МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет будівництва і архітектури

І. А. Саченко

ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Конспект лекцій
для здобувачів першого (бакалаврського) рівня вищої освіти
за спеціальностями 122 «Комп’ютерні науки» та
126 «Інформаційні системи і технології»

Київ 2024

УДК 004.2
С 12

Рецензент О.О. Терентьєв, д-р техн. наук, професор

Затверджено на засіданні навчально-методичній раді КНУБА протокол № 11 від 31 травня 2024 року.

Саченко І. А.

С 12 Проєктування інформаційних систем: конспект лекцій/ Саченко І.А. – Київ: КНУБА, 2024. – 88 с.

Містить введення в предмет, огляд змісту курсу, постановку проблеми, пояснення ключових термінів, понять та методів, що використовуються у проєктуванні інформаційних систем.

Призначено для здобувачів першого (бакалаврського) рівня вищої освіти спеціальностей 122 «Комп’ютерні науки» та 126 «Інформаційні системи і технології»

УДК 004.2

Скорочення та умовні позначення

БД – база даних
ЖЦ – життєвий цикл
ЗП – завдання на проєктування
ІТ – інформаційні технології
ІС – інформаційна система
КІС – корпоративна інформаційна система
НВ – нефункціональні вимоги
ООП – об’єктно-орієнтоване програмування
ООП – об’єктно-орієнтований підхід
ПЗ – програмне забезпечення
ПП – програмний продукт
ПС – програмна система
ТЗ – технічне завдання
ТП – Технологія проєктування
САПР – система автоматизованого проєктування
СУБД – система управління базою даних
ФВ – функціональні вимоги
ФС – формальна специфікація
ISO – International Organization of Standardization
RAD – Rapid Application Development
RUP – Rational Unified Process

Зміст

ВСТУП…..……...7
Лекція №1…….....8
1. Призначення, завдання, функції, класифікація ІС……8
1.1 Завдання і функції ІС……8
1.2 Класифікація ІС……9
1.2.1 Класифікація систем за рівнем та сферою діяльності……..10
1.2.2 Класифікація систем за засобами вирішення інформаційної проблеми……....10
1.2.3 Класифікація систем за характером збереженої інформації……...11
1.2.4 Класифікація систем за масштабом……11
1.2.5 Класифікація систем за сферою обслуговування……..13
1.2.6 Класифікація систем в залежності від сфери застосування……...13
Лекція№2……..…….....16
2. Корпоративні інформаційні систем……...16
2.1 Основні характеристики сучасних КІС……16
2.2 Поділ корпоративних інформаційних систем на класи……...17
2.2.1 Системи ERP (Enterprise Resource Planning System) - планування ресурсів.підприємства……...17
2.2.2 Системи CRM (Customer Relationship Management System) - управління відносинами з клієнтами……...18
2.2.3 Системи MES (Manufacturing Execution System) – спеціалізоване прикладне програмне.забезпечення……19
2.2.4 Системи WMS (Warehouse Management System) - управління складськими.ресурсами……21
2.2.5 Система EAM (Enterprise Asset Management) - управління фондами підприємст……..24
2.2.6. Система HRM (Human Resource Management) - управління персоналом……24
Лекція №3-4……..26
3. Життєвий цикл ІС та його структура……..26
3.1 Поняття життєвого циклу системи……26
3.2 Стадії та етапи життєвого циклу системи……26
3.3 Найбільш поширені стандарти регламентації ЖЦ ІС…..…….28
3.4 Основні процеси життєвого циклу ПЗ……..32
4.1 Аналіз та формування вимог до системи (формування концепції)……...36

4.2 Формування головних вимог до системи……..37
Лекція №5……40
5. Моделі життєвого циклу інформаційної системи……..40
5.1 Каскадна модель життєвого циклу інформаційної системи……...41
5.1.1 Головні етапи каскадної моделі……..41
5.1.2 Переваги каскадної моделі……..44
5.1.3 Недоліки каскадної моделі……..44
5.1.4 Область застосування..…….46
5.2 Спіральна модель життєвого циклу……...46
5.2.1 Переваги спіральної моделі…….47
5.2.2 Недоліки спіральної моделі…….48
5.3 Поетапна (ітераційна) модель з проміжним контролем……..48
5.3.1 Переваги поетапної (ітераційної) моделі……...49
5.3.2 Основні недоліки поетапної (ітераційної) моделі…….49
5.4 Гнучке розроблення програмного забезпечення Agile……49
5.4.1 Переваги Agile……..50
5.4.2 Недоліки Agile випливають з його переваг……...50
Лекція №6……51
6. Технічне завдання. Технічний проєкт…….51
6.1 Місце технічного завдання в життєвому циклі АС……..51
6.2 Склад і зміст технічного завдання…….52
6.3 Ескізний проєкт та технічний проєкт (проєкт) ІС……54
Лекція №7……58
7. Сучасні методології проєктування інформаційних систем……...58
7.1 Методологія RAD……58
7.1.1 Головні особливості методології RAD……...58
7.1.2 Головні засади методології RAD……59
7.1.3 Об’єктно-орієнтований підхід у методології RAD……...59
7.1.4 Візуальне програмування і методологія RAD……...60
7.1.5 Подієве програмування і методологія RAD……...61
7.1.6 Життєвий цикл інформаційної системи за методологією RAD…...61
7.1.7 Обмеження методології RAD……..63
7.2 Методологія RUP…….64
7.2.2 Підходи RUP…….64
7.3 Стандарти проєктування інформаційних систем. Методологія CDM……66
7.3.1 Стандарти та методики……66
7.3.2 Методика CDM фірми Oracle……..67
7.3.3 Структура життєвого циклу згідно з методологією CDM……68
7.3.4 Особливості методики CDM……...69
Лекція №8……70

8. Уніфікована мова візуального моделювання Unified Modeling Language……70
8.1 Синтаксис і семантика основних об'єктів. UML Класи……...72
8.1.1 Діаграма класів……73
8.1.2 Діаграми використання……75
8.1.3 Діаграми послідовностей……76
8.1.4 Діаграми станів……77
8.1.5 Діаграми діяльності……..78
8.1.6 Діаграми компонентів……..79
Лекція №9……80
9. Дослідна експлуатація і введення в дію інформаційних систем……..80
9.1 Етапи впровадження ІС……..80
9.2 Супроводження і модернізація інформаційних систем……...82
Список літератури……...84

Вступ

Проєктування інформаційних систем – дисципліна, яка є нормативною у напрямку бакалаврської підготовки за напрямом "Комп'ютерні науки" та «Інформаційні системи і технології» належить до циклу професійної та практичної підготовки.

Метою дисципліни «Проєктування інформаційних систем» – є придбання студентами базової профільної підготовки за фахом, формування у здобувачів освіти теоретичних знань, та практичних навичок у галузі проєктування інформаційних систем з використанням сучасних методів та засобів створення інформаційних систем під час розробки, налагодження, тестування та подальшої експлуатації. Здобуті знання з дисципліни повинні стати базою вивчення дисциплін професійно-орієнтованого циклу.

Дисципліна фахово надає базову підготовку для студентів в галузі проєктування інформаційних систем, а саме: методами, принципами, технологіями, інструментальними засобами, стандартами та шаблонами проєктування. Дозволяє вирішувати задачі аналізу та проєктування інформаційних систем, їх модернізацію (реновацію), та реінжинірингу.

Завершивши вивчення дисципліни студент повинен бути здатним до проєктної діяльності в професійній сфері за фахом, вміти будувати і використовувати моделі та методи для опису об'єктів та процесів, здійснювати їх якісний аналіз та декомпозицію, застосовувати їх під час розробки та впровадження систем.

За результатами вивчення дисципліни студент повинен знати:

- задачі, функції, види та класифікацію ІС;
- стандарти проєктування ІС та оформлення проєктної документації;
- системний підхід до проєктування ІС, топології та архітектури інформаційних систем;
- методи аналізу, вимоги до ІС, формування вимог до ІС;
- методології і технології проєктування ІС;
- основи структурного і об'єктно-орієнтованого підходу до аналізу і проєктування ІС;
- основні етапи проєктування ІС, визначення і сферу застосування CASE засобів і технологій ІС;
- інструментальні засоби технологій проєктування ІС;
- реінжиніринг ІС.

Лекція №1.
Основні визначення

1. ***Проєктування*** – процес створення проєкту, прототипу, прообразу майбутнього об'єкта, стану та способів його виготовлення.

Проєктування – це комплекс робіт який складається з пошуку, досліджень, розрахунків та розрахування з метою отримання опису достатнього для створення нового об'єкту

2. ***Експлуатація*** – стадія життєвого циклу об’єкту проєктування (системи), на якій реалізується, підтримується і відновлюється його якість.

3. ***Інформаційна система (ІС)*** – система обробки даних в будь-якій предметній галузі із засобами накопичення, збереження, оновлення, пошуку та видачі інформації.

4. ***Технічне завдання*** – основний (підсумовуючий) документ вихідних даних на проєктування, який встановлює призначення системи, галузь її застосування, характеристику (короткий опис), технічні вимоги, етапи розробки і терміни їх виконання, обґрунтування ефективності застосування, перелік документів, що підлягають розгляду замовником, особливості приймальних випробувань.

1. Призначення, завдання, функції, класифікація ІС.

Як система, ІС природно володіє основними властивостями систем, такими як ієрархічність, централізація і децентралізація, цілісність та незалежність.

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

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

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

1.1. Завдання і функції ІС

Перша група пов'язана з суто інформаційним забезпеченням основної діяльності: відбір необхідних повідомлень, їх обробка, зберігання, пошук і видача суб'єкту основної діяльності із заздалегідь заданою повнотою, точністю та оперативністю в найбільш прийнятній формі.

Друга група завдань пов'язана з обробкою отриманої інформації (даних) у відповідності з тими чи іншими алгоритмами або програмами з

метою підготовки рішень завдань, що стоять перед суб'єктом основної діяльності (так званих "користувальницьких" завдань).

Завдання першої групи – це завдання інформатизації суспільства "вшир".

Завдання другої групи – завдання інформатизації суспільства "вглиб".

Для вирішення поставлених завдань ІС повинна виконати наступні ***функції***:

1. Відбір повідомлень з внутрішнього і зовнішнього середовища, необхідних для реалізації основної діяльності;
2. Введення інформації в ІС;
3. Зберігання інформації в пам'яті ІС, її актуалізація і підтримка цілісності;
4. Обробка, пошук і видача інформації у відповідності з заданими вимогами.

Обробка може включати і підготовку варіантів вирішення користувальницьких прикладних задач за відповідними алгоритмами/ програмами.

1.2. Класифікація ІС

Інформаційні системи можна прокласифікувати за різними ознаками. Найбільш поширені класифікаційні ознаки та їх типи систем на сьогодні у використанні наведені у таблиці 1.

Таблиця 1

Класифікація ІС

Ознаки класифікації	Типи систем
За рівнем або сферою діяльності	- державні; - територіальні (регіональні); - галузеві; - об'єднань, підприємств або установ; - Технологічних процесів.
Засоби вирішення інформаційної проблеми	- ручні; - механічні; - автоматизовані; - автоматичні.
За масштабом	- одиночні; - групові; - корпоративні; - глобальні
За сферою обслуговування (по характеру обробки інформації)	- інформаційно-довідкові; - інформаційно вирішуючі.
За характером збереженої	- фактографічні

інформації (БД)	- документальні.
За способом організації	- на основі архітектури файлсервер; - на основі архітектури клієнт сервер; - на основі багаторівневої архітектури; - системи на основі інтернет-технологій
По сфері застосування	- інтегровані (корпоративні); - АСУТП, - організаційного управління; - САПР.

1.2.1. Класифікація систем за рівнем та сферою діяльності

Державні ІС – призначені для вирішення найважливіших народногосподарських проблем країни. На базі використання обчислювальних комплексів та економіко-математичних методів у них складають перспективні та поточні плани розвитку країни, ведуть облік результатів та регулюють діяльність окремих ланцюгів народного господарства, розробляють державний бюджет та контролюють його виконання і т. ін.

Територіальні (регіональні) ІС – призначені для управління адміністративно-територіальним регіоном. Сюди належать АС області, міста, району. Ці системи виконують роботи з обробки інформації, яка необхідна для реалізації функцій управління регіоном, формування звітності й видачі оперативних даних місцевим і керівним державним та господарським органам.

Галузеві інформаційні системи – призначені для управління підвідомчими підприємствами та організаціями. Галузеві системи діють у промисловості та в сільському господарстві, будівництві на транспорті і т. ін. У них розв'язуються задачі інформаційного обслуговування апарату управління галузевих міністерств і їх підрозділів.

Інформаційні системи управління підприємствами або виробничими об'єднаннями – це системи із застосуванням сучасних засобів автоматизованої обробки даних, економіко-математичних та інших методів для регулярного розв'язування завдань управління виробничо господарською діяльністю підприємства.

Інформаційні системи управління технологічними процесами – керують станом технологічних процесів (робота верстата, домни тощо). Перша й головна відмінність цих систем від розглянутих раніше полягає передусім у характері об'єкта управління - машини, прилади,

обладнання, а для державних, територіальних та інших систем - це колективи людей.

1.2.2. Класифікація систем за засобами вирішення інформаційної проблеми

В ручних інформаційних системах проблема вирішується ручним способом. Характеризується відсутністю технічних засобів обробки інформації і виконанням всіх процесів людиною. Наприклад, бібліотечна система, складається з каталогу, що містить впорядковану певним чином коротку інформацію про літературу та її місце збереження і власне сховище, де література знаходиться на вказаних в каталозі місцях, але пошук літератури її доставка виконується вручну.

В механізованих системах – інформаційна проблема виконується механічними пристроями.

В автоматизованих системах інформаційна проблема розв’язується за участю технічних засобів та людини. Головна роль в виконанні формалізованих процесів відводиться комп’ютеру. Автоматизовані системи відповідає сучасному терміну «інформаційна система».

В автоматичних системах інформаційна проблема розв’язується технічними засобами без участі людини.

1.2.3. Класифікація систем за характером збереженої інформації

Розділяють фактографічні та документальні системи

Фактографічні інформаційні системи оперують фактичними відомостями, представленими у вигляді спеціальним чином організованих сукупностей формалізованих записів даних. Центральне функціональна ланка фактографічної інформаційної системи - СУБД.

Фактографічні інформаційні системи використовуються не тільки для реалізації довідкових функцій, але і для вирішення задач обробки даних. Під обробкою даних розуміється спеціальний клас розв'язуваних на ЕОМ задач, пов'язаних з введенням, зберіганням, сортуванням, відбором і угрупованням записів даних однорідної структури. Ці завдання передбачають представлення користувачам підсумкових результатів обробки у вигляді звітів табличної форми.

Основним завданням документальних інформаційних систем є накопичення та надання користувачу документів, змісту, тематики, реквізити і т.д., які адекватні його інформаційним потребам. Тому можна дати наступне визначення документальної ІС – єдине сховище документів з інструментарієм пошуку і відбору необхідних документів.

1.2.4. Класифікація систем за масштабом

За масштабом інформаційні системи підрозділяються на наступні групи

- ✓ одиничні;
- ✓ групові;
- ✓ корпоративні;
- ✓ глобальні.

Одиничні інформаційні системи реалізуються, як правило, на автономному персональному комп'ютері (мережа не використовується). Така система може містити кілька простих додатків, зв'язаних загальним інформаційним фондом, і розрахована на роботу одного користувача або групи користувачів, які поділяють за часом одне робоче місце. Подібні додатки створюються за допомогою так званих настільних або локальних систем управління базами даних (СКБД). Серед локальних СУБД найбільш відомими є Clarion, Clipper, FoxPro, Paradox, dBase і Microsoft Access.

Групові інформаційні системи орієнтовані на колективне використання інформації членами робочої групи і найчастіше будуються на базі локальної обчислювальної мережі. При розробці таких додатків використовуються сервери баз даних (звані також SQL-серверами) для робочих груп. Існує досить велика кількість різних SQL-серверів, як комерційних, так і вільнорозповсюджуваних. Серед них найбільш відомі такі сервери баз даних, як Oracle, DB2, Microsoft SQL Server, InterBase, Sybase, Informix.

Сервер (англ. Server - «служка») – у комп'ютерній термінології термін може стосуватися окремого комп'ютера чи програми. Головною ознакою в обох випадках є здатність машини чи програми переважну кількість часу працювати автономно, без втручання людини реагуючи на зовнішні події відповідно до встановленого програмного забезпечення. Втручання людини відбувається під час встановлення серверу і під час його сервісного обслуговування. Часто це роблять окремі адміністратори серверів з вищою кваліфікацією.

Сервер як комп'ютер – це комп'ютер у локальній чи глобальній мережі, який надає користувачам свої обчислювальні і дискові ресурси, а так само доступ до встановлених сервісів; найчастіше працює цілодобово, чи у час роботи групи його користувачів.

Сервер як програма – програма, що надає деякі послуги іншим програмам (клієнтам). Зв'язок між клієнтом і сервером зазвичай здійснюється за допомогою передачі повідомлень, часто через мережу, і використовує певний протокол для кодування запитів клієнта і відповідей сервера.

Серверні програми можуть бути встановлені як на серверному, так і на персональному комп'ютері, щоразу вони забезпечують виконання певних служб (наприклад, сервер баз даних чи веб-сервер). Комп'ютер або програма, що установлена на цьому комп'ютері, здатні автоматично розподіляти інформацію чи файли під керуванням мережної ОС або у відповідь на запити, прислані у режимі online користувачами, і таким чином надавати послуги іншим комп'ютерам мережі (клієнтам).

Корпоративні інформаційні системи є розвитком систем для робочих груп, вони орієнтовані на великі компанії і можуть підтримувати територіально рознесені вузли або мережі. В основному вони мають ієрархічну структуру з декількох рівнів. Для таких систем характерна архітектура клієнт-сервер зі спеціалізацією серверів або ж багаторівнева архітектура. При розробці таких систем можуть використовуватися ті ж сервери баз даних, що і при розробці групових інформаційних систем. Однак у великих інформаційних системах найбільшого поширення набули сервери Oracle, DB2 і Microsoft SQL Server. Для групових і корпоративних систем істотно підвищуються вимоги до надійності функціонування і збереження даних. Ці властивості забезпечуються підтримкою цілісності даних, посилань і транзакцій в серверах баз.

Глобальні ІС охоплюють територію держави чи континенту. Прикладом такої інформаційної системи є глобальна мережа Інтернет.

1.2.5. Класифікація систем за сферою обслуговування (по характеру обробки інформації)

Інформаційно-довідкові системи роблять введення, систематизацію, зберігання, видачу інформації за запитом користувача без складних перетворень даних. (Наприклад, ІС бібліотечного обслуговування, резервування та продажу квитків на транспорті, бронювання місць у готелях та ін.). Обширний клас інформаційно-довідкових систем заснований на гіпертекстових документах і мультимедіа. Найбільший розвиток такі інформаційні системи отримали в мережі Інтернет.

Інформаційно - вирішуючі системи здійснюють, крім того, операції переробки інформації за певним алгоритмом. За характером використання вихідної інформації такі системи прийнято ділити на керуючі і ті, що радять.

1.2.6. Класифікація систем в залежності від сфери застосування

- Автоматизовані система управління (АСУ)
- Автоматизовані системи організаційного управління
- Автоматизовані системи управління технологічними процесами

- Системи підтримки прийняття рішень
- Система автоматизованого проєктування (САПР)
- Корпоративні (інтегровані) системи.

АСУ – це багаторівневі ієрархічні автоматизовані системи, які забезпечують комплексну автоматизацію управління на всіх рівнях і охоплюють весь цикл робіт від проєктування до збуту продукції. Призначена для забезпечення ефективного функціонування керованого об'єкта (системи) шляхом автоматизованого виконання заданих функцій. Ступінь автоматизації функцій управління визначається виробничою необхідністю і можливостями формалізації процесу управління.

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 та організаційні завдання.

АС управління технологічними процесами (ТП) – 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.

Системи підтримки прийняття рішень (СППР) – це інтерактивна комп'ютерна система, яка призначена для підтримки різних видів діяльності при прийнятті рішень із слабо структурованих або неструктурованих проблем. Системи підтримки прийняття рішень використовуються для управління об’єктами в слабко структурованих предметних областях.

Інтерес до СППР, як перспективної галузі використання обчислювальної техніки та інструментарію підвищення ефективності праці в сфері управління економікою, постійно зростає. У багатьох країнах розробка та реалізація СППР перетворилася на дільницю бізнесу, що швидко розвивається. Третє покоління ІС будується як СППР (в англомовній літературі використовується позначення DSS (Decision Support Systems)). Такі системи мають не тільки спільну БД, а й спільну базу моделей для розв'язування задач.

Вони орієнтовані не на автоматизацію функцій особи, яка приймає рішення (ОПР), а на сприяння в пошуку ефективного рішення. СППР

орієнтовані передусім на розв'язання слабоформалізованих задач управління підприємствами, що виникають у зв'язку з високим рівнем різноманітних невизначеностей ринкового середовища. Особлива увага в СППР приділяється діалогу та "дружності" в інтерфейсі до ОПР.

DSS (Decision Support Systeq) – являють собою інший тип інформаційних систем, в яких за допомогою досить складних запитів проводиться відбір та аналіз даних в різних розрізах: тимчасових, географічних і за іншими показниками.

Системи штучного інтелекту – це штучні системи, створені людиною на базі ЕОМ, що імітують розв'язування людиною складаних творчих завдань.

1. інтелектуальні інформаційно-пошукові системи (системи типу «запитання - відповідь»), які в процесі діалогу забезпечують взаємодію кінцевих користувачів – не програмістів з базами даних та знань професійними мовами користувачів, близьких до природних;

2. розрахунково-логічні системи, які дають змогу кінцевим користувачам, що не є програмістами та спеціалістами в галузі прикладної математики, розв'язувати в режимі діалогу з ЕОМ свої задачі з використанням складаних методів і відповідних прикладних програм;

3. експертні системи, які дають змогу впровадити ефективну комп'ютеризацію областей, у яких знання можуть бути подані в експертній описовій формі, але використання математичних моделей утруднене або неможливе.

ІС автоматизованого проєктування (САПР) – призначені для автоматизації функцій інженерів-проєктувальників, конструкторів, архітекторів, дизайнерів при створенні нової техніки або технології.

Система автоматизованого проєктування́ (САП або САПР) або автоматизована система́ проєктування (АСП) – автоматизована система, призначена для автоматизації технологічного процесу проєктування виробу, кінцевим результатом якого є комплект проєктно-конструкторської документації, достатньої для виготовлення та подальшої експлуатації об'єкта проєктування. Реалізується на базі спеціального програмного забезпечення, автоматизованих банків даних, широкого набору периферійних пристроїв.

САПР виконує такі функції:

• конструкторська частина – розробка повного комплекту конструкторської документації;

• технологічна частина – розрахунок і проєктування технологічних схем, технологічного оснащення, транспорту;

• архітектурно-будівельна частина – розрахунок і проектування металевих і залізобетонних конструкцій;

• санітарно-технічні системи – проєктування теплопостачання, опалення і вентиляції виробничих і адміністративних корпусів, а також водопостачання і каналізації;

• електротехнічні системи – розрахунок і проєктування електропостачання, електросилового устаткування, світлотехнічної частини проєктів, телемеханізації електропостачання;

• гідротехнічні спорудження – розрахунок і проєктування напірного і безнапірного гідротранспорту відвальних хвостів, стійкості укосів;

• системи автоматизації – розробка схем зовнішніх з'єднань, електричних і трубних проводок щитів автоматики;

• кошторисна частина – складання локальних і зведених кошторисів, відомостей матеріалів, специфікацій, комплектація обладнання.

Інтегровані (корпоративні) ІС – використовуються для автоматизації всіх функцій фірми і охоплюють весь цикл робіт від планування діяльності до збуту продукції. Інтегровані (корпоративні) ІС - 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.

Запитання для самоконтролю:

1. Дайте визначення терміну «проєктування».
2. Що таке поняття інформаційної системи.
2. Наведіть основні класифікаційні ознаки інформаційних систем.
3. Дайте визначення терміну «система».
4. Які основні функції має виконувати інформаційна система?
5. Яке призначення систем автоматизованого проєктування?

Лекція №2.
Корпоративні інформаційні системи

Корпоративна інформаційна система (КІС) – це інформаційна система, яка підтримує автоматизацію функцій проєктування та управління на підприємстві (в корпорації) і поставляє інформацію для прийняття рішень. У ній реалізована управлінська ідеологія, яка об'єднує бізнес стратегію підприємства і прогресивні інформаційні технології.

Повноцінна КІС повинна забезпечити інформаційну прозорість підприємства, формувати єдиний Інформаційний простір, який об'єднує

інформаційні потоки, що йдуть від виробництва до нього, з даними фінансово-господарських служб і видавати необхідні повідомлення для всіх рівнів управління підприємства (рис. 2.1.).

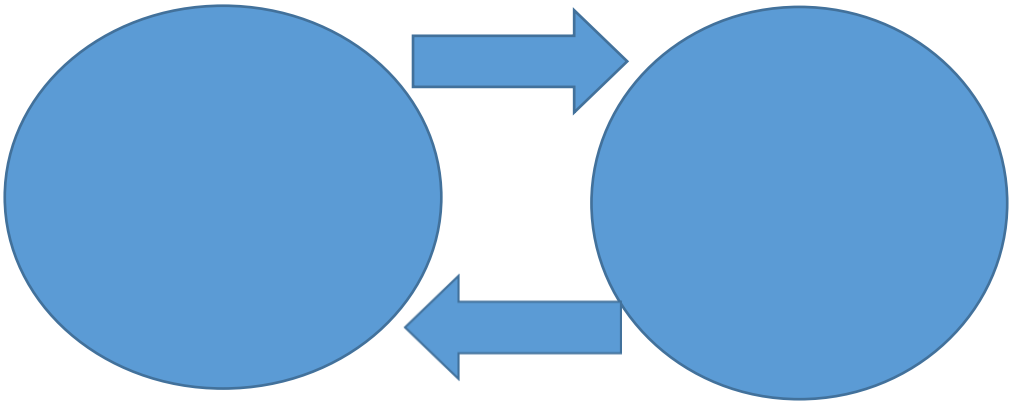

Рисунок 2.1 Взаємозв'язок КІС з виробництвом

2.1. Основні характеристики сучасних КІС

Масштабність. Це одна із важливих характеристик інформаційних систем такого класу, враховуючи масштаби діяльності корпорації. Масштабна ІС повинна функціонувати на масштабній програмно-апаратній платформі (сервери, операційні системи, системи комунікації, СУБД), що потребує значних зусиль спеціалістів з проектування й упровадження таких систем. Оскільки варіантів конфігурації базового устаткування і програмного забезпечення може бути багато, то КІС має бути багатоплатформною.

•**Багатоплатформне обчислювання**. В КІС виникає потреба в тому, щоб прикладна програма працювала на кількох платформах. При цьому мають бути забезпечені однакові інтерфейс і логіка роботи на всіх платформах, маючи на увазі подібність схем екрана, елементів меню і діалогової інформації, що надається користувачеві різними платформами; інтегрованість з користувацьким операційним середовищем; однакова поведінка на різних платформах; узгоджена підтримка незалежно від платформи тощо. Реалізувати прикладну програму одночасно в кількох середовищах нелегко. Тому з'явилися інтегровані програмні середовища розробки (frameworks), які значно полегшують перенесення прикладних програм між різними середовищами. До них належать Windows Open Systems Architecture (WOSA); Win 32, загальне відкрите програмне середовище UNIX COSE і App Ware Foundation та інші.

•**Робота в неоднорідному обчислювальному середовищі.** Важливою перевагою КІС є можливість роботи в мережах, до яких входять комп'ютери, що працюють під управлінням різних операційних систем або побудовані на різних обчислювальних платформах. При цьому має бути забезпечена взаємодія всіх робочих обчислювальних платформ і операційних систем, які використовуються.

•**Розподілені обчислення**. Це один із видів роботи в клієнт-серверній архітектурі, коли отримані з клієнтських машин дані чи запити розподіляються поміж кількома машинами, наприклад між кількома серверами, що збільшує пропускну здатність для користувача і дає можливість багатозадачної роботи. Це сприяє максимальному використанню обчислювальних ресурсів, зниженню витрат і підвищенню ефективності системи. Забезпечення розподіленої роботи і віддаленого доступу до документів — це обов'язкова вимога до інформаційних систем корпоративного рівня. Останніми роками невід'ємною складовою частиною цієї вимоги стала підтримка роботи в архітектурі Internet/Intranet.

2.2. Поділ корпоративних інформаційних систем на класи

• ERP(Enterprise Resource Planning System) - планування ресурсів підприємства;

• CRM (Customer Relationship Management System) – управління відносинами з клієнтами;

• MES (Manufacturing Execution System) – спеціалізоване прикладне програмне забезпечення;

• WMS (Warehouse Management System) – система управління складом;

• EAM (Enterprise Asset Management) – система управління фондами підприємства;

• HRM (Human Resource Management) – система управління персоналом.

2.2.1. Системи ERP (Enterprise Resource Planning System) планування ресурсів підприємства

Призначена для побудови єдиного інформаційного простору підприємства (об'єднання всіх відділів і функцій), ефективного управління всіма ресурсами компанії, пов'язаними з продажами, виробництвом, обліком замовлень.

Будується ERP-система за модульним принципом і, як правило, включає в себе модуль безпеки для запобігання як внутрішніх, так і зовнішніх крадіжок інформації. Проблеми ж виникають в основному через неправильність роботи або споконвічного побудови плану впровадження системи.

Наприклад, урізані інвестиції в навчання персоналу роботі в системі суттєво знижують ефективність. Тому впроваджують ERP-системи (рис. 2.2.) як правило не відразу в повному обсязі, а окремими модулями (особливо на початковій стадії).

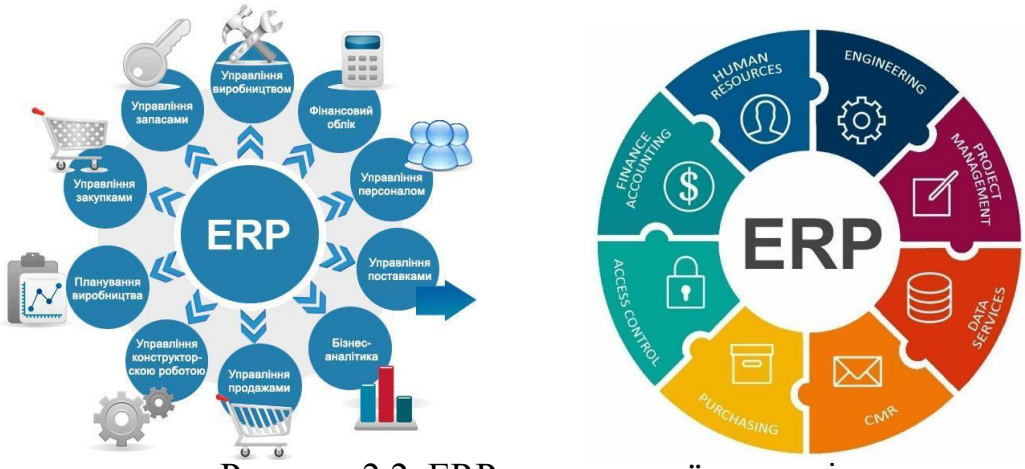

Рисунок 2.2. ERP-системи та їх модулі

2.2.2. Системи CRM (Customer Relationship Management System) - управління відносинами з клієнтами

Управління відносинами з клієнтами – поняття що охоплює концепції, котрі використовуються компаніями для управління їхніми взаємовідносинами зі споживачами, включаючи збір, зберігання й аналіз інформації про споживачів, постачальників, партнерів та інформації про взаємовідносинами з ними (рис. 2.3.).

Рисунок 2.3.Системи CRM

Сучасна CRM направлена на вивчення ринку і конкретних потреб клієнтів. На основі цих знань розробляються нові товари або послуги і таким чином компанія досягає поставлених цілей і покращує свій фінансовий показник (рис. 2.4.).

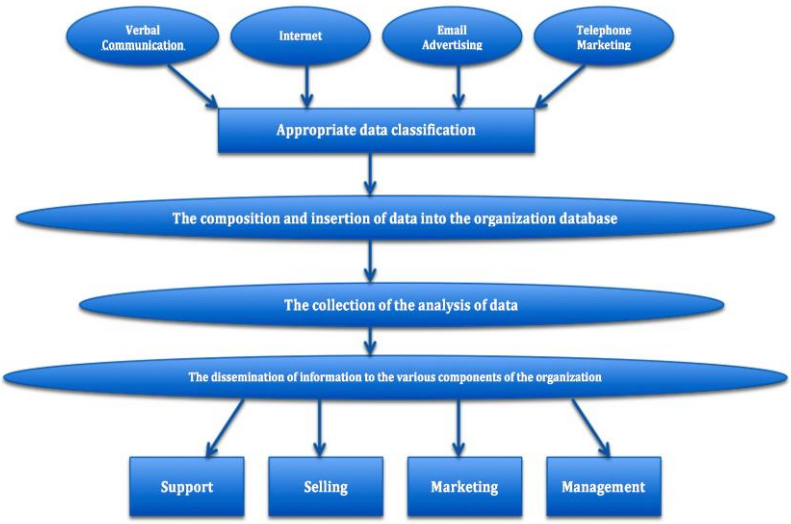

Рисунок 2.4. Направлення CRM на вивчення ринку

Принципи CRM-систем:

✓ Наявність єдиного сховища інформації, звідки в будь-який момент доступні усі відомості про всі випадки взаємодії з клієнтом;

✓ Синхронізація управління множинними каналами взаємодії;

✓ Постійний аналіз зібраної інформації про клієнтів та прийняття відповідних організаційних рішень – наприклад, «сортування» клієнтів на основі їхньої значимості для компанії.

Можливості CRM-систем:

✓ Швидкий доступ до актуальної інформації про клієнтів;

✓ Оперативність обслуговування клієнтів та проведення операцій;

✓ Формалізація схем взаємодії з клієнтами, автоматизація документообігу;

✓ Швидке отримання всіх необхідних звітних даних та аналітичної інформації;

✓ Зниження операційних витрат менеджерів;

✓ Контроль роботи менеджерів;

✓ Узгоджена взаємодія між співробітниками і підрозділами;

✓ Управління бізнес-процесами – дозволяє автоматизувати послідовні операції, які виконуються співробітниками організації;

✓ Управління контактами, історія взаємодії з клієнтами – це єдина база даних всіх контрагентів компанії (клієнтів, постачальників, конкурентів) з внесеною раніше докладною інформацією про них, про їх співробітників і т. д.

2.2.3. Системи MES (Manufacturing Execution System) – спеціалізоване прикладне програмне забезпечення. Системи класу MES (рис .2.5.) призначені для виробничого середовища підприємства.

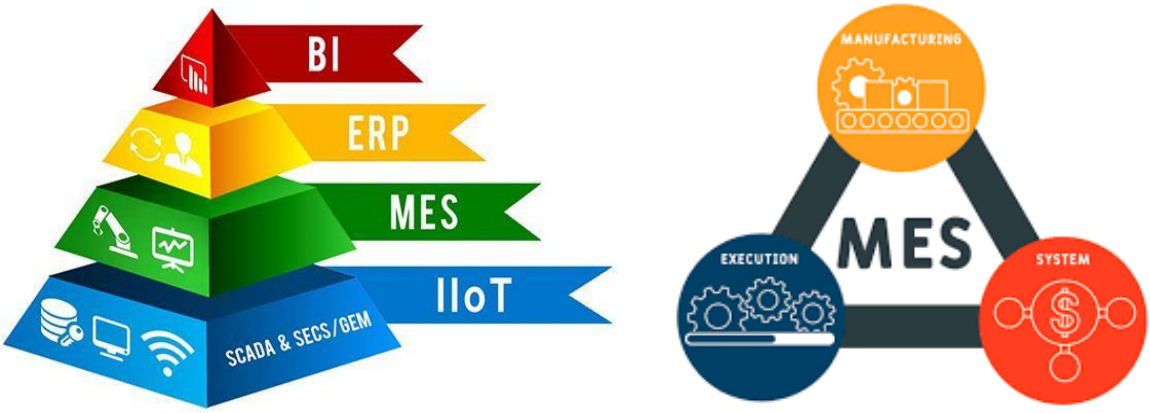

Рисунок 2.5. MES

Системи цього класу відстежують і документують весь виробничий процес, відображають виробничий цикл в реальному часі. На відміну від ERP, яка не має безпосереднього впливу на процес, за допомогою MES стає можливим коригувати (або повністю перебудовувати) процес стільки разів, скільки це буде потрібно. Інакше кажучи, системи такого класу призначені для оптимізації виробництва і підвищення його рентабельності. Збираючи та аналізуючи дані, одержувані, наприклад, від технологічних ліній, вони дають більш детальне уявлення виробничої діяльності підприємства (від формування замовлення до відвантаження готової продукції), покращуючи фінансові показники підприємства. Всі головні показники, які входять в основний курс економіки галузі (віддача основних фондів, обіг грошових коштів, собівартість, прибуток і продуктивність) детально відображаються в ході виробництва (рис. 2.6.). Фахівці називають MES мостом між фінансовими операціями ERP-систем і оперативною діяльністю підприємства на рівні цеху, ділянки або лінії. Зазначимо, що сьогодні в країнах Заходу в MES вкладаються чималі гроші.

Рисунок 2.6 Функціональний склад MES

2.2.4. Системи WMS (Warehouse Management System) - управління складськими ресурсами

Warehouse Management System - система управління, що забезпечує автоматизацію та оптимізацію всіх процесів складської роботи профільного підприємства (рис. 2.7.).

Рисунок 2.7.

WMS-система – це потужний софт, що дозволяє максимально автоматизувати роботу складу, легко й ефективно ним керувати. Контроль за усіма бізнес-процесами дає змогу раціонально використовувати площу приміщення, правильно організувати розміщення товару, рух складського транспорту. Завдяки системі працівник достеменно знає, де знаходиться певний товар, не витрачає час на пошук, що значно скорочує час кожної операції.

WMS-система: що це таке?

Якісна WMS це система керування складом, яка повністю оптимізує його роботу. Вона дозволяє упорядкувати усі процеси, що

пов’язані з прийманням, зберіганням, переміщенням, відвантаженням вантажу. Зокрема, софт контролює:

- ✓ розвантаження та розміщення товару;
- ✓ комплектування, пакування, відвантаження товару;
- ✓ необхідність та доцільність поповнювати запаси;
- ✓ розташування товару;
- ✓ логістику складу;
- ✓ безпеку тощо.

Всі операції здійснюються за допомогою таких компонентів:

✓ Клієнт (інтерфейс) – працівник вводить дані, відправляє запити щодо виконання певних операцій, відстежує розташування товару тощо. Усі дії здійснюються через комп’ютер, смартфон, термінал тощо.

✓ Сервер бази даних – зберігає усю інформацію щодо роботи складу.

Бізнес-логіка – обробляє запит клієнта, використовуючи базу даних, після чого надає відповідь, яка надходить на екран користувача. Контролювати роботу складу у режимі реального часу допомагають різноманітні датчики, сенсори, сканери, інформація з яких надходить у базу даних. Вони дозволяють не лише відстежувати розташування вантажу, але й запобігати крадіжкам, псуванню товару (датчики кліматконтролю), вчасно виявляти пожежу тощо.

Важлива роль у роботі WMS належить штрих-кодам та радіочастотним міткам, які маркують товар. Останні дозволяють легко знайти вантаж у будь-якій точці складу, навіть якщо його випадково перемістили в інше місце.

Типи WMS

Складська система WMS може постачатися як без функцій адресного зберігання, так і з ними. У першому випадку йдеться про базові програми, які ведуть облік товару, що прибуває чи відвантажується зі складу, а також допомагають створювати первинні документи. Системи з адресним зберіганням – складний софт, здатний впоратися з великою кількістю різноманітних завдань.

Системи WMS також можуть бути автономні, хмарні або інтегровані:

Автономні – дані зберігаються локально, на сервері (комп’ютері) компанії.

Хмарні – дані зберігаються у хмарі.

Інтегровані – система WMS є частиною більш складного софту, наприклад ERP (керує роботою не лише складу, а усього

підприємства) або SCM (програма для управління ланцюгом постачання).

Також варто зазначити, що WMS часто складається з модулів, від яких можна відмовитись або додати, залежно від потреб компанії. У цьому аспекті виділяють такі типи систем керування складом:

✓ Базова версія – варіант для невеликих компаній. Софт має обмежені можливості, база даних вміщує незначну кількість позицій.

✓ "Коробкова" – це вже готова система, призначена для компаній, які володіють складом середніх розмірів (1-10 тис. м2), проте товар розвантажують / відвантажують небагато. Така система не підлаштовується під роботу компанії, усі бізнес-процеси треба узгодити з нею.

✓ Адаптовані/кастомні системи – варіант для великих компаній зі складською інфраструктурою. Платформа повністю розробляється з нуля під потреби певного підприємства.

✓ Модульні – складні системи для складів з високим товарообігом та асортиментом. Вони складаються з модулів, які можна легко замінити в залежності від потреб бізнесу.

Таким чином, ринок пропонує бізнесу різні типи WMS-систем, які можна обирати, залежно від потреб підприємства.

Основні переваги WMS:

✓ Забезпечує прозорість усіх процесів, зокрема їх відстеження у режимі онлайн.

✓ Зменшує кількість помилок, що виникають через людський фактор.

✓ Прискорює усі бізнес-операції.

✓ Скорочує витрати завдяки розміщенню товару згідно з терміном придатності, розробці оптимальних шляхів його перевезення.

✓ Забезпечує якісне керування персоналом завдяки структурованій роботі.

✓ Дозволяє оцінити роботу персоналу, використовуючи неупереджені дані.

✓ Запобігає ситуаціям, коли потрібний товар закінчується на складі.

✓ Дозволяє збільшити кількість товару, що можна зберігати на складі.

✓ Поліпшує продаж завдяки швидкому виконанню замовлень, що сприяє популярності компанії серед клієнтів, просуває бренд.

✓ Надає можливість приймати правильне рішення завдяки отриманню інформації у режимі онлайн.

✓ Запобігає крадіжкам, втраті товару.

Щоб скористатися усіма перевагами, важливо навчити персонал працювати з системою. І ось тут можуть виникнути проблеми, оскільки не всі співробітники на практиці здатні опанувати софт без серйозної підготовки. Будьте готові забезпечити у компанії період адаптації.

Серед інших недоліків WMS можна виділити:

✓ Необхідність дотримуватись усіх вимог щодо отримання й відвантаження товару. Якщо працівник його не побачить й не поставить штрих код, вантаж загубиться, що може призвести до серйозних проблем.

✓ Завеликий вибір WMS систем на ринку. Компанії не завжди розуміють, який саме софт їм потрібен, й можуть помилитись з вибором. Внаслідок цього доведеться перелаштовувати усі бізнес-процеси, щоб адаптувати їх під нову систему. Щоб запобігти цьому, варто ретельно обдумати усі аспекти, проконсультуватись з нашим спеціалістом.

✓ Неправильно налаштовані процеси. Більшість компаній несхвально відгукуються про WMS системи лише у випадку, коли технічні фахівці неправильно налаштували систему, тому дуже важливо не помилитися у виборі розробника.

Функції та можливості WMS

WMS система складу може складатися з різних модулів, які дозволяють з часом розширити її можливості. Основні функції:

- Приймання, переміщення, зберігання, комплектація, відвантаження товару.
- Організація роботи складу.
- Аналіз й управління складськими запасами та інвентаризація товару.
- Автоматизація документообігу, зокрема, формування звітності. Це дозволяє менеджерам відстежувати якість роботи складу, виявляти слабкі місця, пропонувати варіанти для їх вирішення.
- Управління персоналом. Система дозволяє отримати дані щодо кожного співробітника компанії, з’ясувати якість його роботи.

2.2.5. Система EAM (Enterprise Asset Management) - управління фондами підприємства

Система управління основними фондами підприємства, що дозволяє скоротити простої устаткування, витрати на техобслуговування, ремонти і матеріально-технічне постачання (рис. 2.8.). Являє собою необхідний інструмент в роботі фондомістких галузей (енергетичних, транспортних, ЖКГ, добувної промисловості та інше).

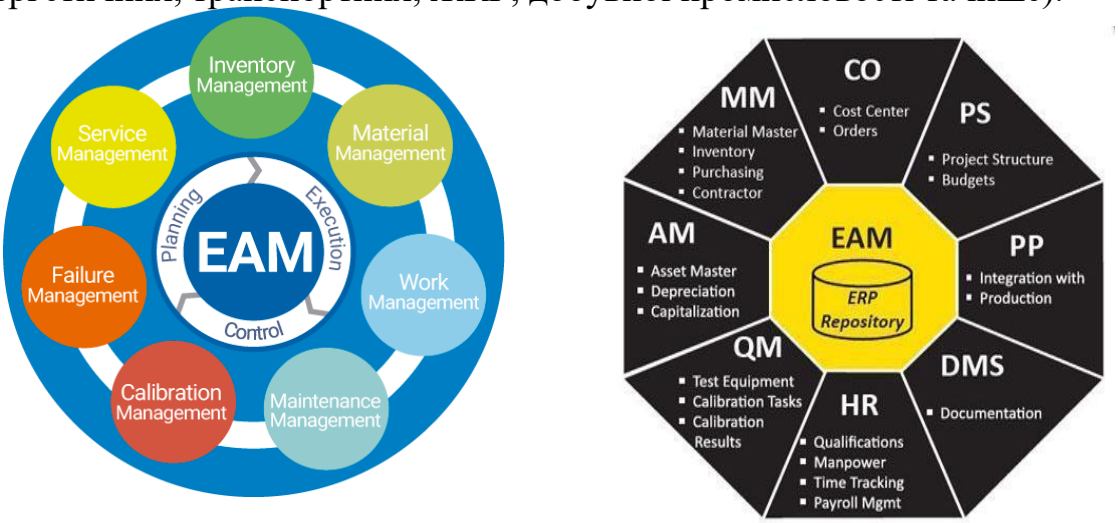

Рисунок 2.8. Система управління основними фондами підприємства

Основні фонди - це засоби праці, які багаторазово беруть участь у виробничому процесі, зберігаючи при цьому свою натуральну форму, поступово зношуючи, переносячи свою вартість по частинах на знов створену продукцію. У бухгалтерському та податковому обліку відображені в грошовому вираженні основні фонди називаються основними засобами. Зараз модулі EAM входять також до складу великих пакетів ERP-систем (таких як mySAP Business Suite, IFS Applications, Oracle E Business Suite та ін.)

2.2.6. Система HRM (Human Resource Management) - управління персоналом

Система управління персоналом - є однією з найважливіших складових частин сучасного менеджменту. Основна мета таких систем - залучення та утримання цінних для підприємства кадрових фахівців (рис. 2.9.).

HRM-системи вирішують два головні завдання:

- упорядкування всіх облікових і розрахункових процесів, пов'язаних з персоналом;
- зниження відсотка текучки співробітників.

- Таким чином, HRM - системи в певному сенсі можна назвати «CRM-системами навпаки», залучати та утримувати не покупців, а власних співробітників компаній.

Рисунок 2.9. Система управління персоналом

Зрозуміло, методи тут застосовуються зовсім інші, але загальні підходи схожі.

Функції HRM-систем:

- пошук персоналу;
- підбір та відбір персоналу;
- оцінка персоналу;
- навчання та розвиток персоналу;
- управління корпоративною культурою;
- мотивація персоналу;
- організація праці.

Запитання для самоконтролю:

1. Назвіть поняття корпоративної інформаційної системи.
2. Наведіть основні характеристики сучасних КІС.
3. Як класифікуються інформаційні системи за сферою застосування?
4. Яке призначення ERP систем.
5. Які основні задачі MES систем.
6. Охарактеризуйте функції HRM систем.

Лекція №3-4.
Життєвий цикл ІС та його структура

3.1. Основні поняття життєвого циклу системи

Життєвий цикл інформаційної системи – період часу, який починається з моменту прийняття рішення про необхідність створення інформаційної системи і закінчується в момент її повного вилучення з експлуатації.

Поняття життєвого циклу є одним з базових понять методології проєктування інформаційних систем.

Методологія проєктування інформаційних систем описує процес створення і супроводу систем у вигляді життєвого циклу (ЖЦ) ІС, представляючи його як деяку послідовність стадій та процесів.

Проведена класифікація інформаційних систем ілюструє всю різноманітність рівнів їх побудови, призначень, предметних галузей, сферою застосування, типом реалізації та інше. Проте, існують загальні стабільні умови проектування систем різного класу, що орієнтовані на наймасовішу частину ІС, мають загальні теоретичні та методологічні підстави, багато спільних рис а також типів зв'язків (функціональних, інформаційних, зовнішніх) між елементами структури систем. Це дозволяє сформулювати єдині принципи і шляхи побудови інформаційних систем.

3.2. Стадії та етапи життєвого циклу системи

Повний життєвий цикл інформаційної системи включає в себе, як правило, декілька стадій (рис. 3.1.):

- аналіз та формування вимог до системи;
- проєктування;
- реалізація ;
- тестування ;
- введення в дію ;
- експлуатація та супровід.

Процес розроблення ПЗ має забезпечити шлях від усвідомлення потреб замовника до передачі йому готового продукту.

Він складається з таких етапів:

- **визначення вимог** – збір та аналіз вимог замовника виконавцем та подання їх у нотації, що зрозуміла як замовнику, так і виконавцю;
- **проєктування** – перетворення вимог до розроблення у послідовність проєктних рішень щодо способів реалізації вимог: формування загальної архітектури програмної системи та принципів її

прив’язки до конкретного середовища функціонування; визначення детального складу модулів кожної з архітектурних компонент;

- **реалізація** – перетворення проєктних рішень у програмну систему, що реалізує означені рішення;
- **тестування** – перевірка кожного з модулів та способів їх інтеграції; тестування програмного продукту в цілому (так звана верифікація); тестування відповідності функцій працюючої програмної системи вимогам, що були до неї поставлені замовником (так звана валідація);
- **експлуатація та супроводження готової системи**.

Рисунок 3.1. Стадії та етапи життєвого циклу системи.

Підготовча робота починається з вибору моделі ЖЦ ПЗ, що відповідає масштабові, значимості і складності проєкту. Процес розроблення має відповідати обраній моделі. Розробник повинен вибрати, адаптувати до умов проекту і використовувати погоджені із замовником стандарти, методи й засоби розроблення, а також скласти план виконання робіт.

Аналіз вимог до системи розглядає функціональні можливості, вимоги користувача, вимоги до надійності і безпеки, вимоги до зовнішніх інтерфейсів тощо. Вимоги до системи оцінюються відповідно до критеріїв реалізації і можливості перевірки при тестуванні.

Проєктування архітектури системи полягає у визначенні компонентів її устаткування, ПЗ й операцій, що виконуються персоналом.

Аналіз вимог до ПЗ визначає: функціональні можливості, включаючи характеристики продуктивності і середовища

функціонування компонента; зовнішні інтерфейси; специфікації надійності і безпеки; ергономічні вимоги; вимоги до даних; вимоги до інсталяції та введення системи; вимоги до документації користувачів; вимоги до експлуатації і супроводу.

3.3. Найбільш поширені стандарти регламентації ЖЦ ІС

Основним нормативним документом, який регламентує склад процесів ЖЦ ПЗ, є міжнародний стандарт ДСТУ ISO/IEC/IEEE 12207:2018 Information Technology - Software Life Cycle Process (Прийнятий в якості стандарту ISO/IEC/IEEE 12207:2017, ISO 12207: 1995 IDT - Інформаційні технології. Процеси життєвого циклу програмних засобів) - Стандарт на процеси і організацію життєвого циклу. Поширюється на всі види замовного ПЗ. Стандарт не прив'язаний до певної моделі ЖЦ. Стандарт не містить опису фаз, стадій і етапів. Процеси стандарту будуть розглянуті далі.

Стандарт визначає: загальні набори завдань, характеристики якості, критерії оцінки та інші характеристики, необхідні для всебічного охоплення проектних ситуацій. Наприклад, при визначенні вимог до системи необхідно передбачити:

- розглянута/вивчена та проаналізована область застосування системи;
- специфікації вимог до системи описували: функції та можливості системи, бізнес, організаційні вимоги і вимоги користувача, безпекові заходи, захищеність, людські фактори, ергономіку, зв'язки, операції і вимоги супроводу; проектні обмеження та кваліфікаційні вимоги;
- документацію кваліфікаційних вимог до системи були.

При виконанні аналізу вимог щодо ПЗ передбачено одинадцять класів характеристик якості, які використовуються пізніше при гарантуванні якості.

При цьому розробник повинен встановити і документувати такі вимоги до ПЗ:

- функціональні та інші можливі специфікації (включаючи виконання, фізичні характеристики та умови середовища експлуатації), при яких повинна бути створена одиниця програмного забезпечення;
- зовнішні зв'язки (інтерфейси) з одиницею ПЗ;
- специфікації надійності, пов'язані з методами функціонування і супроводу ПЗ, включаючи специфікації, пов'язані з впливом навколишнього середовища і ймовірністю травм персоналу;

- специфікації захищеності, включаючи специфікації, пов'язані із спотворенням точності інформації;
- людські фактори специфікацій з інженерної психології (ергономіки), включаючи фактори пов'язані з ручним керуванням, взаємодією людини та обладнання, обмеженнями щодо персоналу, а також фактори, що потребують концентрованої людської уваги;
- типи даних і вимоги до бази даних;
- вимоги щодо встановлення та приймання поставленого програмного продукту в місцях його функціонування і супроводу (експлуатації);
- вимоги до документації користувача.

ISO / IEC 15288 Systems engineering. System life cycle processes Системна інженерія. Процеси життєвого циклу систем).

ГОСТ 34.601-90 – поширюється на автоматизовані системи і встановлює стадії та етапи їх створення. Крім того, в стандарті міститься опис змісту робіт на кожному етапі. Стадії й етапи роботи, закріплені в стандарті, більшою мірою відповідають каскадної моделі життєвого циклу.

Rational Unified Process (RUP) пропонує ітеративну модель розробки, що включає чотири фази: початок, дослідження, побудова та впровадження. Кожна фаза може бути розбита на етапи (ітерації), в результаті яких випускається версія для внутрішнього або зовнішнього використання. Проходження через чотири основні фази називається циклом розробки, кожен цикл завершується генерацією версії системи. Якщо після цього робота над проектом не припиняється, то отриманий продукт продовжує розвиватися і знову мине ті ж фази.

Суть роботи в рамках RUP - це створення і супровід моделей на базі UML. Microsoft Solution Framework (MSF) подібна з RUP, так само включає чотири фази: аналіз, проектування, розробка, стабілізація, є ітераційної, припускає використання об'єктно-орієнтованого моделювання.

Зміст основних процесів життєвого циклу ПЗ (ДСТУ ISO/IEC/IEEE 12207:2018) поданий в табличному виді (Таблиця 2).

Таблиця 2

Зміст основних процесів життєвого циклу ПЗ (ДСТУ ISO/IEC/IEEE 12207:2018)

Процес (виконавець процесу)	Дії	Вхід	Результат
Придбання **(Замовник)**	▪ ініціювання ▪ підготовка заявочних пропозицій ▪ підготовка угоди ▪ контроль діяльності постачальника ▪ приймання ІС	▪ рішення про початок робіт по впровадженню ІС ▪ результати обстеження діяльності замовника ▪ результати аналізу ринку ІС / тендер ▪ план поставки /розробки ▪ комплексний тест ІС	▪ техніко-економічне обґрунтування впровадження ІС (ТЕО) ▪ технічне завдання на ІС ▪ договір на поставку/розробку ▪ акти приймання етапів ▪ роботи ▪ акт приймально/здавальних випробувань
Постачання **(Розробник ІС)**	▪ ініціювання ▪ відповідь на заявочні пропозиції ▪ підготовка договору ▪ планування виконання ▪ контроль виконання ▪ постачання	▪ технічне завдання на ІС ▪ рішення керівництва про участь в розробці ▪ результати тендеру ▪ план УП ▪ розроблена ІС і документація	▪ рішення про участь в розробці ▪ комерційні пропозиції / конкурсна заявка ▪ договір на поставку /розроблення ▪ план УП ▪ реалізація / коригування ▪ акт приймально давальних випробувань
Розроблення **(Розробник ІС)**	▪ підготовка ▪ аналіз вимог до ІС ▪ проєктування архітектури ІС ▪ розробка вимог до ПЗ ▪ проєктування архітектури ПЗ ▪ детальне проєктування ПЗ ▪ кодування і тестування ПЗ ▪ інтеграція ПЗ і кваліфікаційне тестування ПЗ ▪ інтеграція ІС і кваліфікаційне тестування ІС	▪ технічне завдання на ІС ▪ технічне завдання на ІС, модель ЖЦ ▪ технічне завдання на ІС ▪ підсистеми ІС ▪ специфікації вимоги до компонентів ПЗ ▪ архітектура ПЗ ▪ матеріали детального проєктування ПЗ план інтеграції ПЗ, тести ▪ архітектура ІС, ПЗ, документація на ІС, тести	▪ використовувана модель ЖЦ, стандарти розробки ▪ план робіт ▪ склад підсистем, компоненти обладнання ▪ специфікації вимоги до компонентів ПЗ ▪ склад компонентів ПЗ, інтерфейси з БД, план інтеграції ПЗ ▪ проєкт БД, специфікації інтерфейсів між компонентами ПЗ, вимоги до тестів ▪ тексти модулів ПЗ, акти автономного тестування ▪ оцінка відповідності 1)комплексу ПЗ вимогам ТЗ; ▪ 2) ПЗ, БД, технічного комплексу та комплекту документації

Відповідно до стандарту ISO/IEC/IEEE 12207 в структуру ЖЦ слід включати такі групи процесів(рис. 3.2.):

I. *Договірні процеси*: придбання (внутрішні рішення або рішення зовнішнього постачальника); постачання (внутрішні рішення або рішення зовнішнього постачальника).

II. *Процеси підприємства*: управління навколишнім середовищем підприємства; інвестиційне управління; управління ЖЦ ІС; управління ресурсами; управління якістю.

III. *Проєктні процеси*: планування проєкту; оцінка проєкту; контроль проєкту; управління ризиками; управління конфігурацією; управління інформаційними потоками; прийняття рішень.

IV. *Технічні процеси*: визначення вимог; аналіз вимог; розробка архітектури; впровадження; інтеграція; верифікація; перехід; атестація; експлуатація; супровід; утилізація.

V. *Спеціальні процеси*: визначення та встановлення взаємозв'язків виходячи із завдань і цілей.

1. *Основні процеси охоплюють* – процеси придбання; постачання, розроблення; експлуатація; супроводження ПЗ.

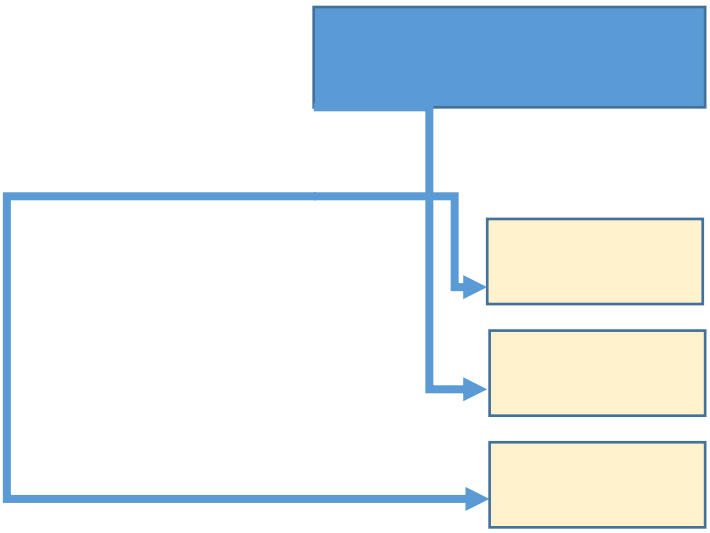

Рисунок 3.2. Процеси ЖЦ ПЗ

2. *Допоміжні процеси* – призначені для підтримки виконання основних процесів, забезпечення якості проекту, організації верифікації, перевірки та тестування ПЗ.

3. *Організаційні процеси* – охоплюють процеси створення інфраструктури; управління; навчання; удосконалення. До них примикає процес адаптації, який визначає основні дії, необхідні для адаптації стандарту до умов конкретного проєкту. Організаційні процеси

визначають дії і завдання, що виконуються замовником та розробником проєкту для управління їхніми процесами.

3.4. Основні процеси життєвого циклу ПЗ

I.***Процес придбання*** - складається з дій і завдань замовника, який купує ПЗ.

Цей процес охоплює такі дії:

1. ініціювання придбання;
2. підготовку заявочних пропозицій;
3. підготовку та коригування договору;
4. нагляд за діяльністю постачальника;
5. приймання і завершення робіт.

Ініціювання придбання включає такі задачі:

a) визначення замовником своїх потреб у придбанні, розробці або вдосконаленні системи;

b) аналіз вимог до системи;

c) прийняття рішення щодо придбання, розробки або удосконалення існуючого ПЗ;

d) перевірка наявності необхідної документації, гарантій, сертифікатів, ліцензій і підтримка в разі придбання програмного продукту (інформаційної системи);

e) підготовка та затвердження плану придбання, який включає вимоги до системи, тип договору та відповідальність сторін.

Заявочні пропозиції повинні містити:

a) вимоги до системи;

b) перелік програмних продуктів; умови та угоди; технічні обмеження (наприклад, середовище функціонування системи).

Їх направляють обраному постачальнику (або декільком постачальникам/розробникам у разі проведення тендера).

Постачальник/розробник – організація, яка укладає договір із замовником на поставку системи, ПЗ або програмної послуги на умовах, обумовлених в договорі.

Підготовка та коригування договору включають етапи:

a) визначення замовником процедури вибору постачальника, яка включає критерії оцінки пропозицій можливих постачальників;

b) вибір конкретного постачальника на основі аналізу пропозицій;

c) підготовлення та укладення договору з постачальником;

d) внесення змін (при необхідності) в договір в процесі його виконання.

Нагляд за діяльністю постачальника - здійснюється відповідно до дій, які передбачені в процесах спільної оцінки та аудиту.

У процесі приймання готуються і виконуються необхідні тести. В разі задоволення всіх умов приймання здійснюється завершення робіт за договором та подальше приймання їх в експлуатацію.

II. ***Процес постачання*** - включає такі дії: ініціювання поставки;

1. підготовку відповіді на заявочні пропозиції;
2. підготовка договору;
3. планування;
4. виконання та контроль;
5. перевірку та оцінку;
6. постачання і завершення робіт.

Ініціювання поставки полягає у розгляданні постачальником заявочних пропозицій і прийнятті рішення погодитися з поставленими вимогами та умовами або запропонувати свої.

Планування включає:

a) прийняття рішення постачальником щодо виконання робіт своїми силами або із залученням субпідрядника;

b) розробку постачальником плану УП, який містить організаційну структуру проєкту, розмежування відповідальності, технічні вимоги до середовища розробки і ресурсів, управління субпідрядниками тощо

III. ***Процес розробки*** – передбачає дії і завдання, які виконуються розробником, і охоплює роботи зі створення ПЗ і його компонентів відповідно до заданих вимог, включаючи оформлення проєктної та експлуатаційної документації, підготовку матеріалів, необхідних для перевірки працездатності та відповідної якості програмних продуктів, матеріалів, необхідних для організації навчання персоналу тощо.

Процес розробки включає:

1. підготовчу роботу;
2. аналіз вимог до системи;
3. проєктування архітектури системи
4. аналіз вимог до ПЗ;
5. проєктування архітектури ПЗ;
6. детальне проєктування ПЗ;
7. кодування і тестування ПЗ;
8. інтеграцію ПЗ;
9. кваліфікаційне тестування ПЗ;
10. інтеграцію системи;
11. кваліфікаційне тестування системи;
12. встановлення ПЗ;
13. приймання ПЗ.

Підготовча робота розпочинається з вибору моделі ЖЦ ПЗ, яка відповідає масштабу, значущості та складності проекту. Дії і завдання процесу розробки повинні відповідати обраній моделі. Розробник повинен вибрати, адаптувати до умов проекту і виконати узгодженні із

замовником стандарти, методи і засоби розробки, а також скласти план виконання робіт.

Аналіз вимог до системи передбачає визначення її функціональних можливостей, призначених для користувача вимог, вимог до надійності і безпеки, до зовнішніх інтерфейсів тощо. Вимоги до системи оцінюються виходячи з критеріїв можливості бути реалізованими і можливості перевірки при тестуванні.

Проєктування архітектури системи на високому рівні полягає у визначенні компонентів її обладнання, ПЗ та операцій, які виконуються персоналом, що експлуатує систему. Архітектура системи повинна відповідати вимогам до системи, а також прийнятим проєктним стандартам і методам.

Аналіз вимог до ПЗ передбачає визначення таких характеристик для кожного компонента:

a) функціональних можливостей, включаючи характеристики продуктивності та середовища функціонування компонента;
b) зовнішніх інтерфейсів;
c) специфікацій надійності і безпеки;
d) ергономічних вимог;
e) вимог до використовуваних даних;
f) вимог до встановлення і приймання;
g) вимог до користувальницької документації;
h) вимог до експлуатації і супроводу.

Вимоги до ПЗ оцінюються виходячи з критеріїв відповідності вимогам системи, можливостей системи бути реалізованою та її перевірки при тестуванні.

Проєктування архітектури ПЗ передбачає вирішення завдань:

a) трансформацію вимог до ПЗ в архітектуру, яка визначає на високому рівні структуру ПЗ і склад його компонентів;
b) розробку і документування програмних інтерфейсів ПЗ і баз даних;
c) розробку попередньої версії користувальницької документації;
d) розробку і документування попередніх вимог до тестів і плану інтеграції ПЗ.

Архітектура компонентів ПЗ повинна відповідати вимогам, пред'явленим до них, а також прийнятим проєктним стандартам і методам.

Детальне проєктування ПЗ передбачає:

a) опис компонентів ПЗ та інтерфейсів між ними на більш низькому рівні, достатньому для їх подальшого самостійного кодування і тестування;
b) розробка і документування детального проєкту бази даних;

c) оновлення (при необхідності) користувальницької документації;

d) розробка і документування вимог до тестів і плану тестування компонентів ПЗ;

e) оновлення плану інтеграції ПЗ.

Кодування і тестування ПЗ реалізує завдання:

a) розробка (кодування) і документування кожного компонента ПЗ і бази даних, а також сукупності тестових процедур і даних для їх тестування;

b) тестування кожного компонента ПЗ і бази даних на відповідність запропонованим до них вимогам (результати тестування компонентів повинні бути задокументовані);

c) оновлення (при необхідності) користувальницької документації;

d) оновлення плану інтеграції ПЗ

Інтеграція ПЗ – об'єднання розроблених компонентів ПЗ відповідно до плану інтеграції та тестування агрегованих компонентів.

Для кожного з агрегованих компонентів розробляються набори тестів і тестові процедури, призначені для перевірки кожного з кваліфікаційних вимог при подальшому кваліфікаційному тестуванні.

Кваліфікаційна вимога – набір критеріїв або умов, які необхідно виконати, щоб кваліфікувати програмний продукт як такий, що відповідає своїм специфікаціям і готовий до використання в умовах експлуатації.

Кваліфікаційне тестування ПЗ – проводиться розробником у присутності Замовника для демонстрації того, що ПЗ задовольняє своїм специфікаціям і є готовим до використання в умовах експлуатації. При цьому також перевіряються повнота технічної і призначеної для користувача документації, її адекватність компонентам ПЗ. Інтеграція системи полягає у об'єднанні всіх її компонентів, включаючи ПЗ та устаткування. Після інтеграції система піддається кваліфікованому тестуванню на відповідність сукупності вимог до неї. При цьому також виконують оформлення і перевірку повного комплекту документації до системи.

Встановлення ПЗ – виконується розробником відповідно до плану в середовищі і на обладнанні, які передбачені договором. У процесі встановлення перевіряється працездатність ПЗ і баз даних. Якщо ПЗ, яке встановлюють, замінює існуючу систему, розробник повинен забезпечити їх паралельне функціонування відповідно до договору.

Приймання ПЗ передбачає оцінку результатів кваліфікаційного тестування ПЗ і системи, документування результатів оцінки, які проводяться замовником за допомогою розробника.

Розробник виконує остаточну передачу ПЗ замовнику відповідно до договору, забезпечуючи при цьому необхідне навчання та підтримку.

IV. ***Процес експлуатації***

Цей процес охоплює дії і завдання оператора – організації, яка експлуатує систему. Даний процес включає:

1. підготовчу роботу;
2. експлуатаційне тестування;
3. експлуатацію системи;
4. підтримку користувачів.

Підготовча робота включає проведення оператором таких завдань:

a) планування дій і робіт, які виконуються в процесі експлуатації, і встановлення експлуатаційних стандартів;

b) визначення процедур локалізації та ліквідації проблем, які виникають в процесі експлуатації.

Експлуатаційне тестування здійснюється для кожної чергової редакції програмного продукту, після чого вона передається в експлуатацію.

Експлуатація системи виконується в призначеному для цього середовищі відповідно до користувальницької документації.

Підтримка користувачів полягає в наданні допомоги і консультацій при виявленні помилок в процесі експлуатації ПЗ.

V. ***Процес супроводження*** – передбачає дії і завдання, які виконує супроводжуюча організація (служба супроводу). Даний процес активується при змінах (модифікаціях) програмного продукту і відповідної документації, викликаних проблемами, що виникли (потребами в модернізації або адаптації ПЗ).

Під супроводом розуміють внесення змін до ПЗ з метою виправлення помилок, підвищення продуктивності або адаптації до умов роботи (або до вимог), які змінилися.

Зміни, які вносяться до існуючого ПЗ (ІС), не повинні порушувати цілісність. Процес супроводу включає перенесення ПЗ в інше середовище (міграцію) і закінчується зняттям ПЗ з експлуатації.

Процес супроводу реалізує:

1. підготовчу роботу;
2. аналіз проблем і запитів на модифікацію ПЗ;
3. модифікацію ПЗ;
4. перевірку і приймання;
5. перенесення ПЗ в інше середовище;
6. зняття ПЗ з експлуатації.

4.1. Аналіз та формування вимог до системи (формування концепції)

Поштовхом для генерації, будь яких бізнес-ідеї завжди є незадоволеність існуючими програмними продуктами (системами), або взагалі їх відсутність. Таким чином, необхідно проаналізувати

проблеми, яку вирішує програмне забезпечення (ІС) і для яких споживачів.

Досить непростим є питання, як правильно сформулювати проблему, тому що неточності в формулюванні проблеми призводять до неможливості її аналізу. Виділяють наступні вимоги до формулювання проблеми:

1. *Проблема повинна бути суттєвою*: має бути чітко та однозначно зрозуміло, що проблема існує. Більшість проблем на теперішній час мають свої рішення. Але ці рішення можуть мати свої недоліки. Довго, дорого, складно і таке інше. Якщо проблема існує і в неї є рішення, яким споживачі задоволені, то дуже важко запропонувати їм краще альтернативне рішення.

2. *Проблема має бути реалістичною та конкретною*. Треба уникати загальних формулювань, глобальних проблем та неоднозначних характеристик, які потребують додаткових критеріїв визначеності. Наприклад: недостатня функціональність системи.

3. *У формулюванні проблеми має бути однозначність*. Під однозначністю розуміється, що розглядається одна проблема, а не декілька. Проблема може бути складною та мати комплексну структуру, але вона при формулюванні повинна бути єдиною.

Для аналізу сформульованої проблеми необхідно:

- побудувати дерево проблем;
- Провести аналіз зацікавлених сторін у вирішенні цієї проблеми.

Необхідно враховувати, що першочерговість етапів залежить від міри обізнаності в проблемі. Якщо мається суттєва експертність у визначеній проблематиці, то можна починати з побудови дерева проблем, а потім консультуватися з зацікавленими сторонами. Якщо експертності недостатньо, то спочатку необхідно визначитись з зацікавленими сторонами проекту.

Зацікавлені сторони – це особа або група осіб, які зацікавленні в створенні, впровадженні програмного продукту, який дозволяє вирішити в певній мірі визначену ними проблему.

Основна мета початкового етапу створення ІС, на стадії аналізу, є формування вимог до ІС.

Ціллю реалізації цього етапу є створення моделі поведінки системи. Він базується на ***аналізі проблеми*** *функціонування діючої системи,* ***дослідженню предметної області, пошуку і оцінці варіантів її удосконалення та зводиться до формування головних вимог до системи.***

4.2. Формування головних вимог до системи

Першим етапом цієї стадії є аналіз проблеми системи.

Зміст аналізу проблеми:

- Чи існує проблема?

- Точне формулювання проблеми.
- Аналіз логічної структури проблеми.
- Розвиток проблеми (у минулому і в майбутньому).
- Зовнішні зв’язки проблеми (з іншими проблемами).
- Принципова можливість розв’язання проблеми.

На наступному етапі потрібно визначити цілі, тому що як формалізовані, так і слабко структуровані проблеми необхідно звести до такого вигляду, коли вони стають завданнями відшукання відповідних засобів для досягнення заданих цілей. Коли йдеться про цілі, то слід з’ясувати, чого ми насправді бажаємо.

Будується та аналізується в першій ітерації дерево цілей побудови системи (рис. 4.1.).

Аналіз проблеми, цілей та визначення вимог до майбутньої системи базується на дослідженні предметної області.

В рамках цього етапу здійснюється:

- Попереднє виявлення вимог, що пред'являються до майбутньої системи;
- Визначення оргштатної і топологічної структур підприємства (об’єкта комп’ютеризації);
- Визначення переліку цільових завдань (функцій) системи;
- Аналіз розподілу функцій по учасникам процесу функціонування та визначається технологія їх реалізації;
- Аналіз інформаційного забезпечення.

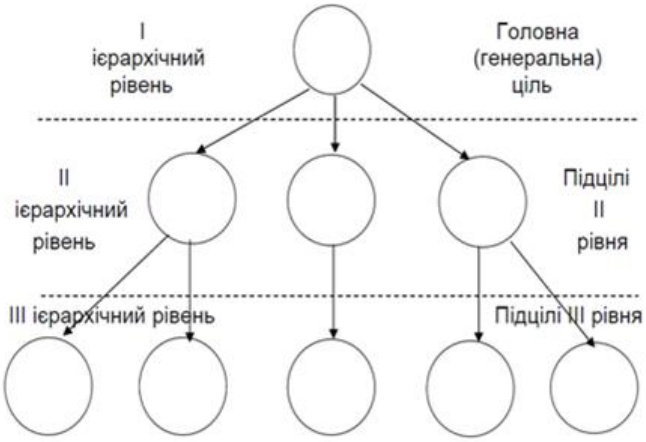

Рисунок 4.1. Ієрархія цілей системи

Початковим етапом процесу створення ІС є моделювання бізнес процесів, що протікають в організації та реалізують її цілі і завдання.

Модель організації, описана в термінах бізнес-процесів і бізнес-функцій, дозволяє сформулювати основні вимоги до ІС. На даному етапі

здійснюється обробка результатів обстеження та побудова моделей діяльності підприємства наступного типу:

- Моделі "як є" , що представляє собою " знімок" стану справ на підприємстві (оргштатної структура, взаємодії підрозділів, прийняті технології, автоматизовані і неавтоматизовані бізнес-процеси і т.д.) на момент обстеження і дозволяє зрозуміти, що робить і як функціонує дане підприємство з позицій системного аналізу, а також на основі автоматичної верифікації виявити ряд помилок і вузьких місць і сформулювати ряд пропозицій щодо поліпшення ситуації.

В процесі аналізу інформаційного забезпечення визначається:

- склад інформації (перелік інформаційних одиниць, необхідних для розв'язання комплексу задач);
- структуру інформації та закономірності її перетворення, тобто правила формування показників і документів;
- характеристики руху інформації (обсяг та інтенсивність потоків, маршрути руху, часові характеристики);
- характеристики якості інформації (систему кількісних оцінок значущості, повноти, своєчасності, вірогідності інформації);
- способи перетворення інформації;
- уніфіковану систему первинної документації;
- методичні й інструктивні матеріали для ведення документів.

З аналізом зв'язані аналіз предметної області та моделювання варіантів побудови системи (планування сценаріїв) **моделі "як має бути"**, що інтегрує перспективні пропозиції керівництва і співробітників підприємства, експертів і системних аналітиків і дозволяє сформувати бачення нових раціональних технологій роботи підприємства (рис. 4.2.).

Рисунок 4.2. Оптимізація процесів

Формуються визначальна документація така як: ***“Концепція системи”*** та ***“Технічне завдання”****, яке* ***є підставою для розробки інформаційної системи і подальшої приймання її в експлуатацію***. Вони визначають основні вимоги до самої системи та процесу її розробки і розробляється для системи в цілому. Додатково можуть розроблятись технічні завдання на окремі частини ІС.

Запитання для самоконтролю:

1. Які саме поняття життєвого циклу інформаційної системи.
2. Які нормативні документи, які регламентують ЖЦ інформаційних систем.
3. Назвіть основні процеси ЖЦ програмного забезпечення.
4. Які основні функції має виконувати інформаційна система.
5. Назвіть етапи формування вимог до інформаційних систем.
6. На чому базується аналіз проблем, цілей та визначення вимог до майбутньої системи?

Лекція №5.
Моделі життєвого циклу інформаційної системи

Модель життєвого циклу – це певна структура, що визначає послідовність та взаємозв’язки між процесами, що передбачені у межах ЖЦ інформаційної системи (рис. 5.1.).

Модель життєвого циклу ПЗ включає в себе:

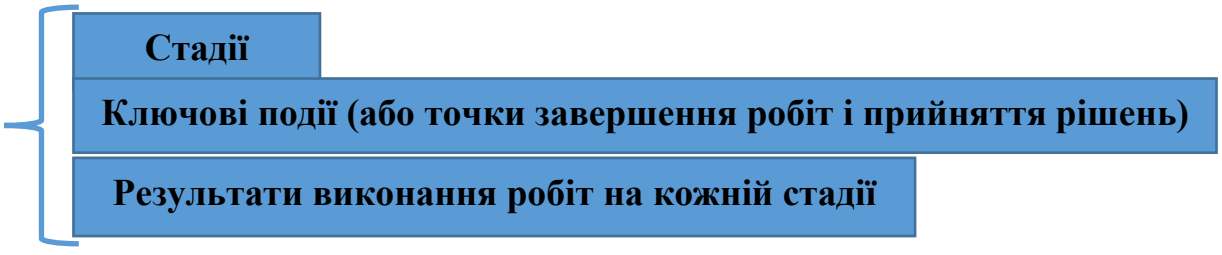

Рисунок 5.1. Структура моделі життєвого циклу ПЗ

Модель ЖЦ будь-якого конкретного ПЗ визначає характер процесу його створення, який має вигляд сукупності впорядкованих у часі, взаємопов'язаних та об'єднаних у стадії робіт, виконання яких необхідно і досить для створення ПЗ, що відповідає заданим вимогам.

Життєвий цикл складається з етапів, на кожному з яких породжується конкретний набір технічних рішень і документів, що відображають їх (при цьому для кожного етапу результативними є документи і рішення, прийняті на попередньому етапі).

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

Ключові положення і базові визначення моделі ЖЦ викладені в стандарті ISO/IEC 12207. Слід відмітити, що вказаний стандарт описує тільки структури процесів. У ньому немає деталізації методів і дій для вирішення завдань, що входять до процесів ЖЦ ІС. Водночас сама модель ЖЦ залежить від специфіки ІС та умов, де вона створюється і буде працювати.

На сьогоднішній день найбільшого поширення набули три головні моделі ЖЦ:

- каскадна (так звана «модель водоспаду» або «waterfall»);
- поетапна;
- спіральна**.**

5.1. Каскадна модель життєвого циклу інформаційної системи

Найбільш широко відомою і вживаною довгий час залишалася класична або так звана каскадна або водоспадна (waterfall) модель життєвого циклу. Вона була вперше чітко сформульована в стандартах міністерства оборони США (автор Уїнстон Ройс). Ця модель припускає послідовне виконання різних видів діяльності, починаючи з вироблення вимог і закінчуючи супроводом, з чітким визначенням меж між етапами, на яких набір документів, виробленою на попередній стадії, передається в якості вхідних даних для наступної. Дуже часто класичний життєвий цикл називають каскадною або водоспадною моделлю, підкреслюючи, що розробка розглядається як послідовність етапів, причому перехід на наступний, ієрархічно нижній етап відбувається тільки після повного завершення робіт на поточному етапі, тобто вся розробка розбивається на етапи, перехід до наступного етапу відбувається виключно за умови повного завершення всіх робіт попереднього етапу.

Кожен етап завершується оформленням повного комплекту документації. Склад і зміст цієї документації передбачає, що реалізація проекту може бути продовжена іншою командою розробників.

5.1.1. Головні етапи розроблення згідно з каскадною моделлю

Враховуючи досвід реальних розробок, у каскадній моделі прийнято виділяти декілька сталих етапів, що майже не залежать від предметної області, а саме (рис. 2.5.):

- аналіз вимог замовника;
- проєктування;
- розробка;
- тестування і дослідна експлуатація;
- здача/приймання готового продукту.

Рисунок 5.2. Структурна схема каскадної моделі розробки ІС

Аналіз. Результатом ***першого етапу*** є ТЗ (технічне завдання/завдання на розробку).

Етап аналізу передбачає докладне дослідження бізнес-процесів (вимог і функцій, визначених на етапі вибору стратегії) і інформації, необхідної для їх виконання (сутності, їх атрибутів і зв'язків (відносин)).

Вся інформація про систему, зібрана на етапі концептуального прийняття рішення, формалізується і уточнюється на етапі аналізу. Особливу увагу слід приділити повноті переданої інформації, аналізу інформації на предмет відсутності суперечностей, а також пошуку невживаною взагалі або інформації, що дублюється. Як правило, ***Замовник*** не відразу формує вимоги до системи в цілому, а формулює вимоги до окремих її компонентів, це природно, адже ***Замовник*** не є фахівцем в певних напрямках, що зумовлює поетапність формування вимог в процесі аналізу із залученням фахівців з предметної області. Тому необхідно приділити увагу узгодженості цих компонентів. Фактично ці вимоги визначають повне завдання на розробку. Воно має бути узгоджене з усіма сторонами (суб’єктами).

Проєктування (моделювання). Результат ***другого етапу*** – комплект проєктної документації (ПД), що містить необхідні дані для реалізації проєкту.

Головна мета проєктування полягає у відображенні функцій, отриманих на етапі аналізу, в модулі майбутньої інформаційної системи. Визначення модулів розкривається в технічній специфікації програм. При проєктуванні модулів визначають розмітку меню, вид вікон, гарячі клавіші і пов'язані з ними виклики.

Реалізація (розробка). Результат ***третього етапу*** – готовий програмний продукт.

На етапі розробки здійснюється тісна взаємодія проєктувальників, розробників і груп тестерів. У випадку інтенсивної розробки тестувальник фактично є членом групи розробників.

Проєктувальник на цьому етапі виконує функції «енциклопедичного фахівця», оскільки постійно відповідає на питання розробників, що стосуються технічної специфікації та здійснює авторський нагляд за реалізацією проєктних рішень. Найчастіше на етапі розробки міняються інтерфейси користувача. Це обумовлено у тому числі і тим, що модулі періодично демонструються Замовникові, тим самим визначається оптимальний front end користувача. Істотно можуть мінятися і запити до даних.

Взаємодія тестувальника і розробника без централізованої передачі частин (розділів, томів) проєктної документації допустима, але тільки у випадку, якщо необхідно терміново перевірити якусь зміну. Дуже часто етап розробки і етап тестування взаємозв'язані і йдуть паралельно.

Тестування. На ***четвертому етапі*** виявляють приховані недоліки, які виявились за умов експлуатації, а також вносять відповідні коригування до програмного забезпечення.

На цьому етапі можна використовувати складні та спрощені схеми тестування. Коли генерація модуля завершена, виконується автономний тест, який переслідує дві основні цілі:

- виявлення відмов модуля (жорстких збоїв);
- відповідність модуля специфікації (наявність всіх необхідних функцій, відсутність зайвих функцій).

Після того, як автономний тест пройшов успішно, група модулів, що згенерували, проходить тести зв'язків, які повинні відстежити взаємний вплив модулів

Впровадження (введення в дію). Головне завдання ***п'ятого етапу*** – переконати замовника у тому, що всі вимоги виконані повною мірою.

Етап класичного життєвого циклу націлений на дві дії: передача розробленого продукту Замовнику і супровід процесу подальшої експлуатації цього продукту.

Згідно із статистичними даними, 3/4 процесу супроводу пов’язано з удосконаленням ПЗ, залишковий темнім припадає на адаптацію та виправленням помилок в рівних частинах.

На практиці ЖЦ реальної системи може істотно відрізнятись, бути складнішим і довшим. Він може мати довільну кількість циклів уточнення, змін і доповнень вже реалізованих проєктних рішень, до того ж саме у таких циклах відбувається розвиток ІС й модернізація її компонентів.

На наступному етапі група модулів тестується на надійність роботи, проводяться тести імітації відмов системи та тести напрацювання на відмову.

Перша група тестів показує, наскільки добре система відновлюється після збоїв програмного забезпечення, відмов апаратного забезпечення.

Друга група тестів визначає ступінь стійкості системи при штатній роботі і дозволяє оцінити час безвідмовної роботи системи. У комплект тестів, в обов’язковому порядку, задля визначеності стійкості повинні входити тести, що імітують пікове навантаження на систему.

Експлуатація. Перехідний місток від процесу тестування, система вводиться в експлуатацію не повністю, а поступово.

Введення в експлуатацію проходить три фази:

- первинне завантаження інформації;
- накопичення інформації;
- вихід на проєктну потужність.

Акт готовності системи до експлуатації - документ, який підтверджує відповідність реалізованої системи проєктним рішенням та фіксує факт проведення попередніх етапів тестування та пусконалагоджувальних робіт, а також формалізує відсутність зауважень з боку відповідальних та зацікавлених осіб(сторін), щодо прийняття системи в експлуатацію.

5.1.2. Переваги каскадної моделі

Серед переваг каскадної моделі передусім можна вказати на такі:

- на кожному етапі формується повний та узгоджений набір ПД. На завершальних етапах також розробляється документація, що охоплює всі передбачені стандартами різновиди забезпечення ІС (організаційне, методичне, інформаційне, програмне, апаратне).
- чітка послідовність етапів дозволяє планувати терміни виконання робіт та відповідні витрати.

Каскадний підхід добре проявив себе під час розробки певного класу ІС. В першу чергу це системи, де з самого початку можна окреслити всі вимоги, що дає розробникам свободу щодо майбутніх шляхів її реалізації.

5.1.3. Недоліки каскадної моделі

Кількість недоліків каскадної моделі досить значна, а саме:

- повільність щодо отримання результатів;
- помилки на будь-якому етапі впливають на подальшу роботу;
- у межах каскадної моделі складно організувати паралельне ведення робіт;
- має місце певна інформаційна перенасиченість на всіх етапах проєкту;
- ускладнено управління проєктом;
- значний рівень ризику та ненадійність інвестицій.

Щоб зрозуміти область можливого використання каскадної моделі розглянемо вказані недоліки більш детально.

Затримка в отриманні результатів – це головний недолік каскадної моделі, що є наслідком послідовного підходу. Адже у цьому випадку узгодження результатів проводиться тільки після завершення чергового етапу робіт.

Крім того, узгодження результатів часто проводиться вибірково, після завершення кожного етапу, вимоги до ІС «заморожені» у вигляді технічного завдання на весь час її створення. Тому користувачі можуть надати свої пропозиції та зауваження тільки після завершення робіт над системою. У разі неточного формулювання вимог або їхньої зміни за період створення ПО користувачі отримують систему, що не задовольняє їхнім потребам.

До того ж задіяні на початок розробки моделі об’єктів можуть застаріти (внаслідок змін у законодавстві, в організаційній структурі об’єкта тощо). Це стосується всіх складових проєкту: функціональної, інформаційної моделей, інтерфейсу користувача й документації.

Складність паралельного виконання робіт. За каскадної моделі робота над проєктом будується як низка послідовних кроків. Навіть у тому разі, коли розробку окремих частин (підсистем) можна виконувати паралельно, зробити це у рамках каскадної схеми досить важко. Ці складнощі пов’язані з необхідністю постійного узгодження різних частин проєкту. Парадокс полягає у тому, що чим більш незалежними є частини, тим ретельніше має виконуватись синхронізація. А внаслідок цього – тим сильніше залежать одна від одної групи розробників.

Інформаційна перенасиченість. Цей недолік є наслідком суттєвої залежності між різними групами розробників. Проблема полягає в тому, що при внесенні змін до будь-якої частини проєкту треба повідомити про них всіх розробників, що пов’язані з цією частиною. Як наслідок – швидко зростають витрати на документування змін, опрацювання цих змін, погіршується оперативність узгоджень й т. і. А в цілому, – збільшується обсяг інформації, що є наслідком особливостей схеми управління проєктом, що має в певному розумінні «паразитну» природу. Питання перенасиченості різко загострюється за умов ротації груп розробників. Адже нові спеціалісти крім вивчення нового матеріалу

повинні опанувати велику кількість старої інформації, що пов’язана з історією змін, коригувань та узгоджень. Цей факт вкрай негативно впливає на ефективність проєктування.

Складність управління проєктом під час використання каскадної схеми здебільшого обумовлена строгою послідовністю стадій розроблення й наявністю складних взаємозв’язків між різними частинами проєкту. Послідовність розроблення проєкту призводить до того, що одні групи розробників повинні очікувати результати роботи інших команд. Отже, потрібно адміністративне втручання для узгодження термінів роботи та складу переданої документації.

Значний рівень ризику. Чим складнішим є проект, тим довшою буде тривалість кожного з його етапів, тим складнішими будуть взаємозв’язки між частинами проєкту. Внаслідок специфіки каскадної моделі така ситуація може призвести до великої кількості повернень до попередніх етапів після виявлення змін та їх погоджень.

Фактично це означає, що існує ймовірність значно збільшити терміни виконання проєкту, а таке збільшення (з фінансової точки зору) означає високий рівень ризику інвестицій.

Нарешті, занадто велика кількість змін (особливо в предметній області або у вимогах замовника) можуть взагалі призупинити проєкт, не довівши його до остаточної реалізації. Тому можна стверджувати, що складні проєкти, що розробляються за каскадною схемою, мають підвищений рівень ризику.

Незважаючи на наполегливі рекомендації компаній – експертів в області проєктування і розробки ПЗ, багато компаній продовжують використати каскадну модель на практиці замість якого-небудь варіанту ітераційної моделі.

Головні причини, по яких каскадна модель зберігає свою популярність: звичка, розробка невеликих проектів, ілюзія зниження ризиків учасників проєкту (замовника і виконавця), проблеми впровадження при використанні ітераційної моделі.

5.1.4. Область застосування

Каскадний підхід добре зарекомендував себе при розробці:

a) однорідних ІС, де кожен додаток становить єдине ціле;

b) ІС, для яких на самому початку розроблення можна досить точно й повно сформулювати всі вимоги, щоб надати розробникам свободу реалізувати їх якнайкраще з технічного погляду. До цієї категорії потрапляють ІС зі складними обчисленнями, системи, що працюють у реальному часі тощо.

5.2. Спіральна модель життєвого циклу

На відміну від каскадної спіральна модель ЖЦ передбачає ітераційний процес розробки ІС. До того ж у рамках цієї моделі зростає роль початкових етапів ЖЦ (аналізу та проектування), – саме на цих

етапах перевіряється та обґрунтовується реалізація технічних рішень через створення прототипів.

Ітерація – це основний елемент концепції спіральної моделі. Кожна ітерація є завершеним циклом розробки, що призводить до випуску діючої версії виробу (або певної його частини). В подальшому від ітерації до ітерації цей продукт вдосконалюється й наприкінці перетворюється у завершену систему (рис. 5.3.).

Рисунок 5.3.– Спіральна модель життєвого циклу

Таким чином кожне коло спіралі відповідає за створення частки або версії програмного продукту, на цьому ж колі уточнюються мета й характеристики проєкту, вимоги до якості, плануються роботи для наступного витку спіралі. На кожній ітерації послідовно конкретизуються деталі проєкту, внаслідок чого вибирається обґрунтований варіант. У подальшому цей варіант доводиться до остаточної реалізації. Використання спіральної моделі дає змогу здійснювати перехід до наступного етапу реалізації проєкту, не чекаючи повного завершення поточного етапу робіт, – їх можна буде виконати у наступній ітерації.

Головне завдання кожної ітерації – якнайшвидше отримати діючий продукт, який можна показати користувачам системи. Ця обставина робить набагато простішим процес внесення уточнень і доповнень до проєкту.

5.2.1. Переваги спіральної моделі

Спіральний підхід до розробки програмного забезпечення дозволяє подолати більшість недоліків каскадної моделі. Крім того, він забезпечує безліч додаткових можливостей, роблячи процес розробки гнучким. Переваги ітераційного підходу такі:

1. Ітераційна розробка полегшує коригування проєкту у разі змін у вимогах замовника.

2. У межах спіральної моделі окремі елементи ІС інтегруються до кінцевого продукту поступово. Тобто фактично інтеграція триває безперервно. Оскільки інтеграція починається з меншої кількості елементів, то виникає набагато менше проблем під час її проведення.

3. При реалізації проєкту за спіральною схемою зменшується рівень ризиків. Ця перевага є наслідком попередньої, оскільки ризики можна виявити вже на етапі інтеграції. Через це рівень ризиків є максимальним на початку розробки проєкту. З часом, по мірі просування розробки, очікуваний рівень ризиків зменшується.

4. Ітераційний характер організації робіт забезпечує гнучкість в управлінні проєктом. Це дає змогу вносити тактичні зміни в процесі виконання робіт на будь-якому етапі.

5. Ітераційний підхід краще пристосований до повторного використання компонентів. Це обумовлено тим, що простіше знайти загальні частини проєкту, коли вони вже частково розроблені, ніж намагатися відшукати їх на початковій стадії проєкту. Досвід свідчить, що аналіз проєкту вже після декількох ітерацій дозволяє виявити компоненти багаторазового використання, які на наступних ітераціях будуть удосконалюватися.

6. Спіральна модель дає змогу отримати більш надійну та сталу систему. Це є наслідком того, що з розвитком системи помилки та слабкі місця виявляються і коригуються на кожній ітерації.

7. Ітераційний підхід дає змогу удосконалювати процес розробки. Аналіз, проведений наприкінці кожної ітерації, дозволяє оцінити: що має бути змінено в організації розробки та як її поліпшити на наступній ітерації.

5.2.2. Недоліки спіральної моделі

Головна проблема спірального підходу – визначення моментів переходу між етапами.

Для її розв’язання необхідно вводити тимчасові обмеження на кожен із етапів ЖЦ. В іншому разі процес розробки може перетворитися в нескінченне удосконалення зробленого. Тому завершення ітерації має здійснюватися тільки відповідно до плану, навіть якщо не весь запланований обсяг робіт вже закінчено. Щодо самого плану, то він

складається на підставі статистичних даних з попередніх проєктів та особистого досвіду розробників.

5.3. Поетапна (ітераційна) модель з проміжним контролем. Ітераційний підхід до моделі життєвого циклу

Розвиток каскадної та спіральної моделей призвів до їхнього природнього зближення. Результатом такого зближення стала поява сучасного ітераційного підходу, який фактично становить раціональне поєднання цих двох моделей. Кожна лінійна послідовність створює інкремент (версію) ПЗ, що поставляється.

Основний недолік каскадного підходу – відсутність «гнучкості». Саме цей недолік нівелюється каскадно-зворотнім підходом, в якому дозволені повернення до попередніх стадій і перегляд або уточнення раніше прийнятих рішень. Каскадно-зворотній підхід реалізує ітераційний характер розробки ПЗ.

Тим самим розробка ІС ведеться ітераціями з циклами зворотного зв'язку між етапами (Рис. 5.4). Міжетапні коригування дозволяють враховувати реально існуючий взаємовплив результатів розробки на різних етапах; час життя кожного з етапів розтягується на весь період розробки.

В результаті на кожній ітерації можна аналізувати проміжні результати робіт і реакцію на них усіх зацікавлених осіб, включаючи користувачів, і вносити зміни, що коригують, на наступних ітераціях. Кожна ітерація може містити повний набір видів діяльності від аналізу вимог, до введення в експлуатацію чергової частини ПЗ.

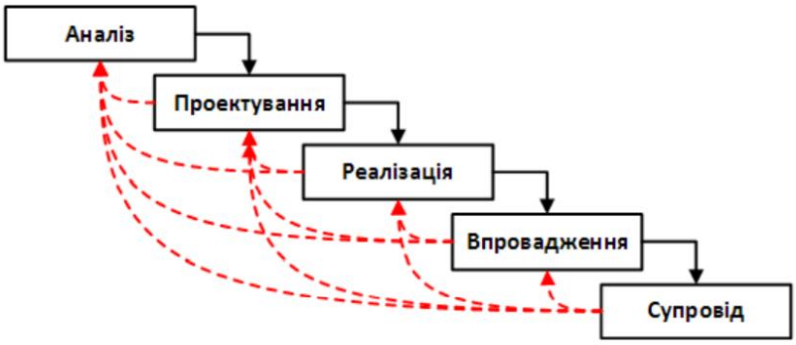

Рисунок 5.4. Поетапна модель життєвого циклу інформаційної системи

5.3.1. Переваги поетапної (ітераційної) моделі

1. Замовникові немає необхідності чекати повного завершення розробки системи, щоб отримати про неї представлення. Компоненти, отримані на перших кроках розробки, задовольняють найбільш критичним вимогам (оскільки зазвичай мають найбільший пріоритет) і їх можна оцінити на самій ранній стадії створення системи.

2. Замовник може використати компоненти, отримані на перших кроках розробки, як прототипи і провести з ними експерименти для уточнення вимог до тих компонент, які розроблятимуться пізніше.

3. Зниження ризиків за рахунок раннього виявлення конфліктів між вимогами, використовуваними моделями і реалізацією проєкту;

4. Динамічне формування вимог;

5. Гнучке управління вимогами;

6. Організація ефективного зворотного зв'язку команди розроблення з ***Замовником*** (створений ПП реально відповідає потребам споживача);

7. Швидкий випуск мінімально життєздатного продукту.

5.3.2. Основні недоліки поетапної (ітераційної) моделі

1. Проблеми з архітектурою і накладні витрати (при роботі з хаотичними вимогами і без чіткого глобального плану може постраждати архітектура ІС, для доопрацювання якої можуть знадобитися додаткові ресурси);

2. Відсутність фіксованого бюджету і термінів виконання;

3. Необхідне сильне залучення Замовника (кінцевих користувачів) у процес розроблення, що не завжди можливо чи зручно.

5.4. Гнучке розроблення програмного забезпечення Agile

Agile – це гнучкий підхід до організації роботи та система установок (Рис. 5.5.). Даний підхід зосереджений на особистій відповідальності, тісному контакті та командній роботі.

Гнучке розроблення за принципом Agile найчастіше використовується, щоб покращувати клієнтський досвід, швидше постачати продукт на ринок, скорочувати кількість нераціональних і неефективних дій та підвищити якість ПП в цілому. Особливо ефективним є застосування Agile при роботі в умовах високої невизначеності.

Являє собою сукупність різних підходів до розробки ПЗ. Включає серії підходів до розробки програмного забезпечення, орієнтованого на

використання ітеративної розробки (в Scrum ітерації називаються спринтами), динамічне формування вимог і забезпечення їхньої реалізації в результаті постійної взаємодії всередині самоорганізованих робочих груп, що складаються з фахівців різного профілю. Окрема ітерація являє собою мініатюрний програмний проект. Однією з основних ідей Agile є взаємодія всередині команди і з замовником напряму.

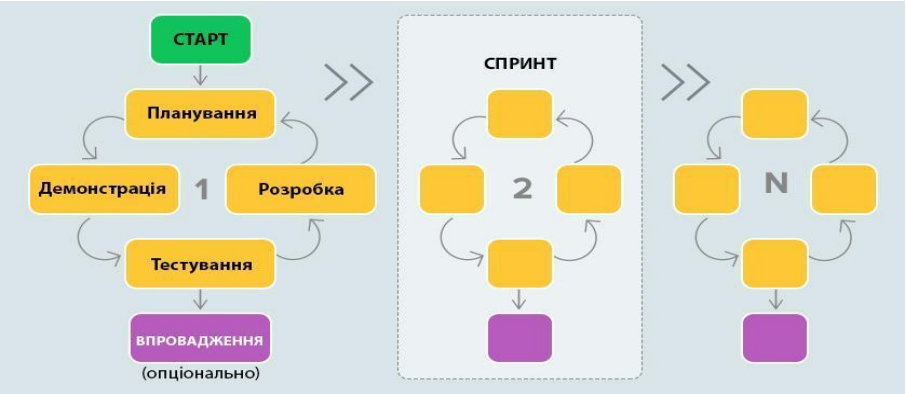

Рисунок 5.5. Модель гнучкого розроблення програмного забезпечення Agile

5.4.1. Переваги Agile

1. Люди і їх взаємодія є важливішими за процеси та інструменти;
2. Працююча ІС є важливішим за вичерпну документацію;
3. Співпраця з замовником є важливішою за узгодження умов контракту;
4. Готовність до змін є важливішою за дотримання початкового плану.

Принцип взаємодії визначає те, що команда взаємодіє з замовником і, навпаки, замовник з командою. Такий підхід дозволяє обмінюватися досвідом між замовником та учасниками команди розроблення, а також кожному з них брати участь у процесі прийняття рішень. Як наслідок, зниження ризиків втрати коштів і часу, а також підвищення здатності команди до вирішення нестандартних складних завдань з високим рівнем невизначеності.

5.4.2. Недоліки Agile

Випливають з його переваг:

1. Так взаємодія всіх з усіма може призвести до хаотичного процесу розроблення, що впливає на всі етапи.
2. Велика кількість мітингів і обговорень, що може збільшити час розробки продукту.

3. Складно планувати процеси, так як вимоги постійно змінюються.

4. Рідко використовується для реалізації великих проєктів.

Саме тому, використовуючи підходи Agile, потрібно зважати на певні обмеження:

a) команди розроблення повинні бути невеликими;

b) учасники повинні бути мотивованими та (що більш важливо) компетентними

c) має бути встановлено чіткі

a) обмеження за часом, кожна ітерація має бути короткою з максимально зрозумілими цілями, а кінцевий результат повинен бути очевидним Agile прогнозує результат на більш короткий період у порівнянні зі стандартними стратегіями розроблення, тому дуже добре вирішує проблему невизначеності.

Для підвищення ефективності необхідно використовувати наступне правило - чим вище невизначеність, тим коротшою має бути ітерація.

Запитання для самоконтролю:

1. Назвіть модель життєвого циклу.
2. Назвіть найбільш поширені моделі ЖЦ ІС.
2. Які головні етапи розроблення інформаційних систем.
3. Назвіть область застосування каскадної моделі ЖЦ ІС.
4. Які саме особливості застосування спіральної моделі ЖЦ?
5. Які основні недоліки спіральної моделі.
6. Назвіть переваги поетапної (ітераційної) моделі.
7. Назвіть недоліки Agile.

Лекція №6.
Технічне завдання. Технічний проєкт

6.1. Місце технічного завдання в життєвому циклі АС.

Стадія технічне завдання являється частиною життєвого циклу створення автоматизованих систем (АС), зокрема інтегрованих автоматизованих систем і регламентується вітчизняними ДСТУ та міждержавними ГОСТ стандартами. Згідно ДСТУ/ГОСТ створення автоматизованих систем складається з таких основних стадій:

1. Формування вимог до АС.
2. Розробка концепції АС.
3. Технічне завдання.
4. Ескізний проєкт.
5. Технічний проект (ПРОЄКТ)
6. Робоча документація.
7. Введення в дію.

8. Супровід.

Формування вимог до АС та створення технічного завдання відноситься до передпроєктних робіт, а технічний проєкт та робоча документація - до проєктних.

На кожній із стадій розробляється ряд документів, які регламентуються згідно з ДСТУ(ГОСТ 34.201-89) та рядом супутніх нормативних документів.

На стадії Технічне завдання розробляється однойменний документ, який регламентується ДСТУ(ГОСТ 34.201-89). Згідно з ДСТУ(ГОСТ 34.201-89), технічне завдання (ТЗ) на автоматизовану систему управління контролю, проєктування та інш. є основним документом, що визначає вимоги й порядок створення розвитку, модернізації автоматизованої системи, відповідно до якого проводиться її розробка й приймання при запровадженні в дію.

ТЗ на АС розробляють на систему в цілому, призначену для роботи самостійно або в складі іншої системи.

Проєкт ТЗ на АС розробляє розробник з участю Замовника на основі технічних вимог, вихідних даних.

При конкурсній організації робіт варіанти ТЗ розглядаються ***Замовником***, який або вибирає кращий варіант, або на основі порівняльного аналізу підготовлює із участю потенційного розробника АС кінцевий варіант ТЗ.

Технічне завдання (ТЗ) — документ, що встановлює основне призначення, показники якості, техніко-економічні та спеціальні вимоги до виробу, обсягу, стадії розроблення та складу конструкторської документації.

Завдання на проєктування – це документ, де розписують вимоги ***Замовника***, які він пред’являє до рішень і характеристик об’єкта, а також до його параметрів, вартості і заходам по зведенню. Розробляється документ на підставі технічних умов, ДБН і містобудівних умов.

Складає завдання Замовник (за участю проєктувальника – сертифікованого спеціаліста) і погоджує його з інвестором та проєктувальником. Узгодження здійснюється шляхом підписання та завірення печаткою підприємств.

Технічне завдання є підсумковим документом вихідних даних для подальшого проєктування споруди (архітектурного комплексу), конструювання технічного пристрою (приладу, машини тощо), розробки автоматизованої системи чи інформаційної системи, створення програмного продукту, проведення науково-дослідних робіт (НДР) і дослідно-конструкторських робіт (ДКР).

Технічне завдання на надання послуг встановлює основні вимоги до робіт в певній сфері діяльності, обов'язки замовника і виконавця, показники якості, терміни проведення і звітності та економічні

показники послуг. Цей документ використовують окремо або у складі договору (контракту) на виконання робіт.

Наявність ТЗ зумовлена розподілом праці між/на стадіях (етапах) життєвого_циклу виробу. Функції ТЗ може виконувати інший документ (договір, угода, контракт, протокол тощо), який містить необхідні та достатні вимоги для виконання роботи з відповідним об'єктом і визнаний сторонами такого документу.

6.2. Склад і зміст технічного завдання

Згідно з ДСТУ(ГОСТ 34.201-89), технічне завдання повинне складатися з наступних розділів:

1. Загальні відомості.

1.1. Повне найменування системи та її умовне позначення.

1.2. Шифр теми або шифр номер договору;

1.3. Найменування підприємств об’єднань розробника та замовника (користувача) системи та їх реквізити;

1.4. Перелік документів, на основі яких створюється система, ким і коли затверджені ці документи;

1.5 Планові терміни початку та закінчення роботи по створенню системи;

1.6. Відомості про джерела та порядок фінансування робіт;

1.7. Порядок оформлення та пред’явлення замовнику результатів робіт по створенню системи її частин),по виготовленню та наладці окремих засобів(технічних, програмних, інформаційних) та програмно-технічних комплексів системи.

2. Призначення та цілі створення розвитку системи.

2.1.Призначення системи.

2.2. Цілі створення системи.

3. Характеристика об’єкту автоматизації.

4. Вимоги до системи.

4.1. Вимоги до системи в цілому;

4.1.1. Вимоги до структури та функціонуванню системи;

4.1.2. Вимоги до чисельності та кваліфікації персоналу системи та режиму його роботи;

4.1.3. Показники призначення;

4.1.4. Вимоги до надійності;

4.1.5. Вимоги безпеки;

4.1.6. Вимоги до ергономіки та технічної естетики;

4.1.7. Вимоги до транспортабельності для пересувних АС;

4.1.8. Вимоги до експлуатації, технічному обслуговуванню, ремонту та збереженню компонентів системи;

4.1.9. Вимоги до захисту інформації від несанкціонованого доступу;

4.1.10. Вимоги до по збереженню інформації при аваріях;

4.1.11. Вимоги до захисту від впливу зовнішніх дій;

4.1.12. Вимоги до патентної чистоти

4.1.13. Вимоги по стандартизації та уніфікації;

4.1.14. Додаткові вимоги.

4.2. Вимоги до функцій(задач), що виконуються системою;

4.2.1. Перелік функцій, задач або їх комплексів в тому числі що забезпечують взаємодію частин системи; функціональних підсистем;

4.2.2. Часовий регламент реалізації кожної функції, задачі або комплексу задач;

4.2.3. Вимоги до якості реалізації кожної функції задачі або комплексу задач, до форми представлення вихідної інформації, характеристики необхідної точності та часу виконання, вимоги одночасності виконання групи функцій, достовірності видачі результатів;

4.2.4. Перелік та критерії відмов для кожної функції, по якій задаються вимоги надійності.

4.3. Вимоги до видів забезпечення.

4.3.1. Вимоги до математичного забезпечення;

4.3.2. Вимоги до інформаційного забезпечення;

4.3.3. Вимоги до лінгвістичного забезпечення;

4.3.4. Вимоги до програмного забезпечення;

4.3.5. Вимоги до технічного забезпечення;

4.3.6. Вимоги до метрологічного забезпечення;

4.3.7. Вимоги до організаційного забезпечення;

4.3.8. Вимоги до методичного забезпечення;

4.3.9. Вимоги до інших видів забезпечення.

5. Склад та зміст робіт по створенню (розвитку) системи.

6. Порядок контролю та прийому системи.

7. Вимоги до складу та змісту робіт по підготовці об'єкту автоматизації до вводу системи в дію.

8. Вимоги до документування.

9. Джерела розробки.

10. Додатки (при необхідності).

6.3. Ескізний проєкт та технічний проєкт ІС

Ескізний проєкт передбачає розробку попередніх проєктних рішень по системі.

Ескізний проєкт (технічну пропозицію) подають у вигляді проєктної документації, що описує архітектуру системи, структуру її підсистем, склад модулів. Тут же вказуються пропозиції на вибір базових програмно-апаратних засобів, які повинні враховувати прогноз розвитку підприємства.

Відносно апаратних засобів і особливо ПЗ такий вибір найчастіше є вибір фірми-постачальника необхідних засобів. У проєкті може бути запропоновано кілька варіантів вибору. При аналізі з'ясовуються можливості покриття функцій, які автоматизуються, наявними

програмними продуктами й, отже, обсяги робіт з розробки оригінального ПЗ. Такий аналіз необхідний для попередньої оцінки тимчасових і матеріальних витрат на автоматизацію. Облік ресурсних обмежень дозволяє уточнити досяжні масштаби автоматизації.

Виконання стадії ескізного проєктування не є строго обов'язковою, якщо основні проєктні 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 проєкту задається в ТЗ на систему.

Як правило, на етапі ескізного проєктування визначаються:

- функції ІС;
- функції підсистем, їх цілі та очікуваний ефект від впровадження;
- склад комплексів задач і окремих завдань;
- концепція інформаційної бази і її укрупнення структура;
- функції системи управління базою даних;
- склад обчислювальної системи і інш. 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 проєктних рішень і достатньому для подальшого виконання робіт зі створення системи.

На основі технічного завдання (і ескізного проєкту) розробляється технічний проєкт ІС.

Технічний проєкт системи (проєкт) - це технічна документація, яка містить загальносистемні проєктні рішення, алгоритми рішення задач, а також оцінку економічної ефективності автоматизованої системи управління і перелік заходів з підготовки об’єкта до впровадження (Таблиця 3.).

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

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 ії експлуатації, а також для підтримки рівня експлуатаційних характеристик (якості) системи.

Розроблена документація повинна бути відповідним чином оформлена, узгоджена і затверджена.

Таблиця 3

Зміст технічного проєкту

Зміст технічного проєкту	
Розділ	**Зміст**
Пояснювальна записка	• підстави для розробки системи; • перелік організацій розробників; • коротка характеристика об’єкта із зазначенням основних техніко-економічних показників його функціонування і зав’язків з іншими об’єктами; • короткі відомості про основні проєктні рішення.
Функціональна й організаційна структура системи	• обґрунтування виділяються підсистем, їх перелік та призначення; • перелік завдань, що вирішуються в кожній підсистемі, з короткою характеристикою їх змісту; • схема інформаційних зав’язків між підсистемами і між завданнями в рамках кожної підсистеми.
Постановка завдань і етапи вирішення	• організаційно-економічна сутність задачі (найменування, мету рішення, короткий зміст, метод, періодичність і час виконання завдання, способи збору і передачі даних, зв'язок задачі з іншими задачами, характер використання результатів рішення, в яких вони використовуються) • економіко-математична модель задачі (структурна і розгорнута форма подання) • вхідні оперативна інформація (характеристика показників, діапазон зміни, форми подання) • нормативно-довідкова інформація (НДІ) (зміст і форми подання) • інформація, що зберігається для зв'язку з іншими завданнями • інформація, що накопичується для наступних варіантів розв'язання задачі • інформація щодо внесення змін (система внесення змін і перелік інформації, яка підлягає змінам) • алгоритм вирішення задачі (послідовність етапів розрахунку, схема, розрахункові формули) • контрольний приклад (набір заповнених даними форм вхідних документів, умовні документи з накопичуваної і збереженої інформацією, форми вихідних документів)

Організація інформаційної бази	• джерела надходження інформації та способи її передачі • сукупність показників, що використовуються в системі • склад документів, терміни і періодичність їх надходження • основні проєктні рішення по організації фонду ДНІ • склад ДНІ, включаючи перелік реквізитів, їх визначення, діапазон зміни і перелік документів ДНІ • перелік масивів ДНІ, їх обсяг, порядок і частота
	• коригування інформації • структура фонду НДІ з описом зв'язку між його елементами; вимоги до технології створення і ведення фонду • методи зберігання, пошуку, внесення змін і контролю • визначення обсягів і потоків інформації НДІ • контрольний приклад по внесенню змін до НДІ • пропозиції щодо уніфікації документації
Система математичного забезпечення	• обґрунтування структури математичного забезпечення • обґрунтування вибору системи програмування • перелік стандартних програм
Принцип побудови комплексу технічних засобів	• опис і обґрунтування схеми технологічного процесу обробки даних • обґрунтування і вибір структури комплексу технічних засобів і його функціональних груп • обґрунтування вимог до розробки нестандартного обладнання • комплекс заходів щодо забезпечення надійності функціонування технічних засобів
Розрахунок економічної ефективності системи	• зведений кошторис витрат, пов'язаних з експлуатацією систем • розрахунок річної економічної ефективності, джерелами якої с оптимізація виробничої структури господарства (об’єднання), зниження собівартості продукції за рахунок раціонального використання виробничих ресурсів і зменшення втрат, поліпшення прийнятих управлінських рішень
Заходи з підготовки об’єкта до впровадження системи	• перелік організаційних заходів щодо вдосконалення бізнес- процесів • перелік робіт по впровадженню системи, які необхідно виконати на стадії робочого проєктування, із зазначенням термінів і відповідальних осіб

Запитання для самоконтролю:

1. Опишіть технічне завдання в життєвому циклі АС.
2. Хто із суб'єктів бере участь в розробці ТЗ?
3. Назвіть різницю між ЗП і ТЗ.
4. Який склад і зміст технічного завдання.
5. Охарактеризуйте технічний проєкт системи та його склад.
6. На якій стадії здійснюється створення програмного продукту і розробка всієї супровідної документації?

Лекція №7.
Сучасні методології проєктування інформаційних систем

7.1. Методологія RAD

На ранніх стадіях розвитку інформаційних технологій практично вся розробка ІС проводилось на традиційних мовах програмування. Згодом, внаслідок зростання складності систем, з появою розвинутого графічного інтерфейсу такий підхід втратив свою актуальність. Стала нагальною потреба у нових засобах, що спрямовані на скорочення термінів розробки та високий рівень автоматизації щодо ІС. Ця обставина стала причиною виникнення цілого напряму в сфері розроблення програмного забезпечення – створення інструментальних засобів для швидкого розроблення додатків і засобів автоматизації практично всіх етапів життєвого циклу інформаційних систем.

Багато з цих інструментів засновані на так званій методології RAD.

7.1.1. Головні особливості методології RAD

Методологія створення інформаційних систем, заснована на використанні засобів швидкого розроблення додатків або RAD (від англ. Rapid Application Development) охоплює всі етапи життєвого циклу сучасних інформаційних систем.

За своєю сутністю та змістом методологія RAD – це комплекс інструментальних засобів, що дозволяють оперувати з певним набором графічних об'єктів, які функціонально відображають окремі інформаційні компоненти додатків.

Методологія заснована на трьох головних елементах (рис. 7.1.)***:***

1) невелика команда програмістів (від 2 до 10 осіб). Команда розробників повинна складатися з групи професіоналів, які мають досвід в аналізі, проєктуванні, генерації коду і тестуванні ПЗ із використанням

CASE-засобів. Члени колективу мають також уміти трансформувати в робочі прототипи пропозиції кінцевих користувачів;

2) короткий, але ретельно підготовлений виробничий графік (від 2 до 6 місяців);

3) ітераційній моделі – коли розробники, працюючи над додатком, періодично уточнюють у замовника його вимоги та реалізують їх у конкретному продукті.

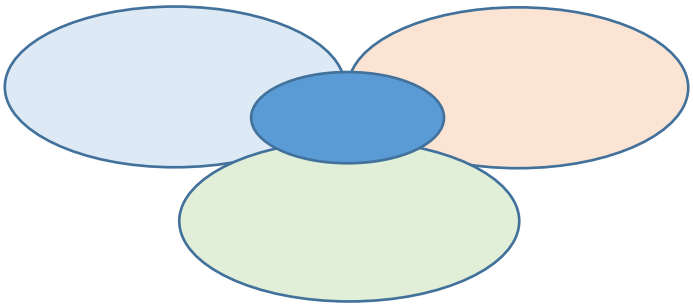

Рисунок 7.1. Цілі методології RAD

7.1.2. Головні засади методології RAD:

✓ ітераційна розробка програмного забезпечення;

✓ допустимість часткового завершення робіт на довільному етапі ЖЦ;

✓ обов'язкове залучення користувачів до процесу розробки ІС;

✓ обов'язкове використання CASE-засобів з метою забезпечення цілісності проєкту;

✓ застосування засобів управління конфігурацією, що робить простішим внесення змін до проекту, а також супровід готової системи;

✓ активне використання генераторів коду;

✓ впровадження прототипів задля кращого розуміння й задоволення потреб користувача;

✓ паралельне тестування, розробка й розвиток проєкту;

✓ ведення розроблення невеликою, але, добре організованою командою професіоналів (від 2 до 10 осіб);

✓ чітке планування та контроль за виконанням робіт.

7.1.3. Об'єктно-орієнтований підхід у методології RAD

Засоби RAD дозволили реалізувати відмінну від традиційної технологію створення додатків, коли інформаційні об'єкти формуються як певні діючі моделі (прототипи). Функціонування цих прототипів узгоджуються з користувачем після чого розробник переходить до формування закінчених додатків.

Поява такого підходу значною мірою є результатом застосування принципів об’єктно-орієнтованого проєктування. Ці принципи дають змогу подолати одну з головних проблем, що виникають під час розроблення складних систем – розрив між реальною предметною областю та середовищем, яке її імітує.

Використання об’єктно-орієнтованих принципів дозволяє створити модель предметної області у вигляді сукупності об’єктів, тобто сутностей, що поєднують дані та методи їхньої обробки у вигляді процедур. Кожен об’єкт має власну поведінку й моделює певний об’єкт реального світу. Із цього погляду об’єкт є досить універсальним і водночас має велику автономність програмному коду.

В об’єктному підході до проєктування акцент переходить до певних характеристик системи (фізичної чи абстрактної), що є предметом подальшого програмного моделювання. Самі об’єкти у цьому разі мають цілісність, яка не може бути порушена. Внаслідок цього властивості, що характеризують об’єкт і його поведінку, залишаються незмінними. Об’єкт може змінювати тільки стан, управлятися або мати певні відношення з іншими об’єктами.

Великої популярності об’єктно-орієнтоване програмування (ОПП) набрало з появою засобів візуального програмування, які реально забезпечили злиття (тобто інкапсуляцію) даних з процедурами поведінки об’єктів. Це зробило можливим створювати програмні системи, максимально наближені до реальних і досягти найвищого рівня абстракції.

Під час розроблення програм з використанням інструментів RAD використовується велика кількість готових об’єктів. Такі об’єкти зберігаються, як правило, у загальнодоступному сховищі. Водночас RAD-технологія надає також можливість створення нових об’єктів. Такі об’єкти можна писати як на основі існуючих, так і незалежно від них.

Інструментальним засобам RAD притаманний розвинутий, дружній графічний інтерфейс, що дозволяє розробляти прості програми практично без написання коду. Це є значною перевагою RAD, оскільки зменшує рутинну роботу щодо розробки інтерфейсів. Висока швидкість створення інтерфейсної частини дає змогу швидко отримувати прототипи та спрощує взаємодію з користувачами.

Інструменти RAD дають змогу розробникам сконцентруватися на сутності реальних процесів, для яких створюється ІС, що це призводить до підвищення якості кінцевого продукту.

7.1.4. Візуальне програмування і методологія RAD

Застосування принципів ООП дало змогу створити принципово нові інструменти проєктування додатків, що отримали назву «засобів

візуального програмування». Такі інструменти RAD дають змогу створювати складні інтерфейси користувача практично без написання коду програми. Тобто розробник може на будь-якому етапі бачити, що буде взято за підґрунтя майбутніх рішень.

Візуальні засоби оперують, насамперед, зі стандартними об'єктами інтерфейсу – вікнами, списками, текстами, які легко можна пов'язати з базами даних і відобразити інформацію на екрані монітора. Інша група об'єктів є стандартними елементами управління, наприклад кнопки, перемикачі, прапорці, меню, тощо. За допомогою таких елементів здійснюється управління програмою та даними, що відображаються. Усі ці об'єкти можуть бути представлені стандартним чином засобами певної мови програмування, а об'єкти – збережені для подальшого використання.

На сьогодні існує багато візуальних засобів розробки додатків.

Незважаючи на певну специфіку, всі вони діляться на дві категорії: ***універсальні та спеціалізовані***.

З **універсальних** систем візуального програмування найбільш поширеними є C, Java, Visual Basic і багато інших. Універсальними вони вважаються тому, що допомагають у розробці додатків практично будь-якого типу, зокрема – ІС. До того ж програми, які розробляються за допомогою універсальних систем, можуть взаємодіяти майже з усіма відомими системами управління базами даних.

Спеціалізовані засоби орієнтовані тільки на створення додатків до баз даних. Варто відзначити, що спеціалізовані засоби прив'язані здебільшого до конкретних систем управління базами даних. Як приклад таких систем можна вказати Power Builder фірми Sybase (орієнтований на роботу з СУБД Sybase Anywhere Server), Oracle від однойменної фірми, тощо.

Оскільки створення прототипів і розробка інтерфейсу багато в чому мають спільні риси, програміст отримав можливість мати постійний зворотній зв'язок із кінцевими користувачами, які можуть не тільки спостерігати за створенням програми, а й активно брати участь у цьому процесі, коригувати результати, висувати додаткові вимоги. Цей факт істотно сприяє скороченню термінів розроблення і є важливим чинником, що значно збільшує кількість прихильників технологій RAD.

7.1.5. Подієве програмування і методологія RAD

Логіка додатків, побудованих засобами RAD, є подієво - орієнтованою. Це означає, що кожен об'єкт зі складу програми, може створювати події та реагувати на події, що генеруються іншими об'єктами. Прикладами подій можуть бути відкриття і закриття вікон,

клацання на кнопці, натискання клавіші на клавіатурі, рух миші, зміни у базі даних, тощо.

Розробник реалізує логіку програми через визначення обробника для кожної події, тобто процедури, що виконується об'єктом у разі виникнення тієї чи іншої ситуації. Наприклад обробник події «клацання на кнопці» може відкрити діалогове вікно. Отже, управління об'єктами здійснюється за допомогою подій.

7.1.6. Життєвий цикл інформаційної системи за методологією RAD

Згідно методології RAD життєвий цикл ПЗ має чотири фази:

1) аналіз і планування вимог;

2) проєктування;

3) побудова;

4) впровадження

На фазі аналізу і планування вимог визначають функції системи, виділяють з них найбільш пріоритетні, описують інформаційні потреби.

Визначення вимог виконують зазвичай користувачі за допомогою фахівців-розробників. Тут же обмежують масштаб проєкту, визначають терміни реалізації для кожної з фаз. Крім того, визначають саму можливість реалізації проекту у межах доступного фінансування, існуючих апаратних засобах тощо.

Результатом цієї фази є:

1) список і пріоритетність функцій ІС;

2) попередні моделі ІС (функціональні та інформаційні) .

На фазі проєктування користувачі під керівництвом фахівців-розробників та за допомогою CASE-засобів беруть участь у створенні прототипу системи. Тут уточнюють і доповнюють вимоги, що не були враховані на попередній стадії, а саме:

- ✓ докладніше розглядають процеси системи. Аналізують і (в разі потреби) коригують функціональну модель. Кожен процес розглядають докладно. У разі потреби для кожного елементарного процесу створюють спеціальний прототип;
- ✓ визначають вимоги розмежування доступу до інформації;
- ✓ визначають набір необхідної документації;
- ✓ оцінюють кількість функціональних елементів ІС, і приймають рішення щодо декомпозиції ІС на підсистеми, які здатна реалізувати одна команда розробників за прийнятний для RAD-проектів час (60-90 днів);
- ✓ за допомогою CASE-засобів проєкт ділиться між різними командами (формується функціональна модель).

Результатом фази проєктування є:

- ✓ інформаційна модель ІС загального рівня;
- ✓ функціональні моделі ІС та кожної з її підсистем;
- ✓ інтерфейси між підсистемами, визначені за допомогою CASE-засобів;
- ✓ прототипи екранів, звітів, діалогів.

Усі моделі та прототипи мають бути отримані на тих CASE-засобах, які будуть використовуватись у подальшому під час створення системи. Ця вимога спричинена тим, що в традиційному підході під час передавання інформації щодо проєкту може відбутися неконтрольоване спотворення даних.

Застосування єдиного середовища зберігання інформації щодо проєкту дозволяє уникнути такої небезпеки.

На відміну від традиційного підходу, де використовуються специфічні засоби прототипування, у підході RAD кожен прототип розвивається до закінченої частини майбутньої системи. Отже, до наступної фази передається більш повна і актуальна інформація.

На фазі побудови виконується безпосередньо реалізація програми. На цій стадії розробники створюють реальну систему на підставі моделей та вимог нефункціонального значення, отриманих з попередніх етапів. Частина програмного коду формується за допомогою автоматичних генераторів. Кінцеві користувачі на цій фазі оцінюють проміжні результати та вносять коригування, якщо система на поточному етапі перестає задовольняти їх вимогам. Тестування системи роблять прямо в процесі розробки. По закінченню робіт команди розробників поступово інтегрують свої результати до загального. Таким чином формується повний програмний код ІС, після чого виконується тестування додатків, а з часом – тестування системи в цілому.

Завершення фізичного проєктування системи може виглядати так:

1) визначається необхідність розподілу даних;
2) проводиться аналіз використання даних;
3) робиться фізичне проєктування бази даних;
4) визначаються вимоги до апаратних ресурсів;
5) визначаються способи збільшення продуктивності;
6) завершується розроблення документації проєкту.

Результат фази побудови – готова система, що задовольняє заявленим вимогам.

На фазі впровадження проводиться навчання користувачів, організаційні зміни й паралельно з впровадженням нової системи здійснюється робота з існуючою системою (до повного впровадження нової). Оскільки фаза побудови досить коротка, планування і підготовка

до впровадження мають починатися заздалегідь, ще на етапі проєктування ІС.

Наведена схема розроблення ІС не є абсолютною. Можуть бути різні

варіанти. Це залежить, наприклад, від початкових умов, як то:

– розробляється зовсім нова система;

– вже зроблено обстеження підприємства та є модель його діяльності;

– на підприємстві є діюча ІС, яку:

а) можна використати як прототип;

б) треба інтегрувати до ІС, що з розробляється.

7.1.7. Обмеження методології RAD

Як і будь-яка методологія, RAD не може претендувати на універсальність. Її ефективно впроваджувати для виконання порівняно невеликих систем, що розробляються для певного підприємства.

Методологія RAD мало прийнятна:

1) для створення типових ІС, які не є завершеним програмним продуктом, а становлять сукупність типових елементів ІС. Для систем, де велике значення мають керованість та якість. Для типових проєктів, що супроводжуються централізовано і можуть бути адаптовані до різних програмно-апаратних платформ, СУБД, комунікаційних засобів. До систем, які мають інтегруватись до існуючих розробок, де потрібен високий рівень планування, жорстка дисципліна проєктування, суворе дотримання щодо розроблених протоколів, інтерфейсів;

2) для розробки програм з великим обсягом унікального коду(систем для наукових та інженерних розрахунків, операційних систем, програм для управління технічними об’єктами);

3) для розроблення додатків, що не мають інтерфейсу користувача або він є вторинним (наприклад, для створення додатків драйверів, службового ПО тощо);

4) для розробки ІС, від яких залежить безпека людей (наприклад, систем управління транспортом, військовою технікою, атомними електростанціями).

Це є наслідком того, що ітеративний підхід допускає на початкових етапах використання не повністю працездатних версій системи.

7.2. Методологія RUP

Серед виробників CASE-засобів компанія IBM Rational Software Corp. однією з перших усвідомила стратегічну перспективність розвитку об’єктно орієнтованих технологій аналізу та проєктування програмних систем. Саме ця компанія стала ініціатором уніфікації мови візуального

моделювання в межах консорціуму OMG, що призвело до появи перших версій UML. Саме вона вперше розробила об'єктно-орієнтований CASE-засіб, де було реалізовано мову UML як базову нотацію візуального моделювання. Унаслідок цього з'явилась одна з найпопулярніших технологій проєктування інформаційних систем – Rational Unified Process (RUP). На сьогодні ця методологія у певному розумінні стає міжнародним стандартом від компанії Rational Software, що входить до складу IBM. Авторами UML вважаються співробітники фірми Граді Буч, Айвар Якобсон, Джеймс Рамбо. RUP повністю відповідає стандартам, що визначають проєктні роботи в процесі життєвого циклу ІС.

7.2.2. Підходи RUP

У методології RUP реалізовано такі підходи (рис. 7.2.):

- ✓ ітераційний та інкрементальний (нарощуваний);
- ✓ побудова системи на базі архітектури інформаційної системи;
- ✓ планування і управління проєктом на підставі функціональних вимог до інформаційної системи.

Розроблення інформаційної системи проходить ітераціями. Це окремі

проєкти, що невеликі за обсягом та змістом, включають власні етапи аналізу вимог, проектування, реалізації, тестування, інтеграції.

Закінчуються ітерації створенням працюючої інформаційної підсистеми.

Ітераційний цикл характеризується періодичним зворотним зв'язком і може адаптуватися до ядра системи, що розробляється. Отже, інформаційна система поступово зростає та вдосконалюється.

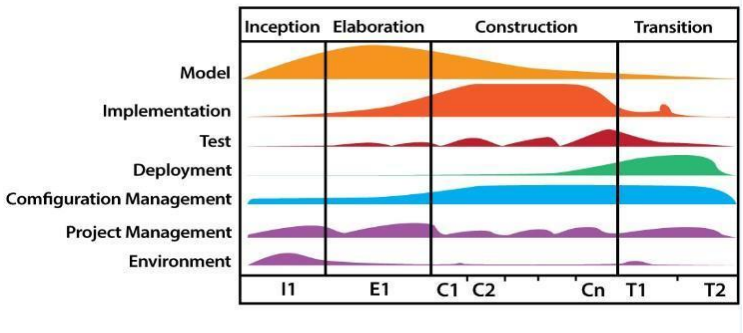

Рисунок 7.2. Методологія RUP

7.2.3. Чотири фази життєвого циклу проекту RUP.

RUP визначає життєвий цикл проєкту, що складається з чотирьох фаз. Ці фази дозволяють процесу, бути представленим на високому рівні, подібно до того як представляються проєкти у «водоспадному» стилі, хоча, по суті, ключем до процесу є ітерації розробки, які

простягаються вздовж всіх фаз. Крім того, кожен етап має одну ключову ціль, та віху в кінці, яка позначає досягнення цілі.

Inception - початкова фаза

Первинною ціллю є адекватна оцінка системи, як база для обчислення початкових розцінок та бюджету. На цьому етапі встановлюються бізнес випадки, які включають бізнес-контекст, фактори успіху (очікувані доходи, визнання на ринку, і т.д.), а також фінансовий прогноз. На додаток до бізнес випадку генерується базова модель прецедентів, план проєкту, попередня оцінка ризику і опис проєкту (основні вимоги до проєкту, обмеження та основні характеристики). Після їх завершення проєкт перевіряється на відповідність наступним критеріям:

- Зацікавлені сторони досягають згоди з визначення масштабів і оцінки вартості/термінів.
- Розуміння вимог як свідчення якості первинних прецедентів.
- Достовірність оцінок вартості/термінів, пріоритетів, ризиків, та процесу розробки.
- Глибина і ширина будь-якого архітектурного прототипу, який був розроблений.
- Встановлення базової лінії за допомогою якої можна порівняти фактичні витрати із запланованими витратами.

Якщо проєкт не пройде цей етап, що називається віхою життєвого циклу, він може бути як скасований так і повторений після переконструювання з метою кращого задоволення критеріїв.

Elaboration - фаза уточнення

Основна мета полягає в пом'якшенні ключових ризиків, виявлених на основі аналізу до кінця цієї фази. Фаза уточнення — фаза де проєкт починає набувати форми. На цьому етапі робиться аналіз предметної області, і архітектура проекту отримує свою базову форму.

Ця фаза має пройти віху життєвого циклу архітектури (LCA), задовольняючи такі критерії:

Модель прецедентів, в якій ідентифікуються прецеденти та актори, та розробляється більшість описів прецедентів. Модель прецедентів повинна бути завершена на 80%.

Опис архітектури програмного забезпечення в процесі розробки програмної системи.

Виконувана архітектура, яка реалізує архітектурно значимі прецеденти.

Бізнес — випадки та список ризиків переглядаються.

Прототипи, що явно зменшили кожен виявлений технічний ризик.

Якщо проєкт не може переступити цю віху, ще є час для того, щоб він був скасований або змінений. Тим не менше, після закінчення цього етапу, проєкт переходить в операцію з високим ступенем ризику, де зміни набагато складніші та згубні, при здійсненні.

Системна архітектура є ключовим елементом розробки, що отримується з аналізу предметної області.

Construction - фаза конструювання

Основна мета полягає в створенні програмної системи. На цьому етапі основна увага приділяється розробці компонентів та інших характеристик системи. Це етап, коли відбувається основна частина кодування. У більших проєктах може бути кілька фаз конструювання, в спробі поділити прецеденти на керовані сегменти, які можуть утворити презентабельні прототипи.

Цей етап створює перший реліз програмного забезпечення. Його завершення позначає віха початкової боєготовності.

Transition - фаза впровадження

Основна мета полягає в переведенні системи з розробки у продукт, зробивши її доступною та зрозумілою для кінцевого споживача. Діяльність у рамках цієї фази включає навчання кінцевих користувачів та обслуговуючого персоналу, бета-тестування системи для перевірки її на відповідність очікуванням користувачів. Продукт також перевіряється на відповідність рівню якості, встановленого в початковій фазі.

Якщо всі вимоги задоволені, досягається віха релізу продукту, і цикл розробки завершується.

7.3. Стандарти проєктування інформаційних систем. Методологія CDM

7.3.1. Стандарти та методики

Важливою умовою ефективного використання інформаційних технологій є впровадження корпоративних стандартів. Такі стандарти декларують угоду щодо єдиних правил організації технології та управління під час реалізації проєкту.

У цьому разі за основу корпоративних стандартів можуть прийматися галузеві, національні й навіть міжнародні стандарти. Останні можна умовно поділити на декілька груп.

За предметом стандартизації. До цієї групи можна віднести функціональні стандарти (на мови програмування, інтерфейси, протоколи) та стандарти щодо організації ЖЦ створення й використання ІС і ПЗ.

За організацією-розробником (затверджувачем). Серед них можна виділити офіційні міжнародні, офіційні національні або відомчі національні стандарти (наприклад ГОСТ, ANSI, IDEFO/1), стандарти міжнародних консорціумів та комітетів зі стандартизації тощо.

За методичним джерелом. Сюди належать різного роду методичні матеріали провідних фірм-розробників програмного забезпечення, фірм-консультантів, наукових центрів, консорціумів зі стандартизації тощо.

Представники з будь-якої групи можуть бути прийнятими за основу для розроблення внутрішніх стандартів. Тут принциповим є вимога до кінцевого результату, а саме: правильно побудовані корпоративні стандарти утворюють цілісну систему, яка включає три види стандартів:

– на продукти й послуги;

– на процеси та технології;

– на форми колективної діяльності, або управлінські стандарти.

Наявність та впровадження стандартів є необхідною умовою для забезпечення якісного проектування та реалізації системи. Однак цілком зрозуміло, що самих стандартів для вирішення цього питання недостатньо.

Для їхньої належної підтримки на рівні розробки необхідна певна методика. Однією з таких методик є CDM (від англ. Custom Development Method), запропонована компанією Oracle. Безумовною перевагою цієї методики є той факт, що вона орієнтована на розробку прикладних ІС відповідно до Міжнародного стандарту ISO/IEC 12207: 1995-08-01 01 та підтримує всі фази життєвого циклу відповідно до концепції вказаного стандарту.

7.3.2. Методика CDM фірми Oracle

Компанія Oracle добре відома на ринку програмного забезпечення як лідер із розроблення потужних СУБД та відповідних інструментів щодо проєктування баз даних. Однак цим сфера інтересів Oracle не обмежується, оскільки одним із важливих напрямів діяльності фірми є розроблення методологічних основ та інструментальних засобів для автоматизації процесів проєктування та реалізації складних інформаційних систем, що орієнтовані на активне використання баз даних. Методика CDM є розвитком давно розробленої методики CASE-Method фірми Oracle, яку реалізовано у CASE-засобі Oracle Designer/2000.

Розглянемо головні складові CASE-технології та інструментального середовища від фірми Oracle.

1) Проєктування має структурне значення, весь процес розроблення системи має вигляд послідовності чітко визначених етапів.

2) Підтримка здійснюється на всіх етапах ЖЦ системи, починаючи від загальних пропозицій і закінчуючи супроводженням готового продукту.

3) Перевага віддається архітектурі «клієнт-сервер», зокрема складним структурам розподілених баз даних.

4) Під час розроблення всі специфікації проєкту зберігаються в спеціальній базі даних (репозиторії). Цей засіб включено до складу ПО

Дизайнер і працює під управлінням СУБД Oracle. Репозиторій дає змогу підключатися до нього великому числу користувачів із різними рівнями прав доступу шляхом стандартних засобів СУБД Oracle.

Унаслідок цього всі дії розробників стають строго узгодженими та стають неможливими незалежні дії кожного з них.

5) Послідовний перехід від одного етапу до іншого автоматизований через використання спеціальних утиліт. За допомогою них за специфікаціями концептуальної стадії можна отримати початковий варіант специфікації рівня проєктування. Надалі генерація доповнень значно спрощується.

6) Стандартні дії етапів проєктування та розроблення автоматизовані. У будь-який момент може бути згенерований певний обсяг звітів за вмістом сховища, які забезпечують документування поточної версії системи на всіх етапах її розроблення. Існують спеціальні процедури, що дозволяють здійснити перевірку специфікацій на повноту й несуперечність.

7.3.3. Структура життєвого циклу згідно з методологією CDM

Методика CDM від Oracle пропонує свій варіант структури життєвого циклу інформаційної системи (рис. 7.3.):

1) формування стратегії. Цей етап передбачає моделювання та аналіз

процесів, які описують діяльність організації, особливості роботи. Кінцева мета – створення моделей процесів, виявлення можливостей їхнього вдосконалення. Етап не обов'язковий, якщо існуючі технології та організаційні структури чітко визначені, добре вивчені й загалом зрозумілі;

2) аналіз. На цьому етапі відбувається розроблення повних концептуальних моделей, які описують інформаційні потреби організації, особливості функціонування. Унаслідок цього формуються моделі двох типів: інформаційні (вони відображають структуру та загальні закономірності предметної області) і функціональні (описують особливості вирішуваних завдань);

3) проєктування. Цей етап передбачає перетворення початкових вимог до системи в докладні специфікації. Спеціальні утиліти, що входять до складу ПО Oracle, значно спрощують цю процедуру;

4) реалізація. На цьому етапі розробляються і тестуються програми, що входять до складу майбутньої ІС. ПО Oracle містить спеціальні генератори додатків, які гранично автоматизують цей етап, зазвичай тривалий і найскладніший. Терміни розробки значно знижуються;

5) впровадження. На цій стадії система встановлюється, підготовляється до початку експлуатації. Відбувається підготовка персоналу;

6) експлуатація. Підтримка системи, планування майбутніх доповнень і змін.

Варто відзначити, що **Oracle не розглядає таку стадію існування** і ПО Oracle, автоматично «підсаджується» на нього. Інша справа, що

зручність засобів розроблення багато в чому виправдовує таку залежність.

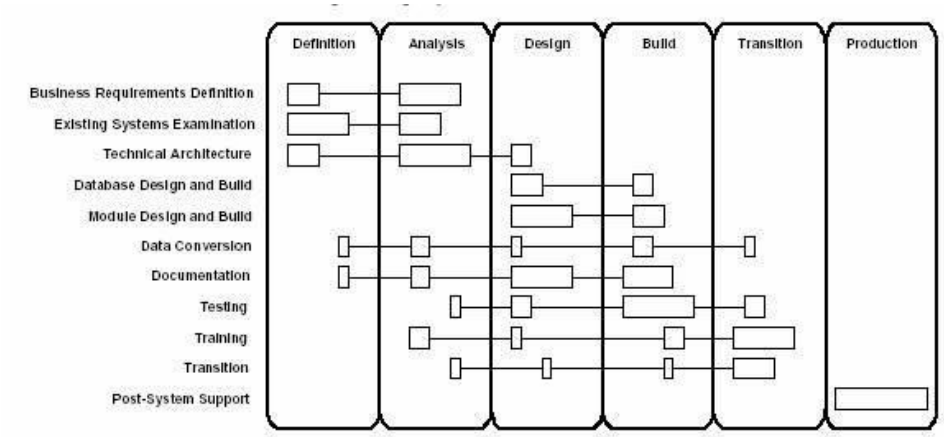

Рисунок 7.3. Методологія CDM

Методика CDM виділяє такі процеси, що мають перебіг протягом

ЖЦ ІС:

– визначення виробничих вимог;
– дослідження існуючих систем;
– визначення технічної архітектури;
– проєктування і побудова бази даних;
– проєктування і реалізація модулів;
– конвертування даних;
– документування;
– тестування;
– навчання;
– перехід до нової системи;
– підтримка й супроводження.

7.3.4. Особливості методики CDM

Серед чисельних особливостей методики головними є такі:

1) за ступенем адаптованості методика CDM пропонує три моделі життєвого циклу: класична – передбачає всі етапи, швидке розроблення – з використанням інструментів моделювання і програмування Oracle, полегшений підхід – для малих проєктів;

2) методикою не передбачено включення додаткових завдань, які не обумовлені в CDM. Зокрема не передбачена прив'язка їх до решти. Крім того, неможливо видалення завдання і відповідних до нього документів, якщо це не передбачено в жодній із трьох моделей життєвого циклу ІС, неможлива зміна послідовності виконання завдань й т. і.;

3) модель життєвого циклу ІС в Oracle CDM, по суті, є каскадною;

4) методика не є обов’язковою, хоча і є фірмовим стандартом, однак в разі використання ПЗ Oracle інший підхід малоймовірний;

5) методика спирається на використання ПЗ розробника від Oracle, пристосування її до інших засобів важко;

6) методика CDM становить певний матеріал, деталізований до рівня шаблонів проєктних документів. Відповідно ці документи розраховані на пряме використання у проектах інформаційних систем, що спираються на інструментальні засоби та СУБД фірми Oracle.

Запитання для самоконтролю:

1 Які сучасні методології проєктування інформаційних систем.

2. Назвіть головні засади методології RAD.

3. Які існують підходи методології RUP .

4. Сформуйте структуру життєвого циклу згідно з методологією CDM.

5. Які існують корпоративні стандарти щодо єдиних правил організації технології та управління під час реалізації проекту.

Лекція №8.
Уніфікована мова візуального моделювання
Unified Modeling Language

Існує безліч технологій і інструментальних засобів, за допомогою яких можна реалізувати в деякому сенсі оптимальний проєкт ІС, починаючи з етапу аналізу і закінчуючи створенням програмного коду системи. У більшості випадків ці технології пред'являють досить жорсткі вимоги до процесу розробки і використовуваних ресурсів, а спроби трансформувати їх під конкретні проєкти виявляються безуспішними. Ці технології представлені CASE-засобами верхнього рівня або CASE-засобами повного життєвого циклу (upper CASE tools або full life-cycle CASE tools). Вони не дозволяють оптимізувати діяльність на рівні окремих елементів проєкту, і, як наслідок, багато розробників перейшли на так звані CASE-засобу нижнього рівня (lower CASE tools). Однак вони зіткнулися з новою проблемою - проблемою організації взаємодії між різними командами, які реалізують проєкт.

Уніфікована мова об'єктно-орієнтованого моделювання Unified Modeling Language (UML) що є засобом досягнення компромісу між цими підходами. Існує достатня кількість інструментальних засобів, що підтримують за допомогою UML життєвий цикл інформаційних систем, і, одночасно, UML є досить гнучким для настройки і підтримки специфіки діяльності різних команд розробників.

Потужний поштовх до розробки цього напрямку інформаційних технологій дало поширення об'єктно-орієнтованих мов програмування в кінці 1980-х - початку 1990-х років. Користувачам хотілося отримати

єдину мову моделювання, який об'єднав би в собі всю міць об'єктно-орієнтованого підходу і давав би чітку модель системи, яка відображатиме всі її значимі сторони. До середини дев'яностих явними лідерами в цій галузі стали методи Booch (Grady Booch), OMT-2 (Jim Rumbaugh), OOSE - Object-Oriented Software Engineering (Ivar Jacobson). Однак ці три методи мали свої сильні і слабкі сторони: OOSE був кращим на стадії аналізу проблемної області та аналізу вимог до системи, OMT-2 був найкращий на стадіях аналізу і розробки інформаційних систем, Booch найкраще підходив для стадій дизайну і розробки.

Все йшло до створення єдиної мови, який об'єднував би сильні сторони відомих методів і забезпечував найкращу підтримку моделювання. Такою мовою виявився UML.

Створення UML почалося в жовтні 1994 р, коли Джим Рамбо і Граді Буч з Rational Software Corporation стали працювати над об'єднанням своїх методів OMT і Booch. Восени 1995 року побачила світ перша чорнова версія об'єднаної методології, яку вони назвали Unified Method 0.8. Після приєднання наприкінці 1995 р до Rational Software Corporation Айвара Якобсона і його фірми Objectory, зусилля трьох творців найбільш були об'єднані і спрямовані на створення UML.

В даний час консорціум користувачів UML Partners включає в себе представників таких грантів інформаційних технологій, як Rational Software, Microsoft, IBM, Hewlett-Packard, Oracle, DEC, Unisys, IntelliCorp, Platinum Technology.

UML є об'єктно-орієнтована мова моделювання, що володіє наступними основними характеристиками:

✓ є мовою візуального моделювання, який забезпечує розробку репрезентативних моделей для організації взаємодії замовника і розробника ІС, різних груп розробників ІС;

✓ містить механізми розширення і спеціалізації базових концепцій мови.

UML - це стандартна нотація візуального моделювання програмних систем, прийнята консорціумом Object Managing Group (OMG) восени 1997 року, і на сьогоднішній день вона підтримується багатьма об'єктно-орієнтованими CASE-продуктами.

UML включає внутрішній набір засобів моделювання (модулів «Ядро»), які зараз прийняті в багатьох методах і засобах моделювання. Ці концепції необхідні в більшості прикладних задач, хоча не кожна концепція необхідна в кожній частині кожної програми. Користувачам мови наданні можливості:

✓ будувати моделі на основі засобів ядра, без використання механізмів розширення для більшості типових програм;

✓ додавати при необхідності нові елементи і умовні позначення, якщо вони не входять в ядро, або спеціалізувати

компоненти, систему умовних позначень (нотацію) і обмеження для конкретних предметних областей.

8.1. Синтаксис і семантика основних об'єктів. UML Класи

Класи - це базові елементи будь-якої об'єктно-орієнтованої системи. Класи є опис сукупностей однорідних об'єктів з притаманними їм властивостями - атрибутами, операціями, відносинами і семантикою. В рамках моделі кожному класу привласнюється унікальне **ім'я**, що відрізняє його від інших класів. Якщо використовується складене ім'я (на початку імені додається ім'я пакета, до якого входить клас), ім'я класу має бути унікальним в пакеті.

Атрибут - це властивість класу, яка може приймати безліч значень. Безліч допустимих значень атрибута утворює домен. Атрибут має ім'я і відображає деяку властивість, моделюються сутності, спільні для всіх об'єктів даного класу. Клас може мати довільну кількість атрибутів.

Операція - реалізація функції, яку можна запитати у будь-якого об'єкта класу. Операція показує, що можна зробити з об'єктом. Виконання операції часто пов'язана з обробкою і зміною значень атрибутів об'єкта, а також зі зміною стану об'єкта.

Зображення класу в UML

Видимість властивості вказує на можливість її використання іншими класами. Один клас може «бачити» інший, якщо той перебуває в області дії першого і між ними існує явне або неявне відношення. У мові UML визначені три рівні видимості:

- ✓ public (загальний) - будь-який зовнішній клас, який «бачить» певний клас, може користуватися його загальними властивостями. Позначаються знаком « + » перед ім'ям атрибута або операції;
- ✓ protected (захищений) - тільки будь-який нащадок даного класу може користуватися його захисними властивостями. Позначаються знаком « # »;
- ✓ private (закритий) - тільки даний клас може користуватися цими властивостями. Позначаються символом « - ».

Ще однією важливою характеристикою атрибутів і операцій класів є область дії. **Область дії** властивості вказує, чи буде вона проявляти себе по-різному в кожному примірнику класу , або одне і те ж значення властивості буде спільно використовуватися всіма екземплярами:

- ✓ instance (екземпляр) - у кожного примірника класу є власне значення даної властивості;
- ✓ classifier (класифікатор) - всі екземпляри спільно використовують загальне значення даної властивості (виділяється на діаграмах підкреслюванням).

Можлива кількість примірників класу називається його кратністю. В UML можна визначати такі різновиди класів:

- ✓ що не містять жодного екземпляру - тоді клас стає службовим (Abstract);

- ✓ містять рівно один екземпляр (Singleton);
- ✓ містять задане число екземплярів;
- ✓ містять довільне число екземплярів.

Принципове призначення класів характеризують стереотипи. Це, фактично, класифікація об'єктів на високому рівні, що дозволяє визначити деякі основні властивості об'єкта (приклад стереотипу - клас «діюча особа»). Механізм стереотипів є також засобом розширення словника UML за рахунок створення на основі існуючих блоків мови нових блоків, специфічних для вирішення конкретної проблеми.

8.1.1. Діаграма класів

Класи в UML зображуються на діаграмах класів, які дозволяють описати систему в статичному стані - визначити типи об'єктів системи і різного роду статичні зв'язки між ними.

Класи відображають типи об'єктів системи.

Між класами можливі різні відносини.

- ✓ залежності, які описують існуючі між класами відносини використання;
- ✓ узагальнення, що зв'язують узагальнені класи зі спеціалізованими;
- ✓ асоціації, що відображають структурні відносини між об'єктами класів.

Залежністю називається відношення використання, згідно з яким зміна в специфікації одного елемента (наприклад, класу «товар») може вплинути на його елемент (клас «рядок замовлення»). Часто залежності показують, що один клас використовує інший як аргумент.

Узагальнення - це відношення між загальною сутністю (батьківським - клас «клієнт») і її конкретним втіленням (нащадком - класи «корпоративний клієнт» або «приватний клієнт»). Об'єкти класу-нащадку можуть використовуватися всюди, де зустрічаються об'єкти батьківського класу, але не навпаки. При цьому він успадковує властивості батька (його атрибути і операції). Операція, яку здійснює клас нащадка з тієї ж сигнатурою, що і у одного із батьківського класу, заміщує операцію з батьківського класу - це властивість називають поліморфізмом. Клас у якого немає батьків, але є нащадки, називається кореневим. Клас, у якого немає нащадків, називається абстрактним.

Асоціація - це відношення, яке показує, що об'єкти одного типу якимось чином пов'язані з об'єктами іншого типу («клієнт» може зробити «замовлення»). Якщо між двома класами встановлено зв'язок, то можна переміщатися від об'єктів одного класу до об'єктів іншого. При необхідності направлення навігації може здаватися стрілкою. Допускається завдання асоціацій на одному класі. В цьому випадку обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу можна зв'язати інші об'єкти з того ж класу. Асоціації може бути присвоєно ім'я, яке описує семантику відносин.

Кожна асоціація має дві ролі, які можуть бути відображені на діаграмі. Роль асоціації має властивість множинності, яке показує, скільки відповідних об'єктів може брати участь в цих питань.

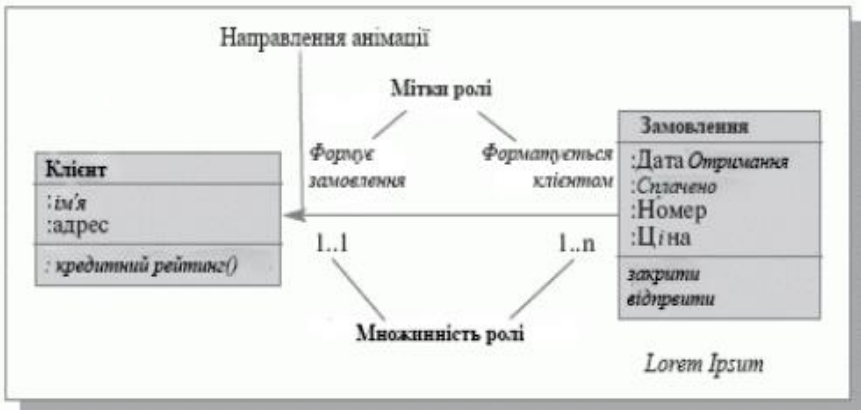

Рисунок 8.1. Модель формування замовлення

Малюнок ілюструє модель формування замовлення (рис. 8.1.). Кожне замовлення може бути створений єдиним клієнтом (множинність ролі 1.1). Кожен клієнт може створити один і більше замовлень (множинність ролі 1..n). Напрямок навігації показує, що кожне замовлення повинен бути « прив'язаний » до певного клієнта.

Такого роду асоціація є простою і відображає відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і жоден з них не є більш важливим, ніж інший. Якщо доводиться моделювати відношення типу «частина-ціле», то використовується спеціальний тип асоціації - агрегування. У такій асоціації один з класів має більш високий ранг (ціле - клас «замовлення ») і складається з декількох менших за рангом класів (частин - клас «рядок замовлення»). В UML використовується і сильніша різновид агрегації - композиція, в якій об'єкт-частина може належати тільки єдиному цілому. У композиції життєвий цикл частин і цілого збігаються, будь-яке видалення цілого обов'язково захоплює і його частини.

Для асоціацій можна задавати атрибути і операції, створюючи за звичайними правилами UML класи асоціацій.

8.1.2. Діаграми використання

Діаграми використання описують функціональність ІС, яка буде видана користувачам системи. «Кожна функціональність» зображується у вигляді «прецедентів використання» (use case) або просто прецедентів. Прецедент - це типове взаємодію користувача з системою, яка при цьому:

- ✓ описує видиму користувачем функцію,
- ✓ може представляти різні рівні деталізації,

✓ забезпечує досягнення конкретної мети, важливої для користувача.

Прецедент позначається на діаграмі овалом, пов'язаним з користувачами, яких прийнято називати діючими особами (акторами, actors). Дійові особи використовують систему (або використовуються системою) в даному прецеденті. Дійова особа виконує деяку роль в даному прецеденті. На діаграмі зображується тільки одна дійова особа, однак реальних користувачів, які виступають у цій ролі по відношенню до ІС, може бути багато. Список всіх прецедентів фактично визначає функціональні вимоги до ІС, які лежать в основі розробки технічного завдання на створення системи.

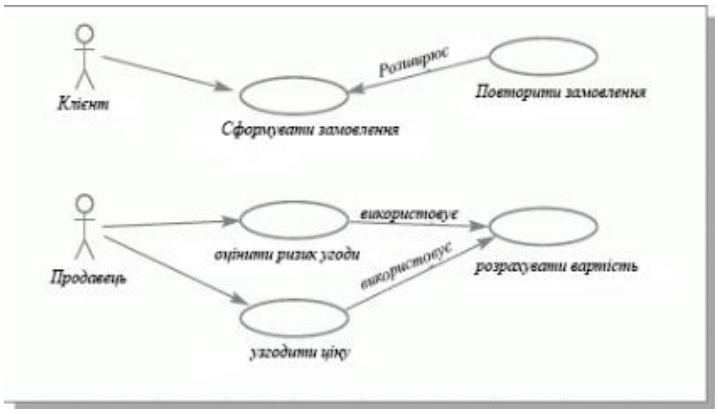

Рисунок 8.2. Діаграма прецедентів

На **діаграмах прецедентів** (рис. 8.2.), крім зв'язків між діючими особами і прецедентами, можливе використання ще двох видів зв'язків між прецедентами: «використання» і «розширення». Зв'язок типу «розширення» застосовується, коли один прецедент подібний до іншого, але має трохи більше функціональне навантаження. Її слід застосовувати при описі змін в нормальному поведінці системи. Зв'язок типу «використання» дозволяє виділити якийсь фрагмент поведінки системи і включати його в різні прецеденти без повторного опису.

Динамічні аспекти поводження системи відображаються наведеними нижче діаграмами.

На відміну від деяких підходів об'єктного моделювання, коли і стан, і поведінка системи відображаються на діаграмах класів, UML відокремлює опис поведінки **в діаграми взаємодії**. В UML діаграми класів не містять повідомлень, які ускладнюють їх читання. Потік повідомлень між об'єктами виноситься на діаграми взаємодії. Як правило, діаграма взаємодії охоплює поведінку об'єктів в рамках одного варіанту використання.

Прямокутники на діаграмі представляють різні об'єкти і ролі, які вони мають в системі, а лінії між класами відображають відносини (або

асоціації) між ними. Повідомлення позначаються ярликами біля стрілок, вони можуть мати нумерацію і показувати повернені значення.

Існують два види діаграм взаємодії : діаграми послідовностей і кооперативні діаграми.

8.1.3. Діаграми послідовностей

Цей вид діаграм використовується для точного визначення логіки сценарію виконання прецеденту. Діаграми послідовностей (рис. 8.3.) відображають типи об'єктів, що взаємодіють при виконанні прецедентів, надсилаючи повідомлення один одному, при цьому будь-які значення асоційовані з цими повідомленнями повертаються. Прямокутники на вертикальних лініях показують «час існування» об'єкта. Лінії зі стрілками і написами назв методів означають виклик методу в об'єкта.

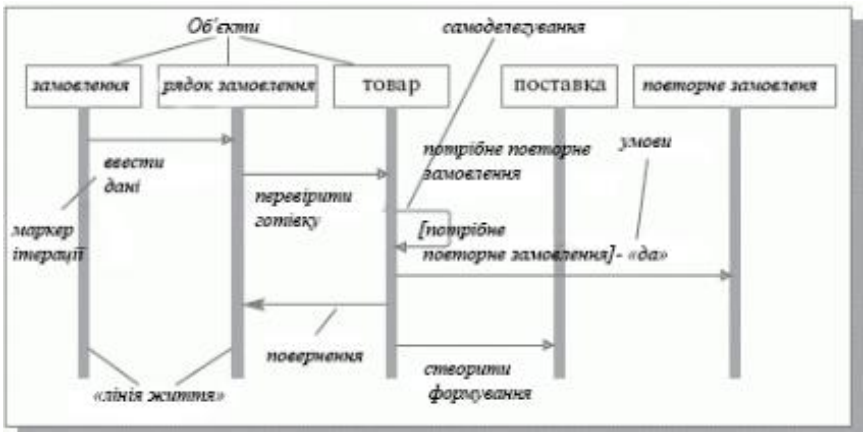

Рисунок 8.3. Діаграма послідовності обробки замовлення

Реалізація обробки замовлення засобами діаграм послідовностей:

- ✓ вводяться рядки замовлення;
- ✓ по кожному рядку перевіряється наявність товару;
- ✓ якщо запас достатній - ініціюється поставка;
- ✓ якщо запас недостатній - ініціюється дозамовлення (повторне замовлення).

Повідомлення з'являються в тій послідовності, як вони показані на діаграмі - зверху вниз. Якщо передбачається відправка повідомлення об'єктом самому собі (самоделегування), то стрілка починається і закінчується на одній «лінії існування».

На діаграми може бути додана керуюча інформація: опис умов, при яких надсилається повідомлення; ознака багаторазової відправки повідомлення (маркер ітерації); ознака повернення повідомлення.

Кооперативні діаграми (рис. 8.4.)

На кооперативних діаграмах об'єкти (або класи) показуються у вигляді прямокутників, а стрілками позначаються повідомлення, якими вони обмінюються в рамках одного варіанту використання. Тимчасова послідовність повідомлень відбивається їх нумерацією.

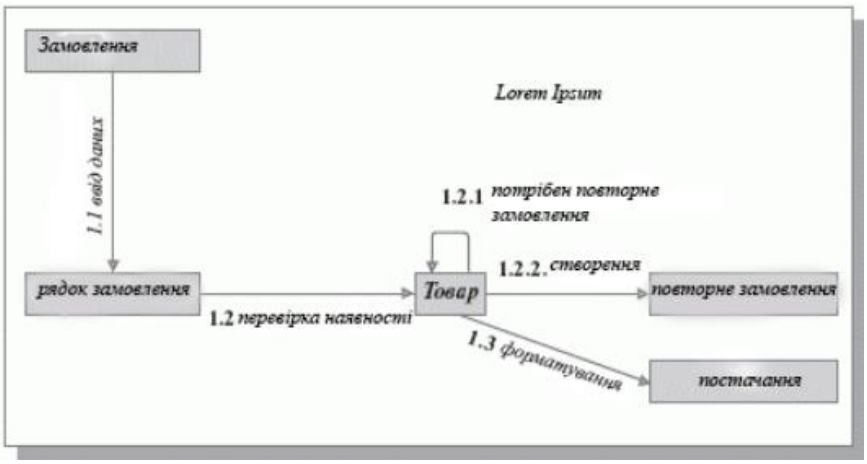

Рисунок 8.4. Кооперативна діаграма проходження замовлення

8.1.4 Діаграми станів

Діаграми станів використовуються для опису поведінки складних систем. Вони визначають всі можливі стани, в яких може перебувати об'єкт, а також процес зміни станів об'єкта в результаті деяких подій. Ці діаграми зазвичай використовуються для опису поведінки одного об'єкта в декількох прецедентах (рис. 8.5.).

Прямокутниками представляються стану, через які проходить об'єкт під час своєї поведінки. Станам відповідають певні значення атрибутів об'єктів. Стрілки являють переходи від одного стану до іншого, які викликаються виконанням деяких функцій об'єкта. Є також два види псевдо-станів: початковий стан, в якому знаходиться щойно створений об'єкт, і кінцевий стан, який об'єкт не покидає, за умови потрапляння до нього. Переходи мають мітки, які синтаксично складаються з трьох необов'язкових частин.

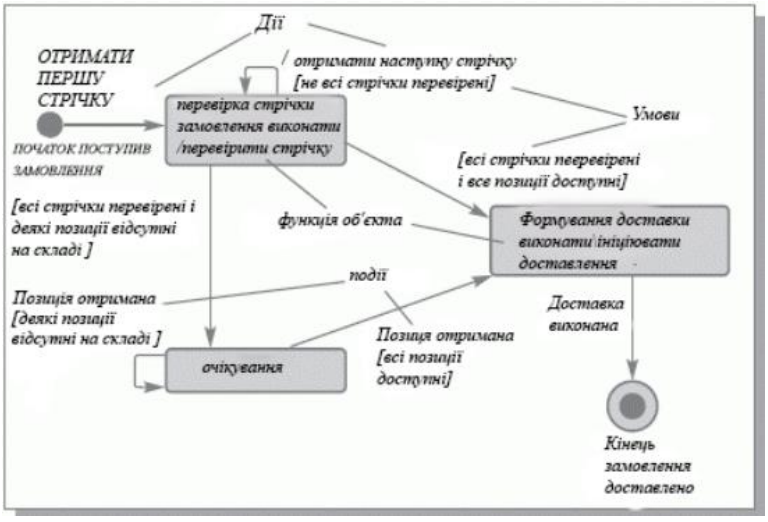

Рисунок 8.5. Діаграма станів об'єкта «замовлення»

8.1.5. Діаграми діяльності

Діаграма діяльності - це окремий випадок *діаграми станів.* На діаграмі діяльності (рис. 8.6.) представлені переходи потоку керування від однієї діяльності до іншої всередині системи. Цей вид діаграм зазвичай використовується для опису поведінки, що включає в себе безліч паралельних процесів. Основними елементами діаграм діяльності є:

- ✓ овали, що зображують дії об'єкта;
- ✓ лінійки синхронізації, що вказують на необхідність завершити або почати кілька дій (модель логічного умови «І»);
- ✓ ромби, що відображають прийняття рішень по вибору одного з маршрутів виконання процесу (модель логічного умови «АБО»);
- ✓ стрілки - відображають послідовність дій, можуть мати мітки умов.

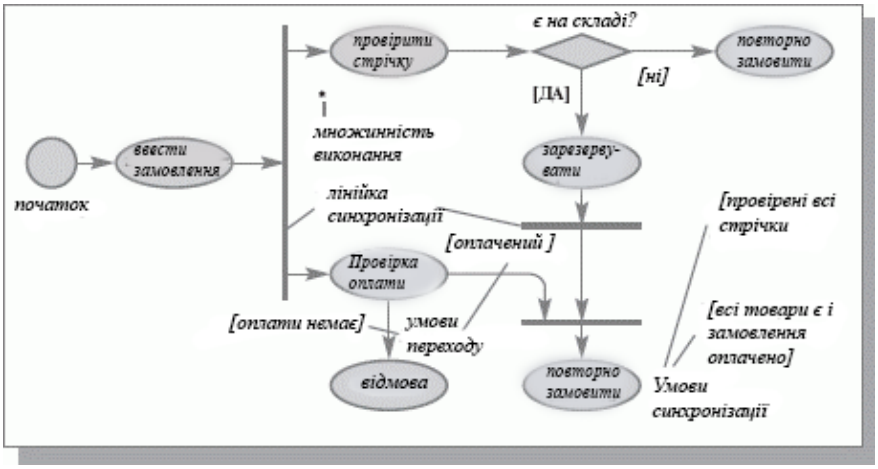

Рисунок 8.6. Діаграма діяльності - обробка замовлення

На діаграмі діяльності можуть бути представлені дії, відповідні кількома варіантами використання. На таких діаграмах з'являється безліч початкових точок, оскільки вони відображають тепер реакцію системи на безліч зовнішніх подій. Таким чином, діаграми діяльності дозволяють отримати повну картину поведінки системи і легко оцінювати вплив змін в окремих випадках використання на кінцеве поведінку системи.

Будь-яка діяльність може бути піддана подальшій декомпозиції і представлена у вигляді окремої діаграми діяльності або специфікації (словесного опису).

8.1.6. Діаграми компонентів

Діаграми компонентів дозволяють зобразити модель системи на фізичному рівні.

Елементами діаграми є компоненти - логічний блок системи, за рівнем абстракції трохи вищий, ніж «клас». Зображується

прямокутником з зображенням вкладки у правому верхньому куті. Кожен компонент є повністю незалежним елементом системи. Різновидом компонентів є вузли. ***Вузол*** - це елемент реальної (фізичної) системи, який існує під час функціонування програмного комплексу і являє собою обчислювальний ресурс, зазвичай володіє як мінімум деяким об'ємом пам'яті, а часто ще й здатністю обробки.

Вузли поділяються на два типи:

- ✓ пристрої - вузли системи, в яких дані не обробляються.
- ✓ процесори - вузли системи, що здійснюють обробку даних.

Для різних типів компонентів передбачені відповідні стереотипи в мові UML.

Компонентом може бути будь-який досить великий модульний об'єкт, такий як таблиця бази даних, підсистема, бінарний виконуваний файл, готова до використання система або додаток. Таким чином, діаграму компонентів можна розглядати як діаграму класів в більшому (менш детальному) масштабі.

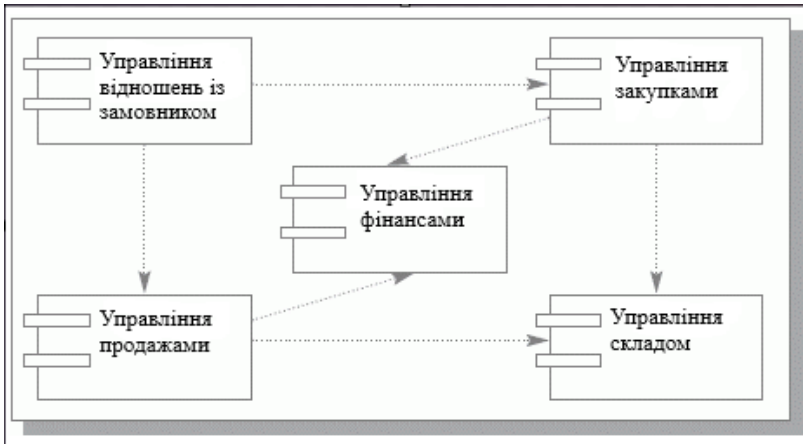

Рисунок 8.7. Діаграма компонентів фрагмента КІС

Компонент, як правило, являє собою фізичну упаковку логічних елементів, таких як класи , інтерфейси і кооперації.

Основне призначення діаграм компонентів - поділ системи на елементи, які мають стабільний інтерфейс і утворюють єдине ціле. Це дозволяє створити ядро системи, яка не буде змінюватися у відповідь на зміни, що відбуваються на рівні підсистем.

На рисунку (рис. 8.7.) показана спрощена схема елементів фрагмента корпоративної системи. «Коробки» представляють собою компоненти - додатки або внутрішні підсистеми. Пунктирні лінії відображають залежності між компонентами.

Кожен компонент діаграми при необхідності має документацію більш детальної діаграми компонентів, діаграми сценаріїв або діаграми класів.

Запитання для самоконтролю:

1. Назвіть синтаксис і семантику основних об'єктів UML.
2. Назвіть класи в UML, діаграми класів.
3. Назвіть види діаграм використання, та зазначте їх функціональність в ІС.
4. Чи можливий опис поведінки складної системи за допомогою UML діаграм?
5. Опишіть зображення моделі системи на фізичному рівні. Охарактеризуйте діаграми компонентів.

Лекція №9.
Дослідна експлуатація і введення в дію інформаційних систем

9.1. Етапи впровадження ІС

Безпосереднє впровадження ІС виконується в два етапи:

✓ дослідна експлуатація, після якої здійснюються приймальні випробування

✓ передача в постійну експлуатацію.

ІС мають впроваджуватися лише на діючому підприємстві.

Наказом замовника, погодженим із розробником, визначаються терміни проведення робіт і склад приймальної комісії. До наказу додають програму робіт і план-графік, у яких визначаються умови і кількість рішень задач, порядок перевірки технічних засобів при розв’язанні задач, порядок усунення недоліків, відмічених при дослідній експлуатації.

Мінімальна тривалість дослідної експлуатації залежно від потрібної частоти вирішення функціональних задач має відповідати таким термінам за ДСТУ (ГОСТ 24.104–85) АСУ «Загальні положення» (табл. 4.).

Таблиця 4

Тривалість дослідної експлуатації

Частота вирішення задачі	**Тривалість дослідної експлуатації**	**Допустима загальна тривалість порушень безперервності дослідної експлуатації**
Один раз на добу і більше	Один місяць	Не більше п’яти діб чи до 15% планової кількості рішень
Від одного разу на місяць до одного разу на добу	Три місяці	Не більше третини планової кількості рішень
Менше одного разу на місяць	Період між двома послідовними рішеннями	Порушень безперервності дослідної експлуатації не повинно бути

Дослідній експлуатації передує обов'язкове складання акту передачі ІС у дослідну експлуатацію.

Під час дослідної експлуатації ведуть журнал дослідної експлуатації, в який записують всі порушення й усі рішення, які прийняті виконавцями.

За результатами дослідної експлуатації складають протокол, у якому виноситься позитивне чи негативне рішення й усі недоліки з термінами їх усунення.

Якщо під час дослідної експлуатації виникли додаткові вимоги замовника, не передбачені в технічному завданні, то вони не є підставою для негативної оцінки результатів дослідної експлуатації і їх можна задовольнити, уклавши додатковий контракт і погодивши терміни.

Після позитивних результатів дослідної експлуатації провадять приймальні випробування, які водночас є здачею проекту в постійну експлуатацію.

Для кожного виду випробувань (попередніх, державних, сертифікаційних та ін.) комплексної системи (підсистеми, компонента) захисту виконавець розробляє "Програму і методику випробувань АС", яка затверджується в установленому порядку. Терміни подання проекту Програми, його розгляду і затвердження погоджуються з Замовником.

Наводиться необхідне для проведення випробувань забезпечення (необхідна нормативна, методична та інша документація, програмні та технічні засоби, метрологічне, спеціальне та інше обладнання, створення інших умов для проведення випробувань), сторона, що його надає, порядок усунення зауважень і т.ін.

Наводиться перелік документів, якими завершуються випробування (етапи випробувань): акт приймання, сертифікат (атестат, експертний висновок) відповідності встановленим критеріям, наказ про введення в експлуатацію тощо.

Приймальній комісії, яка провадить приймальні випробування, подають:

- технічне завдання на систему;
- технічну документацію на систему і на кожну її частину, яка передається в постійну експлуатацію;
- протокол і журнал дослідної експлуатації;
- штатний розклад підрозділів замовника, які обслуговуються ІС;
- акти передачі всіх частин ІС у постійну експлуатацію;
- проєкти програм і методик приймальних випробувань.

Приймальні випробування задач, які мають частоту вирішення один раз на добу і більше, повинні проводитись в режимі нормальної експлуатації, інші випробовуються на тестових прикладах.

Випробування передбачають:

- випробування кожної задачі;
- випробування всіх пред'явлених комісії задач у комплексі з задачами, що вже функціонують;
- випробування задач за наявності помилок в інформації;
- перевірку процедур внесення змін до баз даних чи в нормативно-довідкову інформацію.

За результатом приймальних випробувань складається протокол випробувань.

Показники надійності роботи ІС оцінюють за даними дослідної експлуатації і проведених випробувань. Усі виявлені недоліки та зауваження, а також терміни їх доопрацювання вказують у протоколі погодження. **Після внесення всіх змін складають акт передачі ІС у постійну експлуатацію і акт завершення робіт.**

Уся документація на ІС, а також програмне забезпечення (на машинних носіях) передаються замовнику не менш як у двох примірниках.

9.2. Супроводження і модернізація інформаційних систем

Після передачі ІС у постійну експлуатацію всю відповідальність за її функціонування несе замовник. Гарантійний термін встановлюється до 18 місяців з дня передачі у постійну експлуатацію.

Упродовж цього терміну розробник виконує авторський нагляд і усуває дефекти, виявлені в документації і програмному забезпеченні, за умови дотримання замовником умов експлуатації, викладених у робочій документації.

Роботи можуть виконуватись у два етапи.

1. Гарантійне обслуговування. Виконання робіт відповідно до гарантійних зобов'язань:

- роботи з усунення недоліків, виявлених при експлуатації ІС під час встановлених гарантійних термінів;
- внесення необхідних змін у документацію на ІС;
- внесення змін у програмне, технічне та інші види забезпечення ІС.

2. Післягарантійне обслуговування.

У рамках післягарантійного обслуговування обслуговуючою організацією здійснюється діагностика стану системи.

За результатами діагностики приймається рішення про обсяги необхідних робіт з ремонту і регулювання, переналаштування системи, а саме:

- складається дефектна відомість на складові/елементи, які підлягають заміні/виправленню;

- виготовляється, комплектується і поставляються необхідне устаткування;
- виконуються регулювальні і ремонтні роботи;
- відновлюється застаріла технічна документація.

Виконуються роботи:

- з аналізу функціонування ІС;
- виявлення відхилень фактичних експлуатаційних характеристик ІС від проєктних значень;
- встановлення причин цих відхилень;
- усунення виявлених недоліків і забезпечення стабільних експлуатаційних характеристик ІС;
- внесення необхідних змін у документацію на ІС чи розробка нової модифікації ІС.

Запитання для самоконтролю:

1. Назвіть етапи впровадження ІС.
2. Назвіть стадії дослідної експлуатації. Які суб’єкти задіяні в її проведенні? Яка документація фіналізує дослідну експлуатацію.
3. Опишіть тривалість дослідної експлуатації.
4. Назвіть етапи приймальних випробувань та здачі системи в постійну експлуатацію.
5. Назвіть особливості гарантійних зобов’язання та післягарантійного обслуговування ІС.

Список літератури

1. Задоров В.Б. Системний аналіз об’єктів і процесів: технологічні основи: Навчальний посібник. – Київ:КНУБА,2003. – 276с.
2. Шаховська Н. Б. Проектування інформаційних систем: навчальний. посібник /Н. Б. Шаховська, В. В. Литвин. – Львів: Магнолія-2006, 2011. – 380 с.
3. Пасічник В.В., Литвин В.В., Шаховська Н.Б. Проектування інформаційних систем. Навчальний посібник Львів: 2013.– 380 с
4. 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.
5. Розробка програмних проектів на основі Rational Unified Process(RUP) / Г. Поллис, Л. Огастин, К. Лоу, Дж. Мадхар. – Біном-Прес, 2011. – 256 с.
6. Керівництво з питань проектного менеджменту: Пер. З англ.. / Під ред. С.Д. Бушуєва, - 2-е вид., перероб. – К.: Видавничий дім «Ділова Україна», 2000. – 198 с.
7. Richards M. Fundamentals of Software Architecture: An Engineering Approach / M. Richards, N. Ford. – Sebastopol: O'Reilly Media, 2020. – 432 p.
8. Hoffer, J. A., George, J. F., & Valacich, J. S. Modern Systems Analysis and Design. 9th ed. Pearson. 2020. – p. 528
9. John Gallaugher Information Systems: A Manager's Guide to Harnessing Technology. 2019. – p. 664
10. Axelrod A. Complete Guide to Test Automation: Techniques, Practices, and Patterns for Building and Maintaining Effective Software Projects / A. Axelrod. – NY: Apress, 2018. – 588 p.
11. Sommerville, I. Software Engineering. 10th ed. Pearson. 2015. – p. 816

Інформаційні ресурси в мережі Інтернет

1. Метод інтеграції інформаційних систем при створенні нової техніки [Електронний ресурс] – режим доступу: http://www.irbis-nbuv.gov.ua/cgi-bin/irbis64r_81/cgiirbis_64.exe?C21COM=2&I21DBN=ARD&P21DBN=ARD&Z21ID=&IMAGE_FILE_DOWNLOAD=1&Image_file_name=DOC/2009/09kovsnt.zip
2. Основні етапи проведення реінжинірингу бізнес-процесів [Електронний ресурс] – режим доступу: https://library.if.ua/book/28/1899.html

Навчальне видання

САЧЕНКО Ілля Анатолійович

ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Конспект лекцій