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
С12

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

Затверджено на засіданні навчально-методичній ради КНУБА, протокол № 8 від 30 травня 2024 року.

Саченко І. А.

С12 Проєктування інформаційних систем: конспект лекцій / І.А. Саченко. – Київ: КНУБА, 2024. – 96 с.

Містить введення в предмет, огляд змісту курсу, постановку проблеми, пояснення ключових термінів, понять та методів, використовуваних 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

Зміст

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’єктноорієнтоване програмування
ООП – об’єктноорієнтований 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

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 набуття студентами базової профільної підготовки за фахом, формування у здобувачів освіти теоретичних знань та практичних навичок у галузі проєктування інформаційних систем із застосуванням сучасних методів та засобів створення інформаційних систем під час розроблення, налагодження, тестування та подальшої експлуатації. Здобуті знання з дисципліни повинні стати основою вивчення дисциплін професійно орієнтованого циклу.

Дисципліна фахово надає базову підготовку в галузі проєктування інформаційних систем, а саме ознайомлює з методами, принципами, технологіями, інструментальними засобами, стандартами та шаблонами проєктування. Дозволяє вирішувати завдання з аналізу та проєктування інформаційних систем, їхню модернізації (реновації) 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 їхній якісний аналіз та декомпозицію, застосовувати їх під час розроблення та впровадження систем.

За результатами вивчення дисципліни студент повинен опанувати:

- 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'єктноорієнтованого 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

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

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

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

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

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

Призначення, завдання, функції, класифікація ІС

Як система ІС має основні властивості 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 взаємодійних елементів, об'єднаних певними зв'язками, призначена для досягнення заданої мети найкращим (по змозі) чином.

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

1.1. Завдання і функції ІС

Перша група завдань пов'язана з суто інформаційним забезпеченням основної діяльності: відбір потрібних повідомлень, їхня 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) відповідно до тих чи інших алгоритмів або програмам з метою підготовки вирішення завдань, що стоять перед суб'єктом основної діяльності (так званих «користувацьких» завдань).

Завдання першої групи – це завдання інформатизації суспільства «вшир».

Завдання другої групи – завдання інформатизації суспільства «вглиб».

Для вирішення поставлених завдань ІС повинна виконати такі функції:

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

Обробка може означати також підготовку варіантів вирішення користувацьких прикладних задач за відповідними алгоритмами/ програмами.

1.2. Класифікація ІС

Інформаційні системи можна прокласифікувати за різними ознаками. Найбільш поширені класифікаційні ознаки та сучасні типи використовуваних систем наведено у табл. 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

Закінчення табл. 1

Ознаки класифікації	Типи систем
За сферою обслуговування (за характером обробки інформації)	-інформаційно-довідкові, -інформаційно вирішальні
За характером збереженої інформації (БД)	-фактографічні, -документальні
За способом організації	-на основі архітектури файлсервер, -на основі архітектури клієнт–сервер, -на основі багаторівневої архітектури, -на основі інтернет-технологій
За сферою застосування	-інтегровані (корпоративні), -АСУТП, -організаційного управління, -САПР

Класифікація систем за рівнем та сферою діяльності

Державні ІС призначені для вирішення найважливіших народногосподарських проблем країни. На основі використання обчислювальних комплексів та економіко-математичних методів у них складають перспективні та поточні плани розвитку країни, ведуть облік результатів та регулюють діяльність окремих ланок 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, потрібної для виконання 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 тощо. У них розв'язують задачі з інформаційного обслуговування апарату управління галузевих міністерств і їхніх 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 з 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 керування – машини, прилади, обладнання, а для державних, територіальних та інших систем – це колективи людей.

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 стислу інформацію про літературу та місце її зберігання і сховище, де літературу розміщено на вказаних в каталозі місцях, але пошук літератури, її доставку виконують вручну.

В механізованих системах інформаційну проблему розв’язують за допомогою механічних пристроїв.

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

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

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. Центральна функціональна ланка фактографічної інформаційної системи – СУБД.

Фактографічні інформаційні системи використовують не тільки для виконання довідкових функцій, а й для вирішення задач з обробки даних. Під обробкою даних розуміють 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, реквізитів та ін., адекватних його інформаційним потребам. Тому можна дати таке визначення документальної ІС – це єдине сховище документів з інструментарієм пошуку і відбору необхідних документів.

Класифікація систем за масштабом

За масштабом інформаційні системи поділяються на 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 застосунків, пов'язаних загальним інформаційним фондом, і призначена для роботи одного користувача або групи користувачів, які поділяють за часом одне робоче місце. Подібні застосунки створюють за допомогою так званих настільних або локальних систем керування 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, найчастіше будують на базі локальної обчислювальної мережі. Для розроблення таких застосунків використовують сервери баз даних (звані також SQL-серверами) для робочих груп. Створено досить велику 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 сервера і під час його сервісного обслуговування. Часто це роблять 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 відбувається за допомогою передавання 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 результатом розвитку систем для робочих груп, вони орієнтовані на великі компанії і можуть підтримувати територіально окремо розміщені вузли або мережі. Вони мають переважно ієрархічну структуру з декількох рівнів. Для таких систем характерна архітектура «клієнт–сервер» зі спеціалізацією серверів або багаторівнева архітектура. Для розроблення таких систем можуть використовуватися ті самі сервери баз даних, що й для розроблення 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.

Класифікація систем за сферою обслуговування (за характером обробки інформації)

Інформаційно-довідкові системи виконують 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 основано на гіпертекстових документах і мультимедіа. Найбільшого розвитку такі інформаційні системи набули в мережі Інтернет.

Інформаційно-вирішувальні системи виконують, крім того, операції переробки інформації за певним алгоритмом. За характером використання вихідної інформації такі системи поділяють на керівні і ті, що радять.

Класифікація систем залежно від сфери застосування

- Автоматизовані системи управління (АСУ)
- Автоматизовані системи організаційного управління
- Автоматизовані системи управління технологічними процесами
- Системи підтримки ухвалення 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 ефективного функціонування керованого об'єкта (системи) шляхом автоматизованого виконання заданих функцій. Ступінь автоматизації функцій управління визначається виробничою потрібно 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 такі: оперативний контроль і регулювання, оперативний облік й аналіз, перспективне й оперативне планування, бухгалтерський облік, управління збутом, постачанням та інші економічні й організаційні завдання.

АС управління технологічними процесами (ТП) слугують для автоматизації функцій виробничого персоналу з контролю та керування виробничими операціями. Такі системи зазвичай характеризуються наявністю розвинених засобів вимірювання параметрів технологічних процесів (температури, тиску, хімічного складу тощо), процедур контролю допустимості значень параметрів і регулювання технологічних.

Системи підтримки ухвалення рішень (СППР) – це інтерактивна комп'ютерна система, призначена для підтримки різних

видів діяльності під час ухвалення рішень із слабо структурованих або неструктурованих проблем. Системи підтримки ухвалення рішень використовуються для управління об’єктами в слабкоструктурованих предметних сферах.

Інтерес до СППР як перспективної галузі використання обчислювальної техніки та інструментарію підвищення ефективності праці у сфері управління економікою постійно зростає. У багатьох країнах розроблення та використання СППР перетворилися на сферу бізнесу, що швидко розвивається. Третє покоління ІС створюється 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.

Згадані системи орієнтовані не на автоматизацію функцій особи, яка ухвалює рішення (ОПР), а на сприяння в пошуку ефективного рішення. СППР орієнтовані передусім на розв'язання слабкоформалізованих задач з 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 System) – являють собою інший тип інформаційних систем, в яких за допомогою досить складних запитів виконується відбір й 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 із застосуванням складаних методів і відповідних прикладних програм;

3. Експертні системи, які дають змогу впровадити ефективну комп'ютеризацію тих галузей, 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 під час створення нової техніки або технології.

Система автоматизованого проєктування́ (САП або САПР), або автоматизована система́ проєктування (АСП), – це 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. Застосовується на основі спеціального програмного забезпечення, автоматизованих банків даних, широкого набору периферійних пристроїв.

САПР виконує такі функції:

- конструкторська частина – розроблення повного комплекту конструкторської документації;
- технологічна частина – розрахунок і проєктування технологічних схем, технологічного оснащення, транспорту;
- архітектурно-будівельна частина – розрахунок і проєктування металевих і залізобетонних конструкцій;
- санітарно-технічні системи – проєктування теплопостачання, опалення і вентиляції виробничих й адміністративних корпусів, а також водопостачання і каналізації;
- електротехнічні системи – розрахунок і проєктування електропостачання, електросилового устаткування, світлотехнічної частини проєктів, телемеханізації електропостачання;
- гідротехнічні споруди – розрахунок і проєктування напірного і безнапірного гідротранспорту відвальних хвостів, стійкості укосів;
- системи автоматизації – розроблення 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 й охоплюють весь цикл робіт від планування діяльності до збуту продукції. Вони містять ряд модулів

(підсистем), що працюють в єдиному інформаційному просторі і виконують функції підтримки відповідних напрямів діяльності.

Запитання для самоконтролю

1. Дайте визначення терміна «проєктування».
2. Охарактеризуйте поняття про інформаційну систему.
2. Наведіть основні класифікаційні ознаки інформаційних систем.
3. Дайте визначення терміна «система».
4. Які основні функції має виконувати інформаційна система?
5. У чому полягає 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 постачає інформацію для ухвалення рішень. У ній реалізовано управлінську ідеологію, 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).

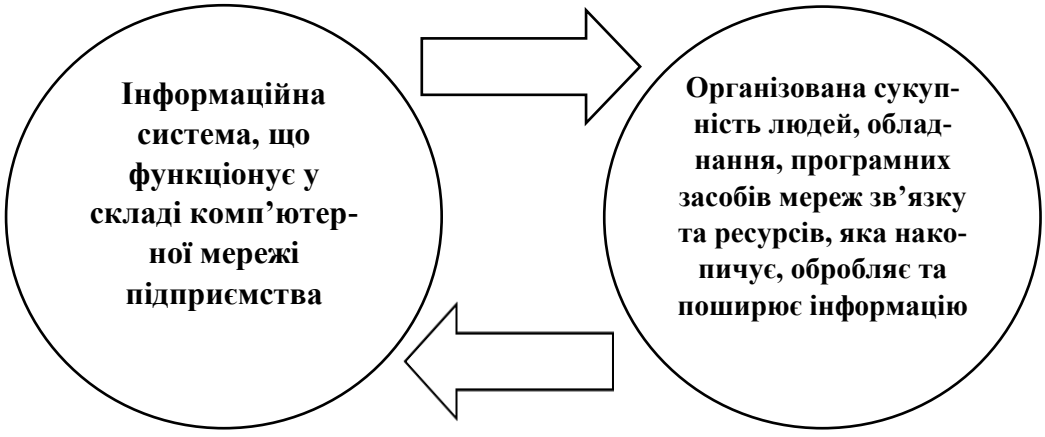

Рис. 2.1. Взаємозв'язок КІС з виробництвом

2.1. Основні характеристики сучасних КІС

Масштабність. Це одна із важливих характеристик інформаційних систем такого класу, зважаючи на масштаби діяльності корпорації. Масштабна ІС повинна функціонувати на масштабній програмно-апаратній платформі (сервери, операційні системи, системи комунікації, СУБД), що потребує значних зусиль фахівців з проєктування 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, з огляду на 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 розроблення (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, 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 досягнута взаємодія всіх робочих обчислювальних платформ і використовуваних операційних систем.

• **Розподілені обчислення**. Це один з 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.

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 продажем, виробництвом, обліком замовлень.

Створюється ERP-система за модульним принципом і, як правило, містить модуль безпеки для запобігання внутрішнім і зовнішнім крадіжкам інформації. Проблеми виникають переважно через неправильність роботи або початкову побудову плану впровадження системи.

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

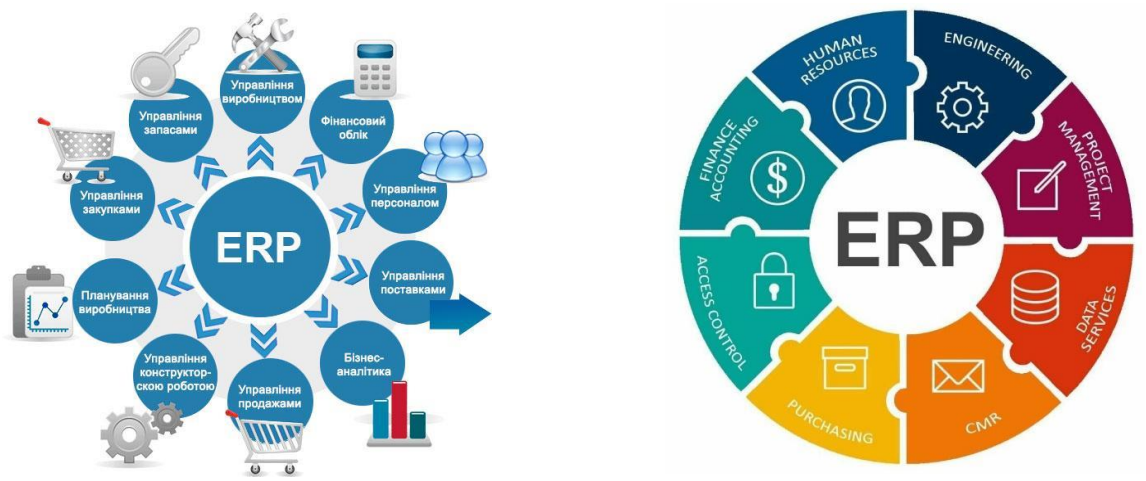

Рис. 2.2. ERP-системи та їхні модулі

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

Управління відносинами з клієнтами – поняття, що охоплює концепції, котрі компанії використовують для управління їхніми взаємовідносинами зі споживачами, зокрема такі, як збір, зберігання й аналіз інформації про споживачів, постачальників, партнерів та інформації про взаємовідносини з ними (рис. 2.3).

Рис. 2.3.Системи CRM

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

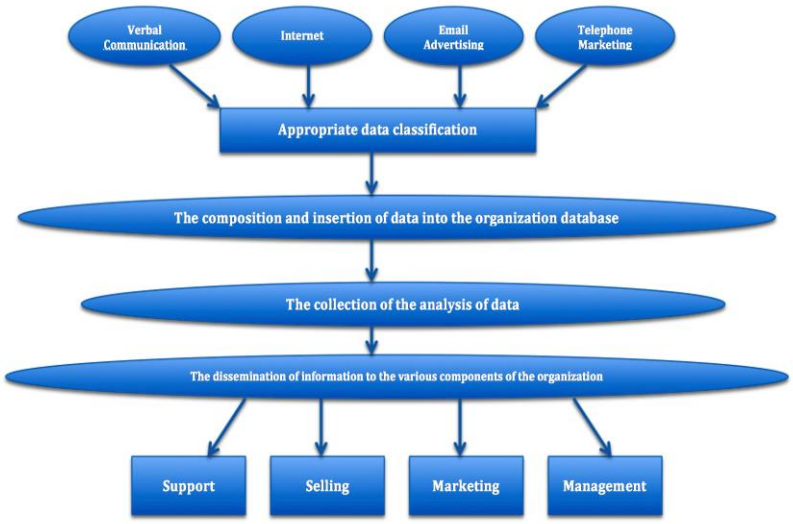

Рис. 2.4. Напрями CRM вивчення ринку

Принципи CRM-систем

✓ Наявність єдиного сховища інформації, завдяки якому 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 ухвалення відповідних організаційних рішень, наприклад, «сортування» клієнтів на основі їхньої значущості 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 потрібних звітних даних та аналітичної інформації.

✓ Зниження операційних витрат менеджерів.

✓ Контроль роботи менеджерів.

✓ Узгоджена взаємодія між співробітниками і підрозділами.

✓ Управління бізнес-процесами дає змогу автоматизувати послідовні операції, які виконують співробітники 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 їхніх співробітників тощо.

Системи MES (Manufacturing Execution System) – спеціалізоване прикладне програмне забезпечення

Системи класу MES (рис. 2.5) призначені для виробничого середовища підприємства.

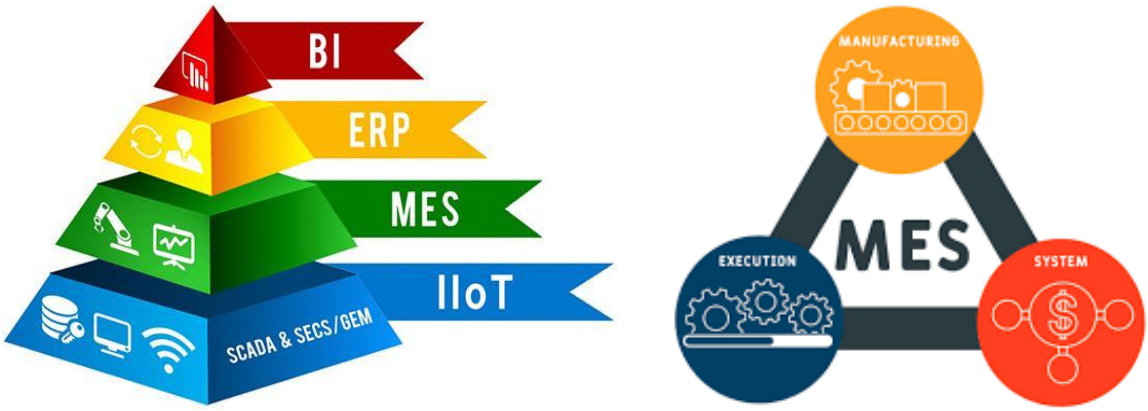

Рис. 2.5. MES

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 про виробничу 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) докладно відображаються в процесі 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 нині в країнах Заходу в MES вкладають чималі гроші.

Рис. 2.6. Функціональний склад MES

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).

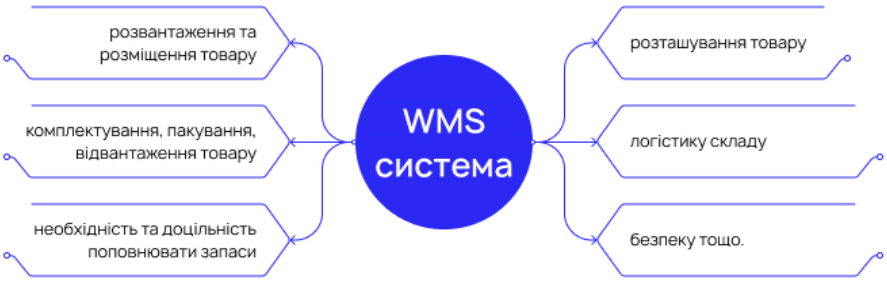

Рис. 2.7. Система управління WMS

WMS-система – це потужний софт, що дає змогу 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 цілком оптимізує його роботу. Вона дає змогу упорядкувати усі процеси, що

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

- ✓ розвантаження та розміщення товару;
- ✓ комплектування, пакування, відвантаження товару;
- ✓ потребу та доцільність поповнення запасів;
- ✓ розміщення товару;
- ✓ логістику складу;
- ✓ безпеку тощо.

Усі операції відбуваються за допомогою таких компонентів:

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

✓ Сервер бази даних – зберігає усю інформацію про роботу складу.

✓ Бізнес-логіка – обробляє запит клієнта, використовуючи базу даних, після чого дає 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 дають змогу не лише відстежувати розміщення вантажу, але й запобігати крадіжкам, псуванню товару (датчики кліматконтролю), вчасно виявляти пожежу тощо.

Важлива роль у роботі WMS належить штрих-кодам та радіочастотним міткам, якими маркують товар. За їхньою допомогою можна легко знайти вантаж у будь-якій точці складу, навіть якщо його випадково перемістили в інше місце.

Типи WMS

Складська система WMS може бути як без функцій адресного зберігання, так і з ними. У першому випадку йдеться про базові програми, які ведуть облік товару, що надходить 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 й усього підприємства) або SCM (програма для управління ланцюгом постачання).

Варто також зазначити, що WMS часто складається з модулів, від яких можна відмовитись або додати, залежно від потреб компанії. У цьому аспекті виділяють кілька типів 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. $м^2$), проте товарів 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. Платформу повністю розробляють з нуля під потреби певного підприємства.

✓ Модульні – складні системи для складів з високим товарообігом та асортиментом. Вони складаються з модулів, які можна легко замінити залежно 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 їхнє 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.

✓ Дає змогу оцінити роботу персоналу, використовуючи неупереджені дані.

✓ Запобігає ситуаціям, коли потрібний товар закінчується на складі.

✓ Дає змогу збільшити кількість товару, який можна зберігати на складі.

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

✓ Дає можливість ухвалювати правильне рішення завдяки отриманню інформації у режимі онлайн.

✓ Запобігає крадіжкам, втраті товару.

Щоби 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 ґрунтовної підготовки. Треба бути готовим забезпечити у компанії період адаптації.

Серед інших недоліків WMS можна виділити кілька.

✓ Потреба дотримуватись усіх вимог щодо отримання й відвантаження товару. Якщо працівник його не побачить й не поставить штрих код, вантаж загубиться, що може призвести до значних 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 зі спеціалістом.

✓ Неправильно налаштовані процеси. Більшість компаній несхвально відгукуються про WMS-системи лише тоді, 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 дають змогу згодом розширити її можливості. Основні функції полягають у такому.

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

Система EAM (Enterprise Asset Management) – управління фондами підприємства

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

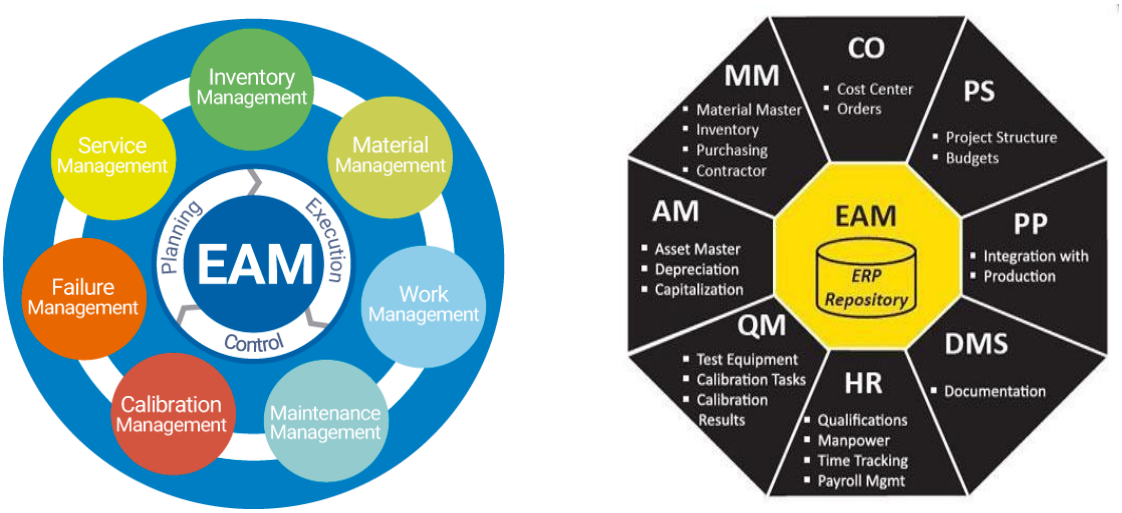

Рис. 2.8. Система управління основними фондами підприємства

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

Система HRM (Human Resource Management) – управління персоналом

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

Рис. 2.9. Система управління персоналом

HRM-системи вирішують два основні завдання:

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

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

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

Функції HRM-систем:

- пошук персоналу;
- добір та відбір персоналу;
- оцінювання 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 класифікують інформаційні системи за сферою застосування?
4. Охарактеризуйте призначення ERP-систем.
5. Назвіть основні задачі MES-систем.
6. Охарактеризуйте функції HRM-систем.

Лекції № 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 ухвалення рішення про створення інформаційної системи і закінчується в момент її цілковитого 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, подаючи його як деяку послідовність стадій та процесів.

Класифікація інформаційних систем ілюструє всю різноманітність рівнів їхньої побудови, призначень, предметних галузей, сфери застосування, типу реалізації та ін. Проте є загальні сталі умови проєктування систем різного класу, що орієнтовані на наймасовішу частину ІС, мають загальні теоретичні та методологічні засади, багато спільних рис, а також типів зв'язків (функціональних, інформаційних, зовнішніх) між елементами структури систем. Це дає змогу сформулювати єдині принципи і шляхи побудови інформаційних систем.

3.2. Стадії та етапи життєвого циклу системи

Повний життєвий цикл інформаційної системи звичайно складається з кількох 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 їхньої інтеграції; тестування програмного продукту в цілому (так звана верифікація); тестування відповідності функцій здатної працювати програмної системи до вимог, що були до неї поставлені замовником (так звана валідація);
- **експлуатація та супровід готової системи**.

Рис. 3.1. Стадії та етапи життєвого циклу системи

Підготовча робота починається з вибору моделі ЖЦ ПЗ, відповідної масштабові, значущості і складності проєкту. Процес розроблення має бути відповідним обраній моделі. Розробник повинен обрати, адаптувати до умов проєкту і використовувати погоджені із замовником стандарти, методи й засоби розроблення, а також скласти план виконання робіт.

Аналіз вимог до системи охоплює функціональні можливості, вимоги користувача, вимоги до надійності і безпеки, вимоги до зовнішніх інтерфейсів тощо. Вимоги до системи оцінюють відповідно до критеріїв реалізації і можливості перевірки під час тестування.

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

Аналіз вимог до ПЗ визначає функціональні можливості, зокрема 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 як стандарт 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 програмного забезпечення. Стандарт не прив'язаний до певної моделі життєвого цикла. Стандарт не містить опису фаз, стадій і етапів. Процеси стандарту будуть розглянуті далі.

Стандарт визначає загальні набори завдань, характеристики якості, критерії оцінювання та інші характеристики, потрібні для всебічного охоплення ситуацій. Наприклад, визначаючи вимоги до системи, слід відобразити таке:

- розглянута/вивчена та проаналізована сфера застосування системи;
- специфікації вимог до системи: функції та можливості системи, бізнес, організаційні вимоги і вимоги користувача, безпекові заходи, захищеність, людські фактори, ергономіка, зв'язки, операції і вимоги до супроводу, проєктні обмеження та кваліфікаційні вимоги;
- документація кваліфікаційних вимог до системи.

Аналіз вимог до ПЗ має містити одинадцять класів характеристик потрібних пізніше у гарантуванні якості.

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

- функціональні та інші можливі специфікації (зокрема виконання, фізичні характеристики та умови середовища експлуатації), за якими повинна бути створена одиниця програмного забезпечення;
- зовнішні зв'язки (інтерфейси) з одиницею ПЗ;

- специфікації надійності, пов'язані з методами функціонування і супроводу ПЗ, зокрема специфікації, пов'язані з впливом навколишнього середовища і ймовірністю травм персоналу;
- специфікації захищеності, зокрема специфікації, пов'язані із спотворенням точності інформації;
- людські фактори специфікацій з інженерної психології (ергономіки), зокрема 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 їхнього створення. Крім того, в стандарті міститься опис змісту робіт на кожному етапі. Стадії й етапи роботи, закріплені в стандарті, більшою мірою придатні для каскадної моделі життєвого циклу.

Rational Unified Process (RUP) пропонує ітеративну модель розроблення, що містить чотири фази: початок, дослідження, побудова та впровадження. Кожна фаза може бути поділена на етапи (ітерації), в результаті яких випускається версія для внутрішнього або зовнішнього використання. Виконані чотири основні фази називають циклом розроблення, кожен цикл завершується генерацією версії системи. Якщо після цього робота над проєктом не припиняється, то отриманий продукт розвивають, проходячи ті самі фази.

Суть роботи в рамках RUP – це створення і супровід моделей на базі UML. Microsoft Solution Framework (MSF) подібна до RUP, так само має чотири фази: аналіз, проєктування, розроблення, стабілізація, є ітераційною, припускає виконання об'єктноорієнтованого моделювання.

Зміст основних процесів життєвого циклу ПЗ (ДСТУ ISO/IEC/IEEE 12207:2018) подано в табличному вигляді (табл. 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 заявкових пропозицій ● підготовка угоди ● контроль діяльності постачальника ● приймання ІС	● рішення про початок робіт з впровадження ІС ● результати обстеження діяльності замовника ● результати аналізу ринку ІС / тендер ● план постачання/ розроблення ● комплексний тест ІС	● техніко-економічне обґрунтування впровадження ІС (ТЕО) ● технічне завдання на ІС ● договір про постачання/розроблення ● акти приймання етапів роботи ● акт приймально-здавальних випробувань
Постачання **(Розробник ІС)**	● ініціювання ● відповідь на заявкові пропозиції ● підготовка договору ● планування виконання ● контроль виконання ● постачання	● технічне завдання на ІС ● рішення керівництва про участь в розробленні ● результати тендеру ● план УП ● розроблена ІС і документація	● рішення про участь в розробленні ● комерційні пропозиції / конкурсна заявка ● договір про постачання /розроблення ● план УП ● реалізація / коригування ● акт приймально-здавальних випробувань
Розроблення **(Розробник ІС)**	● підготовка ● аналіз вимог до ІС ● проєктування архітектури ІС ● розроблення вимог до ПЗ ● проєктування архітектури ПЗ ● детальне проєктування ПЗ ● кодування і тестування ПЗ	● технічне завдання на ІС ● технічне завдання на ІС, модель ЖЦ ● технічне завдання на ІС ● підсистеми ІС ● специфікації вимог до компонентів ПЗ ● архітектура ПЗ	● використовувана модель ЖЦ, стандарти розроблення ● план робіт ● склад підсистем, компоненти обладнання ● специфікації вимог до компонентів ПЗ ● склад компонентів ПЗ, інтерфейси з БД, план інтеграції ПЗ ● проєкт БД, специфікації інтерфейсів між компонентами ПЗ, вимоги до тестів

Закінчення табл. 2

Процес (виконавець процесу)	Дії	Вхід	Результат
Розроблення **(Розробник ІС)**	● інтеграція ПЗ і кваліфікаційне тестування ПЗ ● інтеграція ІС і кваліфікаційне тестування ІС	● матеріали детального проєктування ПЗ, план інтеграції ПЗ, тести ● архітектура ІС, ПЗ, документація на ІС, тести	● 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 структурі ЖЦ мають бути 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; оцінювання проєкту; контроль проєкту; управління ризиками; управління конфігурацією; управління інформаційними потоками; ухвалення рішень.

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

V. *Спеціальні процеси*: визначення та встановлення взаємозв'язків відповідно до завдань і цілей.

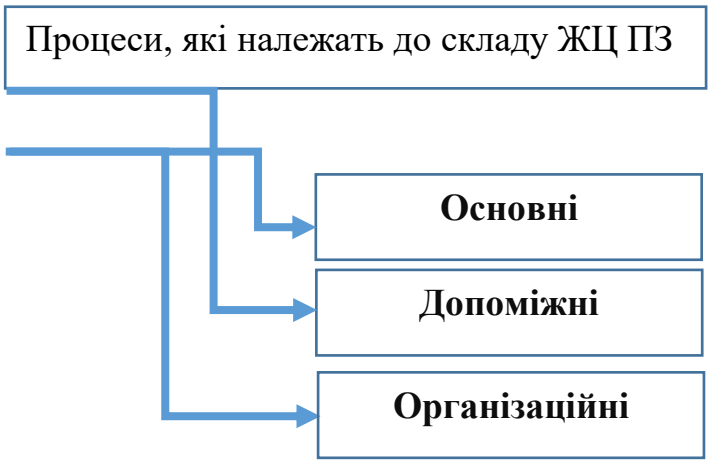

Рис. 3.2. Процеси ЖЦ ПЗ

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

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

3. *Організаційні процеси* охоплюють процеси створення інфраструктури; управління; навчання; удосконалення. До них прилягає процес адаптації, який визначає основні дії, потрібні для адаптації стандарту до умов конкретного проєкту. Організаційні процеси визначають дії і завдання, виконувані 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 заявкових пропозицій;
3) підготовку та коригування договору;
4) нагляд за діяльністю постачальника;
5) приймання і завершення робіт.

Ініціювання придбання полягає в такому:

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

b) аналіз вимог до системи;

c) ухвалення рішення про придбання, розроблення або удосконалення наявного ПЗ;

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

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

Заявкові 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, зазначених в договорі.

Підготовка та коригування договору охоплюють етапи:

a) визначення замовником процедури вибору постачальника, яка містить критерії оцінювання пропозицій можливих постачальників;

b) вибір конкретного постачальника на підставі результатів пропозицій;

c) підготовлення та укладення договору з постачальником;

d) внесення змін (за потреби) в договір в процесі його виконання.

Нагляд за діяльністю постачальника відбувається відповідно до дій, які передбачені в процесах спільного оцінювання та аудиту.

У процесі приймання готують і виконують необхідні тести. В разі задоволення всіх умов приймання відбувається завершення робіт за договором та подальше приймання їх в експлуатацію.

II. ***Процес постачання*** охоплює такі дії:

ініціювання поставки;

a) підготовка відповіді на заявкові пропозиції;

b) підготовка договору;

c) планування;

d) виконання та контроль;

e) перевірка та оцінювання;

f) постачання і завершення робіт.

Ініціювання поставки полягає у розгляді постачальником заявкових пропозицій і ухвалення рішення погодитися з поставленими вимогами та умовами або запропонувати свої.

Планування означає:

a) ухвалення рішення постачальником про виконання робіт своїми силами або із залученням субпідрядника;

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

III. ***Процес розробки*** – це дії і завдання, які виконує розробник, охоплює роботи зі створення ПЗ і його компонентів відповідно до

заданих вимог, зокрема такі, як оформлення проєктної та експлуатаційної документації, підготовка матеріалів, потрібної для перевірки працездатності та відповідної якості програмних продуктів, матеріалів, потрібних для організації навчання персоналу тощо.

Процес розроблення охоплює:

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, відповідної масштабу, значущості та складності проєкту. Дії і завдання процесу розроблення повинні бути відповідними обраній моделі. Розробник має обрати, адаптувати до умов проєкту і виконати узгоджені із замовником стандарти, методи і засоби розроблення, а також скласти план виконання робіт.

Аналіз вимог до системи охоплює визначення її функціональних можливостей, призначених для користувача вимог, вимог до надійності і безпеки, до зовнішніх інтерфейсів тощо. Вимоги до системи оцінюються відповідно до критеріїв можливості бути реалізованими і можливості перевірки під час тестування.

Проєктування архітектури системи на високому рівні полягає у визначенні компонентів її обладнання, ПЗ та операцій, які виконує персонал, що експлуатує систему. Архітектура системи має бути відповідна вимогам до системи, а також прийнятим проєктним стандартам і методам.

Аналіз вимог до ПЗ – це визначення характеристик для кожного компонента:

a) функціональних можливостей, зокрема 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 користувацької документації;

h) вимог до експлуатації і супроводу.

Вимоги до ПЗ оцінюють, виходячи з критеріїв відповідності вимогам системи, можливостей системи бути реалізованою та її перевірки під час тестування.

Проєктування архітектури ПЗ означає вирішення таких завдань:

a) трансформацію вимог до ПЗ в архітектуру, яка визначає на високому рівні структуру ПЗ і склад його компонентів;

b) розроблення і документування програмних інтерфейсів ПЗ і баз даних;

c) розроблення попередньої версії користувацької документації;

d) розроблення і документування попередніх вимог до тестів і плану інтеграції ПЗ.

Архітектура компонентів ПЗ має бути відповідна вимогам до них, а також прийнятим проєктним стандартам і методам.

Детальне проєктування ПЗ полягає в такому:

a) опис компонентів ПЗ та інтерфейсів між ними на більш низькому рівні, достатньому для їхнього подальшого самостійного кодування і тестування;

b) розроблення і документування детального проєкту бази даних;

c) оновлення (за потреби) користувацької документації;

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

e) оновлення плану інтеграції ПЗ.

Кодування і тестування ПЗ означає виконання таких завдань:

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

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

c) оновлення (за потреби) користувацької 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 розробляють набори тестів і тестові процедури, призначені для перевірки кожної з кваліфікаційних вимог з метою подальшого кваліфікаційного тестування.

Кваліфікаційна вимога – набір критеріїв або умов, яких треба дотримати, щоби кваліфікувати програмний продукт, відповідний своїм специфікаціям і готовий до використання в умовах експлуатації.

Кваліфікаційне тестування ПЗ виконує розробник у присутності замовника для демонстрації того, що ПЗ є відповідним своїм специфікаціям і готовий до використання в умовах експлуатації. При цьому перевіряють також повноту технічної і призначеної для користувача документації, її адекватність компонентам ПЗ. Інтеграція системи полягає в об'єднанні всіх її компонентів, зокрема ПЗ та устаткування. Після інтеграції система підлягає кваліфікованому тестуванню на відповідність сукупності вимог до неї. При цьому також виконують оформлення і перевірку повного комплекту документації до системи.

Встановлення ПЗ виконує розробник відповідно до плану в середовищі і на обладнанні, зазначеному в договорі. У процесі встановлення перевіряють працездатність ПЗ і баз даних. Якщо встановлюване ПЗ замінює наявну систему, розробник повинен забезпечити їхнє паралельне функціонування відповідно до договору.

Приймання ПЗ полягає в оцінюванні результатів кваліфікаційного тестування ПЗ і системи, документування результатів оцінювання, які виконує замовник за допомогою розробника.

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

IV. ***Процес експлуатації***

Цей процес охоплює дії і завдання оператора – організації, яка експлуатує систему. Оператор, зокрема, виконує:

1) підготовчу роботу;

2) експлуатаційне тестування;
3) експлуатацію системи;
4) підтримку користувачів.

Підготовча робота означає виконання оператором таких завдань:

a) планування дій і робіт, виконуваних 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 її передають в експлуатацію.

Експлуатація системи відбувається в призначеному для цього середовищі відповідно до користувацької документації.

Підтримка користувачів полягає в наданні допомоги і консультацій у разі виявлення помилок в процесі експлуатації ПЗ.

V. ***Процес супроводження*** – це дії і завдання, які виконує супроводжувальна організація (служба супроводу). Цей процес активується в разі змін (модифікацій) програмного продукту і відповідної документації, спричинених 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 вносять до ПЗ (ІС), не повинні порушувати цілісність. Процес супроводу полягає в перенесенні ПЗ в інше середовище (міграцію) і закінчується зняттям ПЗ з експлуатації.

Процес супроводу охоплює:

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-ідей завжди є незадоволеність наявними програмними продуктами (системами), або

взагалі їхня відсутність. Таким чином, слід проаналізувати проблему, яку вирішує програмне забезпечення (ІС).

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

1. *Проблема повинна бути суттєвою*: має бути чітко та однозначно зрозуміло, що проблема існує. Більшість проблем на теперішній час мають певні рішення. Але ці рішення можуть мати свої недоліки: довго, дорого, складно і таке ін. Якщо проблема є і вона має 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 розуміють, що розглядається одна проблема, а не декілька. Проблема може бути складною та мати комплексну структуру, але вона у формулюванні повинна бути єдиною.

Для аналізу сформульованої проблеми потрібно:

- побудувати дерево проблем;
- провести аналіз зацікавленості сторін у вирішенні цієї проблеми.

Слід мати на увазі, що першочерговість етапів залежить від міри обізнаності з проблемою. Якщо йдеться про суттєву експертність у визначеній проблематиці, то можна починати з побудови дерева проблем, а потім консультуватися з зацікавленими сторонами. Якщо експертності недостатньо, то спочатку треба визначитись з зацікавленими сторонами проєкту.

Зацікавлені сторони – це особа або група осіб, які зацікавленні у створенні, впровадженні програмного продукту, який дає змогу розв’язати певною мірою визначену ними проблему.

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

Метою реалізації цього етапу є створення моделі поведінки системи. Він ґрунтується на ***аналізі проблеми*** *функціонування наявної системи,* ***дослідженні предметної області, пошуку і оцінюванні***

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 є 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 слабкоструктуровані проблеми треба звести до такого вигляду, коли вони стають завданнями з пошуку відповідних засобів для їхнього досягнення. Коли йдеться про цілі, то слід з’ясувати, чого ми насправді бажаємо.

Для цього будують та аналізують в першій ітерації дерево цілей для побудови системи (рис. 4.1).

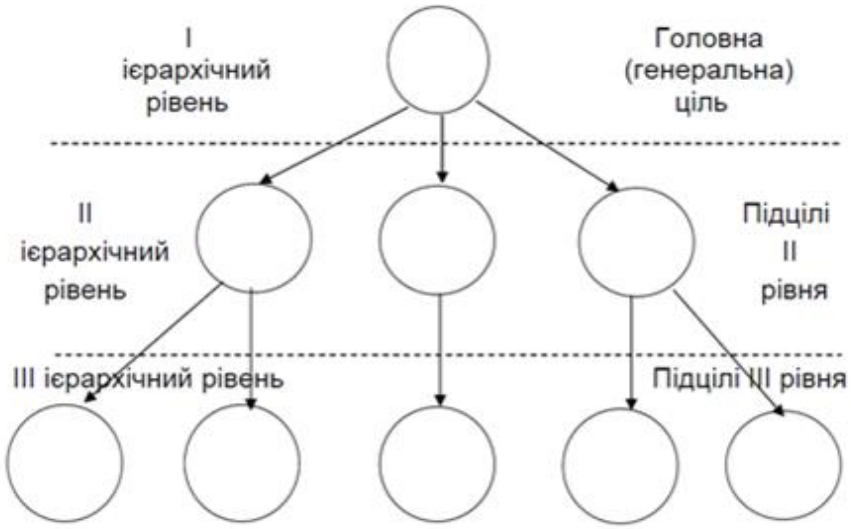

Рис. 4.1. Ієрархія цілей системи

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

В рамках цього етапу виконують таке:

- попереднє виявлення вимог до майбутньої системи;
- визначення оргштатної і топологічної структур підприємства (об’єкта комп’ютеризації);
- визначення переліку цільових завдань (функцій) системи;
- аналіз розподілу функцій між учасниками процесу функціонування та визначення технології їхнього виконання;
- аналіз інформаційного забезпечення.

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

Модель організації, описана в термінах бізнес-процесів і бізнес-функцій, дає змогу сформулювати основні вимоги до інформаційної системи. На цьому етапі виконують обробку результатів обстеження та побудову моделей діяльності підприємства такого типу.

- Модель «як є», що являє собою «знімок» стану справ на підприємстві (оргштатної структури, взаємодії підрозділів, застосовуваних технологій, автоматизованих і неавтоматизованих бізнес-процесів та ін.) на момент обстеження, дає змогу зрозуміти, що робить і як функціонує підприємство з позицій системного аналізу, а також на основі автоматичної верифікації виявити ряд помилок і «вузьких місць» і сформулювати ряд пропозицій щодо поліпшення ситуації.

У процесі аналізу інформаційного забезпечення визначають таке:

- склад інформації (перелік інформаційних одиниць, потрібних для розв'язання комплексу завдань);
- 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 (система кількісних оцінок значущості, повноти, своєчасності, вірогідності інформації);
- способи перетворення інформації;
- уніфіковану систему первинної документації;
- методичні й інструктивні матеріали для ведення документів.

З цим аналізом пов'язаний 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 дає змогу сформувати бачення нових раціональних технологій роботи підприємства (рис. 4.2).

Рис. 4.2. Оптимізація процесів

Формують визначальну документацію, як-от «***Концепція системи»*** та ***«Технічне завдання»****, яке* ***є підставою для розроблення інформаційної системи і подальшого її приймання в експлуатацію***. У ній визначають основні вимоги до системи та процесу її розроблення і розробляють для системи загалом. Додатково можуть бути розроблені технічні завдання на окремі частини ІС.

Запитання для самоконтролю

1. Назвіть поняття життєвого циклу інформаційної системи.
2. Які нормативні документи 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 ґрунтується аналіз проблем, цілей та визначення вимог до майбутньої системи?

Лекція № 5. МОДЕЛІ ЖИТТЄВОГО ЦИКЛУ ІНФОРМАЦІЙНОЇ СИСТЕМИ

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

Модель життєвого циклу ПЗ містить:

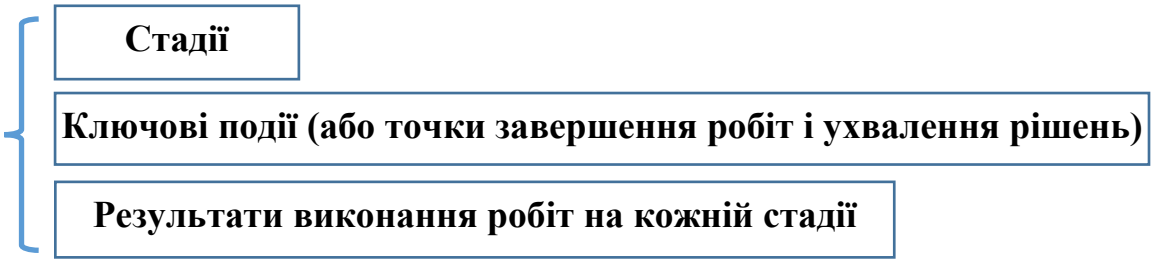

Рис. 5.1. Структура моделі життєвого циклу ПЗ

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 достатньо для створення ПЗ відповідно до поставлених вимог.

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

Сучасні 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 викладено в стандарті ISO/IEC 12207. Слід відмітити, що цей 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, у яких 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 протягом тривалого часу залишалася класична, або так звана каскадна, або водоспадна (waterfall), модель життєвого циклу. Вона була вперше чітко сформульована в стандартах міністерства оборони США (автор Вінстон 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, вироблених на попередній стадії, передається як вхідні дані для наступної. Дуже часто класичний життєвий цикл називають каскадною, або водоспадною, моделлю, підкреслюючи, що розроблення розглядається як послідовність етапів, причому перехід на наступний, ієрархічно нижній етап відбувається тільки після повного завершення робіт на поточному етапі, тобто все розроблення поділяється на етапи, перехід до наступного етапу відбувається тільки за умови цілковитого завершення всіх робіт попереднього етапу.

Кожен етап завершується оформленням повного комплекту документації. Склад і зміст цієї документації означає, що над втіленням проєкту далі може працювати інша команда розробників.

Головні етапи розроблення згідно з каскадною моделлю

Зважаючи на досвід реальних розробок, у каскадній моделі звичайно виділяють декілька сталих етапів, що майже не залежать від предметної області, а саме (рис. 2.5):

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

Рис. 5.2. Структурна схема каскадної моделі розробки ІС

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

Етап аналізу означає докладне дослідження бізнес-процесів (вимог і функцій, визначених на етапі вибору стратегії) та інформації, потрібної для їхнього виконання (сутності, їхніх атрибутів і зв'язків (відносин)).

Всю інформацію про систему, зібрану на етапі концептуального ухвалення рішення, формалізують і уточнюють на етапі аналізу. Особливу увагу слід приділити повноті переданої інформації, аналізу інформації на предмет відсутності суперечностей, а також пошуку невживаної взагалі або інформації, що дублюється. Як правило, замовник не відразу формує вимоги до системи загалом, а формулює вимоги до окремих її компонентів, це природно, адже замовник не є фахівцем з певних напрямів, що зумовлює поетапність формування вимог в процесі аналізу із залученням фахівців з предметної області. Тому потрібно приділити увагу узгодженості цих компонентів. Фактично ці вимоги визначають повне завдання на розроблення. Воно має бути узгоджене з усіма сторонами (суб’єктами).

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

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

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

На етапі розроблення відбувається тісна взаємодія проєктувальників, розробників і груп тестерів. У разі інтенсивного розроблення 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 розроблення змінюють інтерфейси користувача. Це зумовлено, зокрема, і тим, що модулі періодично демонструють замовникові, тим самим

визначаючи оптимальний front end користувача. Істотно можуть змінюватися й запити до даних.

Взаємодія тестувальника і розробника без централізованого передавання частин (розділів, томів) проєктної документації допустима, але тільки якщо треба терміново перевірити якусь зміну. Дуже часто етап розроблення і етап тестування взаємопов'язані і відбуваються паралельно.

Тестування. На ***четвертому етапі*** виявляють приховані недоліки, виявлені під час експлуатації, а також вносять відповідні коригування до програмного забезпечення.

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

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

Після того як автономний тест відбувся успішно, виконують тести зв'язків згенерованих модулів з метою відстежити взаємний вплив модулів.

Впровадження (введення в дію). Головне завдання ***п'ятого етапу*** – переконати замовника у тому, що всіх вимог дотримано 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'язані з удосконаленням ПЗ, решта часу 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 групу модулів тестують на надійність роботи, виконують 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 в умовах штатної роботи і дає змогу оцінити час безвідмовної роботи системи. У комплекті обов'язкових тестів для визначення стійкості мають бути тести, що імітують пікове навантаження на систему.

Експлуатація. Перехідний місток від процесу тестування, систему вводять в експлуатацію не одразу, а поступово.

Введення в експлуатацію складається з трьох фаз:

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

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

Переваги каскадної моделі

Серед переваг каскадної моделі передусім можна вказати на такі:

- на кожному етапі формується повний та узгоджений набір пакета документів. На завершальних етапах також розробляють документацію, що охоплює всі передбачені стандартами різновиди забезпечення ІС (організаційне, методичне, інформаційне, програмне, апаратне);
- чітка послідовність етапів дає змогу планувати терміни виконання робіт та відповідні витрати.

Каскадний підхід добре проявив себе під час розроблення певного класу інформаційних систем. Це системи, у яких від самого початку можна окреслити всі вимоги, що дає розробникам свободу щодо майбутніх шляхів їхньої реалізації.

Недоліки каскадної моделі

Кількість недоліків каскадної моделі досить значна, а саме:

- повільне отримання результатів;
- помилки на будь-якому етапі впливають на подальшу роботу;

- у межах каскадної моделі складно організувати паралельне ведення робіт;
- певна інформаційна перенасиченість на всіх етапах проєкту;
- ускладнене управління проєктом;
- значний рівень ризику та ненадійність інвестицій.

Для того щоби зрозуміти сферу можливого використання каскадної моделі, розглянемо згадані недоліки докладніше.

Затримка в отриманні результатів – це основний недолік каскадної моделі, що є наслідком послідовного підходу, адже узгодження результатів відбувається 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 висловити 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 їхні потреби.

Крім того, обрані на початок розроблення 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 роботу над проєктом будують як низку 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 потребою в постійному узгодженні різних частин проєкту. Парадокс полягає у тому, що чим більш незалежними є частини, тим ретельніше треба виконувати синхронізацію. 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 про внесення змін до будь-якої частини проєкту треба повідомити всіх розробників, що пов’язані з цією частиною. Як наслідок – швидко зростають витрати на документування змін, опрацювання цих змін, погіршується оперативність узгоджень тощо. Загалом збільшується обсяг інформації внаслідок особливостей схеми управління проєктом,

що має в певному розумінні «паразитну» природу. Питання перенасиченості різко загострюється в разі ротації груп розробників, адже нові спеціалісти, крім вивчення нового матеріалу, повинні опанувати велику кількість попередньої 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 зумовлена 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 на результати роботи інших команд. Отже, потрібно адміністративне втручання для узгодження термінів роботи та складу переданої документації.

Значний рівень ризику. Що складнішим є проект, то довшою буде тривалість кожного з його етапів, то 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 їхніх погоджень.

Фактично це означає, що є ймовірність значно збільшити терміни виконання проєкту, а таке збільшення (з фінансового погляду) означає високий рівень ризику інвестицій.

Нарешті, занадто велика кількість змін (особливо в предметній області або у вимогах замовника) можуть взагалі призупинити проєкт, завадити його остаточній реалізації. Тому можна стверджувати, що складні проєкти, розроблювані за каскадною схемою, мають підвищений ступінь ризику.

Незважаючи на наполегливі рекомендації експертів в галузі проєктування і розроблення ПЗ, багато компаній і надалі використовують каскадну модель на практиці замість якого-небудь варіанта ітераційної моделі.

Головні причини, з яких каскадна модель не втрачає своєї популярності: звичка, розроблення невеликих проєктів, ілюзія зниження ризиків учасників проєкту (замовника і виконавця), проблеми впровадження в разі використання ітераційної моделі.

Сфера застосування

Каскадний підхід добре зарекомендував себе у розробленні:

a) однорідних ІС, у яких кожен застосунок 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 зумовлює ітераційний процес розроблення ІС. До того ж у межах цієї моделі зростає роль початкових етапів ЖЦ (аналізу та проєктування) – саме на цих етапах перевіряють та обґрунтовують реалізацію технічних рішень через створення прототипів.

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

Рис. 5.3. Спіральна модель життєвого циклу

Таким чином, кожне коло спіралі є відповідним за створення частки або версії програмного продукту, на цьому ж колі уточнюють мету й характеристику проєкту, вимоги до якості, планують роботи для наступного витка спіралі. На кожній ітерації послідовно конкретизують

деталі проєкту, обирають обґрунтований варіант. У подальшому цей варіант доводять до остаточної реалізації. Використання спіральної моделі дає змогу переходити до наступного етапу виконання проєкту, не чекаючи на повне завершення поточного етапу робіт, – їх можна буде виконати у наступній ітерації.

Основне завдання кожної ітерації – якнайшвидше отримати опрацьований продукт, який можна показати користувачам системи. Ця обставина значно спрощує процес внесення уточнень і доповнень до проєкту.

Переваги спіральної моделі

Спіральний підхід до розроблення програмного забезпечення дає змогу подолати більшість недоліків каскадної моделі. Крім того, він надає безліч додаткових можливостей зробити процес розроблення гнучким. Переваги ітераційного підходу такі.

1. Ітераційне розроблення полегшує коригування проєкту в разі змін у вимогах замовника.

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

3. У разі реалізації проєкту за спіральною схемою зменшується ступінь ризиків. Ця перевага є наслідком попередньої, оскільки ризики можна виявити вже на етапі інтеграції. Через це рівень ризиків є максимальним на початку розроблення проєкту. Поступово, у міру просування розробки, очікуваний рівень ризиків знижується.

4. Ітераційний характер організації робіт сприяє гнучкості 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 зумовлене тим, що простіше знайти загальні частини проєкту, коли вони вже частково розроблені, ніж намагатися виявити їх на початковій стадії проєкту. Досвід свідчить, що завдяки аналізу проєкту вже після декількох ітерацій можна виокремити компоненти багаторазового використання, які на наступних ітераціях будуть удосконалені.

6. Спіральна модель дає змогу створити більш надійну та сталу систему. Це є наслідком того, що в міру розвитку системи помилки та слабкі місця виявляють і коригують на кожній ітерації.

7. Ітераційний підхід дає змогу удосконалювати процес розроблення. За аналізом, виконаним наприкінці кожної ітерації, можна оцінити, що має бути змінене в організації розроблення та як її поліпшити на наступній ітерації.

Недоліки спіральної моделі

Головна проблема спірального підходу – визначення моментів переходу між етапами.

Для її розв'язання потрібно вводити тимчасові обмеження на кожен з етапів ЖЦ, інакше процес розроблення може перетворитися на нескінченне удосконалення зробленого. Тому завершення ітерації має відбуватися тільки відповідно до плану, навіть якщо не весь запланований обсяг робіт вже закінчено. Щодо самого плану, то його складають 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, яку поставляють.

Основний недолік каскадного підходу – брак «гнучкості». Саме цей недолік нівелюється каскадно-зворотним підходом, в якому дозволено повернення до попередніх стадій і перегляд або уточнення раніше ухвалених рішень. Каскадно-зворотниій підхід реалізує ітераційний характер розроблення ПЗ.

Таким чином розроблення ІС ведеться ітераціями з циклами зворотного зв'язку між етапами (рис. 5.4). Міжетапні коригування дають змогу брати до уваги реальний взаємовплив результатів роботи на різних етапах; час життя кожного з етапів розтягується на весь період розроблення.

Отже, на кожній ітерації можна аналізувати проміжні результати робіт і реакцію на них усіх зацікавлених осіб, зокрема користувачів, і

вносити коригувальні зміни на наступних ітераціях. Кожна ітерація може містити повний набір видів діяльності від аналізу вимог до введення в експлуатацію чергової частини програмного забезпечення.

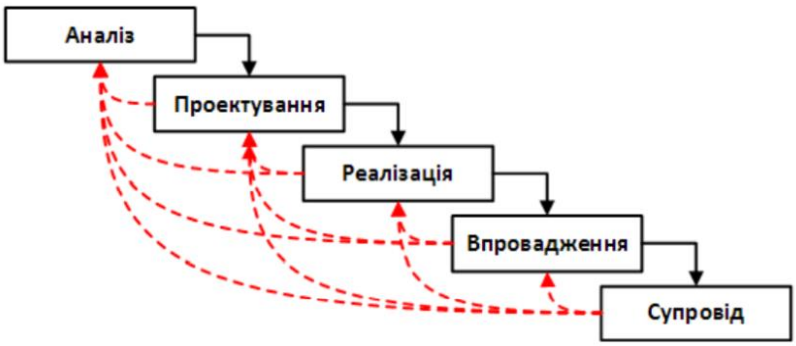

Рис. 5.4. Поетапна модель життєвого циклу інформаційної системи

Переваги поетапної (ітераційної) моделі

1. Замовникові немає потреби чекати на завершення розроблення системи, щоб скласти про неї уявлення. Компоненти, отримані на перших кроках розроблення, задовольняють найбільш критичні вимоги (оскільки зазвичай мають найбільший пріоритет) і їх можна оцінити на ранній стадії створення системи.

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

3. Зниження ризиків завдяки ранньому виявленню 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 є реально відповідним потребам споживача).

7. Швидкий випуск мінімально життєздатного продукту.

Основні недоліки поетапної (ітераційної) моделі

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

2. Брак фіксованого бюджету і термінів виконання.

3. Потрібне значне 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 настанов (рис. 5.5). Такий підхід зосереджений на особистій відповідальності, тісному контакті та командній роботі.

Гнучке розроблення за принципом Agile найчастіше застосовують, щоби покращувати клієнтський досвід, швидше постачати продукт на ринок, зменшити кількість нераціональних і неефективних дій та підвищити якість ПП загалом. Особливо ефективним є застосування Agile під час роботи в умовах високої невизначеності.

Являє собою сукупність різних підходів до розроблення програмного забезпечення. Складається з серії підходів до розроблення програмного забезпечення, орієнтованого на використання ітеративної розробки (в Scrum ітерації називають «спринтами»), динамічне формування вимог і забезпечення їхньої реалізації завдяки постійній взаємодії всередині самоорганізованих робочих груп фахівців різного профілю. Окрема ітерація являє собою мініатюрний програмний проєкт. Однією з основних ідей Agile є взаємодія всередині команди і з замовником безпосередньо.

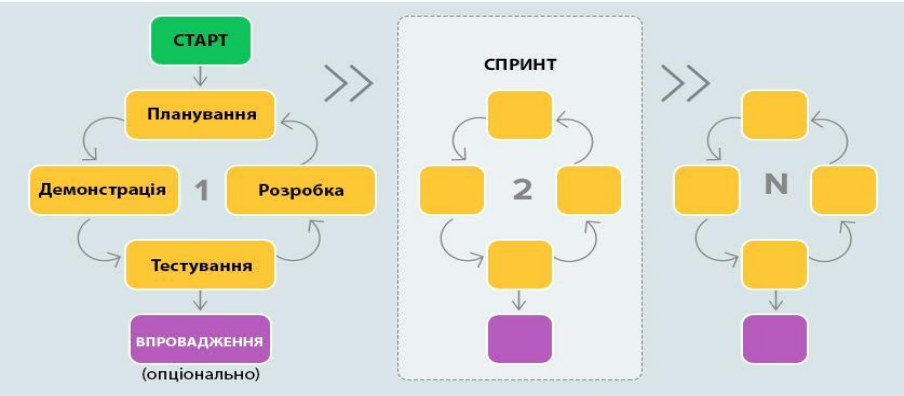

Рис. 5.5. Модель гнучкого розроблення програмного забезпечення Agile

Переваги Agile

1. Люди і їхня взаємодія є важливішими за процеси та інструменти.
2. Робоча ІС є важливішою за вичерпну документацію.
3. Співпраця із 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 дає змогу обмінюватися досвідом між замовником та учасниками команди розроблення, а також кожному з них брати участь у процесі ухвалення рішень. Позитивним наслідком є зниження ризиків втрати коштів і часу, а також підвищення здатності команди до вирішення нестандартних складних завдань з високим рівнем невизначеності.

Недоліки Agile

Випливають з його переваг:

1. Така взаємодія всіх з усіма може призвести до хаотичного процесу розроблення, що впливає на всі етапи.
2. Велика кількість зустрічей і обговорень, що може збільшити час розроблення продукту.
3. Складно планувати процеси, оскільки вимоги постійно змінюються.
4. Рідко застосовують для втілення 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) мають бути встановлено чіткі обмеження за часом, кожна ітерація має бути короткою з максимально зрозумілими цілями, а кінцевий результат повинен бути очевидним. Agile прогнозує результат на більш стислий період порівняно зі стандартними стратегіями розроблення, тому дуже добре вирішує проблему невизначеності.

Для підвищення ефективності треба дотримуватися такого правила: що вища невизначеність, то коротшою має бути ітерація.

Запитання для самоконтролю

1. Наведіть модель життєвого циклу.
2. Назвіть найбільш поширені моделі ЖЦ ІС.
3. Назвіть основні етапи розроблення інформаційних систем.
4. Окресліть сферу застосування каскадної моделі ЖЦ ІС.

5. Охарактеризуйте особливості застосування спіральної моделі ЖЦ.
6. У чому полягають основні недоліки спіральної моделі?
7. Назвіть переваги поетапної (ітераційної) моделі.
8. Назвіть недоліки Agile.

Лекція № 6. ТЕХНІЧНЕ ЗАВДАННЯ. ТЕХНІЧНИЙ ПРОЄКТ

6.1. Місце технічного завдання в життєвому циклі АС

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

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

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

На кожній зі стадій розробляють ряд документів, регламентованих ДСТУ(ГОСТ 34.201-89) та рядом супутніх нормативних документів.

На стадії технічного завдання розробляють однойменний документ, регламентований ДСТУ(ГОСТ 34.201-89). Згідно з цим нормативним документом технічне завдання (ТЗ) на автоматизовану систему управління контролю, проєктування та ін. є основним документом, що визначає вимоги й порядок створення, розвитку, модернізації автоматизованої системи і відповідно до якого виконують її розроблення і приймання та запровадження.

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

Проєкт ТЗ на АС розробляє розробник за участі замовника на основі технічних вимог, вихідних даних.

За конкурснійої організації робіт варіанти ТЗ розглядає замовник, який або обирає кращий варіант, або на підставі результатів порівняльного аналізу підготовлює за участі 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, в якому викладено вимоги замовника до вирішень і характеристик об’єкта, а також до його параметрів, вартості і заходів із зведення. Розробляють документ на підставі технічних умов, ДБН і містобудівних умов.

Складає завдання замовник (за участі проєктувальника – сертифікованого спеціаліста) і узгоджує його з інвестором та проєктувальником. Узгодження відбувається шляхом підписання та засвідчення печатками підприємств.

Технічне завдання є підсумковим документом вихідних даних для подальшого проєктування споруди (архітектурного комплексу), конструювання технічного пристрою (приладу, машини тощо), розроблення автоматизованої чи інформаційної системи, створення програмного продукту, виконання 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 виконання 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 документа.

6.2. Склад і зміст технічного завдання

Згідно з ДСТУ (ГОСТ 34.201-89), технічне завдання повинне складатися з таких розділів.

1. Загальні відомості

1.1. Повне найменування системи та її умовне позначення.

1.2. Шифр теми або номер договору.

1.3. Найменування підприємств, об'єднань розробника та замовника (користувача) системи та їхні реквізити.

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

1.5 Планові терміни початку та закінчення роботи зі створення системи.

1.6. Відомості про джерела та порядок фінансування робіт.

1.7. Порядок оформлення та пред'явлення замовнику результатів робіт зі створення системи (її частин), з виготовлення та налагодження 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'єкта автоматизації

4. Вимоги до системи

4.1. Вимоги до системи загалом.

4.1.1. Вимоги до структури та функціонування 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 до 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, технічного обслуговування, ремонту та збереження компонентів системи.

4.1.9. Вимоги до захисту інформації від несанкціонованого доступу.

4.1.10. Вимоги до збереження інформації під час аварій.

4.1.11. Вимоги до захисту від впливу зовнішніх дій.

4.1.12. Вимоги до патентної чистоти.

4.1.13. Вимоги до стандартизації та уніфікації.

4.1.14. Додаткові вимоги.

4.2. Вимоги до функцій (завдань), виконуваних системою.

4.2.1. Перелік функцій, завдань або їхніх комплексів, зокрема тих, від яких залежить взаємодія частин системи; функціональних підсистем.

4.2.2. Часовий регламент виконання кожної функції, завдання або комплексу завдань.

4.2.3. Вимоги до якості реалізації кожної функції завдання або комплексу завдань, 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 задають 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 із створення (розвитку) системи.

6. Порядок контролю та приймання системи.

7. Вимоги до складу та змісту робіт з підготовки об'єкта автоматизації до введення системи в дію.

8. Вимоги до документування.

9. Джерела розроблення.

10. Додатки (за потреби).

6.3. Ескізний проєкт та технічний проєкт ІС

Ескізний проєкт означає розроблення попередніх проєктних рішень стосовно системи.

Ескізний проєкт (технічну пропозицію) подають у вигляді проєктної документації, у якій описано архітектуру системи, структуру її підсистем, склад модулів. Тут само вказують пропозиції щодо вибору

базових програмно-апаратних засобів, які повинні відображати прогноз розвитку підприємства.

Відносно апаратних засобів, особливо ПЗ, такий вибір найчастіше означає вибір фірми-постачальника потрібних засобів. У проєкті може бути запропоновано кілька варіантів вибору. Аналіз дає змогу з'ясувати можливості покриття функцій, які автоматизують, наявними програмними продуктами, отже, й обсяги робіт з розроблення оригінального ПЗ. Такий аналіз потрібен для попереднього оцінювання тимчасових і матеріальних витрат на автоматизацію. Облік ресурсних обмежень дає змогу уточнити досяжні масштаби автоматизації.

Виконання стадії ескізного проєктування не є строго обов'язковим, якщо основні проєктні рішення визначені раніше або досить очевидні для конкретної ІС й об'єкта автоматизації, то ця стадія може бути вилучена із загальної послідовності робіт.

Зміст ескізного проєкту задається в ТЗ на систему.

Як правило, на етапі ескізного проєктування визначають:

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

За результатами виконаної роботи оформляють, узгоджують і затверджують документацію в обсязі, необхідному для опису повної сукупності ухвалених проєктних рішень і достатньому для подальшого виконання робіт зі створення системи.

На основі технічного завдання (й ескізного проєкту) розробляють технічний проєкт ІС.

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

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

Таблиця 3

Зміст технічного проєкту

Розділ	Зміст
Пояснювальна записка	• підстави для розроблення системи; • перелік організацій розробників; • стисла характеристика об’єкта із зазначенням основних техніко-економічних показників його функціонування і зав’язків з іншими об’єктами; • стислі відомості про основні проєктні рішення
Функціональна й організаційна структура системи	• обґрунтування виділення підсистем, їхній перелік та призначення; • перелік завдань, виконуваних в кожній підсистемі, із стислою характеристикою їхнього змісту; • схема інформаційних зав’язків між підсистемами і між завданнями в межах кожної підсистеми
Постановка завдань і етапи вирішення	• організаційно-економічна сутність задачі (найменування, мета рішення, стислий зміст, метод, періодичність і час виконання завдання, способи збору і передавання даних, зв'язок задачі з іншими задачами, характер використання результатів рішення; • економіко-математична модель задачі (структурна і розгорнута форма подання); • вхідна 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 про 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 накопичуваною і збереженою інформацією, форми вихідних документів)

Закінчення табл. 3

Розділ	Зміст
Організація інформаційної бази	• джерела надходження інформації та способи її передавання; • сукупність показників, використовуваних в системі; • склад документів, терміни і періодичність їхнього надходження; • основні проєктні рішення з організації фонду ДНІ; • склад ДНІ, зокрема перелік реквізитів, їхні визначення, діапазон зміни і перелік документів ДНІ; • перелік масивів ДНІ, їхній 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 внесення 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 й обґрунтування схеми технологічного процесу обробки даних; • обґрунтування і вибір структури комплексу технічних засобів і його функціональних груп; • обґрунтування вимог до розроблення нестандартного обладнання; • комплекс заходів для забезпечення надійності функціонування технічних засобів
Розрахунок економічної ефективності системи	• зведений кошторис витрат, пов'язаних з експлуатацією систем; • розрахунок річної економічної ефективності, джерелами якої є оптимізація виробничої структури господарства (об’єднання), зниження собівартості продукції завдяки раціональному використанню виробничих ресурсів і зменшенню втрат, поліпшення ухвалених управлінських рішень
Заходи з підготовки об’єкта до впровадження системи	• перелік організаційних заходів із вдосконалення бізнес- процесів; • перелік робіт з впровадження системи, які потрібно виконати на стадії робочого проєктування, із зазначенням термінів і відповідальних осіб

На стадії «робоча документація» створюють програмний продукт і розробляють всю супровідну документацію.

Документація повинна містити всі необхідні і достатні відомості для забезпечення виконання робіт з введення ІС в дію та її експлуатації, а також для підтримки рівня експлуатаційних характеристик (якості) системи.

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

Запитання для самоконтролю

1. Опишіть технічне завдання в життєвому циклі АС.
2. Хто із суб'єктів бере участь в розробленні ТЗ?
3. У чому полягає різниця між ЗП і ТЗ?
4. Охарактеризуйте склад і зміст технічного завдання.
5. Охарактеризуйте технічний проєкт системи та його склад.
6. На якій стадії створюють програмний продукт і розробляють супровідну документацію?

Лекція № 7. СУЧАСНІ МЕТОДОЛОГІЇ ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

7.1. Методологія RAD

На ранніх стадіях розвитку інформаційних технологій майже завжди розроблення ІС виконували на традиційних мовах програмування. Згодом внаслідок ускладнення систем, з появою розвинутого графічного інтерфейсу такий підхід втратив свою актуальність. Постала нагальна потреба у нових засобах, що спрямовані на скорочення термінів розроблення та високий рівень автоматизації щодо ІС. Ця обставина стала причиною виникнення цілого напряму у сфері розроблення програмного забезпечення – створення інструментальних засобів для швидкого розроблення застосунків і засобів автоматизації майже всіх етапів життєвого циклу інформаційних систем.

Багато з цих інструментів основані на так званій методології RAD.

Головні особливості методології RAD

Методологія створення інформаційних систем, основана на використанні засобів швидкого розроблення застосунків, 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 дають змогу оперувати з певним набором графічних об'єктів, які функціонально відображають окремі інформаційні компоненти додатків.

Методологія основана на трьох головних елементах (рис. 7.1):

1) невелика команда програмістів (від двох 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) стислий, але ретельно підготовлений виробничий графік (від двох до шести місяців);

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

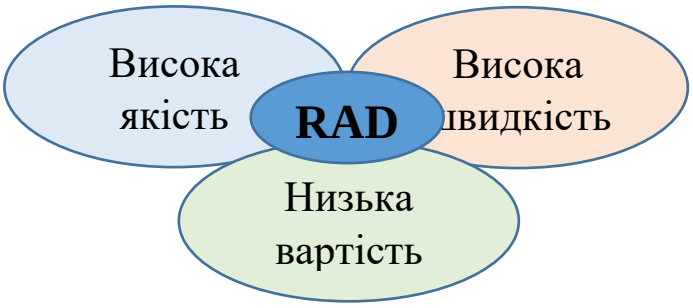

Рис. 7.1. Цілі методології RAD

Головні засади методології RAD:

- ✓ ітераційне розроблення програмного забезпечення;
- ✓ допустимість часткового завершення робіт на довільному етапі ЖЦ;
- ✓ обов'язкове залучення користувачів до процесу розроблення ІС;
- ✓ обов'язкове використання CASE-засобів з метою досягнення цілісності проєкту;

- ✓ застосування засобів управління конфігурацією, що спрощує внесення змін до проєкту, а також супровід готової системи;
- ✓ активне використання генераторів коду;
- ✓ впровадження прототипів задля кращого розуміння й задоволення потреб користувача;
- ✓ паралельне тестування, розроблення й розвиток проєкту;
- ✓ ведення розроблення невеликою, але добре організованою командою професіоналів (від двох до 10 осіб);
- ✓ чітке планування та контроль за виконанням робіт.

Об'єктноорієнтований підхід у методології RAD

Засоби RAD дали змогу реалізувати відмінну від традиційної технологію створення застосунків, коли інформаційні об'єкти формують як певні робочі моделі (прототипи). Функціонування цих прототипів узгоджують з користувачем, після чого розробник переходить до формування закінчених додатків.

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

Використання об'єктноорієнтованих принципів дає змогу 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 програмного 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 такому 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, бути керованим або мати певні відношення з іншими об'єктами.

Великої популярності об'єктноорієнтоване програмування (ОПП) набрало з появою засобів візуального програмування, які реально забезпечили злиття (тобто інкапсуляцію) даних з процедурами поведінки об'єктів. Це дало змогу створювати програмні системи,

максимально наближені до реальних і досягати найвищого рівня абстракції.

Під час розроблення програм за допомогою інструментів RAD використовують велику кількість готових об'єктів. Такі об'єкти зберігаються, як правило, у загальнодоступному сховищі. Водночас RAD-технологія дає можливість для створення нових об'єктів. Такі об'єкти можна писати як на основі наявних, так і незалежно від них.

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

Інструменти RAD дають розробникам змогу сконцентруватися на сутності реальних процесів, для яких створюють ІС, що сприяє підвищенню якості кінцевого продукту.

Візуальне програмування і методологія RAD

Застосування принципів ООП дало змогу створити принципово нові інструменти проєктування додатків, що дістали 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 можна легко пов'язати з базами даних і відобразити інформацію на екрані монітора. Інша група об'єктів є стандартними елементами управління, це, наприклад, кнопки, перемикачі, прапорці, меню тощо. За допомогою таких елементів управляють 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.

Нині є багато візуальних засобів розроблення застосунків.

Незважаючи на певну специфіку, їх поділяють на дві категорії: універсальні та спеціалізовані.

З *універсальних* систем візуального програмування найбільш поширеними є C, Java, Visual Basic і багато інших. Універсальними їх вважають тому, що вони допомагають у розробленні застосунків

практично будь-якого типу, зокрема ІС. До того ж програми, які розробляють за допомогою універсальних систем, можуть взаємодіяти майже з усіма відомими системами управління базами даних.

Спеціалізовані засоби орієнтовані тільки на створення застосунків 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 розроблення інтерфейсу багато в чому мають спільні риси, програміст дістав 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.

Подієве програмування і методологія RAD

Логіка застосунків, побудованих засобами RAD, є подієво-орієнтованою. Це означає, що кожен об'єкт зі складу програми може створювати події та реагувати на події, генеровані 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, виконуваної об'єктом у разі виникнення тієї чи іншої ситуації. Наприклад, обробник події «клацання на кнопці» може відкрити діалогове вікно. Отже, управління об'єктами відбувається за допомогою подій.

Життєвий цикл інформаційної системи за методологією RAD

Згідно з методологією 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 само обмежують масштаб проєкту, визначають терміни реалізації для кожної з фаз. Крім того, визначають саму можливість реалізації проєкту у межах доступного фінансування, наявних апаратних засобів тощо.

Результат цієї фази:

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 взяті до уваги 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 до розмежування доступу до інформації;

✓ визначають набір документації;

✓ оцінюють кількість функціональних елементів ІС і ухвалюють рішення щодо декомпозиції ІС на підсистеми, які здатна реалізувати одна команда розробників за прийнятний для RAD-проєктів час (60-90 днів);

✓ за допомогою CASE-засобів проєкт розподіляють між різними командами (формують функціональну модель).

Результат фази проєктування:

✓ 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 використані у подальшому під час створення системи. Ця вимога пояснюється тим, що в традиційному підході під час передавання інформації щодо проєкту може відбутися неконтрольоване спотворення даних.

Застосування єдиного середовища зберігання інформації про проєкт дає змогу уникнути такої небезпеки.

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

На фазі побудови виконують безпосередню реалізацію програми. На цій стадії розробники створюють реальну систему на підставі моделей та вимог нефункціонального значення, отриманих з попередніх етапів. Частину програмного коду формують за допомогою автоматичних генераторів. Кінцеві користувачі на цій фазі оцінюють проміжні результати та вносять коригування, якщо система на поточному етапі перестає задовольняти їхні вимоги. Тестують систему безпосередньо в процесі розроблення. По закінченню робіт команди розробників поступово інтегрують свої результати до загального. Таким чином формують повний програмний код ІС, після чого виконують тестування застосунків, а згодом – тестування системи загалом.

Завершення фізичного проєктування системи може відбуватися так:

1) визначають потребу в розподілі даних;

2) проводять аналіз використання даних;

3) виконують фізичне проєктування бази даних;

4) визначають вимоги до апаратних ресурсів;

5) визначають способи підвищення продуктивності;

6) завершують розроблення документації проєкту.

Результат фази побудови – готова система, що задовольняє заявлені вимоги.

На фазі впровадження відбувається навчання користувачів, організаційні зміни, паралельно з впровадженням нової системи виконують роботу з наявною системою (до остаточного 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-от:

– розробляється зовсім нова система;

– вже зроблено обстеження підприємства та є модель його діяльності;

– на підприємстві є робоча ІС, яку:

а) можна використати як прототип;

б) треба інтегрувати до розроблюваної ІС.

Обмеження методології RAD

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

Методологія RAD малоприйнятна в таких випадках:

1) для створення типових ІС, які не є завершеним програмним продуктом, а становлять сукупність типових елементів ІС; для систем, у яких 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 наявних розробок, коли потрібен високий рівень планування, жорстка дисципліна проєктування, суворе дотримання розроблених протоколів, інтерфейсів;

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

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

4) для розроблення 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 цілком працездатних версій системи.

7.2. Методологія RUP

Серед виробників CASE-засобів компанія IBM Rational Software Corp. однією з перших усвідомила стратегічну перспективність розвитку об’єктноорієнтованих 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’єктноорієнтований CASE-засіб, у якому реалізовано мову UML як базову нотацію візуального моделювання. Завдяки цьому виникла одна з найпопулярніших технологій проєктування інформаційних систем – Rational Unified

Process (RUP). Нині ця методологія у певному розумінні стає міжнародним стандартом від компанії Rational Software, що належить до складу IBM. Авторами UML вважають співробітників фірми Граді Буча, Айвара Якобсона, Джеймса Рамбо. RUP є цілком відповідною стандартам, що визначають проєктні роботи в процесі життєвого циклу ІС.

Підходи RUP

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

✓ ітераційний та інкрементальний (нарощувальний);

✓ побудова системи на основі архітектури інформаційної системи;

✓ планування і управління проєктом на підставі функціональних вимог до інформаційної системи.

Розроблення інформаційної системи відбувається ітераціями. Це окремі проєкти, невеликі за обсягом та змістом, містять власні етапи аналізу вимог, проєктування, реалізації, тестування, інтеграції.

Закінчуються ітерації створенням робочої інформаційної підсистеми.

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

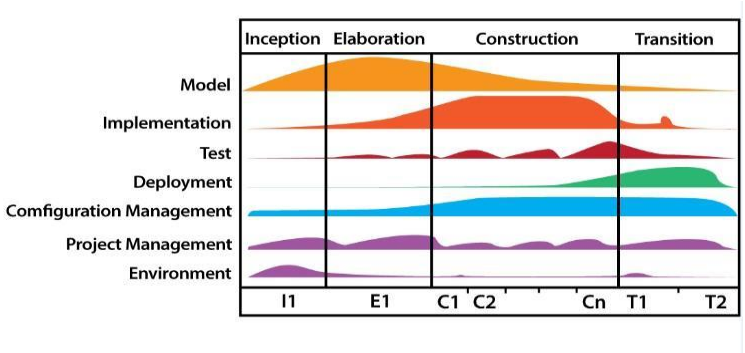

Рис. 7.2. Методологія RUP

Чотири фази життєвого циклу проєкту RUP

RUP визначає життєвий цикл проєкту, що складається з чотирьох фаз. Ці фази дають процесу змогу бути представленим на високому рівні, подібно до того, як представляють 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 метою є адекватна оцінка системи як бази для обчислення початкових розцінок та бюджету. На цьому етапі встановлюють бізнес-випадки, зокрема бізнес-контекст, фактори успіху (очікувані доходи, визнання на ринку тощо), а також фінансовий прогноз. На додаток до бізнес-випадку генерують базовау модель прецедентів, план проєкту, попередню оцінку ризику й опис проєкту (основні вимоги до проєкту, обмеження та основні характеристики). Після цього проєкт перевіряють на відповідність таким критеріям:

- зацікавлені сторони досягають згоди щодо визначення масштабів і оцінки вартості/термінів;
- розуміння вимог як свідчення якості первинних прецедентів;
- достовірність оцінок вартості/термінів, пріоритетів, ризиків та процесу розроблення;
- глибина і ширина будь-якого розроблення архітектурного прототипу;
- 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 підставі аналізу до кінця цієї фази. Фаза уточнення – фаза, коли проєкт починає набувати форми. На цьому етапі виконують аналіз предметної області, і архітектура проекту отримує свою базову форму.

Ця фаза має пройти віху життєвого циклу архітектури (LCA), задовольняючи певні 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 значущі прецеденти.

Бізнес-випадки та список ризиків переглядають.

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. Однак після закінчення цього етапу проєкт переходить в операцію з високим ступенем ризику, коли зміни набагато складніші та згубні в разі здійснення.

Системна архітектура є ключовим елементом розробки, яку отримують з аналізу предметної області.

Construction – фаза конструювання

Основна мета полягає у створенні програмної системи. На цьому етапі основну увагу приділяють розробленню 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 межах цієї фази охоплює навчання кінцевих користувачів та обслуговчого персоналу, бета-тестування системи для перевірки її на відповідність очікуванням користувачів. Продукт також перевіряють на відповідність рівню якості, встановленого в початковій фазі.

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

7.3. Стандарти проєктування інформаційних систем. Методологія CDM

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 виконання проєкту.

У такому разі за основу корпоративних стандартів можуть бути взяті 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. До них належать різноманітні методичні матеріали провідних фірм-розробників програмного забезпечення, фірм-консультантів, наукових центрів, консорціумів зі стандартизації тощо.

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

– на продукти й послуги;

– на процеси та технології;

– на форми колективної діяльності, або управлінські стандарти.

Наявність та впровадження стандартів є обов'язковою умовою для якісного проєктування та створення системи. Однак цілком зрозуміло, що самих лише стандартів для вирішення цього питання недостатньо.

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

Методика CDM фірми Oracle

Компанія Oracle добре відома на ринку програмного забезпечення як лідер із розроблення потужних СУБД та відповідних інструментів проєктування баз даних. Однак цим сфера інтересів Oracle не обмежується, оскільки одним із важливих напрямів її діяльності є розроблення методологічних основ та інструментальних засобів для автоматизації процесів проєктування та реалізації складних інформаційних систем, орієнтованих 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 відбувається на всіх етапах ЖЦ-системи, починаючи від загальних пропозицій і закінчуючи супроводженням готового продукту.

3) Перевагу віддають архітектурі «клієнт – сервер», зокрема складним структурам розподілених баз даних.

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

Дизайнер працює під управлінням СУБД Oracle. Репозиторій дає змогу під'єднатися до нього великій еількості користувачів із різними рівнями прав доступу за допомогою стандартних засобів СУБД Oracle. Завдяки цьому всі дії розробників стають строго узгодженими, а незалежні дії кожного з них неможливими.

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

6) Стандартні дії на етапах проєктування та розроблення автоматизовані. У будь-який момент може бути згенерований певний обсяг звітів про вміст сховища, потрібних для документування поточної версії системи на всіх етапах її розроблення. Існують спеціальні процедури, що дають змогу виконати перевірку специфікацій на повноту й несуперечність.

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. На цьому етапі виконують моделювання та аналіз процесів, які описують діяльність організації, особливості роботи. Кінцева мета – створення моделей процесів, виявлення можливостей їхнього вдосконалення. Етап не обов'язковий, якщо наявні

технології й 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. Далі формують моделі двох типів: інформаційні (вони відображають структуру та загальні закономірності предметної області) і функціональні (описують особливості вирішуваних завдань).

3) Проєктування. На цьому етапі перетворюють початкові вимоги до системи на докладні специфікації. Спеціальні утиліти, що належать до складу ПО Oracle, значно спрощують цю процедуру.

4) Реалізація. На цьому етапі розробляють і тестують програми, що належать до складу майбутньої інформаційної системи. ПО Oracle містить спеціальні генератори застосунків, які гранично автоматизують цей етап, зазвичай тривалий і найскладніший. Терміни розроблення значно скорочуються.

5) Впровадження. На цій стадії систему встановлюють, підготовляють 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 річ, що зручність засобів розроблення багато в чому виправдовує таку залежність.

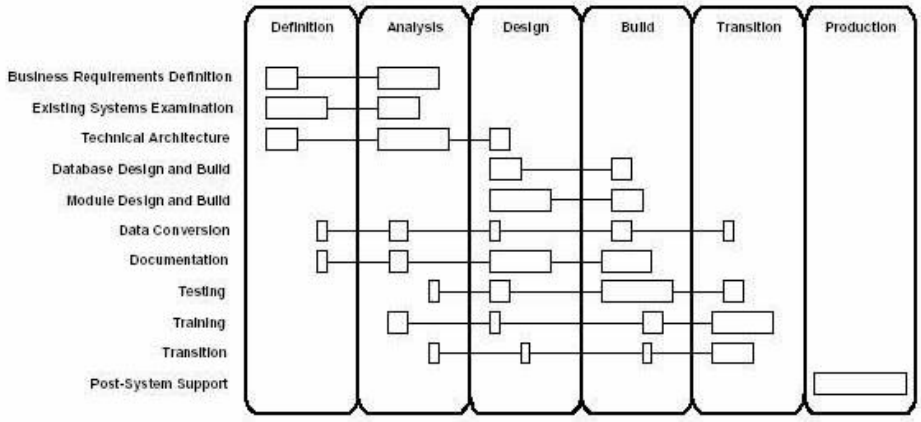

Рис. 7.3. Методологія CDM

Методика CDM виділяє такі процеси, що тривають протягом ЖЦ ІС:

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

Особливості методики CDM

Серед численних особливостей методики основними є такі:

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

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

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

4) методика не є обов'язковою, хоча і становить фірмовий стандарт, однак в разі використання ПЗ Oracle інший підхід видається малоймовірним;

5) методика спирається на використання ПЗ розробника від Oracle, пристосувати 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 проєктах інформаційних систем, що спираються на інструментальні засоби та СУБД фірми Oracle.

Запитання для самоконтролю

1. Назвіть сучасні методології проєктування інформаційних систем.
2. Назвіть головні засади методології RAD.
3. Які підходи методології RUP вам відомі?
4. Сформулюйте структуру життєвого циклу згідно з методологією CDM.
5. Охарактеризуйте корпоративні стандарти щодо єдиних правил організації технології та управління під час реалізації проєкту.

Лекція № 8. УНІФІКОВАНА МОВА ВІЗУАЛЬНОГО МОДЕЛЮВАННЯ UNIFIED MODELING LANGUAGE

Нині застосовують безліч технологій та інструментальних засобів, за допомогою яких можна реалізувати в певному сенсі оптимальний проєкт ІС, починаючи від етапу аналізу і закінчуючи створенням програмного коду системи. У більшості випадків ці технології містять досить жорсткі вимоги до процесу розроблення 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 дають змоги оптимізувати діяльність на рівні окремих елементів проєкту, і, як наслідок, багато розробників перейшли на так звані CASE-засоби нижнього рівня (lower CASE tools), однак зіткнулися з новою проблемою – проблемою організації взаємодії між різними командами, які втілюють проєкт.

Уніфікована мова об'єктноорієнтованого моделювання Unified Modeling Language (UML) є засобом досягнення компромісу між цими підходами. Достатня кількість інструментальних засобів підтримують за допомогою UML життєвий цикл інформаційних систем, одночасно UML є досить гнучким для налагодження і підтримання специфіки діяльності різних команд розробників.

Потужний поштовх до розроблення цього напряму інформаційних технологій дало поширення об'єктноорієнтованих мов програмування

наприкінці 1980-х – на початку 1990-х років. Користувачам хотілося отримати єдину мову моделювання, яка об'єднала б в собі всю міць об'єктноорієнтованого підходу і давала б чітку модель системи, що відображатиме всі її значущі 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 найкращим на стадії аналізу проблемної області та аналізу вимог до системи, OMT-2 був найкращий на стадіях аналізу і розроблення інформаційних систем, Booch найкраще підходив для стадій дизайну і розроблення.

Все йшло до створення єдиної мови, яка поєднувала б сильні сторони відомих методів і забезпечувала найкращу підтримку моделювання. Такою мовою виявилася 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, спільні зусилля трьох творців були спрямовані на створення UML.

В теперішній час консорціум користувачів UML Partners об’єднує представників таких грантів інформаційних технологій, як Rational Software, Microsoft, IBM, Hewlett-Packard, Oracle, DEC, Unisys, IntelliCorp, Platinum Technology.

UML є об'єктноорієнтованою мовою моделювання, їй притаманні такі основні характеристики:

✓ є мовою візуального моделювання, придатною до розроблення 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 досі підтримувана багатьма об'єктно-орієнтованими CASE-продуктами.

UML містить внутрішній набір засобів моделювання (модулів «Ядро»), які нині акцептовані в багатьох методах і засобах моделювання. Ці концепції потрібні в більшості прикладних задач, хоча не кожна концепція потрібна в кожній частині кожної програми. Користувачам мови надано такі можливості:

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

✓ додавати за потреби нові елементи й умовні позначення, якщо вони не входять в ядро, або спеціалізувати компоненти, систему умовних позначень (нотацію) й обмеження для конкретних предметних областей.

Синтаксис і семантика основних об'єктів. UML-Класи

Класи – це базові елементи будь-якої об'єктноорієнтованої системи. Класи являють собою опис сукупностей однорідних об'єктів з притаманними їм властивостями – атрибутами, операціями, відносинами і семантикою. В рамках моделі кожному класу дають 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 набувати безліч значень. Безліч допустимих значень атрибута утворює домен. Атрибут має ім'я і відображає деяку властивість, моделюються сутності, спільні для всіх об'єктів певного 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'язане 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 є явне або неявне відношення. У мові UML визначено три рівні видимості:

✓ public (загальний) – будь-який зовнішній клас, який «бачить» певний клас, може користуватися його загальними властивостями. Позначають знаком « + » перед ім'ям атрибута або операції;

✓ protected (захищений) – тільки будь-який нащадок даного класу може користуватися його захисними властивостями. Позначають знаком « # »;

✓ private (закритий) – тільки цей клас може користуватися такими властивостями. Позначають символом « - ».

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

✓ instance (екземпляр) – у кожного примірника класу є власне значення цієї властивості;

✓ classifier (класифікатор) – всі екземпляри спільно використовують загальне значення властивості (виділяється на діаграмах підкреслюванням).

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

✓ ті, що не містять жодного екземпляра – тоді клас стає службовим (Abstract);

✓ містять рівно один екземпляр (Singleton);

✓ містять задану кількість екземплярів;

✓ містять довільну кількість екземплярів.

Принципове призначення класів характеризують стереотипи. Це, фактично, класифікація об'єктів на високому рівні, що дає змогу визначити деякі основні властивості об'єкта (приклад стереотипу – клас «дійова особа»). Механізм стереотипів є також засобом розширення словника UML завдяки створенню на основі наявних блоків мови нових блоків, специфічних для вирішення конкретної проблеми.

Діаграма класів

Класи в UML зображуються на діаграмах класів, за допомогою яких можна описати систему в статичному стані – визначити типи об'єктів системи і різноманітні статичні зв'язки між ними.

Класи відображають типи об'єктів системи.

Між класами можливі різні відносини.

✓ залежності, які описують наявні між класами відносини використання;

✓ узагальнення, що пов'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 (батьком – клас «клієнт») і її конкретним втіленням (нащадком – класи «корпоративний клієнт» або «приватний клієнт»). Об'єкти класу-нащадка можуть використовуватися всюди, де трапляються об'єкти батьківського класу, але не навпаки. При цьому він успадковує властивості батька (його атрибути і операції). Операція, яку здійснює клас нащадка з тією самою сигнатурою, що й в одного із батьківського класу, заміщує операцію з батьківського класу – цю 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. За потреби напрямок навігації може задаватися стрілкою. Допускається завдання асоціацій на одному класі. В такому разі обидва кінці асоціації належать до одного й того самого класу. Це означає, що з об'єктом деякого класу можна пов'язати інші об'єкти того самого класу. Асоціації може бути надано ім'я, яке описує семантику відносин. Кожна асоціація має дві ролі, які можуть бути відображені на діаграмі. Роль асоціації має властивість множинності, яка показує, скільки відповідних об'єктів може брати участь в цих запитах.

Модель формування замовлення ілюструє рис. 8.1. Кожне замовлення може бути створене єдиним клієнтом (множинність ролі 1.1). Кожен клієнт може створити одне і більше замовлень (множинність ролі 1..n). Напрямок навігації показує, що кожне замовлення повинне бути «прив'язане» до певного клієнта.

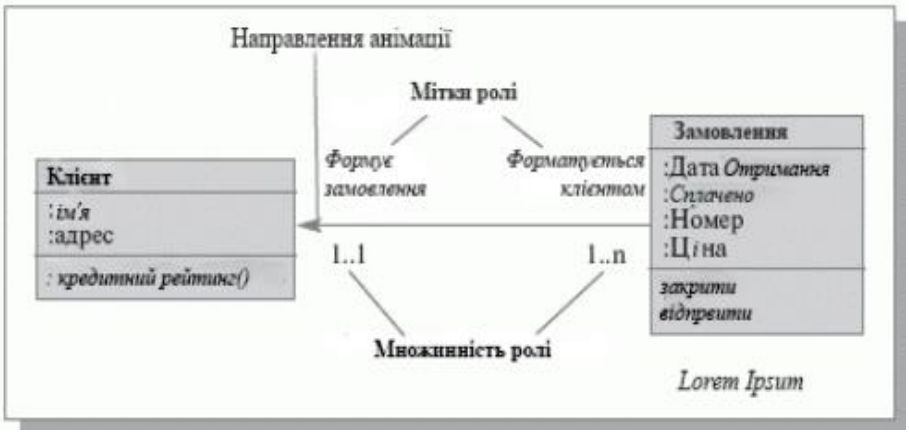

Рис. 8.1. Модель формування замовлення

Такого роду асоціація є простою і відображає відношення між рівноправними сутностями, коли обидва класи перебувають на одному концептуальному рівні і жоден з них не є більш важливим, ніж інший. Якщо доводиться моделювати відношення типу «частина – ціле», то використовують 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 використовують і сильніший різновид агрегації – композицію, 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 й операції, створюючи за звичайними правилами UML класи асоціацій.

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 типова взаємодія користувача з системою, яка при цьому

- ✓ описує видиму користувачем функцію,
- ✓ може представляти різні рівні деталізації,
- ✓ забезпечує досягнення конкретної мети, важливої для користувача.

Прецедент позначають на діаграмі овалом, пов'язаним з користувачами, яких називають дійовими особами (акторами, actors). Дійові особи використовують систему (або використовуються

системою) в прецеденті. Дійова особа виконує деяку роль в прецеденті. На діаграмі зображується тільки одна дійова особа, однак реальних користувачів, які виступають у цій ролі відносно ІС, може бути багато. Список всіх прецедентів фактично визначає функціональні вимоги до ІС, які лежать в основі розробки технічного завдання на створення системи.

На **діаграмах прецедентів** (рис. 8.2), крім зв'язків між дійовими особами і прецедентами, можливе використання ще двох видів зв'язків між прецедентами: «використання» і «розширення». Зв'язок типу «розширення» застосовують, коли один прецедент подібний до іншого, але має трохи більше функціональне навантаження. Його слід застосовувати для опису змін в нормальній поведінці системи. Зв'язок типу «використання» дає змогу виділити якийсь фрагмент поведінки системи і додавати його в різні прецеденти без повторного опису.

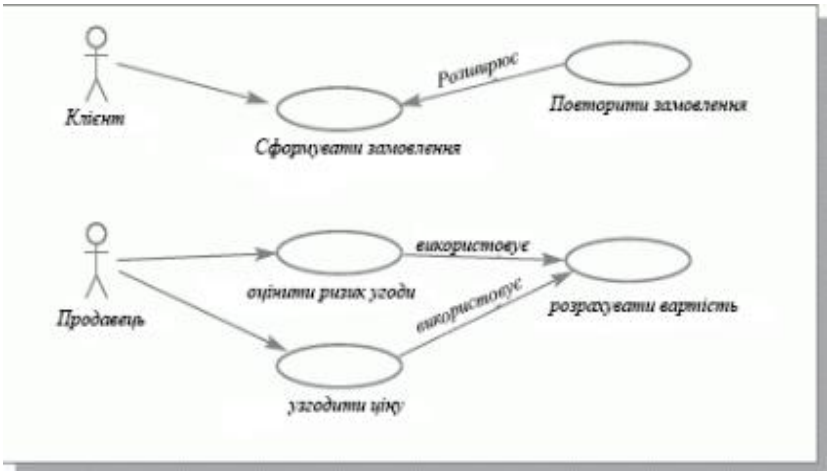

Рис. 8.2. Діаграма прецедентів

Динамічні аспекти поведінки системи відображаються наведеними далі 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 їхнє читання. Потік повідомлень між об'єктами відображається на діаграмах взаємодії. Звичайно діаграма взаємодії охоплює поведінку об'єктів в межах одного варіанта 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.

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

Діаграми послідовностей

Цей вид діаграм використовується для точного визначення логіки сценарію виконання прецеденту. Діаграми послідовностей (рис. 8.3) відображають типи об'єктів, що взаємодіють під час виконання 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.

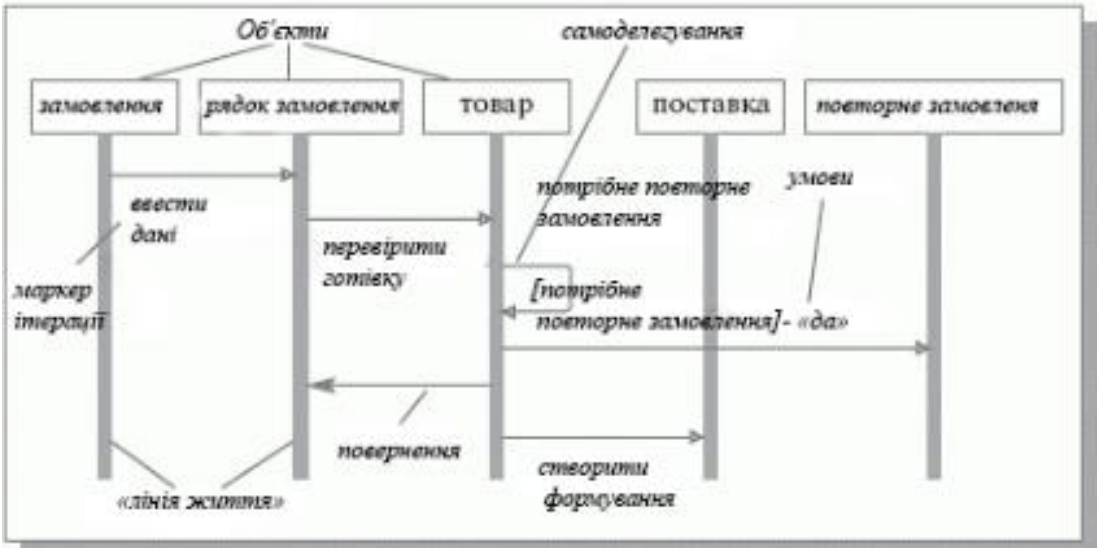

Рис. 8.3. Діаграма послідовності обробки замовлення

Реалізація обробки замовлення засобами діаграм послідовностей:

- ✓ вводять рядки замовлення;
- ✓ за кожним рядком перевіряють наявність товару;
- ✓ якщо запас достатній – ініціюють поставку;
- ✓ якщо запас недостатній – ініціюють дозамовлення (повторне замовлення).

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

На діаграми може бути додана керівна інформація: опис умов, за яких надсилається повідомлення; ознака багаторазового відправлення повідомлення (маркер ітерації); ознака повернення повідомлення.

Кооперативні діаграми (рис. 8.4)

На кооперативних діаграмах об'єкти (або класи) показують у вигляді прямокутників, а стрілками позначають повідомлення, якими вони обмінюються в рамках одного варіанта використання. Тимчасова послідовність повідомлень відбивається їхньою нумерацією.

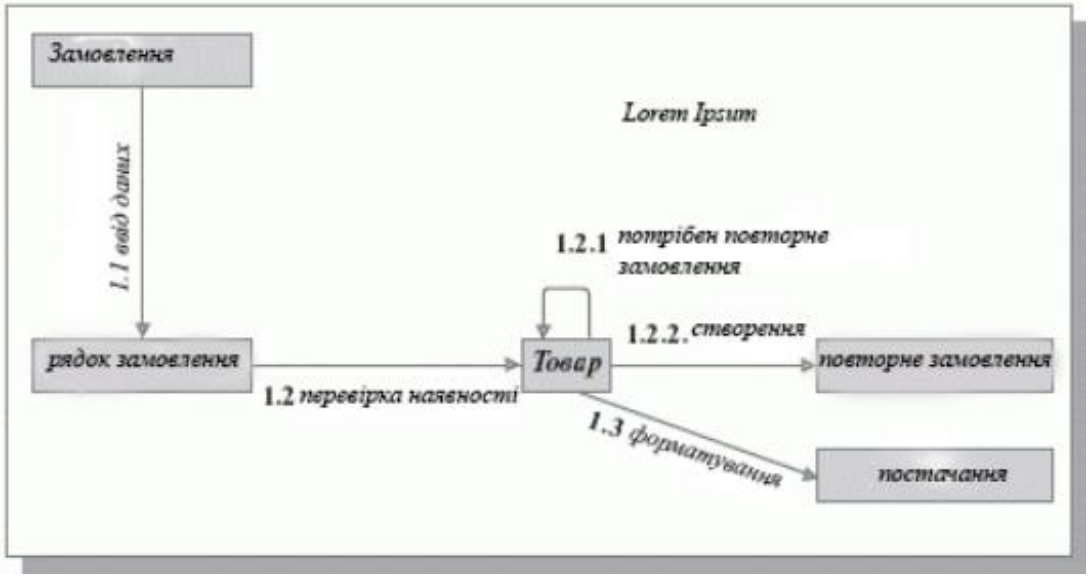

Рис. 8.4. Кооперативна діаграма проходження замовлення

Діаграми станів

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

Прямокутниками відображають стани, через які проходить об'єкт. Станам відповідні певні значення атрибутів об'єктів. Стрілки показують переходи від одного стану до іншого, зумовлені виконанням деяких функцій об'єкта. Є також два види псевдостанів: початковий стан, в

якому перебуває 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.

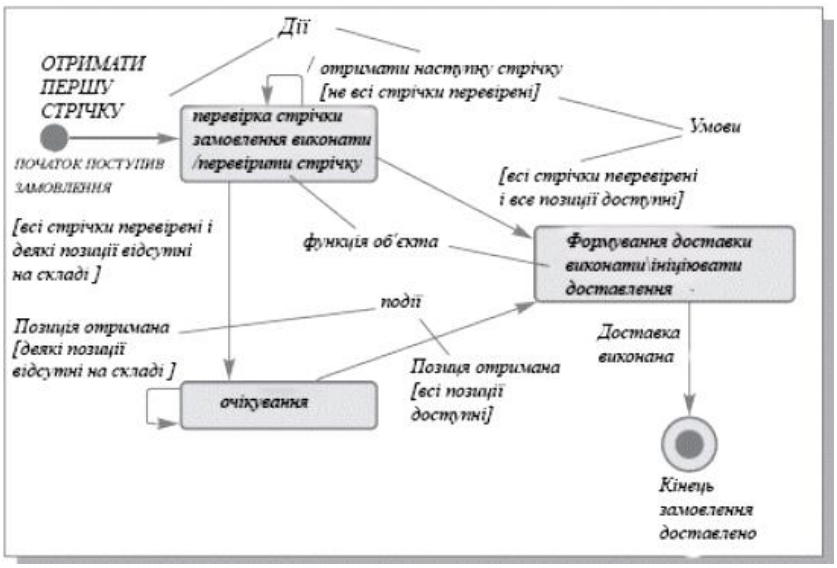

Рис. 8.5. Діаграма станів об'єкта «замовлення»

Діаграми діяльності

Діаграма діяльності – це окремий випадок *діаграми станів.* На діаграмі діяльності (рис. 8.6) представлено переходи потоку керування від однієї діяльності до іншої всередині системи. Цей вид діаграм зазвичай використовують для опису поведінки, що являє собою безліч паралельних процесів.

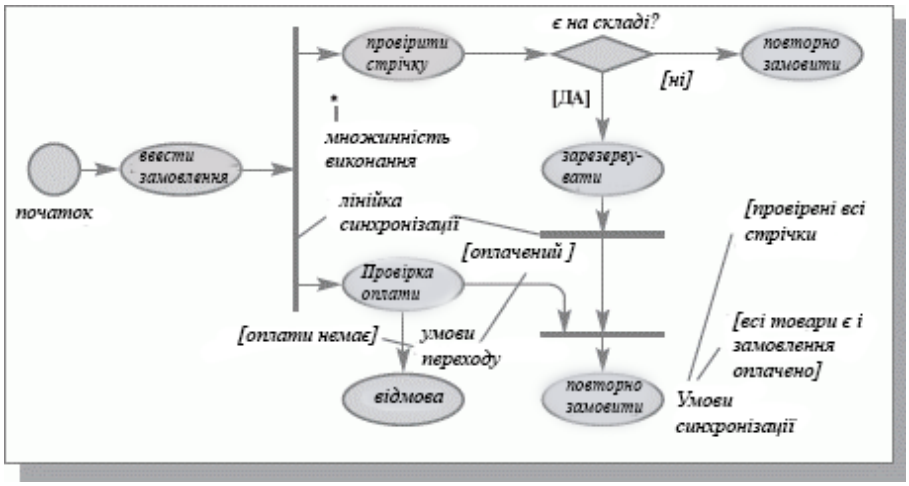

Рис. 8.6. Діаграма діяльності «обробка замовлення»

Основні елементи діаграм діяльності такі:

- ✓ овали, якими зображують дії об'єкта;
- ✓ лінійки синхронізації, що вказують на потребу завершити або розпочати кілька дій (модель логічної умови «І»);
- ✓ ромби, що відображають ухвалення рішень у виборі одного з маршрутів виконання процесу (модель логічної умови «АБО»);
- ✓ стрілки відображають послідовність дій, можуть мати мітки умов.

На діаграмі діяльності можуть бути представлені дії, відповідні кільком варіантам використання. На таких діаграмах з'являється безліч початкових точок, оскільки вони відображають тепер реакцію системи на безліч зовнішніх подій. Таким чином, діаграми діяльності дають змогу отримати повну картину поведінки системи і легко оцінювати вплив змін в деяких випадках використання на кінцеву поведінку системи.

Будь-яка діяльність може зазнати подальшої декомпозиції і бути представленою у вигляді окремої діаграми діяльності або специфікації (словесного опису).

Діаграми компонентів

Діаграми компонентів дають змогу зобразити модель системи на фізичному рівні.

Елементами діаграми є компоненти – логічний блок системи, за рівнем абстракції трохи вищий, ніж «клас». Позначається 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, що зазвичай має як мінімум деякий об'єм пам'яті, а часто ще й здатність до обробки.

Вузли поділяють на два типи:

- ✓ пристрої – вузли системи, в яких дані не обробляються;
- ✓ процесори – вузли системи, що виконують 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.

Компонент зазвичай 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 дає змогу створити ядро системи, яка не буде змінюватися у відповідь на зміни, що відбуваються на рівні підсистем.

На рис. 8.7 наведено спрощену схему елементів фрагмента корпоративної системи. «Коробки» являють собою компоненти – додатки або внутрішні підсистеми. Пунктирні лінії відображають залежності між компонентами.

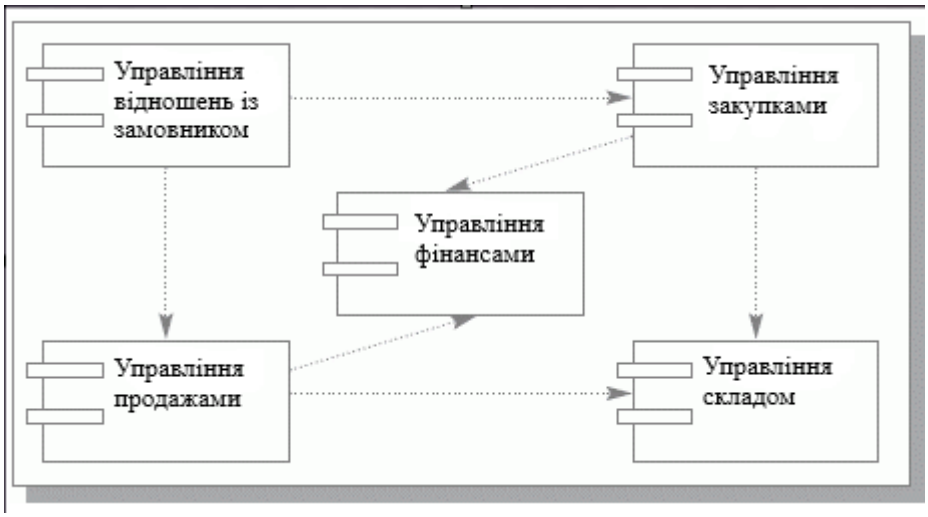

Рис. 8.7. Діаграма компонентів фрагмента КІС

Кожному компоненту діаграми (за потреби) належить документація 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 їхню 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 виконують за два етапи:

✓ дослідна експлуатація, після якої виконують приймальні випробування

✓ передавання в постійну експлуатацію.

ІС впроваджують лише на підприємстві, що працює.

Наказом замовника, погодженим із розробником, визначають терміни виконання робіт і склад приймальної комісії. До наказу додають програму робіт і план-графік, у яких зазначають умови і кількість рішень задач, порядок перевірки технічних засобів під час розв'язування задач, порядок усунення недоліків, виявлених під час дослідної експлуатації.

Мінімальна тривалість дослідної експлуатації залежно від потрібної частоти вирішення функціональних задач має бути відповідна 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 довше. ніж п'ять діб, чи до 15% планової кількості рішень
Від одного разу на місяць до одного разу на добу	Три місяці	Не більш як третина планової кількості рішень
Менш ніж раз 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, в який записують всі порушення й усі рішення, ухвалені виконавцями.

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

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

Після позитивних результатів дослідної експлуатації провадять приймальні випробування, які водночас є здачею проєкту 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 проєкту «Програми», його розгляду і затвердження погоджують з замовником.

Наводять обов'язкове для виконання випробувань забезпечення (нормативна, методична та інша документація, програмні та технічні засоби, метрологічне, спеціальне та інше обладнання, створення інших умов для проведення випробувань), сторону, що його надає, порядок усунення зауважень та ін.

Наводять також перелік документів, якими завершують 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 частіше, повинні відбуватись в режимі нормальної експлуатації, решту випробовують на тестових прикладах.

Випробування охоплюють:

- 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 складають протокол випробувань.

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

Усю документацію на ІС, а також програмне забезпечення (на машинних носіях) передають замовнику не менш як у двох примірниках.

9.2. Супровід і модернізація інформаційних систем

Після передачі ІС у постійну експлуатацію всю відповідальність за її функціонування несе замовник. Гарантійний термін встановлюють до 18 місяців від 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 виконують у два етапи.

1. Гарантійне обслуговування. Виконання робіт відповідно до гарантійних зобов'язань:
    - роботи з усунення недоліків, виявлених у процесі 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 обслуговча організація виконує діагностику стану системи.

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

- складають дефектну відомість на складові/елементи, які підлягають заміні/виправленню;
- виготовляють, комплектують і постачають потрібне устаткування;
- виконують регулювальні і ремонтні роботи;
- відновлюють застарілу технічну документацію.

Виконують такі роботи:

- аналіз 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 змін у документацію на ІС чи розроблення нової модифікації ІС.

Запитання для самоконтролю

1. Назвіть етапи впровадження ІС.
2. Назвіть стадії дослідної експлуатації. Які суб’єкти залучені до її проведення? 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: навч. посіб. / В.Б. Задоров. – Київ : КНУБА, 2003. – 276 с.
2. Шаховська Н.Б. Проектування інформаційних систем: навч. посіб. / Н.Б. Шаховська, В.В. Литвин. – Львів : Магнолія-2006, 2011. – 380 с.
3. Пасічник В.В. Проектування інформаційних систем: навч. посіб. / В.В. Пасічник, В.В. Литвин, Н.Б. Шаховська. – Львів, 2013. – 380 с.
4. Томашевський О.М. Інформаційні технології та моделювання бізнес процесів: навч. посіб. / О.М. Томашевський, М.Б. Цегелик, Г.Г. Вітер, В.І. Дубук. – Київ : Центр учбової літератури, 2005. – 296 с.
5. Розробка програмних проєктів 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. / за ред. С.Д. Бушуєва. – 2-ге вид., перероб. – Київ : Ділова Україна, 2000. – 198 с.
7. Richards M. Fundamentals of Software Architecture: An Engineeri ng 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

Редагування і коректура *Г.В. Кобриної*
Комп’ютерне верстання *Т.І. Кукарєвої*

Підписано до друку 06.12.2024. Формат 60 × 84 1/ 16
Ум. друк. арк 5,58. Обл.-вид. арк 6,0.
Електронний документ. Вид. № 22/І–24.

Видавець і виготовлювач
Київський національний університет будівництва і архітектури

Проспект Повітряних Сил, 31, Київ, Україна, 03037

Свідоцтво про внесення до Державного реєстру суб’єктів
видавничої справи ДК № 808 від 13.02.2002 р.