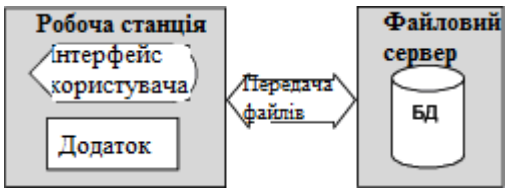



Рисунок 5.2 – ІС з архітектурою "файл - сервер"

***Дворівневий "клієнт - сервер"***

На відміну від раніше розглянутої архітектури, розподілена обробка даних типу "дворівневий клієнт-сервер" (рис. 5.3) передбачає, що на сервері знаходиться БД під управлінням СУБД в архітектурі "клієнт-сервер".

Всі робочі станції (клієнти) посилають запити на дані до серверу, який здійснює вилучення і попередню обробку даних. Одиницею обміну по мережі є запит і релевантна запиту вибірка даних з БД. Істотно зменшується трафік мережі, знімаються обмеження на доступність даних БД різних додатків.

Клієнтська частина додатків стає кілька полегшеної, але в великих ІС зі складною логікою обробки даних виникає проблема "товстого" клієнта. Робоча станція повинна мати достатньо високі технічні параметри для виконання складних додатків.

Недоліком архітектури є наявність дуже високих вимог до технічного комплексу сервера БД, який стає центральною ланкою всієї ІС і визначає її надійність.

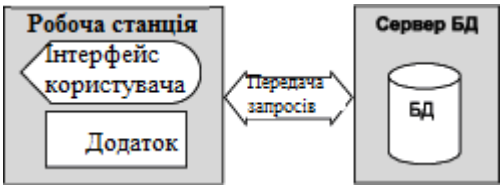



Рисунок 5.3 – ІС з архітектурою "дворівневий клієнт - сервер"

***Багаторівневий "клієнт-сервер"***

На робочій станції встановлені тільки програмні засоби, що підтримують інтерфейс з БД. На сервері БД знаходяться БД під управлінням СУБД, архітектура мережі – "клієнт-сервер". В архітектурі ІС виділений сервер додатків, на якому знаходяться програмні засоби загального користування. Ці сервери виконують всю змістовну обробку даних.

На відміну від дворівневої архітектури, дана архітектура (рис. 5.4) забезпечує ефективне використання додатків загального користування багатьма клієнтами. Клієнти перетворюються в "тонких" клієнтів, при цьому знижуються вимоги до обладнання робочих станцій. Якщо серверів додатків і БД в мережі кілька, архітектура ІС стає багаторівневої клієнт-серверної архітектурою. Наявність самостійних рівнів в інформаційно-технологічної архітектури ІС дає можливість варіювати апаратними та програмними засобами: вибирати операційні системи, СУБД, інтерфейси кінцевих користувачів, типи серверів і робочих станцій.


Рисунок 5.4 – ІС з архітектурою "трирівневий клієнт - сервер"

При побудові великих ІС актуальна проблема створення розподілених систем обробки даних на основі інтеграції неоднорідних апаратно-програмних платформ. Багаторівнева архітектура ІС забезпечує ізоляцію паралельно працюють процесів, в результаті помилки в роботі однієї

програми не впливають на роботу інших програм або операційної системи. Комп'ютерні мережі можуть включати окремі сегменти, для зв'язку яких використовуються стандартні протоколи. Для БД здійснюється адміністрування, реєстрація кожного мав місце доступу до бази даних і виконаних змін в спеціальному журналі БД. Як правило, для великих БД створюються страхові копії, здійснюється "зеркализация" дисків.

# 6 СТРУКТУРНА І ОБ'ЄКТНО ОРІЄНТОВАНОЇ ТЕХНОЛОГІЯ ПРОЕКТУВАННЯ

## 6.1 Структурний підхід до проектування ІС

### 6.1.1 Сутність структурного аналізу і проектування

В основі структурного аналізу лежить виявлення структури як відносно стійкої сукупності відносин, визнання методологічного переважного значення відносин над елементами в системі, часткове відвернення від розвитку об’єктів. Основним заняттям структурного аналізу є структурний елемент, що виконує одну з елементарних функцій, пов’язаних із модельованим предметом, процесом або явищем.

Базовими елементами в процесі використання структурного підходу є модулі. Модуль повинен мати наступні властивості:

- ім’я, за яким до нього можна звертатися як до єдиного цілого;
- приймати і (чи) передавати дані у вигляді параметрів.

Сутність структурного заходу до розробки ІС полягає в її декомпозиції на функції, що автоматизуються: система розбивається на функціональні підсистеми, які, у свою чергу, діляться на підфункції, що підрозділяються на задачі і т. д. Розбиття продовжується аж до конкретних процедур. При цьому система, що автоматизується, зберігає цілісне подання, в якому всі складові компоненти взаємопов’язані.

Мета структурного аналізу полягає в перетворенні загальних, розпливчатих знань про предметну область в точні моделі, що описують різні підсистеми модельованої організації.

Характерні риси структурного аналізу наступні:

1) кількість елементів, виділених на кожному рівні, обмежена (звичайно від 3 до 9, при цьому верхня межа обмежується можливостями людського мозку в сприйнятті певної кількості взаємозалежних об’єктів, а нижня межа вибирається з міркувань здорового глузду);

2) виділення на кожному рівні тільки істотних елементів;

3) використання чітких формальних правил запису;

4) послідовне наближення до кінцевого результату.

Структурний аналіз передбачає дослідження системи за допомогою її графічного модельного подання, яке починається із загального огляду і

подальшої деталізації, коли система набуває ієрархічної структури із усе більшою кількістю рівнів.

Структурний аналіз здійснюється з дотриманням наступних принципів:

– розбиття системи на частини у вигляді "чорних ящиків";
– ієрархічне упорядкування;
– формалізація.

Перший принцип використання так званих *"чорних ящиків"* для розчленовування великих систем на частини дає можливість їх спростити. Перевага використання "чорних ящиків" полягає в тому, що їхньому користувачу не потрібно знати, як вони працюють, – потрібно знати лише їх входи і виходи, а також їх призначення (тобто функції, що вони виконують).

Таким чином, першим кроком спрощення складної системи є її поділ на "чорні ящики" (принцип "поділяй і пануй" – принцип вирішення складних проблем розбиттям їх на множину незалежних завдань, легких для розуміння і вирішення), при цьому цей поділ повинен задовольняти такі критерії:

а) кожний "чорний ящик" повинен реалізовувати одну-єдину функцію системи;
б) функція кожного "чорного ящика" повинна бути легко зрозумілою незалежно від складності її реалізації;
в) зв’язок між "чорними ящиками" повинен вводитися тільки за наявності зв’язку між відповідними функціями;
г) зв’язки між "чорними ящиками" повинні бути якомога простішими для забезпечення незалежності між ними.

Другий принцип *ієрархічного упорядкування* декларує, що упорядкування виділених частин є важливим для розуміння проблеми. Для розуміння складної системи потрібно виділені частини організувати як ієрархічні структури. Система може бути зрозумілою, коли вона розбудована за рівнями, кожний із яких додає нових деталей.

Третій принцип – *формалізації* – полягає в необхідності використання графічних нотацій, які відображають структуру системи, елементи даних, етапи обробки за допомогою спеціальних графічних символів діаграм, а також описують проект системи формальними і природними мовами. Нотації полегшують розуміння складних систем. Відомо, що "одна картинка коштує тисячі слів"

### 6.1.2 Методологія структурного аналізу і проектуваняя

В методології структурного аналізу традиційно використовують моделі, що показують:

- функції, які система повинна виконувати;
- процеси, що забезпечують виконання функцій;
- дані, що необхідні під час виконання функцій, і відношення між ними;
- організаційні структури, що забезпечують виконання функцій;
- матеріальні та інформаційні потоки, що виникають у ході виконання функцій.

Для побудови перерахованих моделей у методологіях структурного аналізу найбільш часто застосовуються такі діаграми:

1) діаграми функціональних специфікацій, діаграми потоків даних, діаграми переходів станів у нотаціях:

- SADT (Structured Analysis and Design Technique);
- Йордана (Yourdon);
- Гейна – Сарсона (Gane – Sarson);
- SAG (Software AG);

2) діаграми моделей даних "сутність – зв’язок" у нотаціях:

- Чена (Chen);
- Беркера;
- SADT;
- SAG;

3) діаграми структури програмного додатку в нотаціях:

- Константайна (Constantine);
- Джексона (Jackson);
- Варньє – Орра (Warnier SAG.

У якості інструментальних засобів структурного аналізу і проектування найбільш часто і ефективно застосовуються наступні.

***BFD (Business Function Diagrams)*** – діаграми бізнес-функцій (функціональні специфікації).

***DFD (Data Flow Diagrams)*** – діаграми потоків даних у нотаціях Гейна – Сарсона, Йордана та інших, що забезпечують аналіз і функціональне проектування інформаційних систем.

***ERD (Entity-Relationship Diagrams)*** – діаграми "сутність – зв’язок" у нотаціях Чена и Баркера.

***STD (State Transition Diagrams)*** – діаграми переходів станів, які засновані на розширеннях Уорда – Меллора (Ward – Mellor) і Хатлі (Hatley) для проектування систем реального часу.

***SSD (System Structure Diagrams)*** – діаграми структури програмного додатка.

***FDD (Functional Decomposition Diagrams)*** – діаграми функціо-нальної декомпозиції.

Сімейство IDEF (Integration Definition for Function Modeling):

***IDEF0*** – методологія функціонального моделювання, що є складовою частиною SADT і що дозволяє описати бізнес-процес у вигляді ієрархічної системи взаємопов'язаних функцій.

***IDEF1*** – методологія аналізу і вивчення взаємозв'язків між інформаційними потоками в рамках комерційної діяльності підприємства.

***IDEF1X*** – методологія інформаційного моделювання, заснована на концепції "сутність – зв'язок" Ченом. Застосовується для розробки реляційних баз даних і використовує умовний синтаксис, що спеціально розроблений для зручної побудови концептуальної схеми і забезпечує універсальне подання структури даних в рамках підприємства, незалежне від кінцевої реалізації бази даних і апаратної платформи; IDEF3 – методологія моделювання потоків робіт підприємства, що дозволяє подати їх сценарії за допомогою опису послідовності змін властивостей об'єкта в рамках даного процесу.

***IDEF4*** – методологія об'єктно-орієнтованого проектування для під-тримки проектів, пов'язаних з об'єктно-орієнтованими реалізаціями.

***IDEF5*** – методологія, що забезпечує наочне подання даних, отриманих у результаті обробки онтологічних запитів, у простій, графічній формі.

Усі перелічені засоби містять графічні і текстові засоби моделювання: перші – для зручності відображення основних компонентів моделі, другі – для забезпечення точного визначення її компонентів і зв'язків.

Методології структурного аналізу і проектування можна класифікувати наступним чином (рис. 6.1).

За відношенням до шкіл виділяють методології програмної інженерії (*Software Engineering, SE*) та інформаційної інженерії (*Informa-tion Engineering, IE*).

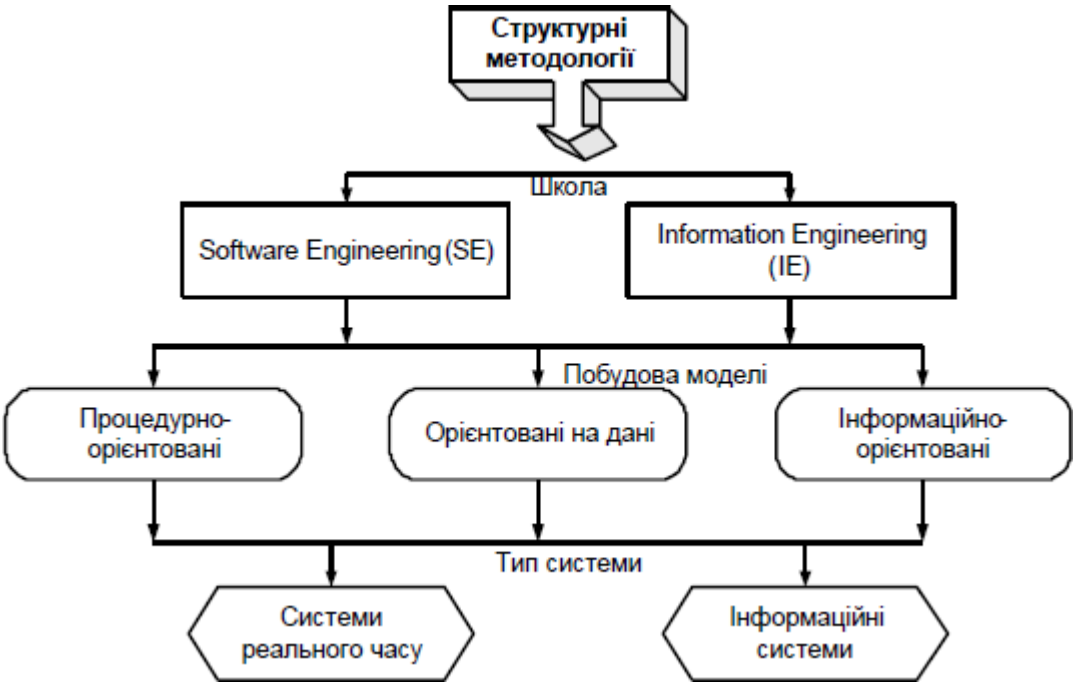



Рисунок 6.1 – Класифікація структурних методологій

***SE*** є нисхідним поетапним підходом до розробки ПЗ, що починається із загального погляду на його функціонування. Потім проводиться декомпозиція на підфункції. Процес повторюється доти, поки підфункції не стануть достатньо малими для можливості їх кодування. У результаті виходить ієрархічна, структурована модульна програма. SE подає універсальні методології розробки ПЗ.

***IE*** представляє нові методології побудови систем взагалі і включає етапи більш високого рівня (наприклад, стратегічне планування). На етапі проектування систем за методологією SE і IE аналогічні.

Залежно від порядку побудови моделі виділяють методології процедурно-орієнтовані, орієнтовані на дані та інформаційно-орієнто- вані. Процедурно-орієнтований підхід регламентує первинність проектування функцій порівняно з даними: вимоги до даних розкриваються через функціональні вимоги. Підхід орієнтований на дані найбільш важливими є вхід і вихід системи, тому структури даних визначаються в першу чергу, а процедурні компоненти є похідними від даних. Третій, інформаційно-орієнтований підхід, є складовою IE-дисципліни, його відмінна риса від підходу, орієнтованого на дані, – можливість працювати з ієрархічними структурами даних.

Залежно від типу проектованих систем виділяють методології для систем реального часу та методології для інформаційних систем. Основна відмінність цих типів систем полягає в наступному: системи реального часу контролюють і контролюються зовнішніми подіями, а інформаційні системи – даними.

Найбільш істотна відмінність між різновидами структурного аналізу і проектування полягає у використовуваних методах і засобах функціонального моделювання. З цього погляду всі різновиди структурного аналізу можуть бути розділені на дві групи: у якій застосовуються методи і технологія DFD (у різних нотаціях) і ті, що використовують SADT-методологію.

За допомогою цих методологій можуть бути побудовані логічні моделі предметної сфери: початкової – як є (as is) і реорганізованої – як повинно бути (to be).

Далі слід розглянути більш детально деякі з найбільш поширених методологій структурного аналізу і проектування.

### 6.1.3 Технологія структурного проектування

У процесі структурного проектування виконуються два види робіт:

1) проектування архітектури ІС, що включає:

- розробку структури й інтерфейсів її компонентів (автоматизованих робочих місць);
- узгодження функцій і технічних вимог до компонентів;
- визначення інформаційних потоків між основними компонентами, зв’язків між ними і зовнішніми об’єктами;

2) детальне проектування, що включає:

- розробку специфікацій кожного компонента;
- розробку вимог до тестів і плану інтеграції компонентів;
- побудову моделей ієрархії програмних модулів і міжмодульних взаємодій, проектування внутрішньої структури модулів.

При цьому відбувається розширення моделі вимог за рахунок:

- уточнення наявних функціональних, інформаційних і, можливо, подієвих моделей вимог, побудованих за допомогою відповідних засобів структурного аналізу;

- побудови моделей автоматизованих робочих місць, що включають інформаційну модель і орієнтовані на неї функціональні моделі (аж до ідентифікації конкретних сутностей інформаційної моделі);
- побудови моделей міжмодульних і внутрішньомодульних взаємодій з використанням техніки структурних карт.

У структурному підході з метою проектування модулів використовується техніка структурних карт (схем), що демонструє, яким чином системні вимоги будуть відбиватися комбінацією програмних структур. При цьому найчастіше застосовують *два види структурних карт*:

1) Структурні карти Константайна, призначені для опису відношень між модулями.
2) Структурні карти Джексона – для опису внутрішньої структури модулів.

*Структурні карти Константайн*а є моделлю відносин ієрархії між програмними модулями. Вузли структурних карт відповідають модулям і областям даних, потоки відображають міжмодульні виклики (у тому числі циклічні, умовні й рівнобіжні). Міжмодульні зв’язки за даними і управлінням також моделюються спеціальними вузлами, прив’язаними до потоків, стрілками вказуються напрямки потоків і зв’язків. Фундаментальні елементи структурних карт Константайна стандартизовані IBM, ISO і ANSI.

*Техніка структурних карт Джексона* зводиться до методології структурного програмування Джексона і полягає в проектуванні діаграм і схем для графічного ілюстрування внутрішньомодульних (а іноді й міждумодульних) зв’язків і документування проекту архітектури ІС. При цьому структурні карти Джексона дозволяють здійснювати проектування нижнього рівня ІС і на цьому етапі є близькими до традиційних блок-схем, що моделюють послідовне, рівнобіжне, умовне й ітераційне виконання їх вузлів.

## 6.2 Об'єктно-орієнтована технологія проектування

### 6.2.1 Сутність об’єктного підходу

Об’єктно-орієнтовані методології засновані на об’єктній декомпозиції предметної області. Предметна область подається у вигляді сукупності об’єктів, що взаємодіють між собою за допомогою передачі повідомлень. В

об’єктній методології використовуються поняття, визначення яких наведене в табл.6.1.

Як об’єкти предметної області можуть розглядатися конкретні предмети, а також абстрактні чи реальні сутності (наприклад, клієнт, замовлення, підприємство тощо). Кожен об’єкт характеризується своїм станом (точніше, набором атрибутів, значення яких визначають стан), а також набором операцій для перевірки і зміни цього стану.

Таблиця 6.1 – Визначення термінів об’єктно-орієнтованої методології

| Термін | Англійський варіант | Значення терміна |
|---|---|---|
| 1 | 2 | 3 |
| ідентичність | identity | Природа об’єкта: те, що відрізняє його від інших об’єктів |
| клас | class | Множина об’єктів зі спільною структурою і поведінкою. Терміни "клас" і "тип" взаємозамінні |
| об’єктно-орієнто-вана декомпозиція | object-oriented de-composition | Процес розбиття системи на частини, відповідні класам і об’єктам предметної області. Практичне застосування методів об’єктно-орієнтованого проектування приводить до об’єктно-орієнтованої декомпозиції, за якої ми розглядаємо світ як сукупність об’єктів, що узгоджено діють для забезпечення необхідної поведінки |
| об’єктно-орієнто-ване програмування | object-oriented pro-gramming (OOP) | Методологія реалізації, за якої програма організовується як сукупність об’єктів, що співробітничають, кожен із яких є екземпляром якого-небудь класу, а класи утворюють ієрархію спадкоємства. При цьому класи зазвичай статичні, а об’єкти дуже динамічні, що заохочується динамічним зв’язуванням і поліморфізмом |
| об’єктно-орієнто-ване проектування | object-oriented design (OOD) | Методологія проектування, що поєднує процес об’єктно-орієнтованої декомпозиції і систему позначень для подання логічної і фізичної, статичної та динамічної моделей проектованої системи. Система позначень складається з діаграм класів, об’єктів, модулів і процесів |
| об’єктно-орієнто-ваний аналіз | object-oriented analy-sis | Метод аналізу, згідно з яким вимоги розглядаються з погляду класів і об’єктів, що є складовими словника предметної області |
| операція | operation | Дія, що виконується одним об’єктом над іншим, щоб викликати реакцію. Усі операції, які можна виконати над яким-небудь об’єктом, зосереджені у вільних підпрограмах і функціях-членах (методах). Терміни "операція" і "метод" взаємозамінні |
| поведінка | behavior | Дії та реакції об’єкта, виражені в термінах передачі повідомлень і зміни стану; видима ззовні і відтворена активність об’єкта |
| стан | state | Сукупний результат поведінки об’єкта: одна зі стабільних умов, у яких об’єкт може існувати, охарактеризованих кількісно; у будь-який конкретний момент часу стан об’єкта включає перелік (зазвичай, статичний) властивостей об’єкта і поточні значення (зазвичай, динамічні) цих властивостей |

Кожен об’єкт є представником певного класу однотипних об’єктів, що визначає його загальні властивості. Усі представники (екземпляри) того самого класу мають той самий набір операцій і можуть реагувати на ті самі повідомлення.

Об’єкти і класи організуються з дотриманням таких принципів:

1. *Принцип інкапсуляції* (приховання інформації) декларує заборону будь-якого доступу до атрибутів об’єкта, крім одного – операції. Відповідно до цього внутрішня структура об’єкта прихована від користувача, а будь-яка його дія ініціюється зовнішнім повідомленням, що обумовлює виконання відповідної операції.

2. *Принцип спадкування* декларує створення нових класів – від загального до окремого. Такі нові класи зберігають усі властивості класів-батьків і при цьому містять додаткові атрибути й операції, що характеризують їхню специфіку.

3. *Принцип поліморфізму* декларує можливість роботи з об’єктом без інформації про конкретний клас, представником якого він є. Кожен об’єкт може вибирати операцію на основі типів даних, прийнятих у повідомленні, тобто реагувати індивідуально на це повідомлення (те саме для різних об’єктів).

Таким чином, *об’єктно-орієнтований підх*ід полягає в поданні системи, що моделюється, у вигляді сукупності класів і об’єктів предметної сфери. При цьому ієрархічний характер складної системи виявляється з використанням ієрархії класів, а її функціонування розглядається як взаємодія об’єктів.

Об’єктно-орієнтовані методології базуються на інтегрованих моделях трьох типів:

1) об’єктної моделі, що відображає ієрархію класів, пов’язаних спільністю структури і поводження і які відображають специфіку атрибутів і операцій кожного з них;
2) динамічної моделі, що відображає часові аспекти й послідовність операцій;
3) функціональної моделі, що описує потоки даних.

Якщо методи структурного проектування мали на меті спрощення системної розробки на основі алгоритмічного підходу, то об’єктно-орієнтовані методи вирішують аналогічне завдання, використовуючи опис класів і об’єктів, тобто чіткі засоби об’єктно-орієнтованого проектування. М. Буч визначив об’єктно-орієнтоване проектування як "методологію проектування, що поєднує в собі процес об’єктної декомпозиції і прийоми подання як логічної і фізичної, так і статичної та динамічної моделей проектованої системи".

Кінцевим результатом процесу об’єктно-орієнтованого проектування повинна стати множина класів об’єктів із приєднаними методами обробки

атрибутів. Якщо в структурному підході моделі даних і операцій розробляються незалежно одна від одної і тільки координуються між собою, то об’єктно-орієнтований підхід передбачає спільне моделювання даних і процесів. При цьому моделі проблемної сфери застосування уточнюються.

Система об’єктно-орієнтованих моделей послідовно розвертається в напрямку від статичної моделі загального подання функціонування ІС до моделі динамічної взаємодії об’єктів, на основі якої можуть бути згенеровані класи об’єктів у конкретному програмно-технічному середовищі.

### 6.2.2 Стандарти об’єктного проектування

На сьогодні для об’єктно-орієнтованого моделювання проблемної сфери широко використовується уніфікована мова моделювання UML (Unified Modeling Language), яка розроблена групою провідних комп’ютерних фірм світу OMG (Object Management Group) і фактично є стандартом з об’єктно-орієнтованих технологій. Мова UML реалізована багатьма фірмами-виробниками програмного забезпечення в рамках CASE-технологій, на-приклад Rational Rose (Rational), Natural Engineering Workbench (Software AG), ARIS Toolset (IDS prof. Scheer) та ін.

Уніфікована мова моделювання є графічною мовою для візуалізації, специфікації, конструювання і документування систем, у яких велика роль належить програмному забезпеченню. За допомогою мови UML можна розробити детальний проект створюваної системи, що відображає:

- концептуальні елементи: системні функції і бізнес-процеси;
- конкретні особливості реалізації: класи, написані спеціальними мовами програмування, схеми баз даних, програмні компоненти багаторазового використання.

Система об’єктно-орієнтованих моделей відповідно до нотацій UML включає наступні діаграми:

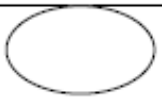
1) *діаграму варіантів використання* (Use-case diagram), що відображає функціональність ІС у вигляді сукупності послідовностей, що виконуються, транзакцій;
2) діаграму класів об’єктів (Class diagram), що відображає структуру сукупності взаємозалежних класів об’єктів аналогічно до ER-діаграми функціонально-орієнтованого підходу;
3) *діаграми станів* (State chart diagrams), кожна з яких відображає динаміку станів об’єктів одного класу і пов’язаних з ними подій;

4) *діаграми взаємодії об'єктів* (Interaction diagrams), кожна з яких відображає динамічну взаємодію об'єктів у рамках одного прецеденту використання;
5) *діаграми діяльності* (Activity diagrams), що відображають потоки робіт у взаємозалежних прецедентах використання (можуть декомпозуватися на більш детальні діаграми);
6) *діаграми пакетів* (Package diagrams), що відображають розподіл об'єктів за функціональними чи забезпечувальними підсистемами (можуть декомпозуватися на більш детальні діаграми);
7) *діаграму компонентів* (Component diagram), що відображає фізичні модулі програмного коду;
8) *діаграму розміщення* (Deployment diagram), що відображає розподіл об'єктів між вузлами обчислювальної мережі.

***Діаграма варіантів використання***

Діаграма варіантів використання виявляє основні бізнес-процеси як послідовності транзакцій, які повинні виконуватися повністю, коли виконання відособленої підмножини дій не має значення без виконання всієї послідовності. Варіанти використання ініціюються із зовнішнього середовища користувачами ІС, так званими акторами. На цьому рівні моделювання не розкривається механізм реалізації процесів. Наведені сутності мають графічні позначення, наведені в табл.6.2.

Таблиця 6.2 – Об'єкти, використовувані в діаграмі варіантів використання

| Найменування об'єкта | Опис | Графічне позначення |
|---|---|---|
| Актор | Зовнішній користувач процесу |  |
| Варіант використання | Бізнес-процес |  |
| Пойменована стрілка | Позначення події |  |

Актор ініціює виконання варіанта використання і отримує від нього результати. Взаємодія (асоціація) актора з варіантом використання здійснюється внаслідок події, яка позначається пойменованою стрілкою. Один актор може брати участь у декількох варіантах використання, а в одному варіанті використання може бути зайнято декілька акторів.

У реалізації варіанта використання можливе виділення декількох потоків подій:

– основний потік подій, який приводить до необхідного результату найбільш коротким шляхом;
– альтернативні потоки подій.

Основний і альтернативний потоки подій у моделі варіантів використання описуються у вигляді неформальних текстових коментарів.

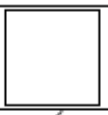
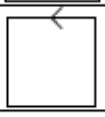
Декілька варіантів використання можуть мати спільну частину, що виділяється в самостійний варіант використання, з яким установлюються відносини використання (uses). З іншого боку, певні варіанти використання можуть бути розширені деталями. У такому разі створюється додатковий варіант використання, з яким встановлюються відносини розширення (extends).

***Діаграми класів об'єктів***

Діаграми класів об'єктів відображають статичну структуру класів об'єктів. Ця діаграма розглядає внутрішню структуру предметної сфери, ієрархію класів об'єктів, статичні зв'язки об'єктів.

Класи об'єктів можуть мати різні стереотипи поведінки: об'єкти-сутності, керівні об'єкти, інтерфейсні об'єкти (табл.6.3). Об'єкти, відображені в діаграмі класів об'єктів, пов'язуються статичними відносинами, які відбивають постійні зв'язки між об'єктами незалежно від виконання конкретного бізнес-процесу. До статичних відносин відносяться узагальнення, агрегація, асоціація об'єктів.

Таблиця 6.3 – Об'єкти, використовувані в діаграмі класів об'єктів

| Найменування об'єкта | Опис | Графічне позначення |
|---|---|---|
| Об'єкт-сутність | Пасивний об'єкт, над яким виконуються операції обробки процесу |  |
| Керівний об'єкт | Активний об'єкт, що координує виконання функцій |  |
| Відносини асоціації | Відносини типу 0.. 1:1;0..1:M; M:N. Відносини можуть бути пойменованими. 0…1 – необов'язковість зв'язку; * – множинність зв'язку |   |
| Відносини узагальнення | Відносини спадкоємства |  |
| Відносини агрегації | Відносини "ціле – частина" |  |

***Діаграми станів***

Діаграми станів відображають поведінку об'єктів одного класу в динаміці, зв'язок станів об'єктів з подіями і визначає:

- які типові стани проходить об'єкт;
- які події ведуть до зміни стану об'єкта;
- які дії об'єкт виконує, коли він отримує повідомлення про зміну стану;
- як об'єкти створюються і знищуються (вхідні і вихідні точки діаграми).

У табл.6.4 наведені використовувані в діаграмі станів поняття і їх графічні позначення.

Таблиця 6.4 –Об'єкти, використовувані в діаграмі станів

| Найменування об'єкта | Опис | Графічне позначення |
|---|---|---|
| Вхідна точка | Визначає подію, яка утворює початковий стан об'єкта. У точку входу не можна перейти зі стану об'єкта |  |
| Стан | Передає ситуацію, протягом якої виконується безперервна діяльність або об'єкт перебуває в стаціонарному положенні. Стан визначається як набір значень атрибутів і відносин, пов'язаних із об'єктом. Ім'я стану повинне бути унікальним лише всередині класу об'єкта, для якого воно визначається |  |
| Перехід станів | Визначає зміну в стані об'єкта, яка відбувається внаслідок події, що виникла в той час, коли об'єкт перебував у даному стані. Кожен перехід станів повинен мати унікальне ім'я |  |
| Вихідна точка | Визначає завершення існування об'єкта. З точки виходу немає переходу стану |  |

З кожним станом пов'язана одна або більше подій, які можуть його змінити. Для стану задаються імена всіх пов'язаних з ним переходів в інші стани.

Перехід станів описується наступними атрибутами.

Призначення – стан об'єкта, в який перейде об'єкт після переходу стану.

Виклик – ім'я події, яка спричиняє перехід станів. Імена подій повинні бути ідентичними у визначенні класу і стану. Події, що спричиняють перехід, можуть бути або зовнішніми, здійснюваними акторами, або внутрішніми,

пов’язаними з поведінкою інших об’єктів, або тимчасовими, пов’язаними із закінченням заданого інтервалу часу.

Умова переходу – це логічний вираз, пов’язаний з атрибутами об’єкта, які повинні бути перевірені для вибору переходу стану. Умова переходу задається в тому випадку, якщо відбувається подія, внаслідок якої може відбутися неоднозначний перехід станів. Умови переходів для одного початкового стану повинні бути взаємовиключними.

Дія – атрибут, що інформаційно описує сутність дії, яка повинна виконуватися під час переходу станів. Цій дії відповідатиме певна процедура, що реалізовує метод класу об’єктів.

Перехід станів графічно позначається лінією, на якій задається принаймні один із наступних атрибутів: виклик, умова переходу, дія.

***Діаграма взаємодії об’єктів***

Для кожного варіанта використання може бути побудована модель динамічної взаємодії об’єктів, яка подається однією з двох форм:

- формою діаграми послідовностей (sequence diagram), що показує послідовність взаємодій на графі;
- формою кооперативної діаграми (collaboration diagram), що показує взаємодію об’єктів у табличній формі.

У діаграмі послідовностей взаємодія об’єктів відтворюється у вигляді стрілки між об’єктами, яка відповідає події або повідомленню від одного об’єкта до другого, який викликає виконання об’єкта, що реагує на подію (повідомлення). Номер стрілки відповідає номеру події в послідовності.

Діаграма кооперативної поведінки подається в табличному вигляді за наступними правилами:

1. У стовпчиках таблиці вказуються об’єкти всіх типів використання, що беруть участь у реалізації варіанта. Порядок розташування активних і пасивних об’єктів довільний і повинен бути зручним для розуміння моделі. Актори прецеденту використання відображаються на правій і лівій межах таблиці.

2. По горизонталі проводяться пойменовані стрілки, що відбивають взаємодію (комунікацію) об’єктів у рамках однієї операції. Ця стрілка означає, що перший об’єкт у рамках виконуваної операції посилає повідомлення другому об’єкту про необхідність виконання дії. У разі отримання повідомлення другий об’єкт виконує дію.

3. На перетині рядків і стовпчика вертикально відображається умовний відрізок часу, протягом якого виконується та або інша дія над об’єктом.

***Діаграма діяльностей***

Діаграми взаємодій не відбивають детально порядку виконання операцій в частині розгалужень, циклічних повторень, паралельності дій.

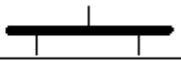
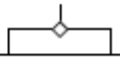
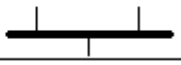
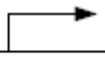
Діаграма діяльностей передбачає відображення даних можливостей. Під діяльністю мається на увазі певна робота, яка може бути розбита на сукупність дій.

Діаграма діяльностей може відбивати взаємодію об'єктів з декількох варіантів використання, що, зокрема, реалізовують окремо стандартні й альтернативні шляхи обробки об'єктів.

Блок, відповідний одній діяльності, може відбивати декілька подій і бути декомпозований аналогічно до блоку структурного підходу.

Поняття і їх графічні позначення, використовувані в діаграмі діяльностей, наведені в табл.6.5.

Таблиця 6.5 – Об'єкти, використовувані в діаграмі діяльностей

| Найменування об'єкта | Опис | Графічне позначення |
|---|---|---|
| Діяльність | Робота, яка може бути розбита на сукупність дій |  |
| Потік | Від діяльності до діяльності |  |
| Поділ потоку | Поділ потоку на діяльності, виконувані паралельно або довільно |  |
| Розв'язання | Вибір одного з потоків залежно від події |  |
| Синхронізація | Перехід до однієї діяльності після виконуваних паралельно |  |
| Ітерація | Повторне виконання діяльності |  |
| Вихід | Визначає завершення існування |  |

***Діаграми пакетів***

В об'єктному підході пакет містить безліч взаємопов'язаних класів об'єктів і відповідає поняттю "підсистема" в структурному підході. Один варіант використання може потребувати класи об'єктів з різних пакетів. Клас об'єктів зазвичай призначається одному пакету, але з позиції досягнення різних підцілей може входити до складу різних пакетів.

Пакетна технологія групування класів об'єктів дозволяє спростити:

- розробку й експлуатацію ІС;
- гнучку адаптацію типових компонентів з позиції їх повторного використання;

– оптимізацію клієнт-серверної архітектури ІС.

Зазвичай ІС розбивається на функціональні й забезпечувальні пакети. Функціональні пакети, відповідні вирішуваним проблемам (завданням), об’єднуються в один спільний пакет "Предметна область". Кожен пакет, у свою чергу, може бути розбитий на підпакети відповідно до семантичної близькості й щільності взаємодії класів об’єктів. Зазвичай пакети предметної області містять ієрархії узагальнення й агрегації. Класи об’єктів, потрібні в декількох підсистемах, виділяються в самостійні пакети. В одному пакеті визначається не більше 20 компонентів, зазвичай 5 – 15.

Із точки зору забезпечувальних пакетів ІС розбивають на п’ять основних пакетів:

– інтерфейс, об’єкти якого реалізують функції взаємодії користувачів з ІС щодо введення-виведення інформації й обміну повідомленнями між підсистемами;
– база даних, об’єкти якої виконують доступ до даних у зовнішній пам’яті;
– управління завданнями, об’єкти якого здійснюють функції диспетчеризації і маршрутизації обробки об’єктів, наприклад у системі управління робочими потоками;
– утиліти, об’єкти якого здійснюють допоміжні функції, наприклад перетворення форматів даних;
– забезпечувальні пакети, тобто ті, що працюють за принципом "клієнт-серверної" архітектури, виконують серверні функції для функціональних об’єктів-клієнтів. Таким чином, забезпечувальні пакети звільняють користувача від знання деталей програмно-технічної реалізації ІС.

***Діаграми компонентів і розміщення***

Діаграма компонентів відображає залежності програмних компонентів, які подаються у вигляді початкових програмних кодів об’єктів, що відкомпілювалися, і виконуваних кодів. Один компонент, як правило, відповідає програмному коду одного пакета класів об’єктів.

Компонент у своєму складі має інтерфейсний клас об’єктів, через який здійснюється доступ до решти класів об’єктів компоненти. За допомогою інтерфейсу об’єкти інших компонентів звертаються не до конкретних об’єктів даного компоненту, а до його інтерфейсного об’єкта. Таким чином спрощується взаємодія компонентів між собою, коли в разі доступу до компоненти з інших компонентів не потрібно знати внутрішню структуру

цієї компоненти. Компонент, до якого здійснюється звернення, може бути не об’єктно-орієнтованою. Достатньо, щоб у такого компонента був лише один інтерфейсний класс об’єктів, який транслює запити до компонента у виклики звичайних процедур. У компонентів може бути декілька інтерфейсів.

### 6.2.3 Технологія об’єктно-орієнтованого проектування

Слід розглянути детальніше зміст операцій, що виконуються у випадку об’єктного підходу до проектування.

*1. Аналіз системних вимог.*

Аналіз системних вимог здійснюється на основі опису предметної області, отриманої в ході робіт з аналізу й проектування бізнес- процесів. Ці матеріали містять опис організаційної структури, структури матеріальних, фінансових і інформаційних потоків, який може бути виконаний як традиційними графічними методами, так і за допомогою об’єктно-орієнтованої методології. На рис. 43 наведена технологія аналізу системних вимог.

*2. Логічне проектування.*

На етапі логічного проектування здійснюється деталізація моделей варіантів використання, класів об’єктів, станів, пакетів і розробка моделей взаємодії об’єктів і діяльностей, які визначають характер методів (процедур) обробки об’єктів. Технологія логічного проектування ІС наведена на рис. 44.

*3. Фізичне проектування.*

На етапі фізичного проектування відбувається деталізація діаграм класів об’єктів і пакетів для їх подальшої реалізації в конкретному програмно-технічному середовищі. Технологія фізичного проектування наведена на рис.6.5.

На етапі реалізації ІС здійснюється генерація кодів класів об’єктів, програмування процедур методів класів об’єктів, заповнення баз даних розміщення компонентів у вузлах обчислювальної мережі (рис.6.6 – 6.8).

Умовні позначення:

D1 – опис предметної сфери;

D2 – діаграма варіантів використання;

D3 – діаграма класів об’єктів;

D4 – діаграма станів об’єктів;

D5 – діаграма пакетів.

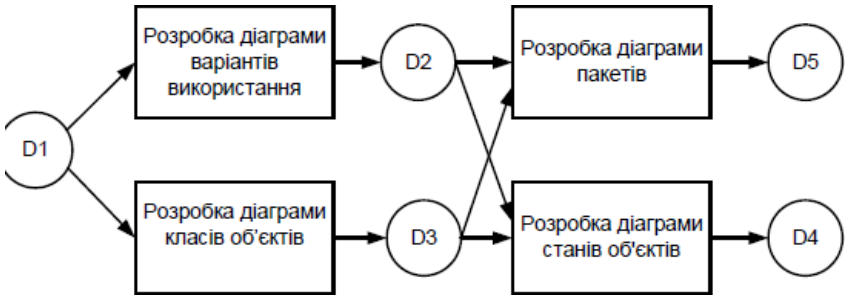



Рисунок 6.5 – Технологія аналізу системних вимог

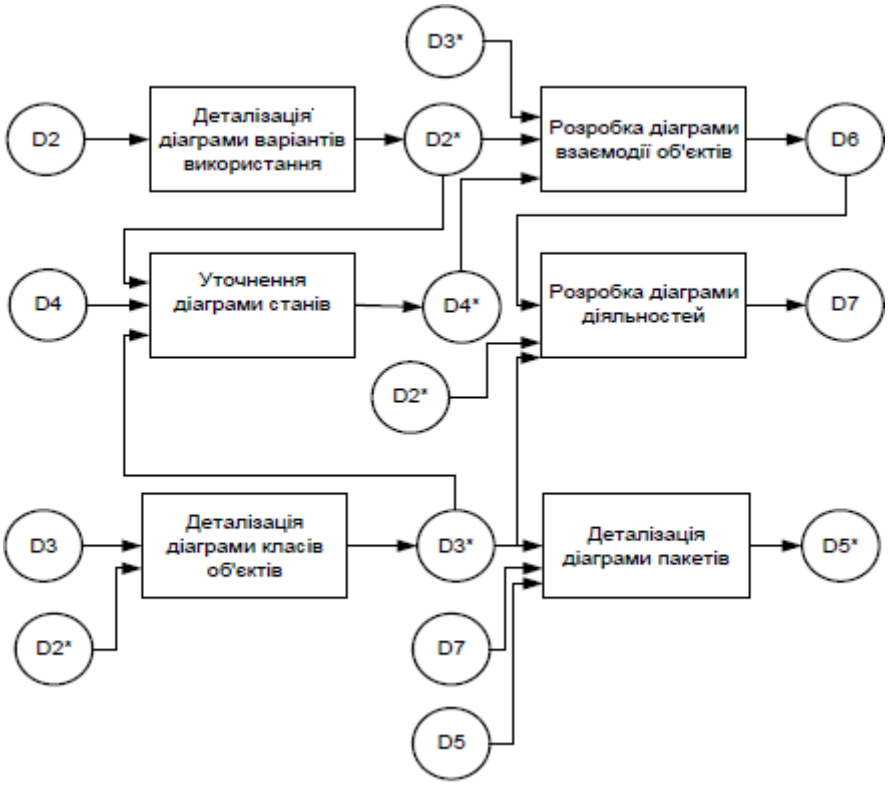



Рисунок 6.6 – Технологія логічного проектування

Умовні позначення:

D2, D2* – діаграма варіантів використання;

D3, D3* – діаграма класів об’єктів;

D4, D4* – діаграма станів об’єктів;

D5, D5* – діаграма пакетів;

D6 – діаграма взаємодії;

D7 – діаграма діяльностей.

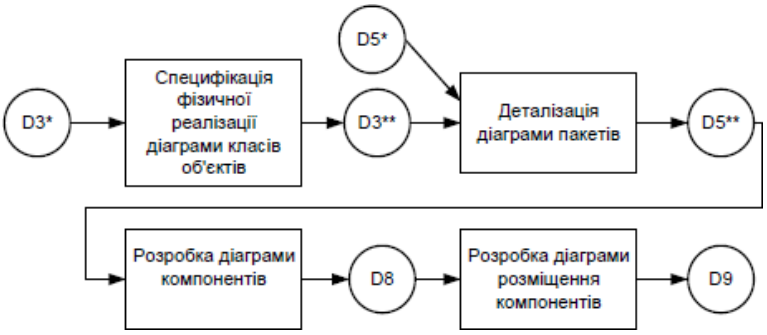



Рисунок 6.7 – Рис. Технологія фізичного проектування

Умовні позначення:
D3*, D3** – діаграма класів об’єктів;
D5* D5**– діаграма пакетів;
D8 – діаграма компонентів;
D9 – діаграма розміщення компонентів.

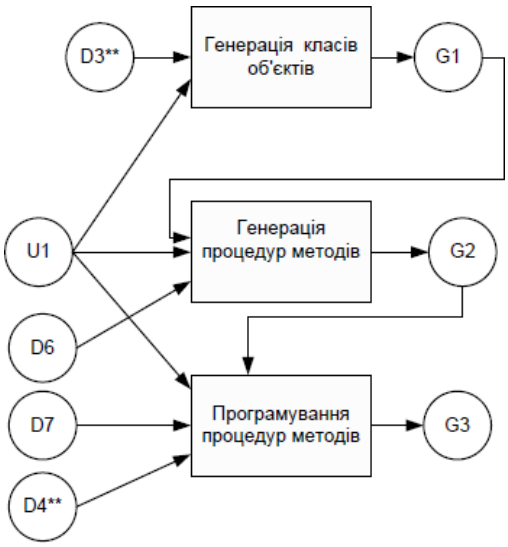



Рисунок 6.8 – Рис. 46. Технологія реалізації ІС

Умовні позначення:
D3* – діаграма класів об’єктів;
D4*– діаграма станів об’єктів;
D6 – діаграма взаємодії;
D7 – діаграма діяльностей;
U1 – універсум об’єктно-орієнтованих мов програмування;
G1 – класи об’єктів;
G2 – шаблони процедур методів класу об’єктів;
G3 – процедури методів.

## 6.3 Порівняння об'єктно-орієнтованого і структурного підходу

1.Функціональние методики розглядають організацію як набір функцій, що перетворює потік інформації що надходить в вихідний потік.

Об'єктні методики розглядають моделюєму організацію як набір взаємодіючих об'єктів-виробничих одиниць. Об'єкт визначається як відчутна реальність – предмет або явище, що мають чітко визначену поведінку.

Основна відмінність функціональної методики від об'єктної методики полягає в чіткому відділенні функцій (методів обробки даних) від самих даних. Крім функціональної декомпозиції існує структура даних, яка перебуває на другому плані. Крім того, не ясні умови виконання процесів обробки інформації, які динамічно можуть змінюватися.

Для перевірки коректності моделювання предметної області між функціональними і об'єктними моделями встановлюються взаємно однозначні зв'язки.

Гідністю функціональних моделей є реалізація структурного підходу до проектування ІС за принципом "зверху-вниз", коли кожен функціональний блок може бути декомпозирований на безліч підфункції і т.д., виконуючи, таким чином, модульне проектування ІС.

При об'єктно-орієнтованому підході змінюється принцип проектування ІС. Спочатку виділяються класи об'єктів, а далі в залежності від можливих станів об'єктів (життєвого циклу об'єктів) визначаються методи обробки (функціональні процедури), що забезпечує найкращу реалізацію динамічної поведінки інформаційної системи.

При виборі методики моделювання предметної області зазвичай в якості критерію виступає ступінь її динамічності. Для більш регламентованих задач більше підходять функціональні моделі, для більш адаптивних бізнес-процесів (управління робочими потоками, реалізації динамічних запитів до інформаційних сховищ) -об'єктно-орієнтовані моделі.

Об'єктна модель природна, оскільки орієнтована на людське сприйняття світу, але менш наочні. Функціональна модель краще систематизована, тому що їй властива процедурна строгість декомпозиції ІС.

Функціональная методики в цілому краще дають уявлення про існуючі функціях в організації, про методи їх реалізації, причому чим вище ступінь деталізації досліджуваного процесу, тим краще вони дають можливість окреслити систему. Під кращим описом в даному випадку розуміється найменша помилка при спробі за отриманою моделі передбачити поведінку

реальної системи. На рівні окремих робочих процедур їх опис практично однозначно збігається з фактичною реалізацією в потоці робіт. На рівні загального опису системи функціональні методики допускають значний ступінь сваволі у виборі загальних інтерфейсів системи, її механізмів і т.д., тобто у визначенні меж системи.

Об'єктний підхід дозволяє добре описати систему на рівні загального опису системи, тому що він заснований на понятті сценарію використання. Ключовим є поняття про сценарії використання як про сеанс взаємодії дійової особи з системою, в результаті якого дійова особа отримує щось, що має для нього цінність.

При об'єктно-орієнтованому підході проектувальник зосереджується на тих функціях системи, які виправдовують її існування. Однак і в цьому випадку завдання визначення меж системи, виділення зовнішніх користувачів є складною.

Технологія потоків даних легко вирішує проблему кордонів системи, оскільки дозволяє за рахунок аналізу інформаційних потоків виділити зовнішні сутності і визначити основний внутрішній процес. Однак відсутність виділених керуючих процесів, потоків і подієвої орієнтованості не дозволяє запропонувати цю методику в якості єдиної.

Об'єктна декомпозиція дає можливість створювати моделі меншого розміру шляхом використання загальних механізмів, що забезпечують необхідну економію засобів вираження. У функціональних моделях при великій кількості рівнів декомпозиції модель може бути великий.

Використання об'єктного підходу істотно підвищує рівень уніфікації розробки та придатність для повторного використання, що веде до створення середовища розробки і переходу до складного створення моделей. У разі успадкування функцій можна абстрагуватися від конкретної реалізації процедур (абстрактні типи даних), які відрізняються для певних підкласів ситуацій. Це дає можливість звертатися до подібних програмним модулям за загальними іменами (поліморфізм) і здійснювати повторне використання програмного коду при модифікації програмного забезпечення. Таким чином, адаптивність об'єктно-орієнтованих систем до зміни предметної області в порівнянні з функціональним підходом значно вище.

Функціональні моделі, як правило, не придатні для повторного використання.

До недоліків об'єктно-орієнтованого підходу відносяться високі початкові витрати. Цей підхід не дає негайної віддачі. Ефект від його

застосування позначається після розробки двох-трьох проектів і накопичення повторно використовуваних компонентів. Структурний підхід не вимагає високих початкових витрат

Підхід від виконуваних функцій інтуїтивно краще розуміється виконавцями при отриманні від них інформації про їх поточну роботу.

# 7 ІНСТРУМЕНТАЛЬНІ ЗАСОБИ ПРОЕКТУВАННЯ ІС

Під засобами проектування інформаційних систем (СП ІС) будемо розуміти комплекс інструментальних засобів, що забезпечують в рамках обраної методології проектування підтримку повного життєвого циклу (ЖЦ) ІС.

Вони включають в себе, як правило, стратегічне планування, аналіз, проектування, реалізацію, впровадження та експлуатацію.

Кожен етап характеризується певними завданнями і методами їх вирішення, вихідними даними, отриманими на попередньому етапі, і результатами. При аналізі СП їх слід розглядати не локально, а в комплексі, що дозволяє реально охарактеризувати їх гідності, недоліки і місце в загальному технологічному циклі створення ІС.

Необхідна апаратна платформа для ІС може формуватися з компонентів різних фірм-виробників. Проте вибрати і скомплексіровать різні інструментальні засоби, кожне з яких може бути одним зі світових лідерів в своєму класі, досить важко, а часом і неможливо.

У загальному випадку стратегія вибору СП для конкретного застосування залежить від наступних факторів:

- характеристик модельованої предметної області;
- цілей, потреб і обмежень майбутнього проекту ІС, включаючи кваліфікацію беруть участь в процесі проектування фахівців;
- використовуваної методології проектування.

Сучасні складні ІС і проекти, що забезпечують їх створення, характеризуються, як правило, такими особливостями:

- складністю предметної області (досить велика кількість функцій, об'єктів, атрибутів і складні взаємозв'язки між ними), що вимагає ретельного моделювання і аналізу даних і процесів;
- наявністю сукупності тісно взаємодіючих компонентів - підсистем, що мають свої локальні завдання і цілі функціонування;
- ієрархічною структурою взаємозв'язків компонентів, що забезпечує стійкість функціонування системи;
- ієрархічної сукупністю критеріїв якості функціонування компонентів і ІС в цілому, що забезпечують досягнення головної мети - створення і подальшого застосування системи;

- відсутністю прямих аналогів, що обмежують можливість використання будь-яких типових проектних рішень і прикладних систем;
- необхідністю досить тривалого співіснування старих додатків і розроблених нових БД і додатків;
- наявністю потреби як в традиційних додатках, пов'язаних з обробкою транзакцій і рішенням регламентних завдань, так і в додатках аналітичної обробки (підтримки прийняття рішень), використовувати нерегламентовані запити до даних великого обсягу;
- підтримкою одночасної роботи чималої кількості локальних мереж, що пов'язуються в мережу масштабу підприємства, і територіально віддалених користувачів;
- функціонуванням в неоднорідній операційному середовищі на декількох обчислювальних платформах;
- роз'єднаністю і різнорідністю окремих мікроколлектівов розробників за рівнем кваліфікації і традиціям, що склалися використання тих чи інших інструментальних засобів;
- - Істотною тимчасової протяжністю проекту, зумовленої, з одного боку, обмеженими можливостями колективу розробників, і, з іншого боку, масштабами організації-замовника і різним ступенем готовності окремих її підрозділів до впровадження ІС.

Крім створення оригінальних і унікальних АІС, досить часто застосовуються універсальні проектні рішення, які повністю або частково можуть бути адаптовані в різних предметних областях.

Спроб використовувати готові рішення і програмне забезпечення, що працюють в інших умовах в ряді, особливо великих, проектів, може виявитися не тільки неефективною, але і згубної для впроваджує ці рішення організації. З іншого боку, використання відкритих стандартів, принципів модульності і інше дозволяє формувати гнучкі, що розширюються та ефективно супроводжувані системи.

У той же час, необхідно відзначити, що основним принципом проектування АІС є використання модульного проектування. Саме воно дозволяє позбутися від багатьох негативних наслідків застосування будь-яких видів АІС.

При реалізації автоматизації процесів, що базуються на взаємодії невеликих самостійних компонентів, все частіше рекомендується

використовувати веб-сервіси. Відповідно до цієї ідеології функціональність кожного блоку через уніфікований інтерфейс повинна бути доступна будь-яким іншим додаткам. З таких блоків легко побудувати складні ланцюжки автоматизованих процесів, при цьому один блок може використовуватися в декількох ланцюжках.

Базові можливості створення і використання веб-служб реалізують всі сучасні програмні платформи, що становлять основу інфраструктури інформаційних технологій багатьох підприємств. При цьому значних переваг вдається досягти в результаті планових модифікацій існуючої ІС шляхом розробки нових інтерфейсів, реінжинірингу окремих модулів, впровадження нових систем автоматизації.

В результаті природним шляхом формується база подальшого розвитку сервісноорієнтірованних ІС.

## 7.1 Комплекс узгоджених інструментальних засобів

Кожна методологія створення ІС підтримується комплексом узгоджених між собою інструментальних засобів, який забезпечує безперервний цикл автоматизації процесів, що виконуються на всіх етапах ЖЦ ІС .. Узгодженість цих коштів забезпечується наявністю інтерфейсів для прямої взаємодії і підтримкою загальноприйнятих стандартів відкритих систем.

Комплекс засобів такого роду дозволяє будувати моделі, що описують діяльність організації, формувати вимоги до ІС, швидко переходити від моделей вимог до ІС до проекту додатків і баз даних. Він забезпечує підтримку швидкої итеративной розробки додатків, їх тестування і інтеграцію в систему. Закладені в методологію і підтримані цими інструментальними засобами принципи, засновані на використанні моделей і повторному використанні специфікацій, забезпечують можливість швидкого внесення змін як на стадіях створення ІС, так і на стадіях супроводу і розвитку.

Створені на базі цього набору засобів розподілені ІВ (додатки та БД) можуть бути реалізовані як в двухзвенной, так і в триланкової архітектурі клієнт-сервер. Цей же набір засобів дозволяє переносити додатки і бази даних на різні платформи без перепрограмування. Програми, створені на базі цього набору засобів, є відкритими і масштабованими. До складу набору входять засоби реінжинірингу, що дозволяють автоматично відновлювати

модель існуючої системи. Відповідно до проекту ця модель може бути використана для побудови моделей нової системи.

Методологія і підтримує її набір інструментальних засобів забезпечують повний контроль і гнучке управління ходом розробки, включаючи:

- підтримку колективної розробки з можливістю паралельного і розподіленого виконання різних робіт;
- можливість переходу до наступного етапу (кроку), не чекаючи повного завершення попереднього;
- застосування методів контролю якості та постійний контроль отриманих результатів;
- підтримку ітеративного характеру розробки (можливість перегляду отриманих результатів і повернення на будь-який з попередніх етапів;
- можливість швидкого внесення змін у вимоги в процесі розробки;
- управління конфігурацією.

Засоби проектування повинні бути:

- в своєму класі інваріантними до об'єкта проектування;
- охоплювати в сукупності всі етапи життєвого циклу ЕІС;
- технічно, програмно і інформаційно сумісними;
- простими в освоєнні і застосуванні;
- економічно доцільними.

Засоби проектування ЕІС можна розділити на два класи: без використання ЕОМ і з використанням ЕОМ.

# 8 МОДЕЛІ ДАНИХ, МОДЕЛІ ПРОЦЕСІВ ТА ЇХ ПРОЕКТУВАННЯ З ДОПОМОГОЮ ERWIN

## 8.1 Методології моделювання даних

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

Найбільш поширеним засобом моделювання даних є діаграми "сутність – зв’язок" (ERD). З їх допомогою визначаються важливі для предметної області об’єкти (сутності), їх властивості (атрибути) і відносини один з одним (зв’язки). ERD безпосередньо використовуються для проектування реляційних баз даних. Нотація ERD була вперше запроваджена П. Ченом (Chen) і отримала подальший розвиток у роботах Беркера.

Методологія IDEF1. Метод IDEF1, розроблений Т. Ремей (T. Ramey), також заснований на підході П. Чена і дозволяє забудувати модель даних, еквівалентну реляційній моделі в третій нормальній формі. Зараз на основі вдосконалення методології IDEF1 створена її нова версія – методологія IDEF1X. IDEF1X розроблена з урахуванням таких вимог, як простота вивчення і можливість автоматизації. IDEF1X-діаграми використовуються рядом поширених CASE-засобів (зокрема, ERWin, Design/IDEF).

### 8.1.1 Діаграма "сутність-зв'язок" (ERD). Метод моделювання даних IDEFIX

Розроблена функціональна модель системи відповідає на питання «Що повинна робити система?» І «За рахунок яких дій може бути досягнутий необхідний результат?». Ця модель також дозволяє концептуально визначити набори даних, що використовуються в системі.

У той же час вона не відповідає на питання «Яким чином організовані дані в системі?». Для відповіді на нього необхідно побудувати інформаційну модель системи.

Традиційно процедуру проектування розбивають на три етапи, кожен з яких завершується створенням відповідної інформаційної моделі.

**Етап 1-й.** *Концептуальне проектування* – створення уявлення (схеми, моделі) БД, що включає визначення найважливіших сутностей (таблиць) і

зв'язків між ними, але не залежить від моделі БД (ієрархічної, мережевої, реляційної) І фізичної реалізації (цільової СУБД).

**Етап 2-й.** *Логічне проектування* – розвиток концептуального уявлення БД з урахуванням прийнятої моделі (ієрархічної, мережевої, реляційної).

**Етап 3-й**. *Фізичне проектування* – розвиток логічної моделі БД з урахуванням обраної цільової СУБД.

Концептуальне та логічне проектування разом називають також інфологичним або семантичним проектуванням.

В даний час для проектування БД активно використовуються CASE-засоби, в основному орієнтовані на використання ERD (Entity – Relationship Diagrams, діаграми «сутність-зв'язок»). З їх допомогою визначаються важливі для предметної області об'єкти (сутності), відносини один з одним (зв'язку) і їх властивості (атрибути). Слід зазначити, що кошти проектування ERD в основному орієнтовані на реляційні бази даних, і якщо існує необхідність проектування іншої системи, скажімо об'єктно-орієнтованої, то краще обрати інші методи проектування.

ERD були вперше запропоновані П. Ченом в 1976 р Основні елементи ERD перераховані нижче.

*Сутність* (таблиця – відношення) – реальний або уявний об'єкт, що має істотне значення для розглянутої предметної області, інформація про який підлягає зберіганню. Якщо висловлюватися точніше, то це не об'єкт, а набір об'єктів (клас) з однаковими властивостями.

*Зв'язок* – деяка асоціація між двома сутностями, значима для розглянутої предметної області.

*Атрибут* (стовпець, поле) – властивість сутності або зв'язку.

Більшість сучасних CASE-засобів моделювання даних, як правило, підтримує кілька графічних нотацій побудови інформаційних моделей. Зокрема система ERwin фірми Computer Associates підтримує дві нотації: IDEF1X і IE (англ. Information Engineering – інформаційне проектування). Дані нотації є взаємно-однозначними, т. Е. Перехід від однієї нотації до іншої і назад виконується без втрати якості моделі. Відмінність між ними полягає лише в формі відображення елементів моделі.

При використанні будь-якого CASE-засобу спочатку будується логічна модель БД у вигляді діаграми із зазначенням сутностей і зв'язків між ними.

*Логічною моделлю* називається універсальне уявлення структури даних, незалежне від кінцевої реалізації бази даних і апаратної платформи. На підставі отриманої логічної моделі переходять до фізичної моделі даних.

*Фізична модель* являє собою діаграму, що містить всю необхідну інформацію для генерації БД для конкретної СУБД або навіть конкретної версії СУБД. Якщо в логічній моделі не має значення, які ідентифікатори носять таблиці і атрибути, тип даних атрибутів, то у фізичній моделі повинно бути повний опис БД відповідно до прийнятого в ній синтаксисом, з зазначенням типів атрибутів, тригерів, збережених процедур і т. д. За однією і тією ж логічної моделі можна створити кілька фізичних.

Перерахований вище порядок дій називається пряме проектування БД (Forward Engineering DB). CASE-засоби дозволяють виконувати також зворотне проектування БД (Reverse Engineering DB), тобто на підставі системного каталогу БД або DDL-скрипта побудувати фізичну і, далі, логічну модель даних.

Крім режимів прямого і зворотного проектування, CASE-засоби зазвичай підтримують синхронізацію між моделлю і системним каталогом БД, т. ї. При зміні моделі вони можуть автоматично внести всі необхідні зміни в існуючу БД і навпаки.

Розвинені CASE-засоби мають також вбудованою підсистемою пошуку та виправлення помилок в моделі. Особливо корисна ця функція при проектуванні великих БД, що містять десятки або сотні таблиць, а також при зворотному проектуванні.

Слід зазначити, що сучасні СУБД мають свої вбудовані засобами візуального моделювання даних. Деякі з них навіть підтримують класичні нотації ERD. Недоліками такого моделювання є побудова тільки фізичної моделі даних і неможливість швидкого переходу на іншу СУБД, якщо таке рішення прийнято. Перевагою цього підходу є більш повне використання потенціалу СУБД, адже розробники СУБД краще за інших знають її особливості та можливості.

Далі розглядається процедура прямого проектування з використанням методології IDEF1X. Методологія IDEF1 була розроблена Т. Ремі. В даний час на основі IDEF1 створена її нова версія - методологія IDEF1X, яка в 1981 р прийнята ICAM як федерального стандарту США.

Мета концептуального проектування – створення концептуальної моделі даних на основі уявлень про предметну область кожного окремого типу користувачів. Концептуальна модель представляє собою опис основних сутностей (таблиць) і зв'язків між ними без урахування прийнятої моделі БД і синтаксису цільової СУБД. Часто на такій моделі будуть відображатися імена сутностей (таблиць) без вказівки їх атрибутів. Подання користувача

включає в себе дані, необхідні конкретному користувачеві для прийняття рішень або виконання деякого завдання.

Послідовність кроків при концептуальному проектуванні:

***Виділення сутностей.***

Перший крок в побудові концептуальної моделі даних полягає у визначенні основних об'єктів (сутностей), які можуть цікавити користувача і, отже, повинні зберігатися в БД. При наявності функціональної моделі IDEF0 прообразами таких об'єктів є входи, управління і виходи. Ще краще для цих цілей використовувати DFD. Прообразами об'єктів в цьому випадку будуть накопичувачі даних. Накопичувач даних є сукупністю таблиць (набором об'єктів) або безпосередньо таблицею (об'єктом). Для більш детального визначення набору основних об'єктів необхідно також проаналізувати потоки даних і весь методичний матеріал, необхідний для вирішення задачі.

Сутність (Entity) – збірне поняття, деяка абстракція реально існуючого об'єкта, процесу або явища, про який необхідно зберігати інформацію.

Можливі труднощі у визначенні об'єктів пов'язані з використанням постановниками задачі:

- приклади і аналогії при описі об'єктів (наприклад, замість узагальнюючого поняття «працівник» вони можуть згадувати його функції або займану посаду: «керівник», «відповідальний», «контролер», «заступник»);
- синонімів (наприклад, «що допускається швидкість» і «встановлена швидкість», «розробка» і «проект», «бар'єрне місце» і «обмеження швидкості»);
- омонімів (наприклад, «програма» може позначати комп'ютерну програму, план майбутньої роботи або програму телепередач).

Далеко не завжди очевидно те, чим є певний об'єкт – сутністю, зв'язком або атрибутом. Наприклад, як слід класифікувати «сімейний шлюб»? На практиці це поняття можна цілком обгрунтовано віднести до будь-якої зі згаданих категорій. Аналіз є суб'єктивним процесом, тому різні розробники можуть створювати різні, але цілком припустимі інтерпретації одного і того ж факту. Вибір варіанту в значній мірі залежить від здорового глузду і досвіду проектувальника.

Кожна сутність повинна володіти деякими властивостями:

- повинна мати унікальне ім'я, і до одного і того ж імені повинна завжди застосовуватися одна й та сама інтерпретація;

– володіти одним або декількома атрибутами, які або належать сутності, або успадковуються через зв'язок;
– володіти одним або декількома атрибутами (первинним ключем), які однозначно ідентифікують кожен екземпляр сутності, т. Е. Роблять унікальною кожен рядок таблиці;
– може мати будь-якою кількістю зв'язків з іншими сутностями.

За методологією IDEF1X, на діаграмі сутність зображується прямокутником. Залежно від режиму уявлення діаграми прямокутник може містити ім'я суті, її опис, список її атрибутів і інші відомості.

Горизонтальна лінія прямокутника розділяє атрибути сутності на два набори – атрибути, складові первинний ключ у верхній частині, і інші (не входять в первинних ключ) – в нижній частині.

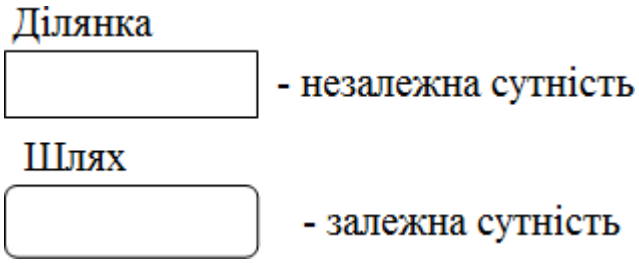



Рисунок 8.1 – Відображення залежної та не залежної сутностей

Сутність в методології IDEF1X *є незалежною* (сильною, батьківської, домінантною, власником), якщо сутність не залежить від існування іншої сутності (іншими словами, кожен екземпляр сутності може бути однозначно ідентифікований без визначення його зв'язків з іншими сутностями, або унікальність екземпляра визначається тільки власними атрибутами ).

Сутність називається *залежною* (слабкою, дочірньої, підпорядкованої), якщо її існування залежить від існування інших сутностей. Термінологія «батьківська» – «дочірня» і «власник» – «підлеглий» також може використовуватися щодо двох залежних сутностей, якщо екземпляри однієї з них (дочірньої, підпорядкованої) можуть бути однозначно визначені з використанням примірників іншої (батьківської, власника), незважаючи на те, що друга сутність в свою чергу залежить від третьої сутності.

***Визначення атрибутів***

Виявлені атрибути можуть бути наступних видів.

*Простий* (атомарний, неподільний) – складається з одного компонента з незалежним існуванням (наприклад, «посаду працівника», «зарплата», «норма непогашеного прискорення», «радіус кривої»).

*Складовий* (псевдоатомарний) – складається з декількох компонентів (наприклад, «ПІБ», «адреса»). Ступінь атомарності атрибутів, що закладається в модель, визначається розробником. Якщо від системи не потрібно вибірки всіх клієнтів з прізвищем Іванов або проживають на вулиці Комсомольській, то складові атрибути можна не розбивати на атомарні.

*Однозначний* – містить тільки одне значення для одного екземпляра сутності (наприклад, у кривої в плані може бути тільки одне значення радіусу, кута повороту, підвищення зовнішньої рейки).

*Багатозначний* – містить кілька значень (наприклад, у одного відділення компанії може бути кілька контактних телефонів).

*Похідний* (який вираховується) – значення атрибута може бути визначено за значеннями інших атрибутів (наприклад, «вік» може бути визначений за «датою народження» і поточну дату, встановленої на комп'ютері).

*Ключовий* – служить для унікальної ідентифікації примірника сутності (входить до складу первинного ключа).

Неключовий (описовий) – не входить в первинний ключ.

*Обов'язковий* – при введенні нового екземпляра в сутність або редагуванні обов'язково вказується допустиме значення атрибута, т. Е. Воно після редагування не може бути невизначеним (NOT NULL).

Після визначення атрибутів задаються їх домени (області допустимих значень), наприклад:

- найменування ділянки – набір з букв російського алфавіту довжиною не більше 60 символів;
- поворот кривої – допустимі значення «Л» (вліво) і «П» (вправо);
- радіус кривої – позитивне число не більше 4 цифр.

Завдання доменів визначає набір допустимих значень для атрибута (декількох атрибутів), а також тип, розмір і формат атрибута (атрибутів).

На підставі виділеного безлічі атрибутів для сутності визначається набір ключів. Ключ – один або кілька атрибутів сутності, службовців для однозначної ідентифікації її примірників або для їх швидкого пошуку. Виділяють такі типи ключів:

- *суперключ (superkey);*

Атрибут або безліч атрибутів, яке єдиним чином ідентифікує екземпляр сутності. Суперключ може містити «зайві» атрибути, які є обов'язковими для унікальної ідентифікації примірника. При правильному проектуванні структури БД суперключ в кожної сутності (таблиці) буде повний набір її атрибутів.

– *потенційний ключ (potential key);*

Суперключ, який не містить підмножини, що також є суперключ даної суті, т. Е. Суперключ, що містить мінімально необхідний набір атрибутів, єдиним чином ідентифікують примірник сутності. Сутність може мати кілька потенційних ключів. Якщо ключ складається з кількох атрибутів, то він називається складовим ключем. Серед усієї безлічі потенційних ключів для однозначної ідентифікації примірників вибирають один, так званий первинний ключ, який використовується в подальшому для встановлення зв'язків з іншими сутностями.

– *первинний ключ (primary key);*

Потенційний ключ, який обраний для унікальної ідентифікації екземплярів всередині суті.

– *альтернативні ключі (alternative key);*

Потенційні ключі, які не вибрані в якості первинного ключа.

Якщо якийсь атрибут (набір атрибутів) присутній в декількох сутності, то його наявність зазвичай відображає наявність зв'язку між екземплярами цих сутностей. У кожної зв'язку одна сутність виступає як батьківська, а інша – в ролі дочірньої. Це означає, що один екземпляр батьківського суті може бути пов'язаний з декількома екземплярами дочірньої. Для підтримки цих зв'язків обидві сутності повинні містити набори атрибутів, за якими вони пов'язані. У батьківській суті це первинний ключ. У дочірньої сутності для моделювання зв'язку повинен бути присутнім набір атрибутів, відповідний первинному ключу батьківської. Однак тут цей набір атрибутів вже є вторинним ключем. Даний набір атрибутів в дочірньої суті прийнято називати зовнішнім ключем (foreign key).

В нотації IDEF1X атрибути зображуються у вигляді списку імен усередині блоку сутності. Атрибути, що визначають первинний ключ, розміщуються нагорі списку і відділяються від інших атрибутів горизонтальною лінією. Попередня ідентифікація атрибутів на прикладі двох сутностей показана на рис. 8.2.


Рисунок 8.1 – Суті і приклад первинного та альтернативного ключів

***Визначення зв'язків.***

Зв'язок (Relationship) – засіб представлення відносини між сутностями.

Найбільш характерними типами зв'язків між сутностями є наступні.

*Зв'язки типу «частина-ціле»*, які визначаються зазвичай дієсловами «складається з», «включає».

*Класифікація зв'язку* (наприклад, «тип-підтип», «безліч-елемент», «загальне-приватне» ).

*Виробничі зв'язки* (наприклад, «начальник-підлеглий»).

*Функціональні зв'язки,* які визначаються зазвичай дієсловами «виробляє», «впливає», «залежить від», «обчислюється по».

Серед них виділяються тільки ті зв'язки, які необхідні для задоволення вимог до розробки системи.

Зв'язок характеризується наступним набором параметрів:

1) ім'ям – вказується у вигляді дієслова і визначає семантику (смислове підгрунтя) зв'язку;
2) кратністю (потужність):

– *один-до-одного (1: 1);*
– *один-до-багатьох (1: N);*
– *багато-до-багатьох (N: M, N = M або N <> M).*

Кратність показує, яка кількість примірників однієї сутності визначається екземпляром іншої. Наприклад, на одній ділянці (описується рядком таблиці «Ділянки») може бути один, два і більше шляхів (кожен шлях описується окремим рядком в таблиці «Шляхи»). В даному випадку зв'язок 1: N. Інший приклад: один шлях проходить через кілька роздільних пунктів і через один роздільний пункт може проходити кілька шляхів - зв'язок N: M;

3) типом;

Ідентифікує (атрибути однієї сутності, звані зовнішнім ключем, входять до складу дочірньої і служать для ідентифікації її примірників, тобто

входять в її первинний ключ) і неідентіфіцірующей (зовнішній ключ мається на дочірньої суті, але не входить до складу первинного ключа).

4) обов'язковістю;

Обов'язкова (при введенні нового екземпляра в дочірню сутність заповнення атрибутів зовнішнього ключа обов'язково і для введених значень повинен існувати екземпляр в батьківської сутності) і необов'язкова (заповнення атрибутів зовнішнього ключа в екземплярі дочірньої сутності необов'язково або введеним значенням не відповідає екземпляр в батьківської сутності).

5) ступенем участі;

Кількістю сутностей, що беруть участь в зв'язку. В основному між сутностями існують бінарні зв'язку, т. ї. асоціації, що зв'язують дві сутності (ступінь участі дорівнює 2). Наприклад, «Ділянка» складається з «Шляхів». У той же час за ступенем участі можливі наступні типи зв'язків:

6) унарна (рекурсивна);

Сутність може бути пов'язана сама з собою. Наприклад, в таблиці «Працівники» можуть бути записи та підпорядкованими, і по їх начальникам. Тоді можлива зв'язок «начальник» - «підлеглий», певна на одній таблиці.

7) тернарна;

Пов'язує три сутності. Наприклад, «Студент» на «Сесії» отримав «Оцінку з дисципліни»;

8) кватернарна.

У методології IDEF1X ступінь участі може бути тільки унарною або бінарної. Зв'язки більшою мірою наводяться до бінарним увазі.

Зовнішній вигляд зв'язку на діаграмах IDEF1X вказує на її потужність, тип і обов'язковість (табл. 8.1).

Таблиця 8.1 –Типи зв'язків

| **Зовнішній вигляд** | **Тип обов’язковості звязку** | **Потужність зв’язку з права** |
|---|---|---|
| ——————— | Обов'язкова, ідентифікуюча | 1 |
| ———————● | Обов'язкова, ідентифікуюча | 0 .. ∞ |
| ———————● Z | Обов'язкова, ідентифікуюча | 0 або 1 |
| ———————● P | Обов'язкова, ідентифікуюча | 1 .. ∞ |
| ———————● | Обов'язкова, ідентифікуюча | <число> |

| | | |
|---|---|---|
| <число> | | |
| ◇---------● | Не обов'язкова, не ідентифікуюча | 0 .. ∞ |
| ----------● | Обов'язкова, не ідентифікуюча | 0 .. ∞ |

Ідентифікуючий зв'язок відображається суцільною лінією, неідентіфіцірующей – пунктирною.

Необов'язковість позначається ромбиком.

На рис. 8.3 – 8.5 наведені приклади відображення зв'язків.

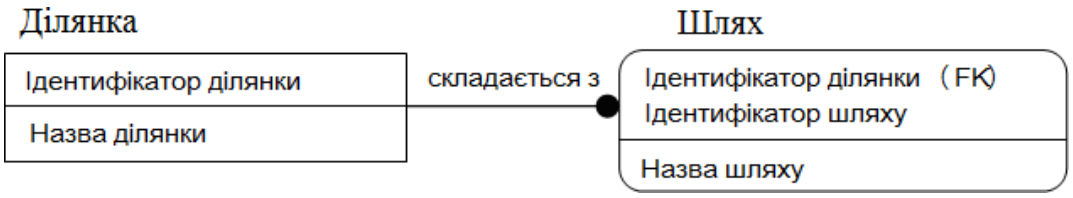



Рисунок 8.3 – Ідентифікуючий зв'язок

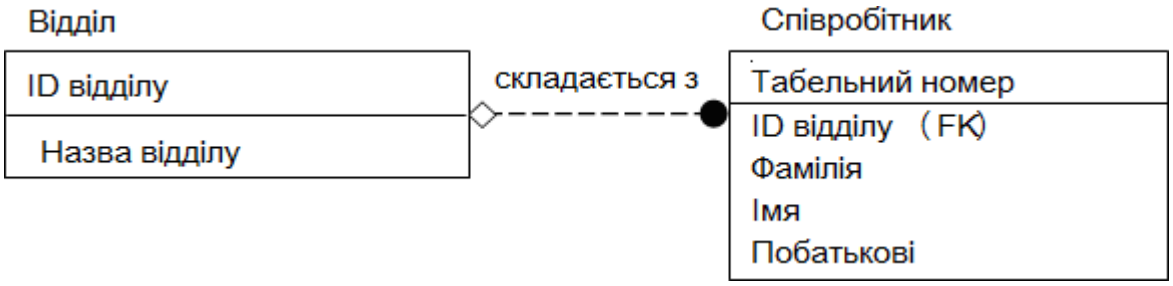



Рисунок 8.4 – Неідентифікуючий звязок

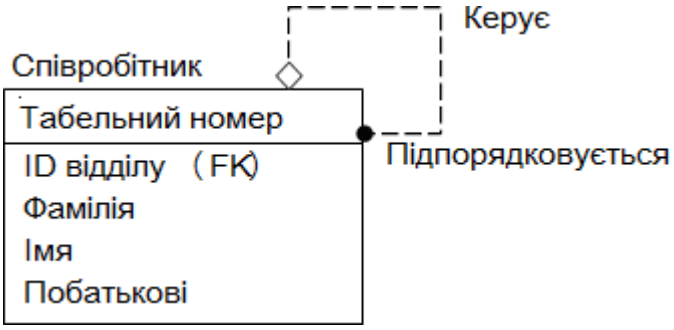



Рисунок 8.5 – Рекурсивний звязок

На рис. 8.4 зв'язок необов'язковий, оскільки деякі співробітники не обов'язково повинні входити в будь-якої відділ (наприклад, директор підприємства), і неідентіфіцірующей, так як табельний номер унікальний для кожного співробітника.

***Визначення суперкласів і підкласів***

У тих випадках, коли дві і більше сутностей по набору атрибутів незначно відрізняються один від одного, можна застосовувати в моделі конструкцію – ієрархію спадкування (категорій), що включає в себе суперклас і підкласи.

*Суперклас* – сутність, що включає в себе підкласи.

Ієрархія спадкування являє собою особливий тип об'єднання сутностей, які поділяють спільні характеристики. Наприклад, в організації працюють службовці, зайняті повний робочий день (постійні службовці) і сумісники. З їхніх спільних властивостей можна сформувати узагальнену сутність (родового предка) «Співробітник» (рис. 8.6), щоб представити інформацію, загальну для всіх типів службовців. Специфічна для кожного типу інформація може бути розміщена в додаткових сутності (нащадках) «Постійний співробітник» і «Сумісник».

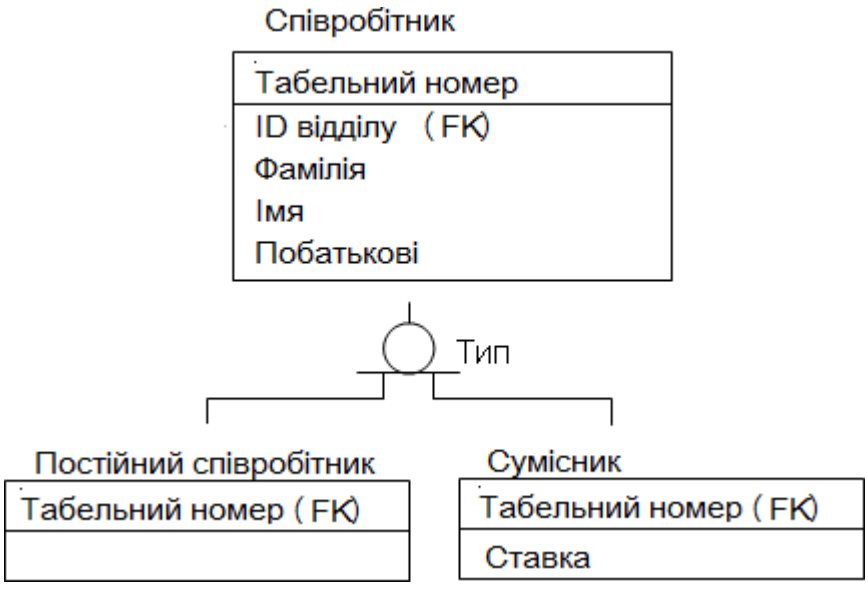



Рисунок 8.6 – Ієрархія спадкування (неповна категорія)

Зазвичай ієрархію спадкування створюють, коли кілька сутностей мають загальні за змістом атрибути, або коли сутності мають загальні за

змістом зв'язку (наприклад, якби «Постійний співробітник» і «Сумісник» мали б подібну за змістом зв'язок «працює в" з сутністю «Організація» ).

Для кожної категорії потрібно вказати дискримінатор – атрибут родового предка, який показує, як відрізнити одну сутність від іншої. У наведеному прикладі діскрімінатор - атрибут «Тип».

Ієрархії категорій діляться на два типи: неповні (рис. 8.6) і повні (рис. 8.7).

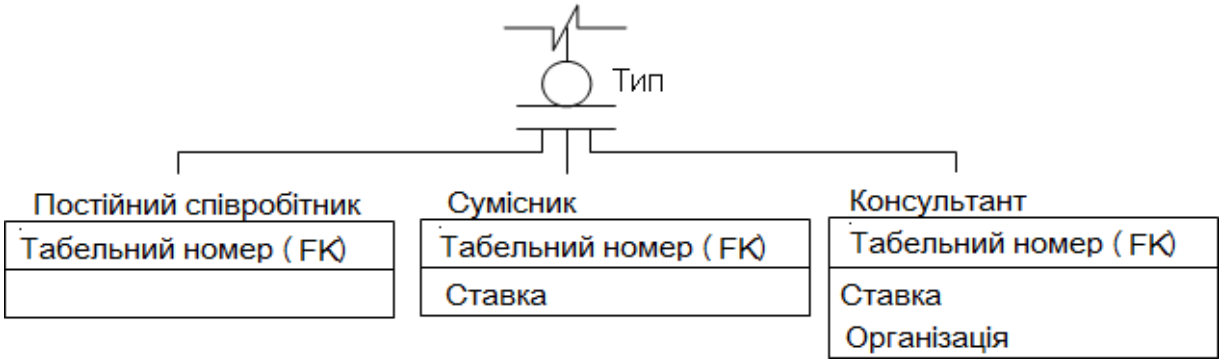



Рисунок 8.7 – Ієрархія спадкування (повна категорія)

## 8.2 Логічне проектування з використанням методології IDEF1X

Мета логічного проектування – розвинути концептуальне уявлення системи з урахуванням прийнятої моделі БД (ієрархічної, мережевої, реляційної).

Приймемо в якості моделі реляционную БД в третій нормальній формі (набір нормалізованих відносин з кратністю зв'язків 1: N). Тому необхідно буде перевірити концептуальну модель за допомогою методів нормалізації та контролю виконання транзакцій.

Транзакція – одну дію або їх послідовність, виконуваних як єдине ціле одним або декількома користувачами (прикладними програмами) з метою здійснення доступу до БД і зміни її вмісту.

***Видалення зв'язків N: M.***

Якщо в концептуальній моделі присутні зв'язку N: M, то їх слід усунути шляхом визначення проміжної сутності. Зв'язок N: M замінюється двома зв'язками типу 1: M, що встановлюються з новоствореної сутністю (рис. 8.8).

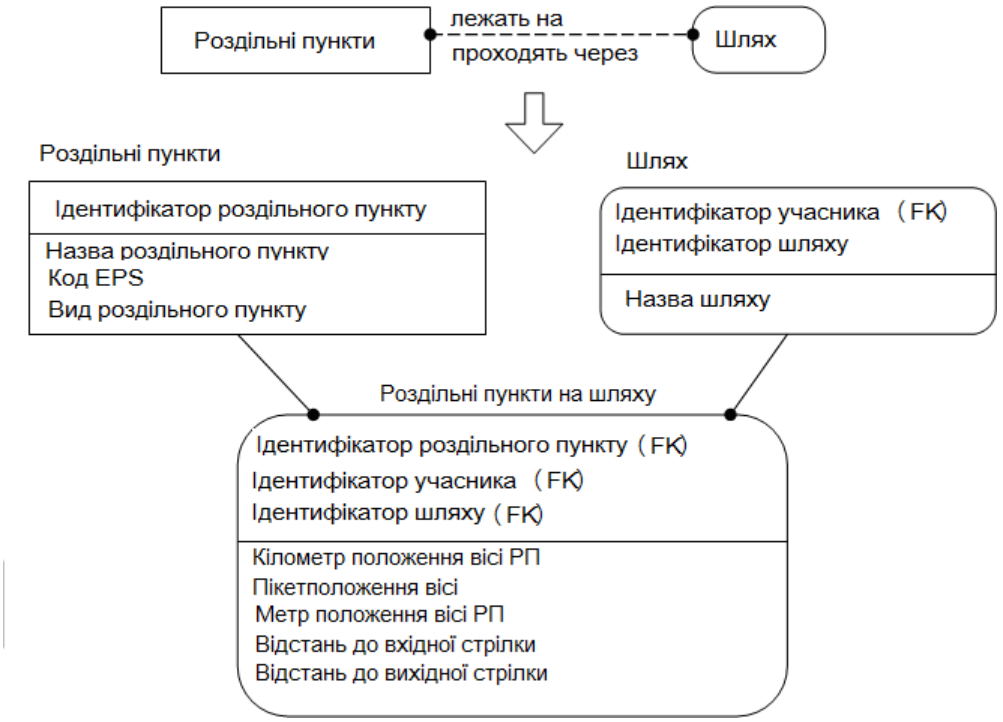



Рисунок 8.8 – Заміна зв'язку N: M

***Видалення зв'язків з атрибутами.***

Зв'язки з атрибутами повинні бути перетворені в суті.

У розробленій концептуальної моделі існував зв'язок N: M ( «Окремі пункти»: «Шляхи»), яка мала власні атрибути. Після усунення зв'язок N: M, її атрибути перейшли в сутність «Окремі пункти на шляху» (див. рис. 8.8, неключових атрибути).

У графічній нотації IDEF1X не передбачено відображення зв'язків з атрибутами, хоча деякі методології ERD допускають їх наявність і відображення.

***Видалення складних зв'язків (зі ступенем участі більше 2).***

Складну зв'язок замінюють необхідною кількістю бінарних зв'язків 1: N з новоствореної сутністю, яка і показує цей зв'язок.

Видалення рекурсивних зв'язків (зі ступенем участі 1).

Рекурсивную зв'язок замінюють, визначивши додаткову сутність і необхідну кількість зв'язків (рис. 8.9).

У даній схемі можна прийняти наступне правило: якщо по суті «Співробітник» атрибути «Табельний номер» і «Табельний номер керівн.» збігаються, то набір атрибутів по суті «Співробітник» характеризує керівника.

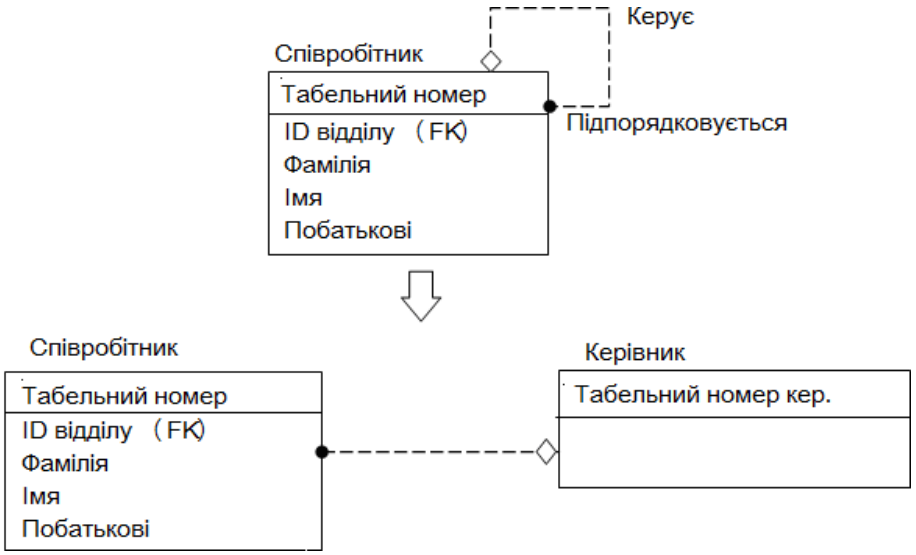



Рисунок 8.9 – Заміна рекурсивної зв'язку

***<u>Видалення багатозначних атрибутів (атрибутів мають кілька значень).</u>***

Багатозначність усувається шляхом введення нової сутності і зв'язку 1: N (рис.8.10).

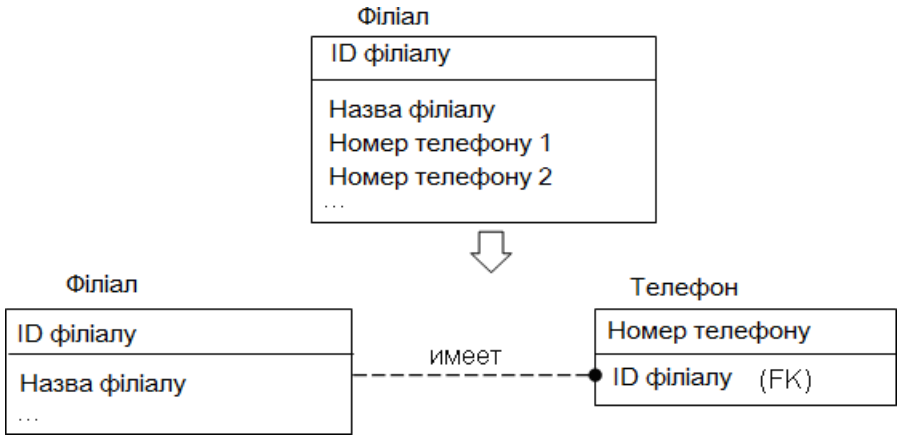



Рисунок 8.10 – Видалення багатозначних атрибутів

Дане перетворення, крім відповідності реляційної моделі даних, також дозволяє зберігати будь-яку кількість телефонів по одному філії.

***<u>Видалення надлишкових зв'язків</u>***

Зв'язок є надлишковою, якщо одна і та ж інформація може бути отримана не тільки через неї, а й за допомогою іншого зв'язку (рис. 8.11).

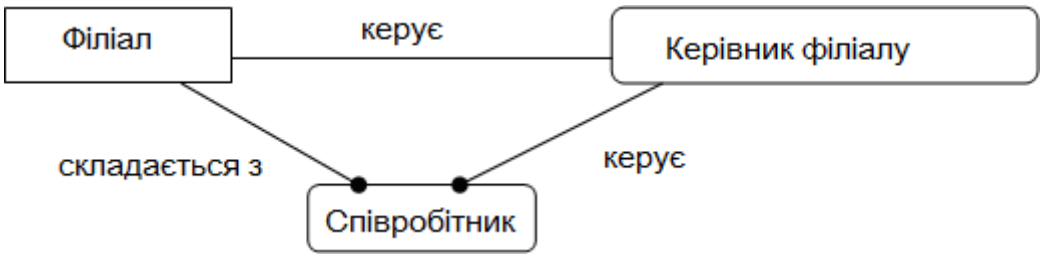



Рисунок 8.11 – Надлишкові зв’язки

У наведеному прикладі один із зв'язків «Керує» можна сміливо видалити (краще між «Керівником філії» і «Співробітником»).

Повторна перевірка зв'язків 1: 1.

У процесі визначення сутностей могли бути створені суті, які насправді є однією. У цьому випадку їх слід об'єднати. Наприклад, у наведеному вище прикладі (рис. 8.12) сутності «Філія» та «Керівник філії» краще об'єднати.

У той же час не завжди можна виконати таке об'єднання (рис. 8.12).

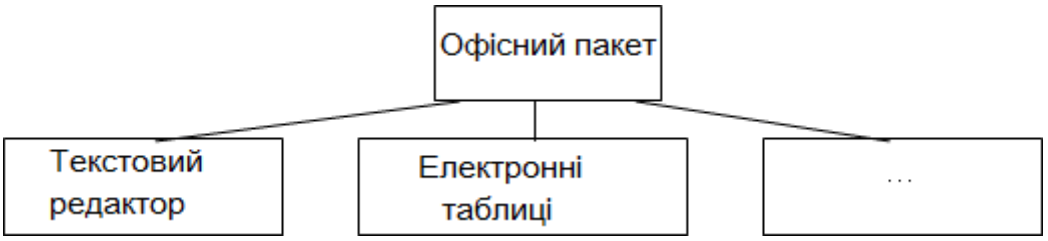



Рисунок 8.12 – Зв'язки 1: 1

Офісний пакет складається з чітко визначеного набору компонентів, причому кожен з них характеризується великою кількістю атрибутів. Крім цього, деякі компоненти можуть бути відсутні в офісному пакеті (наприклад, поштовий клієнт) або вони не входять до його складу (виступають в якості самостійного продукту). В описаних випадках рекомендується не об'єднувати суті.

## 8.3 Фізичне проектування з використанням методології IDEF1X

Мета фізичного проектування - перетворення логічної моделі з урахуванням синтаксису, семантики і можливостей обраної цільової СУБД.

У зв'язку з тим, що методологія фізичного проектування істотно залежить від обраної цільової СУБД, обмежимося лише загальними рекомендаціями.

Аналіз необхідності введення контрольованої надмірності.

При реалізації проекту часто для досягнення більшої ефективності системи потрібно знизити вимоги до рівня нормалізації відносин, т. Е. Внести деяку надмірність даних. Процес внесення таких змін до БД називається денормалізації.

***Перевірка моделі за допомогою правил нормалізації***

Основна ідея нормалізації полягає в тому, щоб кожен факт зберігався в одному місці, т. ї. Щоб не було дублювання даних. Багато з вимог нормалізації, як правило, вже враховуються при виконанні попередніх кроків проектування.

Нижче наводяться короткі відомості з теорії нормалізації.

Проектування реляційної БД являє собою покроковий процес створення набору відносин (таблиць, сутностей), в яких відсутні небажані функціональні залежності.

Функціональна залежність визначається наступним чином. Нехай A і B – довільні набори атрибутів відносини. Тоді B функціонально залежить від A (A → B), в тому і тільки в тому випадку, якщо кожному значенню A відповідає в точності одне значення B. Ліва частина функціональної залежності (A) називається детермінантою, а права (B) - залежною частиною. Зокрема, щодо A може бути первинним ключем, а B - набором неключових атрибутів, так як одного значення первинного ключа в точності відповідає одне значення набору неключових атрибутів.

Якщо в БД відсутні небажані функціональні залежності, то це забезпечує мінімальну надмірність даних, що в свою чергу веде до зменшення обсягу пам'яті, необхідної для зберігання даних. Процес усунення таких залежностей отримав назву нормалізація. Вона виконується у вигляді послідовності тестів для деякого відносини (таблиці, сутності) з метою перевірки його відповідності (або невідповідності) набору обмежень для заданої нормальної форми.

Процес нормалізації вперше був запропонований Е.Ф. Коддом в 1972 р Спочатку було запропоновано три види нормальних форм (1NF, 2NF і 3NF). Потім Р. Бойсом і Е.Ф. Коддом (1974 г.) було сформульовано більш суворе визначення третьої нормальної форми, яке отримало назву нормальна форма Бойса-Кодда (BCNF). Слідом за BCNF з'явилися визначення четвертої (4NF) і

п'ятої (5NF або PJNF) нормальних форм (Р. фагин, 1977 і 1979 г.). На практиці нормальні форми більш високих порядків використовуються вкрай рідко. При проектуванні БД, як правило, обмежуються третій нормальній формою, що дозволяє запобігти можливому виникненню надмірності даних і аномалії оновлень.

1NF. Відношення знаходиться в 1NF, якщо на перетині кожного стовпця і рядка знаходяться тільки елементарні (атомарні, неподільні) значення атрибутів.

Ступінь неподільності (атомарности), т. Е. Рішення про те, чи слід розбивати неатомарний атрибут на атомарні або залишити його псевдоатомарним, визначається проектувальником БД виходячи з конкретних умов. Якщо при обробці таблиць немає необхідності розрізняти атомарні складові псевдоатомарного атрибута, то його можна не ділити (наприклад, атрибути «Прізвище, ім'я, по батькові», «Адреса» і т. Д.).

2NF. Відношення знаходиться в 2NF, якщо воно знаходиться в 1NF, і кожен неключових атрибут характеризується повною функціональною залежністю від первинного ключа.

Повна функціональна залежність визначається наступним чином. В деякому відношенні атрибут B повністю залежить від атрибута A, якщо атрибут B функціонально залежить від повного значення атрибута A і не залежить від будь-якого підмножини повного значення атрибута A.

Наприклад, таблиця «Оцінки іспитів» характеризується наступними набором атрибутів {Номер залікової книжки, Дисципліна, Дата здачі, ПІБ студента, № групи, Оцінка}. Очевидно, що первинним ключем є набір {Номер залікової книжки, Дисципліна, Дата здачі}. Повної функціональної залежністю володіє тільки один неключових атрибут «Оцінка». Атрибути «ПІБ студента» і «№ групи» можуть бути однозначно визначені за частиною первинного ключа - «Номер залікової книжки». Таким чином, потрібно розбиття вихідної таблиці на дві (рис. 8.13).

3NF. Відношення знаходиться в 3NF, якщо воно знаходиться у 2NF і ніякої неключових атрибут функціонально не залежить від іншого неключових атрибута, т. Е. Немає транзитивних залежностей.

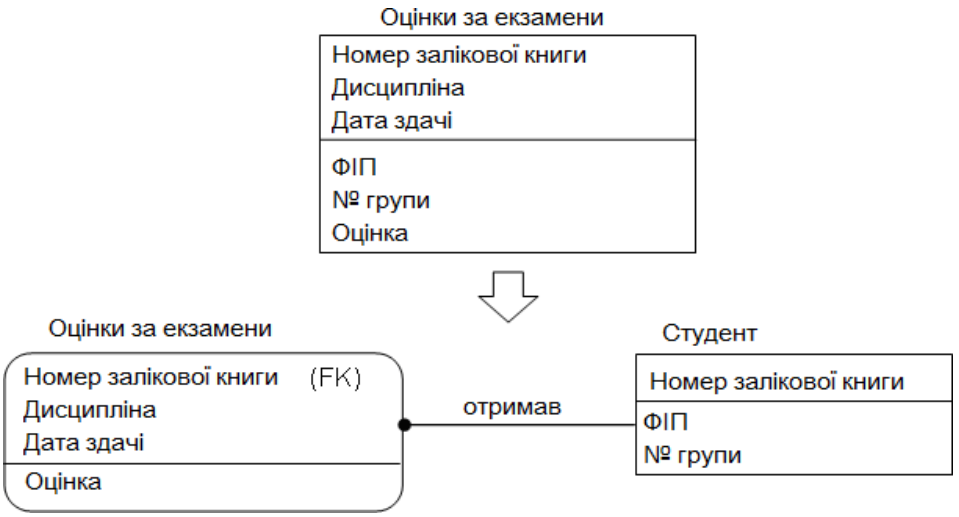



Рисунок 8.13 – Забезпечення повної функціональної залежності

Транзитивная залежність. Якщо для атрибутів A, B і C деякого відносини існують залежності виду A → B і B → C, то атрибут C транзитивній залежить від атрибута A через атрибут B. Наприклад, таблиця «Працівник» характеризується набором атрибутів {Табельний номер, Прізвище, Ім'я, По батькові, Посада, Зарплата, ...}, первинний ключ - {Табельний номер}. У цій таблиці від первинного ключа ( «Табельний номер») залежить неключових атрибут «Посада», а від «Посади» інший неключових атрибут «Зарплата». Для приведення до 3NF необхідно додати нову таблицю (рис. 8.14).

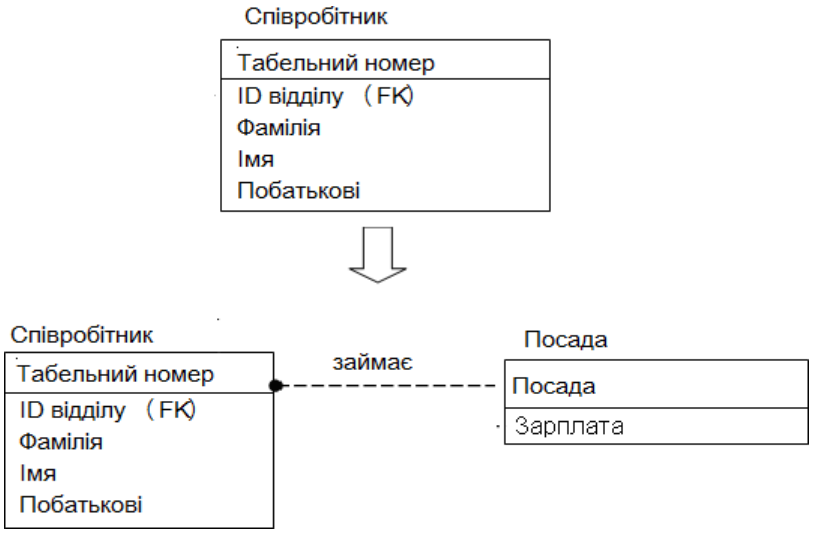



Рисунок 8.14 – Устранение транзитивной зависимости

BCNF. Відношення знаходиться в BCNF, якщо воно знаходиться в 3NF і кожен детермінант відносини є його можливим ключем.

Відступ від BCNF можливо, коли в таблиці є кілька складових потенційних ключів (детермінантів), причому частина атрибутів одного потенційного ключа характеризується повною функціональною залежністю від частини іншого потенційного ключа.

Напри, таблиця «Поставка деталей» характеризується набором атрибутів {ID постачальника, Найменування постачальника, ID деталі, Кількість деталей}, потенційні ключі – {ID постачальника, ID деталі} і {Найменування постачальника, ID деталі}. Мається на увазі, що не може бути двох постачальників з одним найменуванням. Атрибут «ID постачальника» характеризується повною функціональною залежністю від атрибута «Найменування постачальника» і навпаки. Щоб усунути цю небажану залежність, слід один з цих ключів прийняти в якості первинного і додати нову таблицю, куди винести два зазначених атрибута (рис. 8.15).

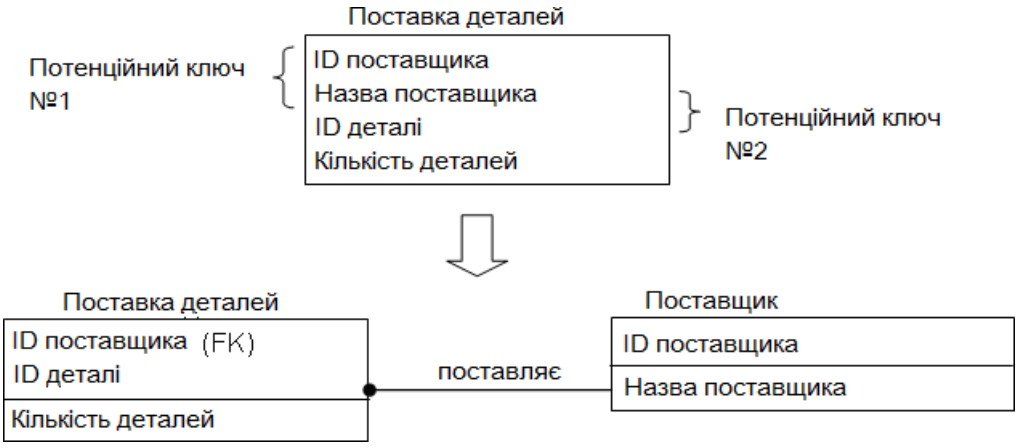



Рисунок 8.15 –Усунення транзитивной залежності

Приведення відношення до BCNF

Відмінність BCNF від 2NF полягає в тому, що виноситься в окрему таблицю атрибут ( «Найменування постачальника») входить до складу потенційного ключа, а не є простим неключових атрибутом ( «ПІБ студента» і «№ групи» на рис. 8.13).

4NF. Відношення знаходиться в 4NF в тому і тільки в тому випадку, якщо в ньому відсутні нетривіальні багатозначні залежності.

Нетривіальна багатозначна залежність. У відношенні з атрибутами A, B і C існує нетривіальна багатозначна залежність, якщо для кожного значення

атрибута А є набір значень атрибута В (А - >> В) і набір значень атрибута С (А - >> З), але між атрибутами В і З немає залежностей.

Наприклад, таблиця «Іспити» характеризується набором атрибутів {Номер групи, Номер залікової книжки, Дисципліна}, первинний ключ - весь набір атрибутів. В даній таблиці є дві багатозначні залежності: номер групи визначає список студентів, які в ній навчаються (Номер групи - >> Номер залікової книжки), і номером групи відповідає список дисциплін навчального плану, за яким потрібно здавати іспити (Номер групи - >> Дисципліна ). У той же час між номером залікової книжки і дисципліною залежність відсутня. Для приведення таблиці до 4NF потрібно розбити її на дві (рис. 8.16).

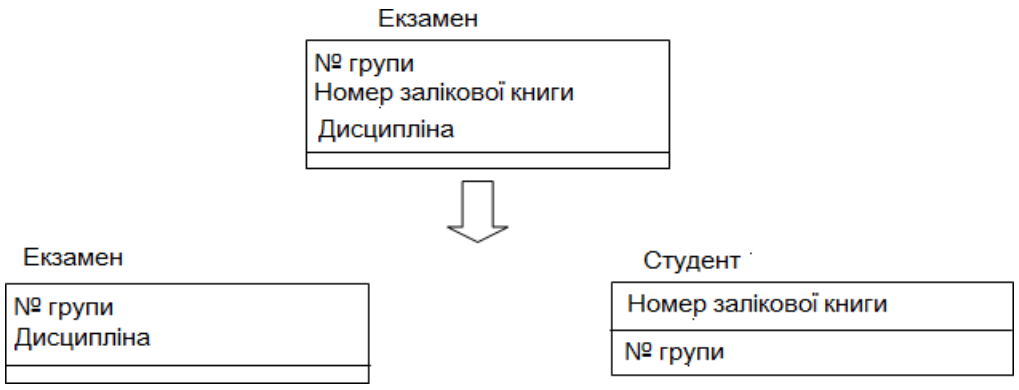



Рисунок 8.16 – Приведення відношення до 4NF

5NF (нормальна форма проекції з'єднання, PJNF). Відношення знаходиться в 5NF, якщо в ньому немає залежностей з'єднання.

Залежність з'єднання. У відношенні з атрибутами А, В і С існує залежність з'єднання, якщо для кожного значення атрибута А є набір значень атрибута В (А - >> В) і набір значень атрибута С (А - >> З), а також існує багатозначна залежність між атрибутами В і С (В - >> З або С - >> В).

Наприклад, таблиця «Дисципліни» характеризується набором атрибутів {Кафедра, Викладач, Дисципліни}, первинний ключ - весь набір атрибутів. В даній таблиці є кілька багатозначних залежностей: на кафедрі працює кілька викладачів (Кафедра - >> Викладач), кафедра веде кілька дисциплін навчального плану (Кафедра - >> Дисципліна), викладач може вести кілька дисциплін (Викладач - >> Дисципліна) і, навпаки , одну дисципліну можуть вести різні викладачі (Дисципліна - >> Викладач). Для усунення залежності з'єднання слід розбити вихідну таблицю на три (рис. 8.17).

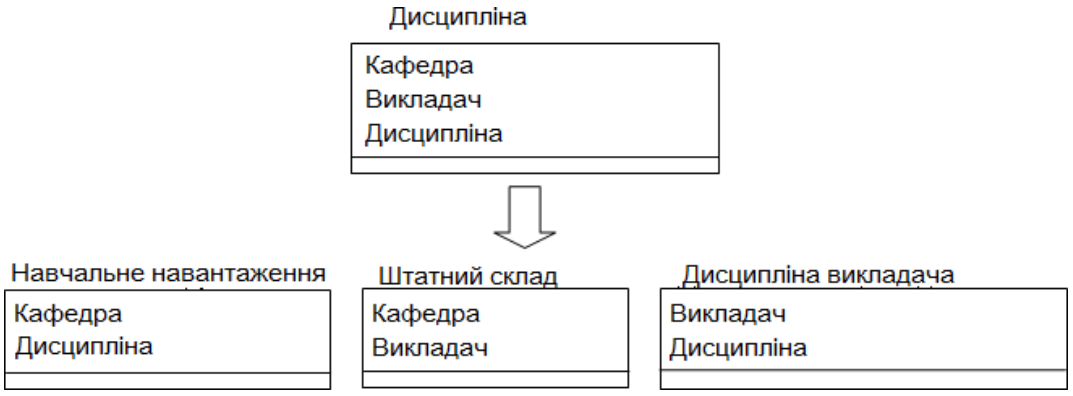



Рисунок 8.17 – Приведення відношення до 5NF

***Перевірка виконання транзакцій.***

Найбільш частими операціями по роботі з БД є введення, видалення і модифікація записів, а також вибірка даних. На основі поточної ERD необхідно перевірити здійсненність і, далі, коректність виконання всіх необхідних операцій по роботі з БД. Приклади транзакцій:

– зміна найменування роздільного пункту;
– видалення завдань розрахунок;
– вибірка даних по роздільним пунктам розрахункового шляху ділянки.

***Визначення вимог підтримки цілісності даних.***

Обмеження цілісності даних є обмеження, які вводяться з метою запобігання приміщення в базу суперечливих даних.

До цих обмежень відносяться.

Обов'язкові дані – атрибути, які завжди повинні містити одне з допустимих значень (NOT NULL). Наприклад, поворот кривої (вліво або вправо) повинен бути обов'язково заданий. Обов'язковими також є всі атрибути, що входять в первинний ключ сутності.

Домени – набори допустимих значень для атрибута. Наприклад, радіус кривої повинен бути позитивним числом не більше 4 цифр або поворот кривої може приймати одне з двох допустимих значень - «Л» (вліво) або «П» (вправо).

Бізнес-правила (бізнес-обмеження) – обмеження, прийняті в даній предметній області. Наприклад, сума довжин перехідних кривих не повинна бути більше довжини всієї кривої, кілометраж початку або кінця кривої повинен бути в межах загального кілометражу шляху.

Посилальна цілісність – набір обмежень, що визначають дії при вставці, оновленні та видаленні записів (примірників сутності).

*Наприклад:*

При наявності обов'язкової зв'язку вставка запису в дочірню сутність вимагає обов'язкового заповнення атрибутів зовнішнього ключа, і введеному значенню повинна відповідати запис батьківської сутності.

Аналогічну вимогу висувається при оновленні зовнішнього ключа в дочірній сутності

Видалення запису з дочірньою суті або вставка запису в батьківську не викликають порушення посилальної цілісності.

Видалення запису в батьківській суті може вимагати видалення всіх пов'язаних записів в дочірньої сутності.

Автоматична підтримка всіх видів обмежень цілісності можлива за рахунок використання операторів SQL.

Посилальна цілісність може бути забезпечена за рахунок використання тригерів. Тригер – це збережена в БД процедура, що викликається автоматично при виконанні видалення (DELETE), вставки (INSERT) або відновлення (UPDATE) записи. Набір команд, що входять в тригер, залежить від прийнятої стратегії (типу тригера) підтримання цілісності. Розглянемо типи тригерів на прикладі видалення запису:

RESTRICT (обмеження дії). Видалення запису з батьківської таблиці забороняється, якщо в дочірній таблиці існує хоча б одна залежна запис.

CASCADE (каскадне видалення або оновлення). При видаленні запису з батьківської таблиці автоматично видаляються всі пов'язані з нею записи дочірньої таблиці. Якщо видаляється запис з дочірньою таблиці виступає в якості батьківської сторони в деякій іншій зв'язку, то операція видалення застосовується до всіх записів дочірньої таблиці зв'язку з цим і т.д .

SET NULL (установка невизначеного значення). При видаленні запису з батьківської таблиці у всіх пов'язаних з нею записах дочірньої таблиці в атрибути зовнішнього ключа записуються невизначені значення (NULL). Такий тип тригера можливий тільки для необов'язкових зв'язків.

SET DEFAULT (установка значення за замовчуванням). При видаленні запису з батьківської таблиці у всіх пов'язаних з нею записах дочірньої таблиці в атрибути зовнішнього ключа записуються заздалегідь певні значення за замовчуванням. Такий тип тригера можливий тільки для необов'язкових зв'язків.

NO CHECK або NONE або IGNORE (без перевірки). При видаленні запису з батьківської таблиці ніяких дій по збереженню посилальної цілісності даних не робиться.

Призначення типу тригера на дії з записами є відповідальною операцією. Вибір невірного типу може привести до порушення посилальної цілісності в БД. Особливої обережності вимагає вибір каскадного видалення, адже при такій відстані по ланцюжку можуть бути видалені сотні і тисячі записів з різних таблиць.

# 9 RAD -МЕТОДОЛОГІЯ ТА CASE-ТЕХНОЛОГІЯ СТВОРЕННЯ Й СУПРОВОДУ ІС

## 9.1 Модель швидкої розробки (RAD)

Метод швидкої розробки додатків (Rapid Application Development, RAD) був використаний у 80-х роках XX століття фірмою IBM. Цей метод був уперше запропонований до уваги розробників ПЗ у книзі Джеймса Мартіна. Завдяки методу RAD користувач задіяний на всіх фазах життєвого циклу розробки проекту не лише під час визначення вимог, але й у процесі проектування, розробки, тестування, а також кінцевого постачання програмного продукту. Участь користувача в процесі розробки стає такою активною завдяки використанню засобів розробки та середовища, що дозволяє дати оцінку продукту на всіх стадіях його розробки. Це забезпечується наявністю засобів розробки графічного, призначеного для користувача інтерфейсу і кодогенераторів.

Характерною рисою RAD є короткий час переходу від визначення вимог до створення цілісної системи. Метод ґрунтується на послідовній ітерації еволюційної системи або прототипів, критичний аналіз яких обговорюється із замовником. У процесі такого аналізу формуються вимоги до продукту. Розробка кожного інтегрованого продукту обмежується чітко окресленим періодом часу, який, як правило, становить 60 днів і називається часовим блоком.

Чинники, що дозволяють створити систему за 60 днів без шкоди для якості, включають:

- використання потужних інструментальних засобів розробки;
- високий рівень повторного використання елементів системи;
- осмислені й виділені ресурси.
- осмислені й виділені ресурси.

Вирішальна рольова участь кінцевого користувача полягає в переміщенні процесу роботи від програмування і тестування до планування та проектування. Користувачам доводиться виконувати великий обсяг роботи на початку життєвого циклу, але за це вони отримують систему, побудовану за коротший проміжок часу.

Фази моделі ЖЦ RAD і участь користувача на всіх фазах процесу проектування наведені на рис. 9.1:

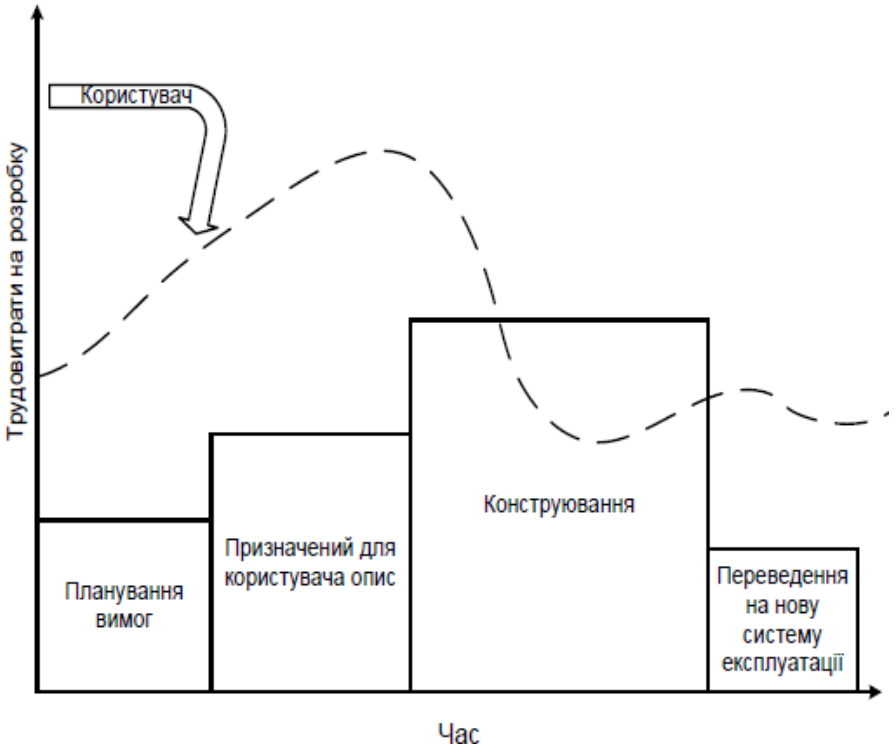



Рисунок 9.1 – Модель швидкої розробки додатків

Планування вимог – збір вимог виконується в процесі використання методу, який називають спільним плануванням вимог (Joint requirements planning, JRP), який є структурним аналізом і обговоренням наявних комерційних завдань; призначений для користувача опис – спільне проектування додатка (Joint application design, JAD) використовується з метою залучення користувачів; на цій фазі проектування системи, яка не є промисловою, команда, яка працює над проектом, часто використовує інструментальні засоби, що забезпечують збирання призначеної для користувача інформації;

Конструювання – ця фаза об’єднує в собі деталізоване проектування, побудову (кодування і тестування), а також постачання програмного продукту замовникові за певний час. Терміни виконання цієї фази значною мірою залежать від використання генераторів коду, екранних форм та інших типів інструментальних засобів;

Переведення на нову систему експлуатації – ця фаза включає проведення користувачами приймальних випробувань, встановлення системи й навчання користувачів.

*Переваги моделі RAD:*

- час циклу розробки для всього проекту можна скоротити завдяки використанню потужних інструментальних засобів;
- потрібна менша кількість фахівців, оскільки розробка системи виконується зусиллями команди, інформованої в предметній області;
- існує можливість здійснити швидкий початковий перегляд продукту;
- завдяки скороченому часу циклу і вдосконаленій технології, а також меншій кількості задіяних у процесі розробників зменшуються витрати;
- залучення замовника на постійній основі зводить до мінімуму ризик того, що він не буде задоволений розробленим продуктом;
- основна увага переноситься з документації на код, при цьому справедливий принцип "отримуєте те, що бачите" (What you see is what you get, WYSIWYG);
- у моделі повторно використовуються компоненти вже існуючих програм.

*Недоліки моделі RAD:*

- якщо користувачі не можуть постійно брати участі в процесі розробки впродовж всього життєвого циклу, це може негативно позначитися на кін-цевому продукті;
- використання моделі може виявитися невдалим у випадку, якщо від-сутні придатні для повторного використання компоненти;
- для реалізації моделі потрібні розробники й замовники, які готові до швидкого виконання дій незважаючи на жорсткі часові обмеження.

Сфера застосування моделі RAD:

- у системах, які підлягають моделюванню (заснованих на використанні компонентних об’єктів), а також в масштабованих системах;
- у системах, вимоги для яких достатньою мірою добре відомі;
- у випадках, коли кінцевий користувач може брати участь у процесі розробки впродовж усього життєвого циклу;
- коли команді, що працює над проектом, знайома предметна область;

- у процесі виконання проектів, розробка яких повинна бути виконана в скорочені терміни, як правило, не більш ніж за два місяці;
- коли можна отримати придатні до повторного використання частини програмних продуктів;
- у системах, які призначені для концептуальної перевірки, є критичними або мають невеликий розмір;
- у системах, у яких не потрібні досягнення високої продуктивності, за невисокого ступеня технічних ризиків;
- в інформаційних системах.

## 9.2 CASE-технологія створення й супроводу ІС.

### 9.2.1 Технологія RUD. Методологія RUP

Методологія Rational Unified Process (RUP) є розробкою компанії Rational. Ця компанія довгий час успішно займається створенням CASE-засобів, використовуваних на різних етапах життєвого циклу продукту від аналізу до тестування і документування. Технологія RUP універсальна, масштабована і налаштовується на застосування в конкретних умовах. Модель життєвого циклу RUP є досить складною ітеративно-інкремент-ною моделлю, що детально відпрацьована, з елементами каскадної моделі.

Основними характеристиками RUP є наступні:

- використання кращих практичних методів, що успішно зарекомендували себе в багатьох проектах розробки ПЗ у всьому світі;
- чітко визначений технологічний процес, що описує структуру життєвого циклу проекту, ролі й відповідальності окремих виконавців, виконувані ними завдання і використовувані в процесі розробки моделі, звіти і т. д.; готовий продукт, що подається у вигляді веб-сайта, який містить усі необхідні моделі й документи із описом процесу.

Підхід до розробки. RUP пропонує розробникам не жорсткі правила, що регламентують виконання всіх дій у ході розробки, а набір досить гнучких методів і підходів, із яких розробник може вибирати те, що понад усе відповідає його завданням і особливостям проекту.

Основними принципами RUP є:

– ітераційна розробка;
– управління процесом на основі прецедентів використання;
– орієнтація на архітектуру.
– Слід розглянути ці принципи детальніше.

*Ітерація* – це закінчений цикл розробки, що приводить до випуску кінцевого продукту або певної його скороченої версії, яка розширюється від ітерації до ітерації, щоб зрештою стати закінченою системою (рис. 24). У разі ітераційного підходу кожна з фаз процесу розробки ПЗ складається з ітерацій, метою яких є послідовне осмислення проблем, що постають, нарощування ефективності рішень і зниження ризиків потенційних помилок у проекті. На виході кожної ітерації створюється закінчена версія дієвого програмного продукту. За такого підходу можна гнучко враховувати нові вимоги або проводити тактичні зміни в ділових цілях, виявляти проблеми і вирішувати їх на більш ранніх етапах розробки, що пов’язане з меншими витратами. Управління на основі прецедентів використання означає, що як метод опису функціональних вимог до системи, а також як природна одиниця для подальшого планування й оцінки виконання робіт застосовуються сценарії використання. Сценарії використання дозволяють легко виявляти реальні потреби майбутніх користувачів системи і відстежувати повноту опису цих вимог.

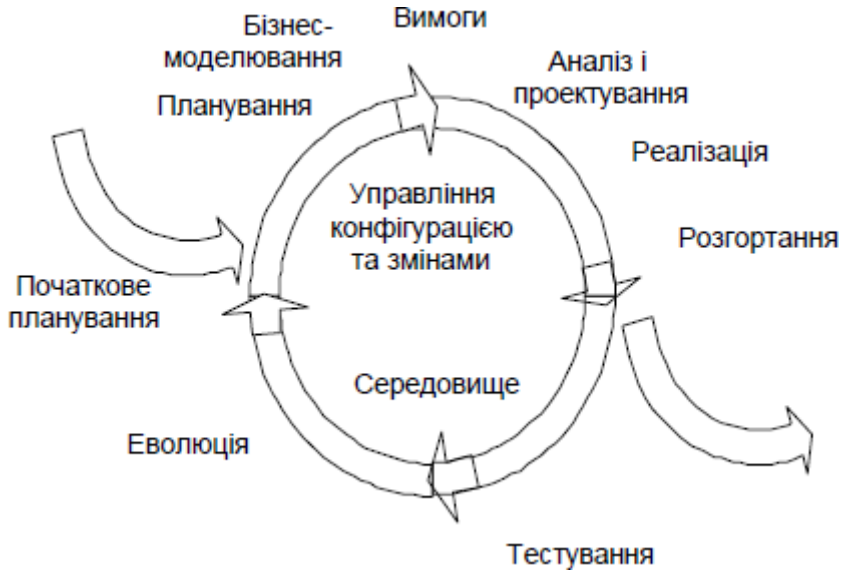



Рисунок 9.2 – Ітерація ЖЦ RUP

Вони гарантують виконання вимог замовника до ПЗ. Крім того, використання завершених сценаріїв як одиниця вимірювання прогресу допомагає уникнути неадекватної оцінки ступеня виконання проекту виконавцем.

Орієнтація на архітектуру передбачає розробку, реалізацію і тестування архітектури на більш ранніх стадіях виконання проекту. Такий підхід дозволяє усувати найнебезпечніші ризики, пов'язані з архітектурою, на ранніх стадіях розробки. Завдяки йому вдається уникнути суттєвих переробок в останню мить, якщо раптом з'ясується, що вибране рішення не забезпечує, наприклад, виконання вимог до безпеки системи.

RUP створювався за методикою, використовуваною в процесі проектування ПЗ. Зокрема, моделювання проводилося за допомогою Software Process Engineering Metamodel (SPEM) – стандарту моделювання процесів, заснованого на Unified Modeling Language (UML). У моделі ЖЦ RUP є два виміри (рис.9.3): динамічне і статичне.

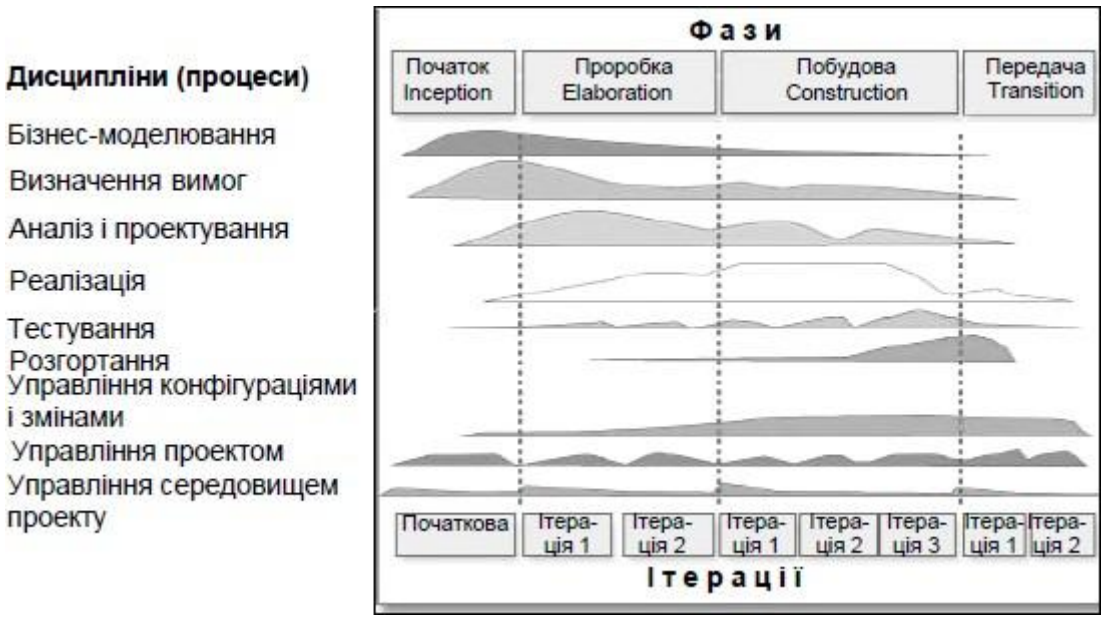



Рисунок 9.3 – Виміри ЖЦ RUP

Динамічна структура. Горизонтальний вимір є динамічною структурою або часовим виміром процесу. Він показує, як процес, виражений у формі циклів, фаз, ітерацій і віх, розгортається в ході життєвого циклу проекту.

*Динамічна структура RUP складається з чотирьох фаз:*

1. *Фаза початку проекту (Inception).* Визначаються основні цілі проекту, бюджет проекту, основні засоби його виконання – технології,

інструменти, ключовий персонал, складаються попередні плани проекту. Основна мета цієї фази – досягти компромісу між усіма зацікавленими особами щодо завдань проекту;

2. *Фаза проробки (Elaboration)*. Основна мета цієї фази – на базі основних, найбільш суттєвих вимог розробити стабільну базову архітектуру продукту, яка дозволяє вирішувати поставлені перед системою завдання і надалі використовується як основа розробки системи;

3. *Фаза побудови (Construction)*. Основна мета цієї фази – детальне прояснення вимог і розробка системи, що задовольняє їх, на основі спроектованої раніше архітектури;

4. *Фаза передачі (Transition)*. Мета фази – зробити систему цілком доступною кінцевим користувачам. Тут відбувається остаточне розгортання системи в її робочому середовищі, підгонка дрібних деталей під потреби користувачів.

Фази можуть підрозділятися на ітерації. Перехід з фази на фазу можливий лише після виконання завдань фази і є контрольною точкою – віхою (milestone) – процесу.

Статична структура. Вертикальний вимір є статичною структурою процесу. Він описує, як елементи процесу (завдання, дисципліни, артефакти і ролі) логічно групуються в дисципліни або робочі процеси (workflows). Для опису осмисленої послідовності виконання робіт і завдань використовуються робочі процеси. Вони описують, хто, що, як і коли виконує в процесі. У RUP входять 6 основних дисциплін і 3 допоміжних.

До *основних дисциплін* належать:

*Моделювання предметної обл*асті (бізнес-моделювання, Business Modeling). Цілі цієї діяльності – зрозуміти бізнес-контекст, в якому повинна буде працювати система (і переконатися, що всі зацікавлені особи розуміють його однаково), зрозуміти можливі проблеми, оцінити їх можливі розв’язання і наслідки для бізнесу організації, в якій працюватиме система;

*Визначення вимог (Requirements*). Цілі – зрозуміти, що повинна робити система, визначити межі системи і основу для планування проекту і оцінок ресурсовитрат у ньому; аналіз і проектування (Analysis and Design). Розробка архітектури системи на основі ключових вимог, створення проектної моделі, наведеної у вигляді діаграм UML, що описують продукт з різних точок зору;

*Реалізація (Implementation*). Розробка початкового коду, компонент системи, тестування та інтеграція компонент;

Тестування (Test). Загальна оцінка дефектів продукту, його якості в цілому, оцінка ступеня відповідності початковим вимогам;

*Розгортання (Deployment)*. Цілі – розгорнути систему в її робочому оточенні і оцінити її працездатність.

До *допоміжних дисциплін* відносяться:

*Управління конфігураціями і змінами (Configuration and Change Management)*. Визначення елементів, що підлягають зберіганню і правил побудови з них узгоджених конфігурацій, підтримка цілісності поточного стану системи, перевірка узгодженості змін, що вносяться;

*Управління проектом (Project Management)*. Включає планування, управління персоналом, забезпечення зв'язків з іншими зацікавленими особами, управління ризиками, відстеження поточного стану проекту;

*Середовище проекту (Environment)*. Настроювання процесу під конкретний проект, вибір і зміна технологій і інструментів, використовуваних у проекті.

На відміну від каскадного підходу, в RUP усі дисципліни виконуються практично на всіх фазах життєвого циклу ПЗ. Однак, залежно від фази, змінюються поточні цілі проекту і співвідношення між обсягами робіт, відповідних різним дисциплінам.

Для забезпечення інструментальної підтримки всіх процесів життєвого циклу розробки і супроводу ПЗ RUP пропонує спеціалізовані інструментальні засоби IBM Rational (рис.9.4).

RequisitePro – дозволяє створювати і оновлювати керовані ним вимоги безпосередньо в програмі або за допомогою редактора Microsoft Word. Для зміни вимог досить відредагувати вміст документа, при цьому сам засіб допоможе відновити, проаналізувати й об'єднати необхідні вимоги. Software Modeler – засіб, що замінив компоненту Rational Rose® XDE, яка раніше використовувалася. Побудоване на платформі Eclipse, вона успішно використовувалося в сімействі продуктів IBM WebSphere® як основа для інтегрованого середовища розробки (IDE) WebSphere Stu-dio Application Developer. Rational Software Modeler надає користувачу інтерфейс для моделювання додатка до початку написання програмного коду.

Application Developer – надає середовище IDE для перетворення моделей в програмний код і фінальні додатки. Rational Application Develo-per також забезпечує користувачів інтегрованим середовищем для тестування готового додатка.

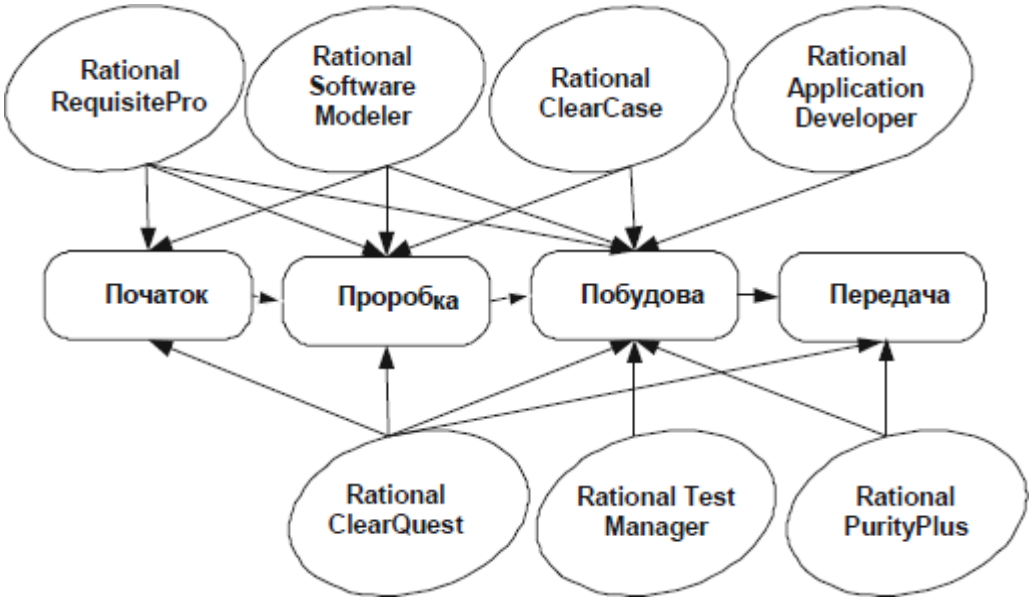



Рисунок 9.4 – Підтримка процесів ЖЦ інструментальними засобами RUP

ClearCase – забезпечує управління конфігурацією бази коду і моделей. WebSphere Studio Application Developer і Rational Rose XDE, взаємодіючи з Rational ClearCase, здатні забезпечити всю команду розробників механізмами управління всіма аспектами процесу розробки.

ClearQuest – полегшує відстеження дефектів та управління операціями. Відстеження дефектів дозволяє виявити помилки і знайти їх первинну причину. Механізм управління операціями дає можливість відстежувати операції над різними компонентами проекту, включаючи програм-ний код, моделі та іншу інформацію. Інтеграція ClearQuest з іншими додатками Rational дозволяє відстежувати дефекти й операції в широкому діапазоні компонентів і вимог, перевіряючи інформацію в будь- якій точці проекту.

PurifyPlus – забезпечує поточний аналіз додатка, що дозволяє простежити, як він використовує ресурси пам’яті. "Витік" пам’яті та інші проблеми такого роду є одним з найпоширеніших дефектів в додатках, за винятком друкарських помилок і помилок в програмному коді, які повинні усуватися за допомогою тестування і спеціальних методик, використовуваних під час застосування інструментальних засобів Ra-tional. Інтеграція цього інструменту з середовищем WebSphere Studio Application Developer забезпечує користувачів інформацією про елементи додатка, що викликають ті або інші проблеми. TestManager – надає єдиний інтерфейс для тестування набору параметрів додатка, дозволяючи проводити це тестування

в автоматичному режимі, що особливо важливо, якщо розробка виконується на різних платформах.

Інструментарій RUP достатньо повний. Це набір рекомендацій і прикладів для всіх стадій і фаз розробки програм. Хоча в основу цих рекомендацій покладений багаторічний досвід розробки програмних систем, не для кожного проекту RUP підходить на сто відсотків. Будь-який програмний проект по-своєму унікальний. Не можна бездумно копіювати чужий процес, створюючи артефакти, що мають незначну цінність. У багатьох невеликих організаціях з розробки програмного забезпечення, особливо в тих, які не мають власної могутньої системи розробки, RUP можна використовувати "як є", але він може бути уточнений, розширений і специфічно настроєний для максимального наближення до потреб організації-розробника.

# 10 РЕІНЖИНІРИНГ ІС

В даний час існує багато різних визначень реінжинірингу (англ. Reengineering) інформаційних систем (ІС). У тому числі - багато схожих і суміжних понять. Загальноприйнятого визначення поки не існує.

Зустрічаються два написання самого терміна: реінжиніринг і реінжиніринг (перше зустрічається частіше).

Під реинжинирингом інформаційної системи (РІС) розуміється аналіз і перепроектування інформаційної системи з метою реалізації її в новій якості. Таким чином, метою РІС встановлюється істотне поліпшення якості інформаційної системи «в рази» (значно більше 100%).

Реінжиніринг інформаційних систем - якісне поліпшення існуючих інформаційних мереж шляхом їх перепроектування, перегляду процесів їх функціонування з єдиною метою: різке збільшення ключових показників продуктивності. Іншими словами, реінжинірингового заходи дозволяють якісно змінити роботу, швидкість її виконання і якість одержуваного результату.

Варто зауважити, що реінжиніринг інформаційних систем не є концепцією сталого поліпшення. Першочергове його завдання - створити відносно стійку структуру чітко визначеної якості з уже наявною, і, звичайно, з урахуванням можливості майбутніх перетворень. При цьому існуюча інформаційна система ні в якому разі не ламається.

Виділяють такі терміни, подібні або суміжні з терміном реінжинірингу інформаційних систем.

Модернізація ІС – щодо невелике поліпшення інформаційної системи (зазвичай не більше 100%), виправлення критичних помилок, при відсутності кардинальних змін системі.

Рефакторинг ІС (англ. refactoring) – повне або часткове перетворення внутрішньої структури програмного забезпечення інформаційної системи (тільки програмного) при збереженні зовнішнього поведінки; переклад на більш сучасну мову програмування.

Редизайн ІС (англ. redesign) – переробка тільки користувальницьких інтерфейсів інформаційної системи без істотного втручання в її пристрій, зазвичай вживають щодо веб-сайтів.

Реверс-інжиніринг ІС (англ. reverse-engineering) – дослідження, відновлення (побудова) структурних моделей інформаційної системи

(наприклад, побудова інформаційної моделі ІС на основі її існуючої бази даних).

Реінжиніринг бізнес-процесів (РБП) – фундаментальне переосмислення і радикальне перепроектування бізнес-процесів для досягнення істотних поліпшень в ключових для сучасного бізнесу показниках результативності.

**Фази реінжинірингу бізнес-процесів**

Реінжиніринг бізнес-процесів полягає в узгодженій оптимізації головних складових системи управління: організаційної структури, функцій, процесів і ресурсів. Щоб оптимізувати систему управління, потрібно її багато разів переконфігурувати, порівняти конструкції, які утворилися, між собою і вибрати найкращу. Виконати це без інструментальної підтримки з боку комп'ютерних технологій неможливо.

У моделях життєвого циклу ІС, передбачалося, що вимоги до організації ІС, як правило, чітко визначаються до моменту початку проектування ІС, тобто організаційна система повинна бути спроектована, а ще краще – повинна функціонувати до моменту розробки ІС. Реінжиніринг бізнес-процесів передбачає, що реорганізація організації не може бути успішно проведена без створення адекватної ІС. Тому ІС не просто автоматизує наявні ділові процеси "як є", а забезпечує підтримку змін організації на принципах "як повинно бути". Унаслідок цього реорганізація організації і проектування ІС відбуваються практично паралельно.

Паралельність життєвого циклу розробки ІС означає також те, що більшість основних бізнес-процесів, що реорганізуються, проектуються одночасно, що викликає необхідність паралельної координації проведених робіт у частині розробки загальних забезпечувальних підсистем. Таким чином, загальносистемні рішення формуються в процесі реалізації вимог за окремими бізнес-процесами. Послідовність етапів проведення бізнес-реінжинірингу наведена на рис.10.1.

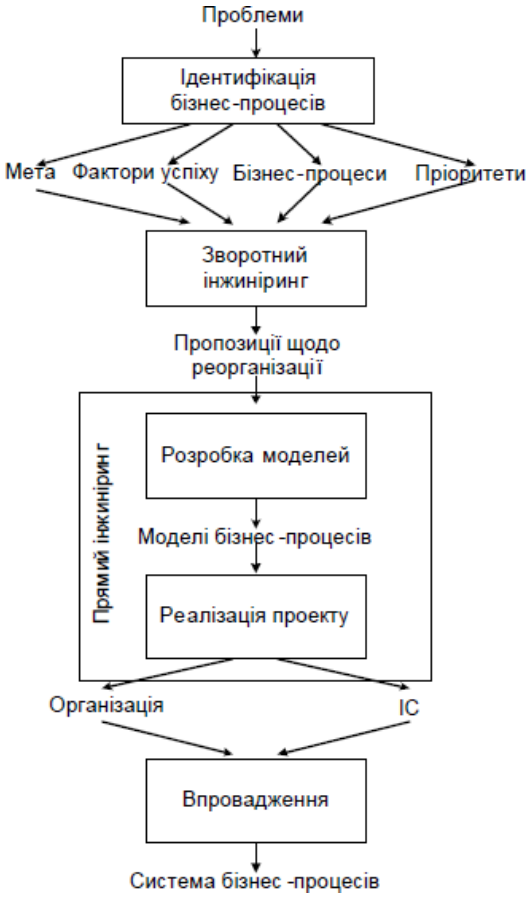



Рисунок 10.1 – Фази проведення бізнес-реінжинірингу

Слід розглянути фази проведення реінжинірингу бізнес-процесів більш докладно.

1. Ідентифікація бізнес-процесів.

На фазі ідентифікації виділяються ключові бізнес-процеси, реорганізація яких забезпечує кардинальне підвищення ефективності функціонування організації.

Роботи з ідентифікації бізнес-процесів ініціюють менеджери верхньої ланки управління підприємством – особи, що ухвалюють рішення. На цьому етапі здійснюється постановка завдань реінжинірингу бізнес-процесів, яка визначається ключовими проблемами підприємства, що викликають необхідність РБП. Наприклад, у зв’язку зі зниженням обсягу продажів або збільшенням кількості рекламацій на продукцію, або високою плинністю кадрів, або низькою завантаженістю устаткування, або міжопераційними простоями, або наднормативними запасами і т. д.

У процесі постановки реінжинірингу бізнес-процесів вирішуються наступні завдання:

- формулювання (уточнення) місії підприємства;
- визначення критичних факторів успіху (7 – 8 факторів) чи критеріїв ефективності організації бізнес-процесів: тривалості, витрат, якості, сервісного обслуговування і т. д.;
- виявлення основних видів бізнес-процесів, які вже існують, так і перспективних (10 – 15 процесів);
- неформальний опис відмінних рис виявлених бізнес-процесів;
- оцінка важливості бізнес-процесів за ступенем впливу на реалізацію критичних факторів успіху;
- визначення обмежень, пов’язаних із рівнем кваліфікації персоналу фірми, технічної оснащеності виробництва, наявністю інформаційних технологій, фінансових ресурсів;
- опис можливих сценаріїв розвитку підприємства: поява нових технологій, ресурсів, зміна поведінки клієнтів, партнерів, конкурентів;
- визначення зовнішніх ризиків забезпечення фінансовими ресурсами, надійності партнерів, кон’юнктури ринку;
- ранжування бізнес-процесів для проведення реінжинірингу бізнес-процесів відповідно до отриманої оцінки важливості, обмежень, ризиків і перспектив.

Для підготовки ухвалення рішень щодо проведення реінжинірингу процесів чи уточнення сформульованих реінжинірингом бізнес-процесів завдань може бути проведена робота з узагальненого аналізу фінансово-господарської діяльності підприємства з використанням сучасних інформаційних технологій.

Так, формулювання місії передбачає вирішення завдань якісного аналізу ділових процесів, пов’язаних із визначенням стратегії поводження підприємства на ринку в частині розширення меж ринку чи глибокого проникнення на ринок, диверсифікованості діяльності чи підвищення якості товарів і послуг, глобалізації чи локалізації діяльності і т. д. Як основні методи формування стратегії підприємства звичайно використовуються методи аналізу ієрархій Сааті, нечіткої логіки Заде, а з інструментальних засобів – статичні експертні системи з можливістю обробки якісних (нечітких) оцінок, такі як Expert Choice, Guru, Level5, Nex-pert Objects та ін.

Оцінка обмежень і ризиків, пов’язаних із проведенням РБП, виконується в ході вирішення завдань бізнес-планування. Зокрема, для виділених бізнес-процесів здійснюється оцінка можливостей підприємства в

плані ефективності розподілу капіталовкладень для різних проектів. Для вирішення цього завдання звичайно використовуються економіко-математичні моделі й методи оптимізації. До найбільш відомих програмних засобів бізнес-планування відносяться ППП Project Expert, "Альт-Инвест", ТЭО-ИНВЕСТ, COMFAR PROSPIN та ін.

Вибір бізнес-процесів виконується на основі аналізу сегментів ринку, за допомогою якого конкретизуються стратегічні цілі підприємства для визначення регіонів, споживачів, каналів розподілу продукції та послуг, а отже, й особливості організації бізнес-процесів. Основними методами досліджень на цьому етапі виступають методи статистичного аналізу і прогнозування, нейронних мереж, інтелектуального аналізу даних сучасних інформаційних сховищ. Найбільш могутніми інструментальними засобами аналізу і прогнозування для виявлення основних сегментів ринку є ППП SAS, Oracle Express, Business Objects, SPSS, NeurOn-Line, Brain Maker, Poly-Analyst та ін. Після ухвалення рішення про проведення РБП виконується планування робіт на всіх стадіях життєвого циклу проекту, виділяються матеріальні, людські, фінансові й часові ресурси, формується організаційна структура проекту і команди РБП, проводиться роз’яснювальна робота з персоналом підприємства з питань майбутньої реорганізації підприємства.

Планування РБП здійснюється за допомогою методів і засобів управління проектами. З використанням програмних засобів управління проектами на зразок TimeLine, Microsoft Project, Primavera будуються мережні календарні графіки виконання робіт. Більш складні програмні продукти, такі, наприклад, як Process Engineer (LBMS), дозволяють визначати ресурси на виконання проектних робіт, вести бібліотеки проектів, здійснювати колективну розробку і моніторинг стану проекту на наступних стадіях РБП. Засоби планування та управління роботами підтримуються також у комплексних технологіях проектування і впровадження корпоративних ІС, наприклад CASE Method (Oracle), Ac-celerated SAP (SAP), Orgware (BAAN).

2. Зворотний інжиніринг.

Зворотний інжиніринг передбачає дослідження бізнес-процесів, які функціонують на підприємстві. Мета етапу полягає в проведенні діагностики "вузьких місць" в організації наявних бізнес-процесів і формулюванні напрямків їх реорганізації. Завдання зворотного інжинірингу спрощується, якщо на підприємстві є документація про процеси, які функціонують після попередньої реорганізації.

На етапі зворотного інжинірингу постановка завдань реорганізації бізнес-процесів уточнюється, сформульовані на фазі ідентифікації бізнес-процесів у загальному вигляді цілі РБП можуть бути скоректовані за результатами дослідження наявної системи організації бізнес-процесів.

Зворотний інжиніринг не повинен викликати одержання надмірно детальної картини існуючих бізнес-процесів, тому що в цьому випадку велика ймовірність не вирішити поставлене завдання. На фазі зворотного інжинірингу будуються, як правило, тільки принципові моделі бізнес-процесів, що дозволяють зрозуміти сутність функціонування підприємства в цілому і виявити напрямки реорганізації бізнес-процесів. За необхідності уточнення деталей яких-небудь ділових процесів до зворотного інжинірингу можна ітеративно повернутися на фазі прямого інжинірингу. На фазі зворотного інжинірингу для розуміння сутності бізнес-про-цесів, які функціонують, широко використовуються методи і засоби структурного аналізу ділових та інформаційних процесів (функціонального чи об’єктного моделювання), які реалізовані в таких відомих ППП, як, наприклад, Design/IDEF, BPWin, ERWin, OOWin, ARIS Toolset, Rational Rose та ін.

Для оцінки ефективності наявних бізнес-процесів використовуються, насамперед, методи і засоби функціонально-вартісного аналізу (ABC – Activity Based Costing), підтримувані, наприклад, у ППП Design/IDEF, Easy ABC+, ARIS ABC та ін. Так, функціонально-вартісний аналіз дозволяє виявити:

- найбільш трудомісткі й витратні функції;
- функції, що не роблять внеску в утворення прибутку;
- функції з низьким коефіцієнтом використання ресурсів.

Для динамічної оцінки наявних бізнес-процесів за наявності розвинутої функціональної інформаційної системи й інформаційного сховища можуть використовуватися методи аналізу реальної статистики. У протилежному випадку застосовуються методи і засоби імітаційного моделювання, наприклад, ППП ReThink, РДО, Pilgrim, Ithink, Workflow Analyser, Service Model та ін.

3. Прямий інжиніринг.

Фаза прямого інжинірингу включає два етапи – розробки моделей нової організації бізнес-процесів і реалізації проекту реінжинірингу бізнес-про-цесів.

Розробка моделей нової організації бізнес-процесів передбачає, що моделі можуть будуватися в декількох варіантах, з яких вибираються

найбільш ефективні варіанти з погляду реалізації критичних факторів успіху. У кінцевому рахунку залишають два варіанти моделей нової організації бізнес-процесів:

- ідеальну модель, що може бути досягнута в перспективі і до якої варто прагнути;
- реальну модель, що може бути досягнута в доступному для огляду майбутньому з урахуванням наявних ресурсів і обмежень.

Причому реальна модель бізнес-процесів повинна бути такою, щоб можна було в перспективі перейти до ідеальної моделі, тобто необхідно, щоб обидві моделі були концептуально близькі одна одній.

Моделювання предметної області бізнес-процесів виконується в два підетапи:

- визначається структура нової організації бізнес-процесів (склад об’єктів, функцій і подій);
- об’єкти і функції бізнес-процесів розподіляються за структурними підрозділами підприємства і визначаються вимоги до інформаційної системи.

Для побудови Для побудови архітектури інформаційної системи на цьому етапі формуються вимоги до того:

- які функції будуть автоматизуватися;
- які для цього будуть потрібні куповані й оригінальні типи програм;
- які необхідні інформаційні об’єкти;
- як вони будуть організовані й розподілені.

На фазі прямого інжинірингу доцільно використовувати програмні засоби моделювання, що дозволяють відображати результати моделювання предметної області в репозиторії, використовуваному на всіх наступних етапах розробки проекту. Під репозиторієм мається на увазі сховище метаінформації про проект. До таких програмних засобів належать або CASE-засоби, наприклад Oracle Designer 2000, SilverRun, Natural Engineering Workbench, Rational Rose та ін., або модельно-орієнтовані засоби компонентного проектування, наприклад, Business Engineer (SAP), Enterprise Modeller (BAAN) ), або спеціалізовані, які дозволяють експортувати моделі в CASE-засоби чи модельно-орієнтовані засоби компонентного проектування, наприклад ARIS Toolset.

Етап реалізації проекту реінжинірингу бізнес-процесів закінчується складанням відповідно до прийнятих стандартів проектної документації або за допомогою CASE-засобів, або спеціалізованих програмних засобів. У

будь-якому випадку повинне забезпечуватися якісне оформлення проектної документації з включенням необхідних графічних ілюстрацій. При цьому може виконуватися автоматизований контроль на відповідність розділів проектної документації, забезпечуватися її своєчасне відновлення.

4. Впровадження проекту реінжинірингу бізнес-процесів. Впровадження проекту РБП, як правило, здійснюється поетапно відповідно до пріоритетів, установлених на етапі ідентифікації бізнес-процесів. Велике значення на етапі впровадження приділяється комплексному тестуванню компонентів проекту, для чого використовуються спеціальні програмні засоби.

Впровадження проекту РБП передбачає його здачу приймальній комісії, у котру входять представники осіб, що ухвалюють рішення, і майбутні менеджери процесів. Перед звітом команди РБП на комісії можлива організація незалежної експертизи проекту з боку спеціально підібраної інспекційної групи.

Після впровадження спроектованих бізнес-процесів у реальну практику дуже важливо організувати аналіз досягнення заданих на початку реін-жинірингу критеріїв (метрик) ефективності функціонування підприємства (benchmarking), на основі яких можна вчасно ухвалювати рішення про необхідність адаптації бізнес-процесів до зовнішнього середовища, що змінюється.

# ПЕРЕЛІК ПОСИЛАНЬ


1. Пасічник В.В., Литвин В.В., Шаховська Н.Б. Проектування інформаційних систем. Навчальний посібник (затв. МОН України) Львів: 2013.– 380 с.
2. Грекул В.И. и др. Проектирование информационных систем. Курс лекций. Интернет-Университет Информационных Технологий, М.2005.
3. Маклаков С.С. BPwin и Erwin. CASE- средства разработки информационных систем. М.:ДИАЛОГ-МИФИ,, 2000.– 256с.
4. Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. М: «Финансы и статистика», 1998.
5. Томашевський О.М. Інформаційні технології та моделювання бізнес-процесів:Навчальний посібник./ О.М.Томашевський, Цегелик М.Б., Вітер Г.Г., В.І.Дубук. К.:Центр учбової літератури, 2005.– 296 с.
6. www.library-odeku.16mb.com
7. Вендров А.М. Проектирование программного обеспечения экономических информационных систем. М: «Финансы и статистика», 2006.
8. Ипатова Э.Р. Методологии и технологии системного проектирования информационных систем, учебник М.: Флинта: МПСИ, 2008
9. Гвоздева В. А. Основы построения автоматизированных информационных систем учебник М. : БИНОМ. Лаборатория знаний, 2008
10. Исаев Г.Н. Проектирование информационных системОмега-Л, 2013.–424
11. Мацяшек Л. А. Анализ требований и проектирование систем. Разработка информационных систем с использованием UML / Л. А. Маця-шек ; пер. с англ. – М. : Издательский дом "Вильямс", 2002. – 432 с.
12. Ларман К. Применение UML и шаблонов проектирования: Введение в объектно-ориентированный анализ и проектирование: Учебное пособие: Пер. с англ. - М.: Вильямс, 2001. – 496 с.
13. Моделирование бизнеса. Методология ARIS / М. Каменова, А. Громов, М. Ферапонтов, А. Шматолюк. – М. : Весть-Мета Технология, 2001. – 328 с.
14. Объектно-ориентированный анализ и проектирование [Элек-тронный ресурс]. – Режим доступа : http://ooad.asf.ru/.
15. DimDim Software: Проектирование и разработка автоматизиро-ванных, информационных и аналитических систем. проектирование [Электронный ресурс]. – Режим доступа : http://www.info-system.ru/.
16. UML2.ru. Сообщество системных аналитиков [Электронный ресурс]. – Режим доступа : http://www.uml2.ru/.

Навчальне електронне видання

РЕМЕНЯК ЛЕСЯ ВАСИЛІВНА

# ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Конспект лекцій

**Видавець і виготовлювач**
Одеський державний екологічний університет
вул. Львівська, 15, м. Одеса, 65016
тел./факс: (0482) 32-67-35
E-mail: info@odeku.edu.ua
Свідоцтво суб'єкта видавничої справи
ДК № 5242 від 08.11.2016