МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ МІСЬКОГО ГОСПОДАРСТВА імені О. М. БЕКЕТОВА

М. Ю. Карпенко, Н. О. Манакова, І. О. Гавриленко

ТЕХНОЛОГІЇ СТВОРЕННЯ ПРОГРАМНИХ ПРОДУКТІВ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

НАВЧАЛЬНИЙ ПОСІБНИК

Харків – ХНУМГ ім. О. М. Бекетова – 2017

УДК [004:681.518](075)
К26

***Автори*:**

***Карпенко М. Ю.*,** кандидат технічних наук, доцент;
***Манакова Н. О.*,** кандидат технічних наук, доцент;
***Гавриленко І. О.*,** асистент

***Рецензенти*:**

***Костенко О. Б.*,** канд. фіз.-мат. наук, доцент кафедри прикладної математики і інформаційних технологій Харківського національного університету міського господарства імені О. М. Бекетова;
***Грицунов О. В.*,** д-р фіз.-мат. наук, професор кафедри мікроелектроніки електронних приладів і пристроїв Харківського національного університету радіоелектроніки (ХНУРЕ)

Рекомендовано до друку
Вченою радою ХНУМГ ім. О. М. Бекетова як навчальний посібник
(протокол № 12 від 28 квітня 2017 р.)

Карпенко М. Ю.

К26 Технології створення програмних продуктів та інформаційних систем : навч. посібник / М. Ю. Карпенко, Н. О. Манакова, І. О. Гавриленко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2017. – 93 с.

У посібнику розглянуто питання щодо сучасних підходів до проектування інформаційних систем, наведено приклади їхньої реалізації через методики проектування від провідних розробників складних програмних продуктів, описані головні стандарти та нормативно-методичне забезпечення процесу проектування на всіх етапах життєвого циклу інформаційної системи. Навчальний посібник може використовуватись студентами та викладачами для формування відповідних компетенцій, що відповідають галузі знань 12 – Інформаційні технології спеціальності 122 – Комп'ютерні науки та інформаційні технології.

УДК [004:681.518](075)

ЗМІСТ

ПЕРЕДМОВА

Сучасний етап розвитку світового суспільства щільно пов'язаний зі стрімким розвитком інформаційних технологій. Складність інформаційних систем (ІС) невпинно зростає, відповідно набувають актуальності питання щодо ефективного управління процесами створення, тестування та впровадження таких систем. Комплексне вирішення цієї проблеми – складна й довготривала робота, що потребує високої кваліфікації задіяних у ній працівників. Адже, незважаючи на великий прогрес, багато питань у сфері автоматизації проектування ІС не піддається повній автоматизації й виконується на інтуїтивному рівні, на досвіді та прогностичних здібностях фахівців, експертних оцінках, експериментальній перевірці якості функціонування системи тощо.

Слід зауважити, що експлуатаційні витрати на впровадження й супроводження складної ІС можуть перевищувати витрати на її розробку. Досвід свідчить, що пошук помилок, які відбувалися на ранніх стадіях аналізу та проектування системи з часом зростає у геометричній прогресії. Наприклад, помилки передпроектувального аналізу, що виявлені на стадії реального проектування, «коштують» приблизно вдвічі дорожче. Виявлення таких помилок на етапі тестування – вже у 10 разів дорожче. А усунення помилок на етапі експлуатації може коштувати у 100 разів дорожче, ніж своєчасне виявлення їх на етапі аналізу системи. Крім того, знаходження системних помилок на етапі експлуатації пов'язано з роботою замовників (тобто реальних користувачів), що негативно впливає на імідж проекту та подальшу експлуатацію системи. Тому вибір методології проектування, засобів супроводження інформаційної системи, тестування її окремих складових на всіх стадіях розробки є складним та відповідальним питанням, від якого залежить успішність та ефективність проекту загалом. Саме виходячи з таких положень й було складено навчальний посібник «Технології створення програмних продуктів та інформаційних систем».

Мета посібника: дати студентам головні теоретичні положення щодо розроблення ІС та їх головного компоненту, – програмного забезпечення (ПЗ), ознайомити із сучасними підходами до вирішення цієї проблеми, зі складом та змістом технологічних операцій щодо створення ІС на різних рівнях проектування та реалізації, із засобами автоматизації проектних робіт, підходами до формалізації процесу проектування, методами управління проектуванням ІС тощо.

Перша частина посібника включає чотири розділи, а саме:

– головні поняття дисципліни. архітектура інформаційних систем;

– життєвий цикл інформаційної системи;

– методологія і технологія розроблення інформаційних систем;

– організація проектування та нормативно-методична підтримка життєвого циклу інформаційних систем.

У першому розділі посібника розглянуто головні поняття щодо ІС, подано їхню класифікацію та сфери застосування, наведено відомості щодо функціональних складових інформаційної систем та їхньої архітектури.

Другий розділ присвячено питанням життєвого циклу ІС та найважливіших процесів, що там відбуваються. У цьому розділі описано взаємозв'язок між процесами життєвого циклу, наповнення та особливості кожного з цих процесів, проаналізовано найпоширеніші моделі життєвого циклу, їх недоліки, переваги та особливості практичного використання.

У третьому розділі посібника «Технології створення програмних продуктів та інформаційних систем» розглянуто головні методології проектування ІС, складові технології проектування ІС, описано ключові принципи та засади структурного підходу до проектування ІС та об'єктно-орієнтованого аналізу, наведено перелік та зміст головних фаз проектування ІС, зокрема: аналіз предметної області, розроблення технічного завдання (ТЗ), технічне й робоче проектування ІС, введення системи в експлуатацію тощо. Також у цьому розділі описано такі методології проектування інформаційних систем, як RAD, RUP, методику CDM від фірми Oracle.

Четвертий розділ першої частини посібника присвячено питанням стандартизації процесів розробки та супроводження ІС. У ньому описано різновиди та класифікацію стандартів для створення ІС, розглянуто головні міжнародні та вітчизняні стандарти проектування ІС, стандарти на процеси життєвого циклу, корпоративні стандарти проектування тощо. Окремим питанням виділено канонічний підхід до проектування ІС та його відмінності.

Навчальний посібник може використовуватись студентами та викладачами для формування відповідних компетенцій, що відповідають галузі знань «12 Інформаційні технології» спеціальності «122 Комп'ютерні науки та інформаційні технології».

1 ГОЛОВНІ ПОНЯТТЯ ДИСЦИПЛІНИ. АРХІТЕКТУРА ІНФОРМАЦІЙНИХ СИСТЕМ

Головні питання: *поняття інформаційної системи, сфери застосування і приклади реалізації ІС. Функціональні підсистеми ІС: контур планування, контур оперативного управління, контур обліку та контролю, контур аналізу. Забезпечуючі підсистеми. Класифікація інформаційних систем. Класифікація за масштабом: поодинокі, групові, корпоративні ІС. Класифікація ІС за сферою застосування: системи обробки транзакцій, системи підтримки прийняття рішень, інформаційно-довідкові системи, офісні інформаційні системи. Класифікація за способом організації архітектури ІС: файл-серверна архітектура ІС, клієнт-серверна архітектура ІС, багаторівнева архітектура ІС, архітектура ІС на основі Інтернет-технологій.*

1.1 Значення та напрями розвитку інформаційних систем

Сьогодні інформаційні ресурси є не менш суттєвими, ніж трудові, матеріальні або енергетичні. Роль інформації настільки значна, що ми можемо говорити про інформаційну економіку (тобто таку, що ґрунтується на інформації), та інформаційну сферу управління (керівників, вчених, спеціалістів і службовців тощо). В такому аспекті управління економікою подібне управлінню живим організмом, де головну функцію покладено на нервову систему.

Функцію такої «нервової системи» в економіці здійснюють інформаційні потоки. Будь-яке порушення в потоках інформації обов’язково дасть взнаки у вигляді перебоїв у роботі конкретного підприємства, або навіть на рівні всього господарства. Без достовірної, повної та своєчасно отриманої інформації неможливо керувати будь-яким виробництвом, своєчасно реагувати на виклики зовнішнього середовища. Саме тому на сьогодні проблема збирання та обробки інформації є одним з ключових завдань управління, вирішити яку можна виключно на базі використання сучасної обчислювальної техніки та новітніх інформаційних технологій. Слід відзначити, що сама обчислювальна техніка все ширше використовується як в управлінні виробничими процесами, так і на рівні економічної системи в цілому. Однак матеріальні витрати на зберігання, передавання та перероблення інформації все частіше перевищують аналогічні витрати, наприклад, на енергетику. І в цьому контексті проблема інформатизації набирає не тільки технічну, а й економічну складову.

Принциповим є той факт, що комп’ютеризація суспільства – це не тільки техніка, а й люди. Нова комп’ютерна ідеологія повинна обіймати не тільки технічні питання, вона має охоплювати у т. ч. підготовку відповідних

спеціалістів і керівників. Керівник, що не володіє комп'ютером, не може сподіватися на прогрес у своїй роботі. Таким чином, комп'ютеризація у загальному розумінні потребує комплексного підходу щодо вирішення широкого спектру проблем, – від суто технічних до освітніх та соціальних. Саме для вирішення таких завдань в Україні було створено Національне агентство по питаннях інформатизації при Президентові України та затверджені закони «Про Національну програму інформатизації» (74/98-ВР від 04.02.98) і «Про концепцію Національної програми інформатизації» (75/98-ВР від 04.02.98).

1.2 Головні поняття інформаційних технологій і систем

1.2.1 Визначення ІТ

В основі комп'ютеризації буд-якого бізнес-процесу лежить інформаційна технологія (ІТ). У широкому розумінні ІТ – це загальний термін, що позначає різні технології оброблення та передачі інформації [1–5]. Зазвичай під ІТ розуміють комп'ютерні технології, а саме – методи застосування обчислювальної техніки (ОТ) і комп'ютерного програмного забезпечення (ПО) під час роботи з інформацією (виконанні функцій: збирання, зберігання, захисту, обробки, передавання й використання даних). До ІТ належать у т. ч. телекомунікація і мікроелектроніка.

Згідно з визначенням ЮНЕСКО, інформаційна технологія – це комплекс взаємопов'язаних наукових й інженерних дисциплін, що вивчають ОТ, способи взаємодії її з людьми та виробничим обладнанням, способи ефективної організації праці людей, а також пов'язані з усім цим соціальні, економічні й культурні проблеми.

За визначенням ІТАА, Американської асоціації з інформаційних технологій (інформаційні технології асоціації Америки), ІТ – це вивчення, проектування, розроблення, впровадження, підтримка або управління комп'ютерними інформаційними системами, включаючи програмні додатки і комп'ютерне апаратне забезпечення. Бурхливе зростання галузі ІТ доводиться на кінець 1990-х років, пов'язаний із появою та розвитком інформаційно-комунікаційних технологій.

Деякі автори дають таке визначення терміну «інформаційна технологія»: цілеспрямована організована сукупність інформаційних процесів з використанням засобів обчислювальної техніки, що забезпечують високу

швидкість обробки даних, пошуку інформації, розосередження даних й, доступ до джерел інформації незалежно від місця їхнього розташування.

Незважаючи на певну відмінність у визначеннях, будь-яка інформаційна технологія спрямована на вирішення завдань за трьома основними напрямками, а саме:

1) персоналізація розрахунків;

2) використання баз даних, експертних систем і баз знань;

3) застосування мереж передавання даних.

Особливості втілення кожного з вказаних напрямів ми розглянемо у наступних розділах нашої роботи.

Слід зауважити, що останнім часом набув популярності відносно новий термін, – «інформаційно-комунікаційні технології» (в подальшому ІКТ). Під ІКТ мають на увазі всі технічні засоби обробки та обміну інформацією [3–5] в поєднанні з відповідним математичним та програмним забезпеченням. Крім того, цей термін формально включає «доцифрові» технології, зокрема паперові. Також поняття ІКТ використовують для опису цифрових технологій у т. ч. способів комунікації (комунікаційні протоколи, комунікаційне обладнання, медіа), способів зберігання та обробки інформації (обчислення, накопичення даних) тощо. Є думка, що термін «інформаційно-комунікаційні технології» став популярним внаслідок об’єднання ІТ та телекомунікаційних технологій.

1.2.2 Визначення та різновиди інформації

Інформація (від лат *informatio* – «навчання», «зведення», «сповіщення») – це відомості, які передаються системою знаків будь-якого роду. Поняття «інформація» прийнято визначати через її властивості, тобто інформація – це явище, яке характеризується наявністю джерела, приймача, каналу зв’язку тощо [1, 2]. До того ж є безліч властивостей інформації, що досі не вивчені наукою. Вищевказане суттєво ускладнює формулювання точного визначення поняття «інформація».

Інформація, зафіксована на матеріальних носіях і зберігається в ІС (бібліотеках, архівах, сховищах, фондах, банках даних, системах знань тощо) має назву інформаційні ресурси.

Інформацію умовно можна розділити на групи за сферою використання [1–3]. Наприклад, економічна інформація – це сукупність відомостей про соціально-економічні процеси, що служать для управління ними й колективами людей. Характеристики економічної інформації: великі обсяги, багаторазове повторення циклів її отримання і перетворення у встановлені періоди (місяць, квартал, рік тощо), Різноманіття джерел і споживачів, значна питома вага рутинних процедур під час її обробки.

Інформація в ІС може бути структурованою або неструктурованою та умовно поділяється на дані та на знання.

Структуровані дані – це інформація у вигляді чисел і тексту, що зберігається в нормалізованих БД. Над такими даними можна виконувати різні операції.

До неструктурованої інформації можна віднести, наприклад, офісні електронні документи в форматі в Word або Excel, PDF, а також малюнки, креслення, графіки, скановані зображення, повідомлення електронної пошти, веб-сторінки, відео та іншу інформацію в електронному вигляді.

На відміну від структурованих даних (які зазвичай знаходяться в середовищі реляційної або багатовимірної СУБД) для зберігання неструктурованої інформації використовують файлову систему ОС або об’єктно-орієнтовані СУБД.

Слід наголосити, що між термінами «інформація» і «дані» є суттєві відмінності. Дані – це окремі факти, що характеризують об’єкти, процеси та явища предметної області, а також їхні властивості. Іншими словами дані – це інформація, що фіксована в певній формі, яка є придатною для подальшої обробки, зберігання та передавання (зберігається в базах даних, обробляється прикладними програмами тощо).

1.3 Поняття інформаційної система та класифікація інформаційних систем

Інформаційна система (Information system, у подальшому ІС) – це сукупність організаційних і технічних засобів для збереження та обробки інформації з метою забезпечення інформаційних потреб користувачів. Згідно до ДСТУ 2392-94:Інформаційна система під ІС мають на увазі

комунікаційну систему, що забезпечує збирання, пошук, оброблення та пересилання інформації.

Інформаційні системи можуть відрізнятись за типами об'єктів управління, характером і обсягом завдань тощо. Відповідно до цього ІС варто класифікувати за певними ознаками, що подано нижче.

1.3.1 Класифікація за рівнем або сферою діяльності

За рівнем (масштабом діяльності) розрізняють такі ІС: державні, регіональні, галузеві, системи об'єднань (підприємств або установ), технологічних процесів.

Державні ІС призначені для складання поточних й перспективних планів розвитку країни, для обліку результатів діяльності окремих ланок народного господарства, для регулювання цими ланками тощо. За допомогою ІС цього рівня розроблюють державний бюджет країни, контролюють його виконання. Прикладами ІС систем державного рівня є: система управління національним банком (АСУ банк), автоматизована система державної статистики (АСДС), автоматизована система планових розрахунків (АСПР). державна ІС фінансових розрахунків (АСФР) при Міністерстві фінансів України, система обробки науково-технічної інформації (АСО НТІ) тощо.

Регіональні ІС орієнтовані на управління певним адміністративно-територіальним регіоном. Це може бути ІС області, міста, району. Такі системи обробляють інформацію, яка потрібна для управління регіоном, також вони формують звітність, надають оперативні дані місцевим державним та господарським органам.

Галузеві системи орієнтовані на управління конкретними підприємствами та організаціями. Ці ІС працюють у промисловості та в сільському господарстві, будівництві й на транспорті тощо. Метою більшості галузевих систем є розв'язання завдань інформаційного обслуговування органів управління галузевих міністерств та їхніх підрозділів. Галузеві ІС поділяють за сферами застосування (промислова, непромислова, наукова тощо).

Інформаційні системи управління підприємствами (далі ІСУП) або виробничими об'єднаннями (далі ІСУ ВО) спрямовані на регулярне розв'язання завдань управління виробничо-господарського діяльністю підприємств із застосуванням сучасних засобів автоматизованої обробки

даних, економіко-математичних, статистичних методів, інструментів моделювання тощо.

Інформаційні системи управління технологічними процесами (ІСУ ТП) мають за мету управління станом технологічних процесів (це може бути робот, плавильна піч, верстат тощо). Такі системи мають дві важливі відмінності:

– особливий об’єкт управління, – для ІСУ ТП це насамперед різне обладнання;

– специфічну форму передавання інформації, – в ІСУ ТП головною формою передавання інформації є сигнал, у той час як в інших ІСУ інформація передається через документи.

1.3.2 Класифікація за рівнем автоматизації процесів управління

Враховуючи мету та завдання, що покладені на ІС, розрізняють: пошукові, довідкові, управлінські (керівні), інтелектуальні інформаційні системи та системи підтримки прийняття рішень. [1–3].

Інформаційно-пошукові системи (ІПС) орієнтовані на розв’язування задач пошуку інформації. Змістова обробка інформації у таких системах незначна, або ж навіть відсутня.

В інформаційно-довідкових системах (ІДС) за результатами пошуку здійснюються певні обчислення (тобто є первинна обробка даних). У більшості випадків така обробка нескладна.

Управлінські (інформаційно-керівні) системи відомі у вітчизняній літературі під назвою «автоматизовані системи організаційного управління». Вони представляють собою організаційно-технічні системи, що забезпечують пошук управлінських рішень на основі автоматизації інформаційних процесів [1–3]. Ці системи мають за мету автоматизоване розв’язання низки управлінських завдань.

До інформаційних систем нового покоління належать системи підтримки прийняття рішень (СППР) та інформаційні системи, що побудовані на штучному інтелекті (інтелектуальні ІС).

СППР – це інтерактивна комп’ютерна система, яка має за мету підтримку прийняття управлінських рішень за умов існування погано структурованих або взагалі неструктурованих проблем. Системи такого класу

здатні суттєво підвищити ефективність праці в управлінні різними секторами економіки. Відповідно зацікавленість до цих систем постійно зростає, а розробка та реалізація СППР де-факто стала окремою галуззю інфо-бізнесу, який останнім часом показує чи не найбільшу динаміку розвитку.

Інтелектуальні ІС відрізняються тим, що в процесі своєї роботи імітують вирішення людиною складних, нетривіальних, навіть творчих завдань. Створенню таких систем сприяло розроблення в теорії штучного інтелекту т. з. логіко-лінгвістичних моделей. Ці моделі дають змогу формалізувати певні змістовні знання про об'єкти та процеси управління й надали використовувати ці знання поряд із «класичними» математичними моделями. Логіко-лінгвістичні моделі (семантичні мережі, фрейми, нейросистеми тощо) іноді об'єднують під загальною назвою «програмно-апаратні засоби в системах штучного інтелекту».

Розрізняють три різновиди інтелектуальних ІС:

– інтелектуальні інформаційно-пошукові системи (типу «запитання – відповідь»); такі системи забезпечують взаємодію кінцевих користувачів (як правило, – непрограмістів) із базами даних та знань у відповідній галузі; спілкування із системою здійснюється у режимі діалогу з використанням спеціальних професійних мов користувачів, що близькі до природніх;

– розрахунково-логічні інтелектуальні системи, які дають змогу кінцевим користувачам та спеціалістами з прикладної математики розв'язувати в діалоговому режимі певні завдання, що пов'язані з використанням складних методів обробки даних із залученням відповідних прикладних програм;

– експертні системи (ЕС), які дають змогу провадити ефективну комп'ютеризацію сфер, де знання можуть бути представлені в описовій, експертній формі, а використання математичних моделей ускладнене або взагалі неможливе.

Значною перевагою ЕС є їх здатність виявляти, накопичувати та узагальнювати знання з різних галузей народного господарства (предметних областей).

1.3.3 Класифікація за ступенем централізації та інтеграції обробки даних

За ступенем централізації обробки даних розрізняють: централізовані ІС, децентралізовані ІС, інформаційні системи колективного використання [1–4].

До централізованих ІС належать такі, де накопичення й обробка інформації здійснюється в єдиному центрі. У цьому разі доступ до самої ІС організовано з одного або багатьох терміналів.

Децентралізовані ІС є більш автономними. Кожна децентралізована ІС певного рівня обслуговує певну верству користувачів. Прикладом децентралізованої ІС може слугувати система обробки статистичних даних. Для неї характерно те, що ІС районного рівня обслуговує певний район, ІС обласного рівня обслуговує певну область тощо.

Інформаційні системи колективного користування відрізняються тим, що доступ до них має велика кількість користувачів.

Залежно від ступеня інтеграції функцій розрізняють: багаторівневі ІС з інтеграцією за рівнями управління (підприємство – об’єднання, об’єднання – галузь й т. і), багаторівневі ІС з інтеграцією за рівнями планування.

Однорівневі інформаційні системи обслуговують переважно окремі підрозділи управління або виробництва. Однорівневою може бути система планово-фінансового відділу, автоматизована системи диспетчера виробництва, працівника складу і т. п.

Багаторівневим інформаційним системам притаманні інтеграція функцій за різними аспектами управління. Так, до багаторівневих здебільшого належать системи автоматизації бухгалтерського обліку, комплексного управління виробництвом, системи ресурсного забезпечення, загальнодержавні системи тощо.

1.3.4 Класифікація за типом ІС

Згідно за цим показником розрізняють фактографічні, документальні й документально-фактографічні ІС.

У документальній ІС об’єктом зберігання виступають власне документи.

Фактографічна ІС зберігає дані про об’єкти або сутності, що становлять певний інтерес для конкретної проблемної сфери (інформація про співробітників, договори, вироби тощо). Відомості про ці сутності можуть знаходитись у великій кількості таблиць, звітів, різних вхідних та вихідних повідомленнях.

1.3.5 Класифікація за сферою застосування

За цим показником ІС поділяють на такі групи:

– СОТ, – системи обробки транзакцій (протоколів);
– СППР, – системи підтримки прийняття рішень;
– ІДС, – інформаційно-довідкові системи;
– ОІС, – офісні інформаційні системи.

Залежно від оперативності обробки даних СОТ ділять на пакетні ІС та оперативні ІС.

Слід вказати, що в інформаційних системах організаційного управління переважає режим оперативної обробки транзакцій (Online Transaction Processing або OLTP). Він дозоляє забезпечити високу оперативність відображення актуального стану предметної області на будь-який момент часу. Щодо пакетної обробки, то на сьогодні вона мало розповсюджена серед сучасних інформаційних системах організаційного управління.

Системи типу OLTP характеризуються регулярним, іноді досить інтенсивним потоком транзакцій. Це можуть бути замовлення, платежі, запити тощо. Тому принциповими вимогами до таких систем є:

– висока продуктивність обробки транзакцій;
– висока надійність передачі інформації за умов віддаленого доступу до БД через систему телекомунікацій.

Системи підтримки прийняття рішень (іноді їх позначають як DSS) відрізняться тим, що отримують на вході та обробляють досить складні запити. За результатами цих запитів СППР відбирають та аналізують дані у різних розрізах: за часом, географією, спеціальними характеристиками процесів тощо. Складність та розмаїття запитів потребує використання у складі СППР розвинутих СУБД (як правило, – реляційного типу)

На відміну від СППР, інформаційно-довідкові системи здебільшого орієнтовані на здатність працювати з інформацією довільної структури. Саме

тому більшість таких систем засновано на використанні гіпертекстових документів та мультимедіа. Найбільшого розвитку такі інформаційні системи отримали в мережі Інтернет.

Клас ОІС націлений переважно на збереження, відтворення та обробку документів, а також на переклад паперових документів до електронного вигляду. Серед функціональних можливостей цих систем головують завдання щодо автоматизації діловодства та управління документообігом.

1.3.6 Класифікація за способом організації

Виходячи зі способу організації інформаційні системи підрозділяються на такі класи:

- на основі архітектури «файл-сервер»;
- на основі архітектури «клієнт-сервер»;
- на основі багаторівневої архітектури;
- на основі Інтернет або Інтранет-технологій.

Зрозуміти обмеження різних архітектур нам допоможе аналіз функціональних компоненти будь-якої інформаційної системи та особливості побудови інформаційних її додатків (табл. 1.2).

Таблиця 1.2 – Типові функціональні компоненти інформаційної системи

Позначення	Найменування	Характеристика
1	2	3
PS	Presentation Services (засоби представлення)	Обслуговує введення даних й запитів від користувача й відображає те, що повідомляє йому компонент логіки подання PL з використанням відповідної програмної підтримки
PL	Presentation Logic (логіка представлення)	Управляє взаємодією між користувачем і системою. Обробляє дії користувача під час вибору команди в меню, при виборі пункту зі списку тощо
BL	Business Logic (прикладна логіка)	Реалізує набір правил для прийняття рішень, обчислень і операцій, які повинен виконати програмний додаток
DL	Data Logic (логіка управління даними)	Оперує з базою даних через мову SQL, які потрібно виконати для реалізації прикладної логіки управління даними

Продовження таблиці 1.2

1	2	3
DS	Data Services (операції з базою даних)	Підтримує дії СУБД, що реалізують логіку управління даними. Наприклад, – маніпулювання даними, визначення даних, фіксацію або відкат транзакцій тощо.
FS	File Services (файлові операції)	Реалізує дискові операції читання і запису даних для СУБД та ін. компонентів. Зазвичай є функціями операційної системи (ОС)

Архітектура «файл-сервер»

В архітектурі «файл-сервер» [2–4] розподіл компонентів діалогу PS і PL на мережевому рівні відсутній, що певною мірою полегшує побудову графічного інтерфейсу. Комп'ютер використовується переважно для функцій відображення. Файл-сервер тільки отримує дані, внаслідок чого додаткові користувачі та їх додатки не суттєво збільшують навантаження на головний сервер.

Об'єктами розробки в файл-серверному додатку є компоненти, що визначають логіку діалогу PL, логіку обробки BL і управління даними DL. Додаток реалізується або у вигляді закінченого модуля, або у вигляді спеціального коду для подальшої інтерпретації.

Головний недолік файл-серверної архітектури: під час виконання запитів до бази клієнта можуть передаватися великі обсяги даних. Це суттєво завантажує мережу, збільшує час реагування та знижує ефективність роботи системи в цілому. Цей недолік значно обмежує використання архітектур типу «файл-сервер». Особливо за умов організації віддаленого доступу до баз даних через повільні канали зв'язку.

Одним із варіантів усунення такого недоліку є віддалене управління файл-серверним додатком у мережі. Для цього у локальній мережі ставлять сервер додатків поєднаний з телекомунікаційним сервером (сервером доступу). Сервер додатків реалізує основний функціонал. А діалог між віддаленими клієнтами бере на себе сервер комунікацій. Однак у такому разі додатки не мають бути надто складними, оскільки це призведе до перевантаження сервера і для вирішення цієї проблеми знадобиться дуже потужна платформа для сервера додатків.

Архітектура «клієнт-сервер»

В архітектурі «клієнт-сервер» [2–4] діє розподіл компонентів програми та баз даних та розташування їх там, де вони будуть працювати найбільш ефективно.

Архітектура «клієнт-сервер» [2–4] дозволяє вирішити проблеми, що притаманні файл-серверним додаткам. Це досягається внаслідок розподілу компонентів програми та розташування їх там, де вони можуть працювати найбільш ефективно. Відмінність архітектури «клієнт-сервер» полягає у тому, що тут є виділені сервера баз даних, які обробляють запити на мові SQL (Structured Query Language). Сам текст запиту формує додаток клієнта та пересилає його на подальшу обробку до сервера. Таким чином сервер баз даних організує пошук, сортування та агрегування інформації. А клієнтська частина робить подальшу обробку результату та його візуалізацію у вигляді форм або звітів. Внаслідок такої організації досягається мінімальний трафік по мережі (передаються здебільшого текстові запити), інформаційна база максимально відокремлена від клієнтських додатків, тому стає можливим незалежне адміністрування баз даних без коригування програм на стороні клієнта.

Більшість конфігурацій типу «клієнт-сервер» використовують дворівневу модель, де клієнт звертається до послуг сервера. Передбачається, що діалогові компоненти PS і PL розміщуються на клієнті. Це спрощує реалізацію графічного інтерфейсу. Компоненти управління даними DS і FS розміщують на сервері, діалог (PS, PL) і логіка (BL, DL) – на клієнті. У дворівневій архітектурі «клієнт-сервер» використовується саме такий варіант, коли додаток працює на клієнті, а СУБД – на сервері.

Схема «клієнт-сервер» пред’являє найменші вимоги до сервера, тому вона має найкращу масштабованість. Однак складні додатки, що активно взаємодіють з БД, можуть завантажити як клієнт, так і мережу. Результати SQL-запиту повинні повернутися клієнту для обробки, оскільки там реалізована логіка прийняття рішень. Це додатково ускладнює адміністрування додатків, що розташовані на різних клієнтських вузлах.

Для зменшення навантаження на мережу і спрощення адміністрування додатками компонент BL також можна розмістити на сервері. У такому разі вся логіка прийняття рішень оформляється у вигляді збережених процедур і виконується на сервері БД.

Збережена процедура – послідовність SQL-операторів для організації доступу до БД. Збережена процедура має унікальне ім'я, може отримувати параметри і виконується на сервері БД. Для підвищення швидкості їх виконання збережені процедури можуть бути відкомпільовані.

Використання збережених процедур покращує рівень захищеності додатків, цілісність БД та безпеку системи в цілому (нема безпосереднього доступу до даних), гарантує актуальність операцій та обчислень, що здійснюються колективно. Крім того, збережені процедури простіше супроводжувати.

Реалізація архітектури «клієнт-сервер» можлива також на основі багатотермінальної системи. У цьому випадку на стороні багатозадачного сервера додатків виконуються програми користувачів. Клієнтські вузли працюють як термінали. Подібна схема притаманна, наприклад, системі Unix.

Дворівневі схеми архітектури «клієнт-сервер» можуть додати певних проблем при реалізації складних додатків з великою кількістю користувачів. Вирішити ці проблеми можна через застосування багаторівневої архітектури.

Багаторівнева архітектура

Ця конфігурація є розвитком архітектури «клієнт-сервер». За класичною формою вона має три рівні:

– нижній, – додатки клієнтів, що виділені для виконання функцій і логіки PS і PL, мають програмний інтерфейс для виклику програм середнього рівня;

– середній, – сервер додатків, де виконується прикладна логіка BL; саме з цього сервера логіка обробки DL здійснює операції з базою даних DS;

– верхній, – віддалений спеціалізований сервер для обробки даних DS і файлових операцій FS (без використання збережених процедур).

Подібну концепцію обробки даних підтримують, зокрема, фірми Oracle, Sun, Borland тощо.

Трирівнева архітектура дає змогу краще збалансувати навантаження на різні вузли та на мережу в цілому, поліпшує спеціалізацію інструментів для розробки додатків, усуває недоліки дворівневої моделі «клієнт-сервер».

Централізація логіки додатків спрощує адміністрування і супровід системи. Чітко поділяються платформи та інструменти для реалізації інтерфейсу і прикладної логіки, що дозволяє фахівцям реалізовувати їх із максимальною віддачею. Нарешті, зміни прикладної логіки та інтерфейсу не залежать одне від одного. Кордони між компонентами PL, BL і DL розмиті, тому прикладна логіка може реалізовуватися на всіх трьох рівнях. Сервер додатків за допомогою монітора транзакцій забезпечує інтерфейс із клієнтами та іншими серверами, може управляти транзакціями та підтримувати цілісність бази даних. Засоби віддаленого виклику процедур найбільше відповідають ідеї розподілених обчислень, оскільки вони забезпечують виклик прикладної процедури з будь-якого вузла мережі, розташованої на іншому вузлі.

Досвід останніх років розвитку систем «клієнт-сервер» показав доцільність використання трьох рівнів архітектури. Продукти для такої трирівневої архітектури (так звані монітори транзакцій) є відносно новими. Ці інструменти здебільшого орієнтовані на середовище Unix, однак прикладні сервери можна будувати також і на базі Microsoft Windows.

На практиці в локальній мережі можуть використовуватися змішані архітектури (дво- та трирівневі), а також архітектури з більшою кількістю рівнів.

Таким чином, багаторівнева архітектура дає змогу підвищити ефективність роботи корпоративної інформаційної системи й оптимізувати розподіл її програмно-апаратних ресурсів. Однак поки що на практиці домінує архітектура «клієнт-сервер».

Інтернет та Інтранет-технології

До недавнього моменту головний акцент щодо розвитку Інтернет та Інтранет-технологій було спрямовано на розроблення інструментальних програмних засобів. У той час як нагальні потреби у розвинутих засобах розроблення додатків, що працюють з базами даних, були забезпечені не належним чином. Компромісним рішенням для вирішення цього питання стало об’єднання Інтернет та Інтранет-технологій з багаторівневою архітектурою. У цьому випаду структура інформаційного додатку набуває вигляду: браузер – сервер додатків – сервер баз даних – сервер динамічних сторінок – веб-сервер.

За допомогою інтеграції Інтернет та Інтранет-технологій із архітектурою «клієнт-сервер», процес впровадження та супроводження ІС стає набагато простішим і при цьому зберігається висока ефективність та можливість спільного використання інформації.

1.3.7 Вимоги до інформаційних систем

Незалежно від класу, структури та галузі використання ІС має відповідати таким вимогам: гнучкість, надійність, ефективність, безпека.

Гнучкість означає здатність системи до адаптації та подальшого розвитку. Це передбачає можливість пристосування інформаційної системи до нових умов та особливих потреб конкретного підприємства. Забезпечити гнучкість можна тільки за умов, якщо на етапі розроблення ІС були задіяні загальновідомі, науково обґрунтовані засоби й методи документування, внаслідок чого зберігається можливість розібратися в структурі системи й внести до неї відповідних змін.

Надійність ІС означає її стабільну роботу за будь-яких умов без спотворення інформації та без втрати даних. Високої надійності можна досягти різними способами: через створення резервних копій, за рахунок операцій протоколювання, підтримання високої якості каналів зв'язку та фізичних носіїв інформації, використання сучасних програмних та апаратних засобів, забезпечення достатнього рівня кваліфікації персоналу.

Система є **ефективною**, якщо вона здатна вирішувати поставлені завдання у мінімальний термін, використовуючи при тому виділені їй ресурси. За будь-яких обставин оцінку ефективності ІС робить замовник. При здійсненні такої оцінки він має співставити обсяг засобів, що вкладені у розробку, та відповідність ІС його очікуванням. Тільки активна участь замовника на всіх стадіях проектування ІС дає змогу досягти високої ефективності та уникнути дискомфорту від впровадження нової технології до реального виробництва.

Активна співпраця із замовником із ранніх етапах проектування дозволяє краще з'ясувати його потреби і, внаслідок цього, зменшити витрати та термін розробки системи. Крім того, контакт із замовником під час розроблення ІС може підштовхнути його до модернізації апаратних засобів, застосування нових методів ведення бізнесу, внесення змін до організаційної структури підприємства тощо. Це, зі свого боку, відповідає потребам, як

замовника, так і проектувальника, оскільки замовник підвищує ефективність свого підприємства, а проектувальник – отримує нові можливості щодо проектування ІС. Ефективність системи також забезпечується оптимізацією даних і методів їхньої обробки, застосуванням оригінальних алгоритмів, ідей, методів проектування тощо.

Слід мати на увазі, що працювати з системою можуть малодосвідчені фахівці у плані інформаційних технологій. Тому оптимізація інтерфейсу грає не останню роль у забезпеченні високої ефективності системи на стадії реальної експлуатації. Інтерфейс має бути інтуїтивно зрозумілим. Структура команд меню та графічних елементів має бути ретельно проаналізована та оптимізована.

Під **безпекою,** передусім, мають на увазі властивість ІС контролювати доступ до інформації з боку різних категорій користувачів, забезпечувати цілісність даних та їхній захист від випадкового пошкодження. Захист інформації від несанкціонованого доступу забезпечується розмежуванням на рівні ресурсів системи, використанням спеціальних програмних засобів та компонентів операційної системи щодо захисту інформації. У великих організаціях доцільно створювати окремі підрозділи для вирішення завдань інформаційної безпеки. У невеликих організаціях можна призначати окремого співробітника, відповідального за таку ділянку роботи.

Також необхідно зважати на можливе порушення безпеки внаслідок недосконалості програмного коду, вірусних атак, похибок в алгоритмах обробки даних тощо. Для вирішення подібних проблем організують постійний моніторинг стану безпеки системи та її оточення (операційного, прикладного, апаратного, спеціальних засобів захисту тощо).

1.4 Питання для самоконтролю

1. Дайте визначення, що таке інформаційна система.
2. Що таке структуровані дані? Наведіть приклади.
3. Чим відрізняються дані від інформації?
4. Класифікація інформаційних систем за сферою застосування.
5. Що таке інформаційно-керівні системи та яка їхня відмінність?
6. Дайте визначення, що таке СППР.
7. Перелічіть три різновиди інтелектуальних інформаційних систем.

8. Види інформаційних систем за ступенем централізації.

9. Що таке документальна система?

10. Різновиди інформаційних систем за способом організації.

11. Функціональні компоненти інформаційних систем.

12. Різновиди архітектур інформаційних систем.

13. Архітектура «файл-сервер», її переваги та недоліки.

14. Архітектура «клієнт-сервер».

15. Перелічіть головні вимоги до інформаційних систем.

2 ЖИТТЄВИЙ ЦИКЛ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Головні питання: *поняття життєвого циклу ІС. Процеси, які відбуваються протягом життєвого циклу. Головні, допоміжні та організаційні процеси життєвого циклу ІС. Взаємозв'язок між процесами життєвого циклу ІС. Структура життєвого циклу інформаційної системи. Початкова стадія. Стадія уточнення. Стадія конструювання. Стадія переходу. Моделі життєвого циклу ІС. Каскадна модель життєвого циклу ІС, її переваги й недоліки. Спіральна модель життєвого циклу ІС, ітерації, переваги спіральної моделі. Проблеми, що виникають під час використання спіральної моделі.*

Будь-яка організація – це сукупність взаємодіючих елементів (підрозділів), що мають свою, досить складну, структуру та розгалужену систему взаємозв'язків. Ці зв'язки можна розділити на три категорії [1–3]:

– функціональні, коли кожен підрозділ виконує певні різновиди робіт у межах єдиного бізнес-процесу;

– інформаційні, коли підрозділи обмінюються інформацією (документами, звітами, базами даних, розпорядженнями тощо);

– зовнішні, коли окремі підрозділи взаємодіють із зовнішнім середовищем, причому як на інформаційному, так і функціональному рівнях.

У цьому контексті процес розробки й впровадження інформаційної системи може бути розглянуто в двох аспектах:

– за змістом дій розробників, коли процес розробки розглядають як певний статичний об'єкт у термінах потоків робіт (виконавці, дії, послідовність дій тощо);

– за часом або за **стадіями життєвого циклу (ЖЦ)**, коли за основу беруть динамічну організацію всього процесу розробки в термінах циклів, стадій, ітерацій та етапів.

Саме другий підхід на сьогодні є домінуючим при розробці складних ІС. Тому в подальшому ми будемо орієнтуватись на процес розробки ІС за другим варіантом.

2.1 Основні фази проектування інформаційної системи

Незалежно від складності кожний проект проходить у своєму розвитку певні етапи. Стосовно ІС таких етапів п'ять [1–3]:

– формування концепції;

– підготовка технічного завдання;

– проектування ІС;

– розробка ІС;

– введення ІС до експлуатації.

Розглянемо докладно склад та відмінності кожної із вказаних етапів.

2.1.1 Концептуальна фаза (формування концепції)

Головним завданням робіт на цьому етапі є визначення мети проекту та його концепції, що передбачає:

– формування цілей та ідеї проекту;

– формування команди для реалізації проекту;

– вивчення вимог замовника мотивацій учасників проекту;

– збір даних та аналіз існуючого стану у контексті тематики проекту;

– визначення ключових вимог і обмежень, обсягів матеріальних, фінансових і трудових ресурсів для реалізації проекту;

– здійснення порівняльної оцінки альтернатив;

– подання пропозицій стосовно проекту, проведення їхньої експертизи та затвердження.

2.1.2 Підготовка технічного завдання

Головне завдання цього етапу – уточнення технічної пропозиції за результатами переговорів із замовником щодо укладання контракту. Зміст етапу, як правило, виглядає так:

– розроблення основного змісту та базової структури проекту;

– розроблення та затвердження технічного завдання;

– планування й подальша декомпозиція[1] базової структури проекту;

– складання кошторису та бюджету проекту, визначення потреб у ресурсах;

– розроблення календарних планів та узагальнених графіків виконання робіт;

– підписання контракту на виконання проекту;

– впровадження засобів комунікації учасників проекту та засобів контролю за процесом виконання робіт.

[1] Декомпозиція – один із головних способів підвищення ефективності робіт. Підсистеми, на які розбивається проект, мають бути слабко пов’язаними за даними й за функціями. Кожна підсистема розробляється окремо.

2.1.3 Проектування

Мета проектування – з'ясувати склад підсистем, що попадають під горизонт проекту, виявити взаємозв'язки між ними, вибрати найбільш ефективні способи виконання проекту й використання ресурсів. Фаза проектування передбачає такі роботи:

- виконання базових проектних робіт;
- розробка індивідуальних технічних завдань;
- концептуальне проектування;
- складання технічних специфікацій та інструкцій;
- представлення проекту, його експертиза та затвердження.

2.1.4 Розроблення

Головні завдання цієї фази – розробити механізми координації та оперативного контролю за роботами, що передбачені проектом, реалізувати весь склад підсистем, забезпечити їх об'єднання, взаємодію та тестування. Основний зміст фази розробки полягає у такому:

- розроблення програмного забезпечення;
- підготовка до впровадження системи;
- перевірка та регулювання головних показників проекту.

2.1.5 Введення системи до експлуатації

На фазі введення системи до експлуатації проводяться: випробування окремих компонент та ІС в цілому, дослідна експлуатація системи в реальних умовах, переговори щодо результатів виконання проекту, можливих змін та нових контрактів.

Ключові роботи цього етапу:

- комплексні випробування;
- підготовка кадрів для експлуатації системи;
- підготовка робочої документації (РД), здавання системи замовнику і введення її в експлуатацію;
- супроводження, підтримка, сервісне обслуговування;
- оцінка результатів проекту та підготовка підсумкових звітів;
- вирішення конфліктних ситуацій і закриття робіт за проектом;

– накопичення дослідних даних для подальших проектів, аналіз досвіду, стану, визначення напрямів розвитку.

Слід зауважити, що пошук помилок на стадії проектування забирає приблизно вдвічі більше часу, ніж на інших фазах, а їх виправлення коштує у п'ять разів дорожче. Саме тому розробку на початкових стадіях проекту варто виконувати дуже ретельно. Нижче вказано типові помилки, до яких вдаються розробники на початкових стадіях проекту:

– помилки у визначенні інтересів замовника;

– концентрація на другорядних інтересах;

– хибна інтерпретація початкового завдання;

– невірне або недостатнє розуміння деталей;

– неповнота функціональних специфікацій;

– вади у визначенні необхідних ресурсів і термінів виконання робіт;

– недостатня перевірка на узгодженість етапів, відсутність контролю з боку замовника.

2.2 Процеси, що мають перебіг у життєвому циклі інформаційної системи

Життєвий цикл – це безперервний процес, що починається з моменту рішення про створення ІС і закінчується після вилучення її з експлуатації [1–3]. Регламентує життєвий цикл інформаційних систем міжнародний стандарт ISO/IEC 12207. [2]

Стандарт ISO/IEC 12207 визначає структуру життєвого циклу, його процеси, дії, завдання, які мають бути виконані під час створення ІС. Згідно цього стандарту, структура життєвого циклу обіймає три групи процесів:

– **головні,** – придбання, постачання, розроблення, експлуатація, супроводження;

– **допоміжні,** що забезпечують виконання головних процесів (документування, забезпечення якості, тестування, атестація, оцінка ефективності, аудит, ліцензування, сертифікація тощо);

– **організаційні**, – управління проектами, створення інфраструктури проекту, навчання.

[2] Скорочення ISO походить від International Organization of Standardization (міжнародна організація за стандартизації), IEC – від International Electrotechnical Commission (міжнародна комісія з електротехніки).

2.2.1 Головні процеси життєвого циклу

Життєвий цикл ІС включає три основні процеси: розробка, експлуатація та супровід (супроводження).

Розробка

Цей процес поєднує всі роботи щодо створення інформаційного та програмного забезпечення та його компонентів відповідно до вимог техзавдання. Розробка інформаційного програмного забезпечення обіймає також такі питання, як:

- оформлення проектної та експлуатаційної документації (ЕД);
- підготовку матеріалів для тестування програмних продуктів;
- розробку матеріалів для навчання персоналу.

Розробка – один із найважливіших процесів життєвого циклу ІС. Як правило, він передбачає стратегічне планування, аналіз, проектування та реалізацію програмних продуктів.

Експлуатація

Умовно роботи на стадії експлуатації можна поділити на **головні** та **підготовчі.** До головних робіт на стадії експлуатації належать:

- експлуатація;
- локалізація проблем та усунення причин їхнього виникнення;
- модифікація (або адаптація) програмного забезпечення;
- підготовка пропозицій щодо удосконалення системи;
- розвиток та подальша модернізація системи.

Підготовчі роботи зазвичай включають:

- конфігурацію бази даних і робочих місць користувачів;
- забезпечення користувачів експлуатаційною документації;
- навчання (або підвищення кваліфікації) персоналу.

Супроводження

Технічна підтримка – дуже важлива складова успішної експлуатації ІС, необхідна умова вирішення поставлених перед нею завдань. Слід зауважити,

що помилки обслуговуючого персоналу можуть призводити до фінансових втрат, які можна порівняти з вартістю самої системи.

Технічна підтримка – досить складний та довготривалий процес. Тому для його успішної реалізації дуже важлива організаційна складова, під час якої слід виконати такі дії:

– виділити ключові складові системи, оптимізувати розподіл ресурсів для їхнього технічного обслуговування (ТО);

– визначити завдання щодо ТО, розділити їх на внутрішні (що вирішуються власними силами) та зовнішні (вирішуються спеціалізованими сервісними організаціями); забезпечити чіткий розподіл функцій та відповідальності;

– провести аудит ресурсів, необхідних для організації ТО, виконати розподіл компетенцій;

– підготувати план організації ТО, в якому конкретизувати етапи дій, терміни виконання, витрати за кожним етапом, відповідальність виконавців.

Якісне технічне обслуговування потребує залучення висококваліфікованих фахівців, які здатні не тільки вирішувати поточні завдання, а й швидко відновлювати дієвість системи в аварійних ситуаціях.

2.2.2 Допоміжні процеси життєвого циклу

Серед допоміжних провідне місце займає процес управління конфігурацією, який підтримує головні етапи життєвого циклу ІС, зокрема – процеси розробки та супроводження.

Під час розробки складних ІС кожен компонент може розроблятися незалежно, мати декілька варіантів або версій реалізації. У такому випадку виникає проблема щодо обліку зв’язків і функцій між ними, створення єдиної структури для забезпечення розвитку всієї системи. Цю проблему вирішує управління конфігурацією, що дозволяє системно контролювати внесення змін до різних компонент ІС на всіх стадіях ЖЦ.

2.2.3 Організаційні процеси

Управління проектом щільно пов’язано з плануванням і організацією робіт, створенням колективу проектувальників, з контролем за термінами та

якістю реалізації всіх етапів розробки ІС. Вирішення цих питань покладено на технічне й організаційне забезпечення проекту, яке передбачає:

- вибір методів та інструментальних засобів для реалізації проекту;
- визначення методів щодо опису всіх проміжних станів розробки;
- розробка методів і засобів для випробувань ПЗ;
- навчання та підвищення кваліфікації персоналу.

Забезпечення якості проекту щільно пов'язане з проблемами **верифікації**[3]**,** перевірки й тестування компонентів ІС.

Перевірка – це процес, мета якого – визначення відповідності параметрів ІС вихідним вимогам. Перевірка певною мірою схожа з тестуванням, яке має за мету визначення розбіжностей між дійсними та очікуваними результатами проекту та оцінкою відповідності характеристик ІС вихідним вимогам.

2.3 Структура життєвого циклу інформаційної системи

Повний життєвий цикл ІС передбачає стратегічне планування, аналіз, проектування, реалізацію, впровадження та експлуатацію. Крім того, життєвий цикл можна розбити на декілька стадій. Один із варіантів такого розподілу запропонувала корпорація Rational Software – провідна фірма на ринку програмного забезпечення засобів щодо розроблення інформаційних систем (зокрема універсального CASE-засобу Rational Rose)[4].

Згідно з методологією Rational Software, ЖЦ ІС має чотири стадії:

- початок;
- уточнення;
- конструювання;
- передавання до експлуатації.

Межі кожної стадії обмежені певними моментами часу, коли необхідно приймати критичні рішення і досягати певних цілей.

[3] Верифікація – процес визначення відповідності поточного стану розробки вимогам до цього етапу.

[4] Первісне значення терміна CASE (Computer Aided Software/System Engineering) обмежувалось тільки питаннями автоматизації розробки програмного забезпечення. Однак у подальшому значення цього терміна стало ширшим. Зараз під терміном «CASE-засобу» розуміють програмні засоби, що підтримують процеси створення і супроводження інформаційних систем, зокрема аналіз і формулювання вимог, проектування прикладного програмного забезпечення та баз даних, генерацію коду, тестування, документування, забезпечення якості, конфігураційне управління та управління проектом, а також інші процеси.

2.3.1 Початкова стадія

На цій стадії визначають область застосування системи та граничні умови для майбутньої розробки. Для вирішення такого завдання необхідно ідентифікувати всі зовнішні об’єкти, з якими буде взаємодіяти система, визначити особливості цієї взаємодії. Крім того, на початковій стадії ідентифікують усі функціональні можливості системи й роблять опис найістотніших із них. Також початкова стадія передбачає:

– критерії успіху розробки;

– оцінки ризику;

– оцінку ресурсів, необхідних для виконання розробки;

– календарний план із зазначенням термінів завершення головних етапів.

2.3.2 Стадія уточнення

На стадії уточнення проводять аналіз прикладної області та розробляють архітектуру майбутньої ІС. На завершальному етапі цієї стадії проводять аналіз архітектурних рішень і способів усунення головних чинників ризику в проекті.

2.3.3 Стадія конструювання

На стадії конструювання розробляють закінчений продукт, готовий до передачі в експлуатацію. Після закінчення цієї стадії оцінюють працездатність програмного забезпечення.

2.3.4 Стадія передавання до експлуатації

На цій стадії готове програмне забезпечення передають користувачу (або користувачам). Слід зауважити, що в процесі експлуатації системи можуть з’явитись різного роду проблеми, що потребують додаткових робіт щодо внесення коригувань до готового продукту. Тому одне із завдань стадії експлуатації – виявлення необхідних коригувань, планування і проведення робіт щодо їх реалізації, а також внесення відповідних змін. Наприкінці стадії передавання до експлуатації визначають ступінь досягнення цілей розробки.

2.4 Моделі життєвого циклу інформаційної системи

Модель життєвого циклу – це певна структура, що визначає послідовність та взаємозв'язки між процесами, що передбачені у межах ЖЦ інформаційної системи.

Ключові положення і базові визначення моделі ЖЦ викладені в стандарті ISO/IEC 12207. Слід відмітити, що вказаний стандарт описує тільки структури процесів. У ньому немає деталізації методів і дій для вирішення завдань, що входять до процесів ЖЦ ІС. І це зрозуміло: регламенти ISO/IEC 12207 є загальними для будь-яких моделей і технологій розробки. Водночас сама модель ЖЦ залежить від специфіки ІС та умов, де вона створюється і буде працювати.

На сьогодні найбільшого поширення набули дві головні моделі ЖЦ:

– каскадна (так звана «модель водоспаду» або «waterfall»);
– спіральна.

2.4.1 Каскадна модель життєвого циклу інформаційної системи

Згідно каскадної моделі всі роботи виконуються. При цьому вся розробка розбивається на етапи, перехід до наступного етапу відбувається після повного завершення всіх робіт попереднього етапу.

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

Головні етапи розроблення згідно з каскадною моделлю

Враховуючи досвід реальних розробок, у каскадній моделі прийнято виділяти декілька сталих етапів, що майже не залежать від предметної області (рис. 2.1), а саме:

– аналіз вимог замовника;
– проектування;
– розробка;
– тестування і дослідна експлуатація;
– здавання готового продукту.

Результатом першого етапу є ТЗ (завдання на розробку). Воно має бути узгоджене з усіма сторонами.

Результат другого етапу – комплект проектної документації (ПД), що містить необхідні дані для реалізації проекту.

Результат третього етапу – готовий програмний продукт.

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

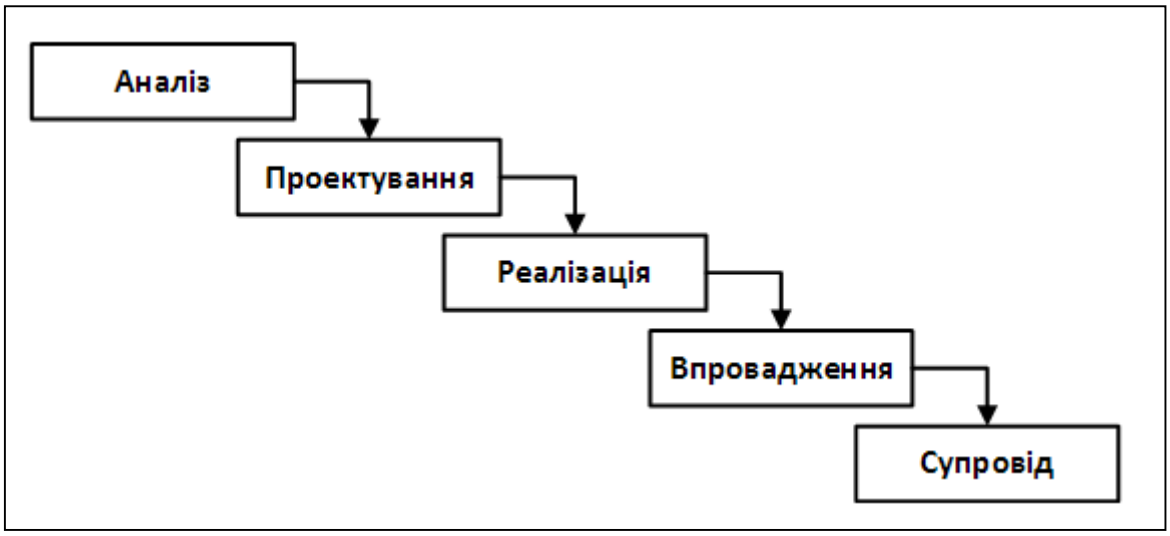

Рисунок 2.1 – Структурна схема каскадної моделі розробки ІС

Головне завдання п’ятого етапу – переконати замовника у тому, що всі вимоги виконані повною мірою.

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

Переваги каскадної моделі

Серед переваг каскадної моделі передусім можна вказати на такі:

– на кожному етапі формується повний та узгоджений набір ПД. На завершальних етапах також розробляється документація, що охоплює всі передбачені стандартами різновиди забезпечення ІС (організаційне, методичне, інформаційне, програмне, апаратне).

– чітка послідовність етапів дозволяє планувати терміни виконання робіт та відповідні витрати.

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

Недоліки каскадної моделі

Кількість недоліків каскадної моделі досить значна, а саме:

– повільність щодо отримання результатів;

– помилки на будь-якому етапі впливають на подальшу роботу;

– у межах каскадної моделі складно організувати паралельне ведення робіт;

– має місце певна інформаційна перенасиченість на всіх етапіх проекту;

– ускладнено управління проектом;

– значний рівень ризику та ненадійність інвестицій.

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

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

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

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

Повернення до попередніх стадій. Цей недолік каскадної моделі є наслідком попереднього. Повернення може стати причиною порушення

графіку робіт та ускладнити взаємовідносини між розробниками ІС, що працюють над різними етапами тощо. Найгіршим є той факт, що недоліки попереднього етапу можуть проявити себе набагато пізніше. Унаслідок цього роботу треба припиняти і повертатись з поточного до попереднього етапу. Тому в реальному житті каскадна схема розробки виглядає, як показано на рисунку 2.2.

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

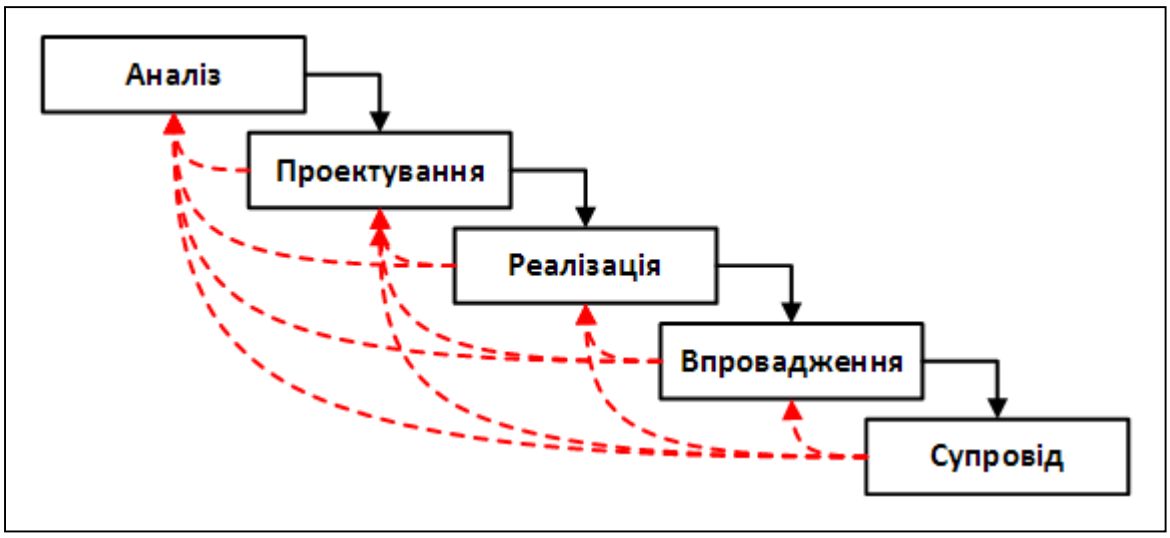

Рисунок 2.2 – Реальний процес розробки за каскадною схемою

Інформаційна перенасиченість. Цей недолік є наслідком суттєвої залежності між різними групами розробників. Проблема полягає в тому, що при внесенні змін до будь-якої частини проекту треба повідомити про них всіх розробників, що пов'язані з цією частиною. Як наслідок – швидко зростають витрати на документування змін, опрацювання цих змін, погіршується оперативність узгоджень й т. і. А в цілому, – збільшується обсяг інформації, що є наслідком особливостей схеми управління проектом, що має в певному розумінні «паразитну» природу.

Питання перенасиченості різко загостюється за умов ротації груп розробників. Адже нові спеціалісти крім вивчення нового матеріалу повинні опанувати велику кількість старої інформації, що пов'язана з історією змін, коригувань та узгоджень. Це факт вкрай негативно впливає на ефективність проектування.

Складність управління проектом під час використання каскадної схеми здебільшого обумовлена строгою послідовністю стадій розроблення й наявністю складних взаємозв'язків між різними частинами проекту.

Послідовність розроблення проекту призводить до того, що одні групи розробників повинні очікувати результати роботи інших команд. Отже, потрібно адміністративне втручання для узгодження термінів роботи та складу переданої документації.

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

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

Нарешті, занадто велика кількість змін (особливо в в предметній області або у вимогах замовника) можуть взагалі призупинити проект, не довівши його до остаточної реалізації.

Тому можна стверджувати, що складні проекти, що розробляються за каскадною схемою, мають підвищений рівень ризику.

Область застосування. Каскадний підхід добре зарекомендував себе при розробці:

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

2) 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.

На базі каскадної моделі життєвого циклу побудовано один із найвідоміших засобів автоматизації процесів розроблення складних інформаційних систем від компанії Oracle, що має назву Oracle Designer. Методика (в літературі вона має назву CRM), що використовується під час роботи цього комплексу, базується на таких положеннях:

1) проектування має структурне походження, весь процес розроблення системи подається у вигляді послідовності чітко визначених етапів;

2) підтримка здійснюється на всіх етапах життєвого циклу системи, починаючи від загальних пропозицій і закінчуючи супроводом готового продукту;

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

4) під час розроблення всі специфікації проекту зберігаються в спеціальній базі даних (репозиторії), який працює під управлінням СУБД Oracle. До репозиторію може підключатись велика кількість користувачів (розробників), унаслідок чого їхні дії є узгодженими;

5) послідовний перехід від одного етапу до іншого автоматизований унаслідок використання спеціальних утиліт. За допомогою них за специфікаціями на концептуальній стадії можна отримати початковий варіант специфікації рівня проектування. Надалі генерація доповнень значно спрощується;

6) усі етапи проектування та розробки автоматизовані; у будь-який момент може бути згенерований довільний обсяг звітів, які забезпечують документування поточної версії системи на всіх етапах її розробки.

Oracle пропонує свій варіант структури життєвого циклу інформаційної системи, а методологія та програмні засоби щодо її реалізації орієнтовані переважно на продукти цієї компанії. Незважаючи на певні недоліки та обмеження, технологія CDM лишається одною з провідних під час розробки складних інформаційних систем. Докладніше до особливостей CDM ми повернемось у розділі 3, коли будемо розглядати головні методики й технології проектування.

2.4.2 Спіральна модель життєвого циклу

На відміну від каскадної спіральна модель ЖЦ передбачає **ітераційний** процес розробки ІС. До того ж у рамках цієї моделі зростає роль початкових етапів ЖЦ (аналізу та проектування), – саме на цих етапах перевіряється та обґрунтовується реалізація технічних рішень через створення т. зв. прототипів.

Поняття ітерації

Ітерація – це основний елемент концепції спіральної моделі. Кожна ітерація є завершеним циклом розробки, що призводить до випуску діючої версії виробу (або певної його частини). В подальшому від ітерації до ітерації цей продукт вдосконалюється й наприкінці перетворюється у завершену систему (рис. 2.3).

Рисунок 2.3 – Спіральна модель життєвого циклу інформаційної системи

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

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

Переваги спіральної моделі

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

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

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

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

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

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

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

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

Недоліки спіральної моделі

Головна проблема спірального підходу – визначення моментів переходу між етапами. Для її розв'язання необхідно вводити тимчасові обмеження на кожен із етапів ЖЦ. В іншому разі процес розробки може перетворитися в нескінченне удосконалення зробленого. Тому завершення ітерації має здійснюватися тільки відповідно до плану, навіть якщо не весь запланований обсяг робіт вже закінчено. Щодо самого плану, то він складається на підставі статистичних даних з попередніх проектів та особистого досвіду розробників.

На базі спіральної моделі була створена потужна методологія швидкого розроблення додатків, що називається RAD (від англ. Rapid Application Development). Вона орієнтована переважно на роботу невеликої команди програмістів, які мають досвід в аналізі, проектуванні, генерації коду й тестуванні ПЗ із використанням CASE-засобів.

Під час проектування за методологією RAD активно використовуються об'єктно-орієнтовані методи опису предметної області. Методологія добре підходить для проектів із коротким та добре опрацьованим графіком робіт. Серед відмінностей RAD можна вказати такі:

– розроблення програм ітераціями;

– допустимість часткового завершення робіт на кожному з етапів ЖЦ;

– обов'язкове залучення користувачів до процесу розробки ІС;

– обов'язкове використання CASE-засобів для забезпечення цілісності проекту;

– використання засобів управління конфігурацією, що полегшують внесення змін до проекту й супровід готової системи;

– використання генераторів коду;

– прототипування, що дозволяє краще з'ясувати й задовольнити потреби кінцевого користувача;

– тестування і коригування проекту одночасно з його розробкою;

– ведення розробки невеликою, але добре керованою командою професіоналів (зазвичай від 2 до 10 осіб);

– професійне керівництво розробкою системи, чітке планування і контроль за виконанням робіт.

Життєвий цикл ПЗ за методологією RAD має чотири фази:

1) аналіз і планування вимог;
2) проектування;
3) побудова;
4) впровадження.

Докладніше до особливостей CDM ми повернемось у розділі 3, коли будемо розглядати методики й технології проектування.

2.4.3 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.

Різні варіанти ітераційного підходу реалізовані в більшості сучасних методів: Rational Unified Process (RUP), Framework Microsoft Solutions (MSF), XR тощо.

1. Метод RUP (запропонований компанією Rational Software у 2003 р.) використовує ітеративну модель розроблення з чотирьох фаз (початок, дослідження, побудова, впровадження), розбитих на ітерації. Кожна ітерація завершується отриманням проміжної, але діючої версії кінцевого продукту. RUP спирається на інтегрований комплекс інструментальних засобів Rational Suite, до складу якого, крім самої технології RUP як продукту, входять такі компоненти, як:

– Rational Rose – засіб візуального моделювання (аналізу і проектування), використовує мову UML;

– Rational XDE – засіб аналізу і проектування, інтегрується з платформами MS Visual Studio .NET, IBM WebSphere Studio Application Developer та ін.

2. MSF (Microsoft Framework, 1993 р.) схожа з RUP, так само включає чотири фази: аналіз, проектування, розробка, стабілізація, є ітераційною і передбачає використання об'єктно-орієнтованого моделювання. MSF, на відміну від, RUP більшою мірою орієнтована на розробку бізнес-додатків.

3. Методологія XP (Extreme Programming або Екстремальне програмування, 1996 р.). Розробка виглядає як ітеративний процес, де фази розбиваються на малі кроки. В основі методології – командна робота й ефективна комунікація між замовником та виконавцем.

2.5 Питання для самоконтролю

1. Вкажіть головні різновиди зв'язків між підрозділами підприємства.

2. Перелічіть фази проектування інформаційної системи.

3. Назвіть головне завдання та перелік робіт концептуальної фази проектування інформаційної системи.

4. У чому полягає зміст етапу «підготовка технічної пропозиції»?

5. Мета й завдання фази «Проектування».

6. Мета, завдання за головні складові фази «Розробка».

7. Мета, завдання, головні складові та відмінності фази «Введення до експлуатації».

8. Типові помилки проектування інформаційних систем.

9. Дайте визначення: що таке «життєвий цикл» інформаційної системи.

10. Перелічіть головні процеси життєвого циклу інформаційної системи.

11. Перелічіть головні процеси стадії «Розробка».

12. Перелічіть головні процеси стадії «Експлуатація».

13. Перелічіть головні процеси стадії «Супровід».

14. Що таке допоміжні процеси життєвого циклу? Перелічіть їх та надайте стислу характеристику.

15. Що передбачає процес «Організація»?

16. Охарактеризуйте структуру життєвого циклу за концепцією Rational Software.

17. Склад робіт початкової стадії життєвого циклу.

18. Склад робіт стадії «Уточнення» життєвого циклу.

19. Склад робіт стадії «Конструювання» життєвого циклу.

20. Склад робіт стадії «Передача до експлуатації» життєвого циклу.

21. Різновиди моделей життєвого циклу інформаційної системи.

22. Каскадна модель життєвого циклу. Характеристика каскадної моделі. Її особливості, переваги та недоліки.

23. Спіральна модель життєвого циклу. Характеристика спіральної моделі. Її особливості, переваги та недоліки.

24. Ієрархічна модель життєвого циклу. Характеристика ієрархічної моделі. Її особливості, переваги, недоліки та сфера використання.

3 МЕТОДОЛОГІЯ І ТЕХНОЛОГІЯ РОЗРОБЛЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Головні питання: *методологія проектування ІС. Поняття методу і технології проектування ІС. Класифікація методів проектування ІС. Головні завдання, що має вирішувати методологія створення інформаційних систем. Складові технології проектування ІС. Основи структурного підходу до проектування ІС. Головні поняття об'єктно-орієнтованого аналізу та програмні засоби щодо його реалізації. Головні фази проектування ІС: аналіз предметної області, розроблення технічного завдання, технічне та робоче проектування ІС, введення системи в експлуатацію. Методика обстеження предметної області. Концептуальна схема предметної області. Сучасні методології проектування інформаційних систем. Методологія RAD. Об'єктно-орієнтований підхід у методології RAD. Подієве програмування і методологія RAD. Життєвий цикл інформаційної системи за методологією RAD. Методологія RUP. Методика CDM фірми Oracle. Структура життєвого циклу згідно з методологією CDM.*

3.1 Головні поняття

Проектування – це пошук такого способу створення системи, що задовольняє вимогам функціональності системи зважаючи на задані обмеження та наявні технології. Проектування інформаційних систем охоплює три головні області:

– проектування об'єктів даних, які будуть реалізовані в базі даних;

– проектування програм, екранних форм, звітів, які будуть забезпечувати виконання запитів до даних;

– проектування певного середовища або технології, тобто топології мережі, конфігурації апаратних засобів, використовуваної архітектури (файл-серверної або клієнт-серверної), паралельної обробки, розподіленої обробки даних тощо.

Мета проектування полягає у забезпеченні ефективного функціонування ІС, а також взаємодії користувачів і розробників ІС. Саме якісне проектування дозволяє створити систему, яка здатна працювати за постійним вдосконаленням своїх технічних, програмних, інформаційних складових і розширювати спектр реалізованих управлінських функцій. У процесі проектування удосконалюється як організація провідної діяльності підприємства, так і організація управлінських процедур.

Основу проекту будь ІС складають такі компоненти:

– методологія проектування;

– технологія проектування;

– стандарти й методики проектування;

– інструментальні засоби проектування (або CASE-засоби).

Між вказаними компонентами діє щільний зв'язок, а саме: реалізація методології здійснюється через конкретні технології, кожна технологія підтримується відповідними стандартами й методиками, а інструментальні засоби забезпечують реалізацію процесів проектування, що описані у методиках та стандартах.

Методологія проектування – це організація процесу розроблення системи й управління цим процесом, яка гарантує дотримання вимог як до самої системи, так і до характеристик щодо процесу її розроблення.

Найважливіші завдання, які має виконувати методологія створення ІС:

– відповідність ІС, що створюється, меті та завданням підприємства, а також вимогам щодо автоматизації бізнес-процесів;

– гарантоване створення системи із заданими параметрами у межах виділеного часу та бюджету;

– підтримка дисципліни супроводження, модифікації та нарощування системи для забезпечення її адаптації до мінливих умов роботи підприємства;

– забезпечення відповідності ІС, що створюється, вимогам відкритості та здатності до масштабування;

– забезпечення спадкоємності розробки, тобто використання у системі, що розробляється, існуючої інформаційної інфраструктури організації.

– забезпечення можливості використання в новій системі елементів інформаційних існуючих технологій, а саме: програмного забезпечення, баз даних, апаратних засобів, телекомунікацій тощо.

Методології, технології та інструментальні засоби проектування (або так звані CASE-засоби) – це основа проекту будь-якої інформаційної системи.

Впровадження методології повинно призводити до зниження складності процесу створення ІС шляхом повного й точного описання цього процесу, а також застосування сучасних методів і технологій створення ІС на всьому життєвому циклі, тобто від ідеї до реалізації.

Головний зміст технології проектування – це технологічні інструкції. Вони включають опис послідовності технологічних операцій, самих операцій, а також умов їхнього виконання.

Технологію проектування прийнято подавати як сукупність із трьох складових:

– послідовності виконання технологічних операцій, що передбачені процесом проектування;

– критеріїв щодо оцінювання результатів виконання технологічних операцій;

– нотацій (тобто графічних і текстових засобів) для описання системи, що проектується.

Кожна технологічна операція для свого виконання потребує матеріальних, інформаційних і трудових ресурсів, серед яких можуть бути:

– дані, що отримані від попередньої операції;

– методичні матеріалами, інструкції, нормативи, стандарти;

– програмні та технічні засоби;

– виконавці (розробники).

Результати здійснення операції мають бути подані у стандартному вигляді, що гарантує його адекватне сприйняття під час виконання наступної технологічної операції (де вони будуть використовуватись як вихідні дані).

Можна сформулювати низку загальних вимог до технології проектування, розробки й супроводження ІС, а саме:

– підтримувати повний життєвий цикл ІС;

– гарантувати досягнення цілей розробки із заданою якістю й у межах встановленого часу;

– забезпечувати можливість розподілу (декомпозиції) великих проектів на кілька складових частин (підсистем), що розробляються окремими групами виконавців із подальшою інтеграцією цих частин;

– надавати можливість проектування підсистем малими групами, що гарантує керованість колективом та підвищує продуктивність внаслідок мінімізації кількості зв’язків із зовнішнім середовищем;

– забезпечувати мінімальний час для отримання працездатної системи;

– давати можливість керувати конфігурацією проекту, вести облік та синхронізацію версій проекту та його складових;

– забезпечувати незалежність проектних рішень від специфіки засобів реалізації, – систем управління базами даних, операційної системи, мов програмування.

3.2 Парадігми проектування інформаційних систем

Існують дві головні парадигми проектування, що реалізують два різних підходи до опису систем [1,2,4]:

1) структурна (процесно-орієнтована);

2) об’єктно-орієнтована.

Перша концепція має в основі на каскадну модель ЖЦ інформаційної системи. Друга концепція заснована на ітеративній моделі ЖЦ.

Розглянемо головні відмінності та особливості використання цих підходів.

3.2.1 Структурний аналіз

Структурний аналіз (від англ. Structured Analysis або SA) і структурне проектування (або SD, від англ. Structured Design) бере свій початок від структурного програмування і класичного системного аналізу [5].

Сутність структурного підходу до розробки ІС полягає в її декомпозиції (розбитті) на функції, що підлягають подальшої автоматизації, а саме: система розбивається на функціональні підсистеми, ті зі свого боку розподіляються на підфункції, завдання тощо. Процес розбиття триває до рівня деталізації процедур. До того ж автоматизована система зберігає цілісне уявлення, коли всі складові компоненти взаємопов’язані та мають загальне підґрунтя.

Під час розробки системи у напряму «знизу-догори», тобто від окремих завдань до всієї системи цілісність втрачається, виникають проблеми у разі інформаційного стикування окремих компонентів.

Усі найпоширеніші методології структурного підходу [6] базуються на певних загальних принципах [7], головними з яких є такі:

– принцип «розділяй і володій» –підхід до вирішення складних проблем шляхом їхнього розбиття на низку менших та незалежних завдань, шо є легкими для сприйняття та практичної реалізації;

– принцип ієрархічного упорядкування – принцип організації складових частин проблеми та упорядкування їх у вигляді ієрархічної деревоподібної структури з додаванням нових деталей на кожному рівні.

Виділення цих двох базових принципів не означає, що інші принципи є другорядними, оскільки ігнорування кожного з них може призвести до непередбачуваних наслідків, зокрема до провалу проекту загалом. Головними з додаткових принципів є такі:

– принцип абстрагування, що полягає у виділенні ключових аспектів системи і абстрагування від несуттєвих;

– принцип формалізації, що полягає в необхідності чіткого методичного підходу до вирішення проблеми;

– принцип несуперечності, що полягає в обґрунтованості й узгодженості елементів;

– принцип структурування даних, він полягає у тому, що всі дані мають бути структуровані та ієрархічно впорядковані.

Методології структурного аналізу використовують каскадну модель життєвого циклу інформаційної системи.

У структурному аналізі використовуються дві головні групи засобів, що ілюструють функції системи та співвідношення між даними. Кожній групі засобів відповідають певні моделі (діаграми). Найпоширенішими серед них є такі:

– SADT (Structured Analysis and Design Technique), – моделі й відповідні функціональні діаграми;

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

– ERD (Entity-Relationship Diagrams) – діаграми «сутність-зв’язок».

На стадії проектування ІС моделі розширюються, уточнюються і доповнюються діаграмами, що відображають структуру програмного

забезпечення: архітектуру ПЗ, структурні схеми програм і діаграми екранних форм.

Вказані моделі в сукупності дають повне описання ІС незалежно від того, чи вона є існуючою чи тільки розробляється. Склад діаграм у кожному конкретному випадку залежить від рівня повноти описання системи. У подальших розділам ми повернемось до особливостей складання та використання діаграм DFD та ERD. Зараз розглянемо перший тип моделей, що є типовим для структурного підходу до проектування систем. Мова буде йти про методології структурного аналізу SADT (від англ. Structured Analysis and Design Technique) та SSADM. Вони базуються на структурному аналізі систем та графічному поданні інформації у вигляді системи функцій, що мають три класи структурних моделей:

– функціональна модель;
– інформаційна модель;
– динамічна модель.

Процес моделювання за методологією SADT складається з таких етапів:

– збір інформації та аналіз інформації щодо предметної області;
– документування отриманої інформації;
– моделювання в термінах IDEF0;
– коригування моделі у процесі ітеративного рецензування.

На сьогодні ця методологія (більш відома як IDEF0, [5]) використовує формалізований процес моделювання ІС та передбачає такі стадії:

– аналіз;
– проектування;
– реалізація;
– об’єднання;
– тестування;
– установка;
– функціонування.

Проектування інформаційних систем за стандартом IDEF0 полягає у послідовній декомпозиції основних функцій організації на окремі бізнес-процеси, роботи або дії з подальшим об’єднанням їх у єдину ієрархічну модель

організації. Декомпозицію можна робити багаторазово, аж до чіткого й вичерпного опису всіх процесів.

Діаграми IDEF0 верхнього рівня прийнято називати батьківськими, нижнього рівня – дочірніми.

Процес, що аналізується, виглядає як прямокутник. Зліва знаходяться вхідні дані, праворуч – вихідні, зверху – керуючі або регламентуючі дії, знизу – об'єкти управління. У діаграмі IDEF0 спочатку описують усі зовнішні зв'язки процесу. Після цього роблять декомпозицію цього процесу й описують внутрішні підпроцеси з позначенням усіх зв'язків. До того ж раніше позначені стрілочками зовнішні зв'язки не повинні загубитися. Вони переносяться до діаграми декомпозиції у відповідні підпроцеси.

Далі можна зробити подальшу декомпозицію кожного підпроцесу, після чого докладно описати усі зв'язки та функції.

Головною перевагою цієї методології є простота й наочність. За недолік можна вказати неможливість описати реакцію процесу на мінливі зовнішні фактори. Для вирішення таких завдань є інші методології.

Методи структурного аналізу удосконалювалися та використовувалися впродовж багатьох років. З часом з'явилися та набули популярності об'єктно-орієнтовані мови програмування. Завдяки цій популярності було розроблено методологію, що має допомагати програмісту розробляти додатки з використанням об'єктно-орієнтованих мов. Ця методологія стала відома як **об'єктно-орієнтований аналіз** та проектування (від англ. Object-oriented analysis and design або **OOAD**).

3.2.2 Об'єктно-орієнтований аналіз та програмні засоби щодо його реалізації

Об'єктно-орієнтований аналіз або OOAD – це підхід до інженерії програмного забезпечення, що розглядає систему як групу взаємодіючих об'єктів. Об'єктно-орієнтований аналіз (Object-oriented analysis, OOA) використовує методи об'єктного моделювання для аналізу функціональних вимог до системи.

Об'єктно-орієнтоване проектування, ООП (Object-oriented design, OOD) має за мету розробити аналітичні моделі процесів для подальшого створення специфікацій для їхньої реалізації. Однією з найважливіших серед

таких специфікацій є технічне завдання (ТЗ). Концептуальною основою ООП є об'єктна модель, яка будується з урахуванням принципів абстрагування, інкапсуляції, модульності, ієрархії, типізації, паралелізму, стійкості.

Головними поняттями об'єктно-орієнтованого підходу є **об'єкт** і **клас. Об'єкт** – це визначена сутність, що відповідає певному предмету або явищу й характеризується класом, станом і поведінкою. Для цих взаємодіючих між собою об'єктів можна створити різні моделі, що характеризують їхню статичну структуру або динамічну поведінку й розгортання в реальній роботі (run-time deployment). Клас – це множина об'єктів, що пов'язані спільною структурою та поведінкою.

Наступну групу важливих понять об'єктного підходу складають **поліморфізм** (здатність класу належати більш ніж одному типу) та **успадкування** (побудова нових класів на основі існуючих із можливістю додавання або коригування даних та методів).

У сучасній практиці використовують велику кількість об'єктно-орієнтованих методів проектування. Один із найпотужніших серед них – це IDEF4 (від англ. Object-Oriented Design). Цей метод становить методологію ООП, що дозволяє відображати структуру об'єктів і принципи їхньої взаємодії у контексті різних нотацій і об'єктних моделей.

Для опису об'єктно-орієнтованих моделей використовують спеціальні мови. Одними з найпопулярніших мов об'єктно-орієнтованого моделювання є UML (від англ. The Unified Modeling Language) та SysML.

Уніфікована мова моделювання UML – це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи (зокрема UML-моделі). Він був створений для визначення, візуалізації, проектування та документування переважно програмних додатків. Методи описання результатів аналізу та проектування UML семантично близькі до методів програмування на сучасних об'єктно-орієнтованих мовах. На підставі UML-моделей можлива генерація програмного коду. Крім того, на UML можна розробити докладний план системи, що містить системні функції і бізнес-процеси, схеми баз даних, програмні компоненти багаторазового використання тощо. Але головними перевагами використання UML є те, що:

1) UML дає змогу розглянути систему з усіх поглядів, що стосуються її розроблення й подальшого розгортання.

2) UML забезпечує підтримку всіх етапів життєвого циклу ІС і пропонує для цього відповідний набір діаграм: структурні (Structure Diagrams); діаграми поведінки (Behavior Diagrams); діаграми взаємодії (Interaction Diagrams).

UML фокусується на трьох архітектурних інтерпретаціях ІС:

– **функціональній** – описує зовнішню поведінку системи з погляду користувача за допомогою use-case діаграм;

– **статичній,** – представляє систему в термінах атрибутів, методів, зв'язків, класів, повідомлень за допомогою CRC (cards, class diagrams, object diagrams);

– **динамічній** – орієнтована на опис сервісних архітектур (SOAs) у вигляді діаграм послідовностей (sequence diagrams), схем співпраці (collaboration diagrams), діаграм стану (statecharts) тощо.

Формальна специфікація останньої версії UML була опублікована в серпні 2005 року. Протягом останнього часу семантика мови істотно змінювалась, удосконалювалась та була розширена до рівня підтримки методології Model Driven Development або MDD. Остання версія UML 2.4.1 була прийнята як міжнародний стандарт ISO/IEC 19505-1, 19505-2.

3.2.3 Проблеми вибору

Доступність різних технологій робить актуальною проблему вибору певного рішення щодо методології проектування системи. Поява нових, більш розвинутих об'єктно-орієнтованих мов мала б (на перший погляд) стимулювати потребу у використанні об'єктного підходу для розроблення бізнес-додатків. Однак на практиці того не сталося. У багатьох випадках використання об'єктно-орієнтованого програмування не є гарантованою перевагою перед традиційною структурною методологією будування системи. У будь-якому разі вибір рішення залежить від специфіки проекту й особливостей тої чи іншої технології. Нижче вказані головні переваги й недоліки структурної та об'єктно-орієнтованої методологій, які можуть вплинути на вибір під час проектування певної системи.

Переваги структурних методологій

Серед головних переваг структурного підходу до проектування варто відзначити такі [1–3]:

– вони засновані на класичному системному аналізі, є процесно-орієнтованими, підходять для описання будь-яких систем, зручні для дослідження предметної області, реінжинірингу бізнес процесів тощо;

– структурні методології дають зрозумілу і просту візуалізацію системи, їх легко сприймають і користувачі, і розробники проекту;

– у структурних методологіях акцент зроблено на командній роботі;

– чітко визначені етапи, що полегшують управління проектом і дозволяють розробляти системи кращої якості;

– структурні методології допускають використання засобів перевірки вимог;

– SSAD і IDEF0 – це класичні, широко розповсюджені методології в області проектування ІС, вони існують впродовж тривалого часу і тому добре перевірені на практиці.

Недоліки структурних методологій

Серед головних недоліків варто вказати на такі [1–4]:

– оскільки структурна методологія є процесно-орієнтованою, то вона ігнорує нефункціональні вимоги до системи, тобто результатом її використання можуть бути не ідеальні рішення з точки зору використання об’єктно-орієнтованих мов програмування на етапі реалізації проекту (методологія краще відповідає структурним мовам);

– оскільки SSAD і SSADM не є ітеративними, то зміна вимог може призвести до перезавантаження практично всього процесу розробки;

– під час використання структурного підходу можуть виникати труднощі у визначенні глибини декомпозиції, тобто моменту, коли доцільно зупинитися та перейти до практичної реалізації моделі.

Переваги об’єктно-орієнтованих методологій

Більшість цих переваг є наслідком об’єктно-орієнтованого програмування, а саме [1–4]:

– спрощення та прискорення програмної реалізації системи на відміну від структурних методологій;

– можливість повторного використання коду в інших проектах (наслідок незалежності об’єктів та інкапсуляції), це скорочує вартість проектування, програмування і тестування системи; також повторне використання коду може сприяти поліпшенню якості подальших проектів;

– відсутність чіткого розподілу між фазами аналізу і розробки, що забезпечує взаємодію між користувачами до завершення проекту;

– відсутність жорстких обмежень щодо впровадження системи; це дозволяє реалізувати проекти, що будуть працювати у різних середовищах;

– програмне забезпечення є добре пристосованим до змін, що надає більший рівень впевненості щодо його коректності, зменшує ризики під час розробки складних ІС;

– під час розробки об’єктів зі складною взаємодією аналітик акцентує увагу на іншому рівні деталізації, ніж це можливо в структурному підході, а стандартизація об’єктів збільшує рівень розуміння проекту.

Недоліки об’єктно-орієнтованих методологій

Зрозуміло, що будь-яка методологія має певні вади та обмеження. Не є винятком і об’єктно-орієнтований підхід. Серед його головних недоліків можна виділити такі [1–4]:

– початкова модель є занадто спрощеною, щоб вважатись за адекватну;

– об’єктно-орієнтований підхід надмірно фокусується на коді;

– недостатньо уваги приділяється командній роботі (на відміну від структурних методологій);

– визначення всіх необхідних для системи класів і об’єктів не є простим завданням і потребує специфічної підготовки розробників ІС;

– спроба поєднати об’єктне програмування з аналізом функцій системи є проблематичним; ці функціональні методи не відповідають OOAD;

– існує перебільшення значущості й універсальності об’єктної методології; не береться до уваги той факт, що в певній ситуації інший підхід міг би дати кращі результати на етапі аналізу та розробки системи;

– об’єктно-орієнтована методологія потребує нового рівня управління проектами, він має охоплювати різні типи аналізу, відмінні від традиційного функціонального підходу з використанням декомпозиції; тому для команд розробників проекту, які мають досвід використання методологій SSAD і SSADM, перехід до OOAD є складним і тривалим процесом;

– прихильники об’єктно-орієнтованого підходу вважають концепцію повторного використання великою перевагою і головною причиною для переходу на OOAD, однак в багатьох реальних проектах нема успішного повторного використання у великих масштабах;

– функціональний опис системи в термінах UML ґрунтується на сценаріях, які відповідають документуванню вимог, що погано відповідають описанню таких взаємодій, як алгоритм, математичні вимоги, не функціональні вимоги, платформа, продуктивність, синхронізація, безпека тощо; використання шаблонів не гарантує якості сценаріїв, яка залежить тільки від навичок розробника;

– об’єктний підхід до моделювання даних часом погано співіснує з використанням реляційних моделей, які присутні в переважній більшості інформаційних систем; саме тому інформаційні системи часто розробляються через комбінацію об’єктно-орієнтованих мов програмування і реляційних баз даних, а не спираються тільки на об’єктні бази даних і мови програмування.

Висновок. Під час розроблення певного додатку насамперед треба визначитись, яка методологія найбільше підходить для вирішення конкретного завдання. Варто мати на увазі, що у практичній роботі можливі ситуації, коли доведеться адаптувати різні методології, і такий підхід буде доцільним. Вважається, що для вирішення простих завдань краще підходять структурні методи, водночас як об’єктно-орієнтовані технології доречніше використовувати для більш високих рівнів абстракції. У ситуаціях, де ймовірність зміни даних вища за зміну функціональності, об’єктні методи також мають чисельні переваги.

3.3 Сучасні методології проектування інформаційних систем

3.3.1 Методологія RAD

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

автоматизації практично всіх етапів життєвого циклу інформаційних систем. Багато з цих інструментів засновані на так званій **методології RAD** [8,9].

Головні особливості методології RAD

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

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

1) невелика команда програмістів (від 2 до 10 осіб). Команда розробників повинна складатися з групи професіоналів, які мають досвід в аналізі, проектуванні, генерації коду і тестуванні ПЗ із використанням CASE-засобів. Члени колективу мають також уміти трансформувати в робочі прототипи пропозиції кінцевих користувачів;

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

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

Головні засади методології RAD:

– ітераційна розробка програмного забезпечення;

– допустимість часткового завершення робіт на довільному етапі ЖЦ;

– обов'язкове залучення користувачів до процесу розробки ІС;

– обов'язкове використання CASE-засобів з метою забезпечення цілісності проекту;

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

– активне використання генераторів коду;

– впровадження прототипів задля кращого розуміння й задовільнення потреб користувача;

– паралельне тестування, розробка й розвиток проекту;

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

– чітке планування та контроль за виконанням робіт.

Об’єктно-орієнтований підхід у методології RAD

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

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

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

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

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

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

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

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

Візуальне програмування і методологія RAD

Застосування принципів ООП дало змогу створити принципово нові інструменти проектування додатків, що отримали назву «засобів візуального програмування». Такі інструменти RAD дають змогу створювати складні інтерфейси користувача практично без написання коду програми. Тобто розробник може на будь-якому етапі бачити, що буде взято за підґрунтя майбутніх рішень.

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

На сьогодні існує багато візуальних засобів розробки додатків. Незважаючи на певну специфіку, всі вони діляться на дві категорії: універсальні та спеціалізовані.

З **універсальних** систем візуального програмування найбільш поширеними є C, Java, Visual Basic і багато інших. Універсальними вони

вважаються тому, що допомагають у розробці додатків практично будь-якого типу, зокрема – ІС. До того ж програми, які розробляються за допомогою універсальних систем, можуть взаємодіяти майже з усіма відомими системами управління базами даних.

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

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

Подієве програмування і методологія RAD

Логіка додатків, побудованих засобами RAD, є подієво-орієнтованою. Це означає, що кожен об'єкт зі складу програми, може створювати події та реагувати на події, що генеруються іншими об'єктами. Прикладами подій можуть бути відкриття і закриття вікон, клацання на кнопці, натискання клавіші на клавіатурі, рух миші, зміни у базі даних, тощо.

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

Життєвий цикл інформаційної системи за методологією RAD

Згідно методології RAD життєвий цикл ПЗ має чотири фази:

1) аналіз і планування вимог;
2) проектування;
3) побудова;
4) впровадження.

1. На **фазі аналізу** і планування вимог визначають функції системи, виділяють з них найбільш пріоритетні, описують інформаційні потреби. Визначення вимог виконують зазвичай користувачі за допомогою фахівців-розробників. Тут же обмежують масштаб проекту, визначають терміни реалізації для кожної з фаз. Крім того, визначають саму можливість реалізації проекту у межах доступного фінансування, існуючих апаратних засобах тощо.

Результатом цієї фази є:

1) список і пріоритетність функцій ІС;

2) попередні моделі ІС (функціональні та інформаційні) .

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

– докладніше розглядають процеси системи. Аналізують і (в разі потреби) коригують функціональну модель. Кожен процес розглядають докладно. У разі потреби для кожного елементарного процесу створюють спеціальний прототип;

– визначають вимоги розмежування доступу до інформації;

– визначають набір необхідної документації;

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

– за допомогою CASE-засобів проект ділиться між різними командами (формується функціональна модель).

Результатом фази проектування є:

– інформаційна модель ІС загального рівня;

– функціональні моделі ІС та кожної з її підсистем;

– інтерфейси між підсистемами, визначені за допомогою CASE-засобів;

– прототипи екранів, звітів, діалогів.

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

інформації щодо проекту може відбутися неконтрольоване спотворення даних. Застосування єдиного середовища зберігання інформації щодо проекту дозволяє уникнути такої небезпеки.

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

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

1) визначається необхідність розподілу даних;

2) проводиться аналіз використання даних;

3) робиться фізичне проектування бази даних;

4) визначаються вимоги до апаратних ресурсів;

5) визначаються способи збільшення продуктивності;

6) завершується розроблення документації проекту.

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

4. На фазі впровадження проводиться навчання користувачів, організаційні зміни й паралельно з впровадженням нової системи здійснюється робота з існуючою системою (до повного впровадження нової). Оскільки фаза побудови досить коротка, планування і підготовка до впровадження мають починатися заздалегідь, ще на етапі проектування ІС.

Наведена схема розроблення ІС не є абсолютною. Можуть бути різні варіанти. Це залежить, наприклад, від початкових умов, як то:

– розробляється зовсім нова система;

– вже зроблено обстеження підприємства та є модель його діяльності;

– на підприємстві є діюча ІС, яку: а) можна використати як прототип або б) треба інтегрувати до ІС, що з розробляється.

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

Методологія RAD мало прийнятна:

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

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

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

4) для розробки ІС, від яких залежить безпека людей (наприклад, систем управління транспортом, військовою технікою, атомними електростанціями). Це є наслідком того, що ітеративний підхід допускає на початкових етапах використання не повністю працездатних версій системи.

3.3.2 Методологія RUP

Серед виробників CASE-засобів компанія IBM Rational Software Corp. однією з перших усвідомила стратегічну перспективність розвитку об’єктно-

орієнтованих технологій аналізу та проектування програмних систем. Саме ця компанія стала ініціатором уніфікації мови візуального моделювання в межах консорціуму OMG, що призвело до появи перших версій UML. Саме вона вперше розробила об'єктно-орієнтований CASE-засіб, де було реалізовано мову UML як базову нотацію візуального моделювання. Унаслідок цього з'явилась одна з найпопулярніших технологій проектування інформаційних систем – Rational Unified Process (RUP) [10]. На сьогодні ця методологія у певному розумінні стає міжнародним стандартом від компанії Rational Software, що входить до складу IBM. Авторами UML вважаються співробітники фірми Граді Буч, Айвар Якобсон, Джемс Рамбо. RUP повністю відповідає стандартам, що визначають проектні роботи в процесі життєвого циклу ІС. У методології RUP реалізовано такі підходи:

– ітераційний та інкрементний (нарощуваний);
– побудова системи на базі архітектури інформаційної системи;
– планування і управління проектом на підставі функціональних вимог до інформаційної системи.

Розроблення інформаційної системи проходить ітераціями. Це окремі проекти, що невеликі за обсягом та змістом, включають власні етапи аналізу вимог, проектування, реалізації, тестування, інтеграції.

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

3.3.3 Стандарти проектування 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 реалізації проекту.

У цьому разі за основу корпоративних стандартів можуть прийматися галузеві, національні й навіть міжнародні стандарти. Останні можна умовно поділити на декілька груп [1, 2].

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

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

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

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

- на продукти й послуги;
- на процеси та технології;
- на форми колективної діяльності, або управлінські стандарти[5].

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

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

Методика CDM фірми Oracle

Компанія Oracle добре відома на ринку програмного забезпечення як лідер із розроблення потужних СУБД та відповідних інструментів щодо проектування баз даних. Однак цим сфера інтересів Oracle не обмежується,

[5] Докладніше до питання класифікації стандартів ми повернемось у четвертому розділі Посібника.

оскільки одним із важливих напрямів діяльності фірми є розроблення методологічних основ та інструментальних засобів для автоматизації процесів проектування та реалізації складних інформаційних систем, що орієнтовані на активне використання баз даних. Методика CDM є розвитком давно розробленої методики CASE-Method фірми Oracle, яку реалізовано у CASE-засобі Oracle Designer/2000 [1, 2]. Розглянемо головні складові CASE-технології та інструментального середовища від фірми Oracle.

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

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

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

4) Під час розроблення всі специфікації проекту зберігаються в спеціальній базі даних (репозиторії). Цей засіб включено до складу ПО Дизайнер і працює під управлінням СУБД Oracle. Репозиторій дає змогу підключатися до нього великому числу користувачів із різними рівнями прав доступу шляхом стандартних засобів СУБД Oracle. Унаслідок цього всі дії розробників стають строго узгодженими та стають неможливими незалежні дії кожного з них.

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

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

Структура життєвого циклу згідно з методологією CDM

Методика CDM від Oracle пропонує свій варіант структури життєвого циклу інформаційної системи:

1) формування стратегії. Цей етап передбачає моделювання та аналіз процесів, які описують діяльність організації, особливості роботи. Кінцева мета – створення моделей процесів, виявлення можливостей їхнього

вдосконалення. Етап не обов'язковий, якщо існуючі технології та організаційні структури чітко визначені, добре вивчені й загалом зрозумілі;

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

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

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

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

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

Варто відзначити, що Oracle **не розглядає** таку стадію існування ІС, як **зняття з експлуатації**. Тобто розробник, який застосовує методику і ПО Oracle, автоматично «підсаджується» на нього. Інша справа, що зручність засобів розроблення багато в чому виправдовує таку залежність.

Методика CDM виділяє такі процеси, що мають перебіг протягом ЖЦ ІС:

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

Особливості методики CDM

На наш погляд, серед чисельних особливостей методики головними є такі:

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

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

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

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

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

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

3.4 Питання для самоконтролю

1. Дайте визначення: що таке проектування інформаційної системи.

2. У чому полягає мета проектування інформаційної системи?

3. Які є компоненти проекту інформаційної системи?

4. Що таке методологія проектування інформаційної системи?

5. Опишіть зв'язок між методологією, технологією та інструментальними засобами проектування.

6. Які має виконати методологія проектування ІС?

7. Перелічіть головні складові технології проектування ІС.

8. Назвіть загальні вимоги, яким має задовільняти технологія проектування, розробки й супроводження інформаційних систем.

9. Назвіть три класи структурних моделей. Наведіть приклади.

10. Назвіть головні етапи моделювання за методологією SADT.

11. Назвіть головні стадії моделювання процесів за методологією IDEF0.

12. Що таке об’єктно-орієнтований аналіз (OOAD), які його відмінності?

13. Вкажіть головні поняття об’єктно-орієнтованого підходу до проектування інформаційних систем.

14. Для чого використовують мову UML? Наведіть приклади.

15. Вкажіть ключові переваги структурного підходу до проектування ІС.

16. Вкажіть ключові недоліки структурного підходу до проектування ІС. У чому полягають обмеження щодо використання цього підходу?

17. Які переваги об’єктно-орієнтованих технологій до проектування ІС?

18. Які недоліки та обмеження об’єктно-орієнтованих технологій до проектування ІС?

19. У чому полягають сутність та особливості методології RAD?

20. Вкажіть головні принципи методології RAD.

21. Назвіть етапи життєвого циклу ІС згідно до ідеології RAD.

22. Яка головна мета й завдання фази аналізу за методологією RAD?

23. Яка мета й завдання фази проектування за методологією RAD?

24. У чому полягають обмеження використання методології RAD?

25. Назвіть головні принципи та ідеї методології RUP.

26. Охарактеризуйте класифікацію стандартів на проектування ІС.

27. Які принципи покладено в основу методології CDM?

28. Охарактеризуйте структуру ЖЦ згідно до методології CDM.

29. Назвіть процеси життєвого циклу згідно з методологією CDM.

30. Назвіть особливості методології CDM, її переваги, недоліки та обмеження щодо використання.

4 ОРГАНІЗАЦІЯ ПРОЕКТУВАННЯ ТА НОРМАТИВНО-МЕТОДИЧНА ПІДТРИМКА ЖИТТЄВОГО ЦИКЛУ ІНФОРМАЦІЙНИХ СИСТЕМ

Головні питання: *різновиди та класифікація стандартів для створення ІС. Міжнародні та вітчизняні стандарти проектування ІС. Стандарти на процеси життєвого циклу. Корпоративні стандарти проектування ІС, Канонічний підхід до проектування та його відмінності. Стадії канонічного підходу до проектування ІС. Типове проектування ІС. Організація тестування та налагодження автоматизованих інформаційних систем, обробка даних під час налагодження програм. Атестація та верифікація інформаційних систем. Проблеми впровадження та супроводження інформаційних систем. Стандарти та методики, що регламентують життєвий цикл інформаційної системи. Міжнародний стандарт ISO/IEC12207 (Information Technology – Software Life Cycle Processes) та його практичне застосування. Канонічне проектування ІС. Склад стадій та етапів канонічного проектування. Методологія RAD (Rapid Application Development) – методологія швидкої розробки додатків. Методика Oracle CDM (Custom Development Method) з розробки прикладних інформаційних систем під замовлення.*

Методології створення ІС описують підтримку певної моделі життєвого циклу (ЖЦ) інформаційної системи. Методології реалізуються через *стандарти*, *методики*, *технології* та *інструментальні засоби* (або CASE-засоби), що підтримують їхню практичну реалізацію [1–4].

4.1 Класифікація та різновиди стандартів для проектування ІС

Комплекс документів, що регламентує діяльність розробників, називають *нормативно-методичним забезпеченням* (НМЗ). Документи, що входять до складу НМЗ – це стандарти, керівні документи, методики, положення, інструкції, шаблони, тощо. Вказані документи регламентують:

– порядок розроблення, впровадження та супроводу системи;

– загальні вимоги до складу системи, до зв’язків між її компонентами, до якості програмного забезпечення;

– різновиди, склад і зміст проектної та програмної документації.

Історично склалося так, що одним із найпоширеніших стандартів щодо створення програмних систем де-факто є ЄСПД, – Єдина Система Програмної Документації (серія ГОСТ 19.XXX). Від самого початку ці стандарти були орієнтовані на клас порівняно простих програм, які могла розробити невелика група програмістів.

Стандарти ЄСПД [1] практично позбавлені змістовної складової і містять формальні вимоги до складу, змісту та оформлення документів, що описують програму на різних стадіях її життєвого циклу. Також в них є вимоги до порядку зберігання й оновлення документації. Деякі автори вважають ці стандарти застарілими (концептуально й за формою), та водночас ЄСПД продовжують активно використовувати під час оформлення ПД.

Зважаючи на **статус** до сучасної бази НМО можна віднести такі міжнародні та вітчизняні стандарти [1, 2]:

1) офіційні (public) стандарти:

– міжнародні:

- ISO/IES (ISO – від International Organization of Standardization, міжнародна організація зі стандартизації; IEC – від International Electrotechnical Commission, міжнародна комісія з електротехніки);
- ANSI (від American National Standards Institute);
- стандарти міжнародних консорціумів та комітетів зі стандартизації (наприклад консорціум OMG);

– стандарти України (ГОСТ, ДСТУ):

- ГОСТ 34.601-90 «Інформаційна технологія. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення»;
- ГОСТ 34.602-89 «Інформаційна технологія. Комплекс стандартів на автоматизовані системи. Технічне завдання на створення автоматизованої системи»;
- ГОСТ 34.603-92 «Інформаційна технологія. Види випробувань автоматизованих систем».

– галузеві стандарти;

– відомчі стандарти.

2) Стандарти «де-факто» – стандарти, що офіційно не затверджені, але фактично діють (наприклад стандартом «де-факто» довгий час були мови взаємодії з базами даних SQL і мова програмування С), фірмові стандарти (наприклад Microsoft ODBC) .

Інші способи класифікації стандартів

1. За об’єктом стандартизації:

– стандарти на продукти й послуги;
– стандарти на процеси й технології;
– стандарти на форми колективної діяльності, або управлінські стандарти.

2. За предметом стандартизації:
1) *функціональні стандарти*:
– стандарти на мови програмування;
– стандарти на інтерфейс, протоколи тощо;
2) *стандарти на організацію життєвого циклу ІС.*

До окремої групи прийнято виділяти корпоративні стандарти.

4.2 Корпоративні стандарти

Реальне застосування будь-якої технології проектування, розробки й супроводу ІС у певній організації або проекті неможливе без застосування низки стандартів (правил або угод), яких мають дотримуватись усі учасники проекту. Для більшості складних проектів доводиться створювати власні комплекти нормативних і методичних документів, що регламентують процеси, етапи, роботи й документи для певних програмних продуктів. Такі стандарти називаються **корпоративними** й становлять угоду щодо єдиних правил організації технології або управління у межах організації. До таких стандартів належать:

– стандарти проектування;
– стандарти на оформлення документації;
– стандарти на інтерфейси користувача.

Стандарт проектування має регламентувати:

– набір моделей (діаграм) на кожному етапі проектування системи та рівень їх деталізації;

– правила фіксації проектних рішень на діаграмах зокрема правила іменування (ідентифікації) об’єктів, набір атрибутів для всіх об’єктів, правила заповнення цих атрибутів на кожній стадії проектування, правила оформлення діаграм, зокрема вимоги до їхньої форми, розмірів, наповнення тощо;

– вимоги до конфігурації робочих місць, налаштування операційної системи, загальні налаштування проекту тощо;

– механізм підтримки спільної роботи над проектом, зокрема: правил інтеграції підсистем проекту, підтримки проекту в актуальному для всіх розробників стані (регламент обміну проектною інформацією, правила та методики синхронізації тощо), правил перевірки проектних рішень на їхню несуперечність тощо.

Стандарт оформлення проектної документації має встановлювати:

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

– вимоги до її оформлення у т. ч зміст розділів, підрозділів, пунктів, таблиць тощо;

– правила підготовки, розгляду, погодження та затвердження документації із вказанням граничних строків виконання кожної стадії;

– вимоги до налаштування технічних засобів підготовки документації;

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

Стандарт інтерфейсу користувача має встановлювати:

– правила оформлення екранних форм (шрифти, колір символів, тла), склад і розташування вікон та елементів управління;

– правила використання засобів вводу (клавіатури, миші тощо);

– правила оформлення текстів довідок;

– перелік повідомлень;

– правила обробки (реакцію) на дії користувача.

За основу корпоративних стандартів можуть прийматися *галузеві*, *національні* або *міжнародні* стандарти. До них можуть належати різного роду методичні матеріали провідних фірм-розробників ПЗ, фірм-консультантів, наукових центрів, консорціумів зі стандартизації тощо.

4.3 Стандарти на процеси життєвого циклу інформаційних систем

4.3.1 Міжнародні стандарти проектування інформаційних систем

ISO/IEC 12207 – базовий стандарт на процеси життєвого циклу ІС, орієнтований на різні типи проектів. У стандарті не передбачено конкретних етапів ЖЦ ІС. Замість того визначено тільки низку процесів. Тому стандарт дає змогу реалізувати довільну модель життєвого циклу, і це є його перевагою.

ГОСТ 34.601-90 – поширюється на автоматизовані інформаційні системи (АІС) і регламентує стадії, етапи їхнього створення, містить описання змісту робіт на кожному з етапів. Стандарт орієнтовано на використання каскадної моделі життєвого циклу.

ISO/IEC 12207: 1995-08-01 й супутні стандарти

Міжнародний стандарт **ISO/IEC 12207** (запропонований у 1995 р. технічним комітетом ISO/IEC JTC1 «Інформаційні технології, підкомітет SC7, проектування програмного забезпечення») є найважливішим нормативним документом, що регламентує життєвий цикл програмного забезпечення. Він визначає структуру життєвого циклу, що містить процеси, дії та завдання, які мають бути виконані під час створення ПЗ. Його регламенти є загальними для будь-яких моделей життєвого циклу, методологій і технологій розроблення ПЗ. Способи виконання дій та завдань, включених до перерахованих процесів, можуть бути довільного типу.

Відповідно до базового міжнародного стандарту ISO/IEC 12207 усі процеси життєвого циклу поділяються на три групи:

1. Головні процеси:

1) придбання, – визначає дії підприємства-покупця;

2) постачання, – визначає дії підприємства-постачальника;

3) розробка, – визначає дії підприємства-розробника;

4) функціонування, – визначає дії підприємства-оператора, що забезпечує обслуговування системи загалом (а не тільки програмного забезпечення) у процесі її функціонування;

5) супровід, – визначає дії персоналу, що забезпечує супроводження програмного продукту, тобто управління, модифікацію, підтримку поточного стану та функціональної придатності; до цього процесу також належить встановлення (інсталяція) ПЗ на обчислювальній системі та його видалення.

2. Д**опоміжні процеси** – процеси, що призначені для підтримки виконання головних процесів, забезпечення якості проекту, організації верифікації, перевірки й тестування ПЗ тощо:

1) процес документування;

2) процес управління конфігурацією;

3) процес забезпечення якості;

4) процес верифікації;

5) процес атестації;

6) процес аудиту;

7) процес спільної оцінки;

8) процес вирішення проблем.

3. Організаційні процеси визначають дії і завдання, що виконують як замовник, так і розробник проекту, а саме:

1) процес управління;

2) процес створення інфраструктури проекту;

3) процес удосконалення;

4) процес навчання.

Характеристики стандарту ISO/IEC 12207:

– *динамічність*: забезпечується послідовністю виконання процесів, коли один процес у разі потреби активізує інший (повністю або частково); це дозволяє реалізувати довільну модель ЖЦ;

– *адаптивність*: стандарт ISO 12207 передбачає виключення процесів, різновидів діяльності й завдань, непридатних у певному проекті.

Нижче наведено орієнтовний опис головних процесів життєвого циклу (табл. 4.1–4.2).

Таблиця 4.1 – Орієнтовний опис процесу придбання. Відношення «Виконавець – Замовник»

Вхід	**Дія**	**Вихід**
1	**2**	**3**
1 Рішення щодо початку робіт із впровадження ІС. 2 Результати обстеження діяльності замовника.	1 Ініціювання. 2 Підготовка заявочних пропозицій. 3 Підготовка договору. 4 Контроль діяльності постачальника.	1 Техніко-економічне обґрунтування впровадження ІС. 2 Технічне завдання на ІС. 3 Договір на постачання/розроблення.

Вхід	Дія	Вихід
1	2	3
3 Результати аналізу ринку ІС. 4 План постачання/ розроблення. 5 Комплексне тестування ІС	5 Приймання ІС	4 Акти приймання етапів роботи. 5 Акт приймально-здавальних випробувань
1 Технічне завдання на ІС. 2 Рішення керівництва про участь у розробці. 3 Результати тендеру. 4 Технічне завдання на ІС. 5 Розроблена ІС і документація.	1 Ініціювання. 2 Відповідь на заявочні пропозиції. 3 Підготовка договору. 4 Планування виконання. 5 Постачання ІС.	1 Рішення щодо участі у розробленні. 2 Комерційні пропозиції/конкурсна заявка. 3 Договір на постачання/розроблення. 4 План управління проектом. 5 Акт приймально-здавальних випробувань.

Таблиця 4.2 – Орієнтовний опис процесу розробки. Відношення «Виконавець – Розробник ІС»

Вхід	Дія	Вихід
1 Технічне завдання на ІС. 2 Технічне завдання на ІС, модель ЖЦ. 3 Технічне завдання на ІС. Підсистеми ІС. 4 Специфікації вимоги до компонентів ПО. 5 Архітектура ПО. 6 Матеріали докладного проектування ПО. 7 План інтеграції ПО, тести. 8 Архітектура ІС, ПЗ, документація на ІС, тести	1 Підготовка. 2 Аналіз вимог до ІС. 3 Проектування архітектури ІС. 4 Розроблення вимог до програмного забезпечення. 5 Проектування архітектури ПЗ. 6 Докладне проектування ПО. 7 Кодування і тестування програмного забезпечення. 8 Інтеграція ПО і кваліфікаційне тестування ПО. 9 Інтеграція ІС і кваліфікаційне тестування ІС	1 Використовувана модель ЖЦ, стандарти розроблення. 2 План робіт. 3 Склад підсистем, компоненти обладнання. 4 Специфікації вимоги до компонентів ПЗ. 5 Склад компонентів ПО, інтерфейси з БД, план інтеграції ПЗ. 6 Проект БД, специфікації інтерфейсів між компонентами ПЗ, вимоги до тестів. 7 Тексти модулів ПЗ, акти автономного тестування. 8 Оцінка відповідності комплексу ПЗ вимогам ТЗ 9 Оцінка відповідності ПО, БД, технічного комплексу та комплекту документації вимогам ТЗ

Для підтримки застосування стандарту ISO/IEC 12207 було розроблено низку технологічних документів, а саме:

– керівництво для ISO/IEC 12207 (ISO/IEC TR 15271 : 1998 Information technology – Guide for ISO/IEC 12207);

– керівництво щодо застосування ISO/IEC 12207 в управлінні проектами (ISO/IEC TR 16326:1999 Software engineering – Guide for the application of ISO/IEC 12207 to project management).

З часом було розроблено стандарт на процеси ЖЦ систем ISO/IEC 15288 (System life cycle processes). Під час створення стандарту було враховано практичний досвід створення систем в урядових, комерційних, військових та академічних організаціях. У розробці стандарту брали участь провідні фахівці з системної інженерії, програмування, управління якістю, людськими ресурсами, безпекою тощо. Стандарт було орієнтовано на широкий клас систем, однак його головне призначення – це підтримка створення інформаційних систем. Згідно стандарту ISO/IEC серії 15288 до структури життєвого циклу рекомендовано включати такі групи процесів:

1. Договірні процеси:

– придбання комплектуючих, програмного забезпечення, контенту тощо (це можуть бути внутрішні рішення або рішення від зовнішнього постачальника);

– постачання (внутрішні рішення або рішення зовнішнього постачальника).

2. Процеси підприємства:

– управління навколишнім середовищем;
– інвестиційне управління;
– управління життєвим циклом ІС;
– управління ресурсами;
– управління якістю.

3. Проектні процеси:

– планування проекту;
– оцінка проекту;
– контроль проекту;
– управління ризиками;
– управління конфігурацією;

– управління інформаційними потоками;
– прийняття рішень.

4. Технічні процеси:

– визначення вимог;
– аналіз вимог;
– розробка архітектури;
– впровадження;
– інтеграція;
– верифікація;
– перехід;
– атестація;
– експлуатація;
– супроводження;
– утилізація.

5. Спеціальні процеси: визначення та встановлення взаємозв'язків відповідно до завдань і цілей проекту.

Стадії створення ІС відповідно до ISO/IEC 15288:

– формування концепції, – аналіз потреб, вибір принципів щодо реалізації проектних рішень;
– розроблення, – проектування самої системи;
– реалізація, – розробка системи на рівні баз даних та програмного коду;
– експлуатація, – введення системи до експлуатації та її подальше використання;
– підтримка, – забезпечення функціонування системи, внесення відповідних змін з метою адаптації до умов зовнішнього середовища;
– зняття з експлуатації, – припинення використання системи, її демонтаж, створення архіву даних тощо.

4.3.2 Канонічне проектування інформаційних систем та його нормативне забезпечення

За своєю сутністю канонічне проектування відображає технологію індивідуального проектування ІС. Серед особливостей канонічного підходу можна виділити такі:

– відображення особливостей «ручної» технології проектування;

– орієнтація на індивідуальне проектування;

– акцент на планування та розподілі робіт на рівні виконавців;

– можливість інтеграції виконання елементарних операцій;

– орієнтація на порівняно невеликі інформаційні проекти;

– використання універсальних інструментальних засобів комп'ютерної підтримки.

Канонічне проектування спрямоване на *мінімальне* використання типових проектних рішень. Адаптація проектних рішень у канонічному проектуванні здійснюється виключно шляхом перепрограмування відповідних програмних модулів.

Організація канонічного проектування ІС заснована на використанні *каскадної* моделі життєвого циклу й передбачає низку стадій та етапів. Принцип розподілу процесу проектування на стадії та етапи спрямований на проектування системи «зверху-вниз», тобто на послідовну розробку спочатку узагальнених, а потім деталізованих проектних рішень.

Оскільки об'єкти автоматизації за складністю різні, то набір завдань для локальних рішень у межах певної системи, стадії та етапи робіт для її створення можуть суттєво відрізнятись за трудомісткістю. Тому допустимо об'єднувати послідовні етапи, вилучати певні етапи на довільній стадії проекту, а також починати виконання наступної стадії до закінчення попередньої. Усі стадії та етапи створення ІС фіксуються в договорах й технічних завданнях на виконання робіт.

Канонічне проектування регламентує низка стандартів, а саме:

ГОСТ 34.003 – Терміни та визначення головних понять у сфері автоматизованих систем;

ГОСТ 34.201 – Види, комплектність і позначення документів при створенні автоматизованих систем;

ГОСТ 34.601 – Стадії створення автоматизованих систем;

ГОСТ 34.602 – Технічне завдання на створення ІС;

ГОСТ 34.603 – Види випробувань автоматизованих систем;

ГОСТ 2.105 – Загальні вимоги до текстових документів.

Стосовно проекту розробки ІС можна виділити три узагальнені стадії проектування:

– передпроектну;
– проектну;
– післяпроектну.

Розглянемо вказані стадії більш детально.

4.3.3 Стадії та етапи канонічного проектування інформаційних систем

Головні етапи робіт (стадії) відповідно до ГОСТ 34.601 і додаткових пояснень, виглядають так:

1. Формування вимог до інформаційної системи

Має за мету виконання таких дій:

– обстеження об’єкту та обґрунтування необхідності створення ІС;
– формування вимог користувача до ІС;
– оформлення звіту щодо виконаної роботи й заявки на розроблення ІС.

2. Розроблення концепції ІС.

Ця стадія складається з таких етапів:

– вивчення об’єкта;
– проведення науково-дослідних робіт (за необхідністю);
– розроблення варіантів концепції ІС відповідно до вимог користувачів;
– оформлення звіту щодо виконаної роботи.

3. Технічне завдання

Це дуже відповідальний етап, що має за мету розроблення й затвердження ТЗ на створення інформаційної системи. Підготовчим етапом для формування технічного завдання є техніко-економічне обґрунтування проекту. Це спеціальний документ, що фіксує результати визначення стратегії впровадження ІС. У цьому документі має бути чітко визначено результати виконання проекту для замовника, а також вказані графіки проведення робіт і графік фінансування на різних стадіях виконання проекту. Додатково у такому документі вказують терміни, період окупності, очікувані зиски та економічний

ефект від реалізації проекту. Зазвичай, у техніко-економічному обґрунтуванні вказують:

– усі ризики та обмеження, що впливають на успішність проекту;

– умови експлуатації майбутньої системи: архітектурні, програмні, вимоги до апаратних засобів та до компонентів ПО і СУБД;

– користувачів системи (можливо, за категоріями);

– функції, що виконуються системою;

– інтерфейси та розподіл функцій між людиною та системою;

– терміни завершення етапів, форма прийому/передачі робіт;

– горизонт проекту;

– можливості подальшого розвитку системи.

За результатами обстежень формується **технічне завдання** на інформаційну систему.

Відповідно до ГОСТ 34.602-89, технічне завдання – це головний документ, що визначає вимоги та порядок створення (розвитку або модернізації) системи, згідно до яких проводиться її розроблення та приймання під час введення до експлуатації.

Розроблення технічного завдання складається із таких розділів:

– загальні відомості;

– призначення та мета створення (розвитку) ІС;

– характеристика об’єктів автоматизації;

– вимоги до ІС;

– склад і зміст робіт щодо створення ІС;

– порядок контролю та приймання ІС;

– вимоги до складу та змісту робіт щодо підготовки та введення ІС до експлуатації;

– вимоги щодо документування;

– джерела розроблення.

Проектна стадія орієнтована переважно на розробку технічного та робочого проектів. Процес розробки технічного завдання передбачає обстеження об’єкта автоматизації (організації або підрозділу) та його діючої системи управління. Для вирішення завдань інформаційного забезпечення

необхідно також проаналізувати інформаційні потоки, форми документації, системи кодування, а також усе, що пов'язано зі структурою баз даних та СУБД, що визначає склад вихідних технологічних вимог.

4. Ескізний проект.

Включає такі етапи робіт:

– розробка попередніх проектних рішень щодо ІС та її складових;
– розробка ескізної документації на систему та на її складові.

Якщо для ІС певного об'єкта автоматизації раніше обрані проектні рішення є очевидними, стадія ескізного проекту може бути виключена з послідовності робіт. Тобто ця стадія не є обов'язковою.

Крім того, на етапі ескізного проекту мають бути визначені:

– цілі та функції ІС та її підсистем;
– склад комплексів задач і окремих завдань;
– концепція та структура інформаційної бази;
– функції СУБД;
– функції та параметри програмних засобів;
– очікуваний ефект від впровадження системи.

Документація з результатами робіт за сукупністю прийнятих проектних рішень погоджується, затверджується і використовується у подальшому проектування для виконання робіт щодо створення ІС.

На підставі технічного завдання (зокрема за наявністю ескізного проекту) розробляється технічний проект.

5. Технічний проект

Включає такі етапи:

– розробка проектних рішень щодо системи та її складових;
– підготовка документації на систему в цілому та на її окремі складові;
– розробка й оформлення документації на постачання комплектуючих до системи і (або) технічних вимог (завдань) на їх розробку;
– розробка завдань на проектування в суміжних частинах проекту об'єкта автоматизації.

На етапі технічного проекту проводяться роботи науково-дослідного та експериментального характеру для вибору головних проектних рішень, а також розраховується економічна ефективність системи. Важливим аспектом розробки технічного проекту є аналіз всієї інформації, що використовується на предмет таких характеристик, як повнота, відсутність дублювання і надмірності, несуперечливість тощо. Також на цьому етапі визначають форми вихідних документів.

Документацію за результатами робіт оформляють відповідно до вимог ГОСТ 34-201.

6. Робоча документація (РД)

Передбачає такі дії:

- розроблення РД на систему й окремі її складові;
- розроблення або адаптацію програм.

Один із головних етапів стадії робочого проектування – розроблення РД на інформаційне забезпечення системи. До складу такої документації входять:

- технічний проект ІС;
- описання баз даних;
- перелік вхідних та вихідних даних і документів.

Стадія технічного проектування завершується підготовкою та оформленням документації на постачання та комплектування ІС, визначенням технічних вимог і складанням ТЗ на розроблення системи.

Стадія «Робоча документація» передбачає створення як програмного продукту, так і повного комплекту супроводжувальної документації. Така документація має надавати всі відомості, що забезпечують виконання робіт на стадіях введення ІС до експлуатації, самої експлуатації ІС, а також відомості щодо підтримки рівня якості системи (дотримання експлуатаційних характеристик).

Післяпроектна стадія має за мету реалізацію заходів щодо впровадження системи, підготовку приміщень і технічних засобів, навчання персоналу. Також проводиться експлуатація системи з вирішенням низки завдань, аналізуються результати випробувань, реалізуються заходи щодо супроводу й доопрацювання тощо.

Стадія 7. Введення до експлуатації

Складається із таких пунктів:

– підготовка об’єкта автоматизації до введення ІС у дію;

– підготовка персоналу, за необхідністю – його перекваліфікація;

– комплектація ІС необхідними програмними й технічними засобами, інформаційним забезпеченням тощо;

– проведення будівельно-монтажних робіт;

– проведення пусконалагоджувальних робіт;

– проведення попередніх випробувань;

– проведення дослідної експлуатації;

– здійснення приймальних випробувань.

Головними різновидами випробувань для ІС є такі:

– попередні випробування;

– дослідна експлуатація;

– приймальні випробування, вони можуть бути розширені додатковими випробуваннями ІС і її складових частин.

Впродовж попередніх випробувань (за відповідною програмою та методикою), проводять випробування системи на її працездатність і відповідність ТЗ, усувають недоліки та вносять зміни до проектної та супровідної документації. Далі проводять дослідну експлуатацію системи, аналізують її результати, здійснюють доробку програмного забезпечення і додаткове налаштування технічних засобів.

На етапі приймальних випробувань увагу концентрують на відповідності ТЗ, аналізують результати комплексних випробувань системи, усувають недоліки, які були виявлені під час цієї роботи. За результатами всіх випробувань оформляють відповідні акти щодо: приймання ІС до дослідної експлуатації, її завершення та приймання системи до постійної експлуатації.

8. Супроводження ІС

Ця стадія передбачає проведення таких дій:

– виконання робіт згідно до гарантійних зобов’язань;

– післягарантійне обслуговування системи.

Головними процесами цієї стадії є здійснення робіт щодо усунення недоліків, виявлених підчас експлуатації системи протягом гарантійного терміну, аналіз роботи системи за реальних умов, виявлення відхилень з'ясування причини їхнього виникнення, усунення причин відхилень і недоліків, забезпечення стабільної експлуатації та характеристик системи.

4.3.4 Типове проектування інформаційних систем

Згідно із самою назвою, методи типового проектування під час розроблення нової системи зорієнтовані на максимальне використання прототипів (типових рішень або ТПР).

Типове проектне рішення розробляється так, щоб бути придатним до багаторазового використання.

Застосування методів типового проектування має свої особливості. Головною умовою для використання таких методів є можливість декомпозиції системи, що проектується, на окремі складові (підсистеми, програмні модулі, комплекси виконуваних завдань тощо), для реалізації яких можна знайти типові проектні рішення, що можуть бути адаптовані до потреб певного підприємства.

Крім власне функціональних елементів (програмних, апаратних тощо), типове рішення має включати необхідний комплекс документації з детальним описом ТПР (зокрема процедур налаштування).

За рівнем декомпозиції системи можна виділити такі класи ТПР:

– елементні ТПР – рішення, зроблені для окремого елемента (задачі, різновиду забезпечення тощо);

– підсистемні ТПР – рішення, що розраховані на окремі підсистеми;

– об'єктні ТПР – рішення, що розраховані на певний набір підсистем ІС.

Типові рішення кожного із класів мають як певні переваги, так і недоліки. Розглянемо найбільш типові з них.

До переваг елементних ТПР можна віднести реалізацію модульного підходу до проектування ІС. З іншого боку, це призводить до великих витрат на доопрацювання ТПР певних елементів, а також до витрат на об'єднання різних елементів унаслідок їхньої поганої сумісності.

Підсистемні ТПР також дають змогу реалізувати модульний підхід до проектування ІС. Крім того, вони дають змогу здійснювати параметричне налаштування компонентів на об’єкти різних рівнів управління. Взаємопов’язані компоненти та високий ступінь інтеграції елементів ІС призводять до мінімізації витрат на проектування і на програмування. Однак у разі декількох виробників програмного забезпечення з’являються проблеми в об’єднанні різних функціональних підсистем. Окрім цього, з погляду безперервного реінжинірингу процесів адаптивність ТПР є недостатньою.

Об’єктні ТПР мають такі переваги, як:

– масштабованість (можливі конфігурації інформаційної системи для різної кількості робочих місць);

– методологічна єдність, тобто загальна методологічна база для всіх компонентів ІС;

– сумісність усіх компонент ІС;

– відкритість архітектури інформаційної системи; це дає змогу розгортати ТПР на платформах різного типу;

– розвинуті можливості щодо конфігурування, здатність використовувати необхідні підмножини компонентів системи.

Щодо недоліків об’єктних ТПР, то серед них можна виділити проблеми реалізації типового проекту на оригінальному об’єкті управління. За певних обставин це призводить до необхідності зміни організаційної структури об’єкта автоматизації.

Під час реалізації типового проектування застосовуються два головні підходи, а саме: **параметрично-орієнтоване** і **модельно-орієнтоване** проектування.

Етапами параметрично-орієнтованого проектування є:

– постановка завдань та оцінка можливості використання для їхнього вирішення пакетів прикладних програм (ППП); для оцінки відповідності ППП поставленим завданням використовують спеціальну систему критеріїв;

– аналіз доступних ППП (з огляду на критерії відповідності);

– вибір і придбання необхідних ППП;

– налаштування параметрів придбаних ППП, адаптація та допрограмування функцій.

Серед критеріїв оцінки ППП виділяють [11] такі групи:

– загальне призначення та функціональні можливості пакета;
– відмінні ознаки й властивості пакету;
– вимоги до технічних і програмних засобів;
– документування пакету;
– фінансові фактори, доступність ППП та економічна доцільність його використання;
– особливості інсталяції пакету;
– особливості експлуатації пакету;
– допомога постачальника щодо впровадження та підтримки пакету;
– оцінка якості пакету та досвід його використання;
– перспективи подальшого розвитку та оновлення функцій пакету.

Відзначимо, що кожна із вказаних груп критеріїв у подальшому може бути деталізована до рівня сукупності приватних показників. Це дає додаткову інформацію для кожного аспекту аналізу обраного ППП. Значення критеріїв визначаються з використанням методів експертного оцінювання.

На практиці часто використовують ще один підхід до реалізації типового проектування, – так зване модельно-орієнтоване проектування. Сутність цього підходу полягає у прискореній адаптації наявних характеристик типової ІС (з огляду на модель об’єкта автоматизації) з використаннями спеціального програмного інструментарію.

За таким підходом технологія проектування повинна мати кошти як для роботи з моделлю певного підприємства, так і з моделлю типової ІС.

У репозиторії типової інформаційної системи зберігається модель об’єкта автоматизації, яка є основою для конфігурації програмного забезпечення. Крім того, у репозиторії міститься базова модель ІС і типова (або референтна) моделі її певних класів. Базова модель ІС описує бізнес-процеси, організаційну структуру, бізнес-об’єкти, бізнес-функції, для підтримки яких призначені програмні модулі типової ІС. Типові моделі мають за мету описання конфігурації ІС для тих чи інших галузей й типів виробництва.

Модель певного підприємства може бути побудована або внаслідок вибору фрагментів типової моделі з огляду на особливості об’єкта

автоматизації (на кшталт BAAN Enterprise Modeler), або з використанням автоматизованої адаптації цих модулів з огляду на думки експертів (SAP Business Engineering Workbench). Модель підприємства, за якою здійснюється автоматичне конфігурування та налаштування ІС, зберігається в репозиторії. За потреби ця модель може бути відкоригована та адаптована до певних вимог.

Впровадження типової ІС починається з аналізу результатів передпроектного обстеження підприємства, що мають бути оформлені у вигляді вимог до певної ІС. Для оцінки таких вимог може бути використана методика ППП. Результатом та метою наступного етапу є формування попередньої моделі ІС, яка повинна повною мірою відображати особливості реалізації ІС для певного об'єкта. Попередня модель – це основа для вибору типової моделі системи. Також така модель потрібна, щоб визначити перелік компонентів, для реалізації яких будуть необхідні інші програмні засоби або інструментальні засоби, що є у складі типової ІС. Загалом, під час реалізації типового проекту має місце виконання таких операцій [11]:

- встановлення глобальних параметрів системи;
- визначення структури об'єкта автоматизації;
- визначення структури головних даних;
- завдання переліку реалізованих функцій і процесів;
- описання інтерфейсів;
- описання звітів;
- налаштування авторизації доступу;
- налаштування системи архівування.

Завдяки чисельним перевагам, типове проектування час широко представлено й активно використовується в сучасних засобах.

4.4 Питання для самоконтролю

1. Що регламентують стандарти, керівні документи, методики, положення, інструкції та шаблони проектування?

2. У чому полягає особливість стандартів ЄСПД?

3. Що таке ISO/IES?

4. Назвіть головні різновиди класифікації стандартів.

5. Які є різновиди стандартів за об’єктом стандартизації?

6. Які є різновиди стандартів за предметом стандартизації?

7. Для чого потрібні корпоративні стандарти? Вкажіть на їх відмінність.

8. Вкажіть різновиди корпоративних стандартів.

9. Що має регламентувати стандарт проектування?

10. Що має встановлювати стандарт оформлення проектної документації? Наведіть приклади.

11. Що має встановлювати стандарт інтерфейсу користувача? Наведіть приклади.

12. Охарактеризуйте головні положення стандарту ISO/IEC 12207.

13. Вкажіть головні процеси життєвого циклу за стандартом ISO/IEC 12207.

14. Перелічіть та охарактеризуйте головні процеси за стандартом ISO/IEC 12207.

15. Перелічіть та охарактеризуйте допоміжні процеси за стандартом ISO/IEC 12207.

16. Перелічіть та охарактеризуйте організаційні процеси за стандартом ISO/IEC 12207.

17. Перелічіть низку технологічних документів згідно зі стандартом ISO/IEC 12207.

18. Охарактеризуйте договірні процеси у контексті стандарту ISO/IEC 12207.

19. Охарактеризуйте проектні процеси у контексті стандарту ISO/IEC 12207.

20. Охарактеризуйте технічні процеси у контексті стандарту ISO/IEC 12207.

21. Охарактеризуйте спеціальні процеси у контексті стандарту ISO/IEC 12207.

22. Назвіть особливості та головні відмінності канонічного підходу до проектування інформаційних систем.

23. Вкажіть ключові стандарти, на яких базується канонічний підхід до проектування інформаційних систем.

24. Охарактеризуйте стадії та головні етапи канонічного підходу до проектування інформаційних систем.

25. Мета та головні завдання, що має вирішити складання технічного завдання на проектування системи.

26. Охарактеризуйте склад розділів технічного завдання на проектування інформаційної системи.

27. Які роботи має включати ескізний проект?

28. Які етапи має включати технічний проект?

29. Які діє передбачає етап «Робоча документація»?

30. Які діє передбачає етап «Введення до експлуатації»? Охарактеризуйте їх та наведіть приклади.

31. Які діє передбачає етап «Супровід»? Охарактеризуйте їх та наведіть приклади.

32. У чому полягає типове проектування інформаційних систем? Охарактеризуйте його особливості, переваги, сферу використання.

33. Вкажіть головні підходи, що використовуються при типового проектування інформаційних систем.

34. Охарактеризуйте головні операції, що присутні під час реалізації типового проектування інформаційної системи.

ВИСНОВОК

У першій частині навчального посібника «Технології створення програмних продуктів та інформаційних систем» розглянуто головні теоретичні положення щодо розробки ІС на всіх стаціях їхнього життєвого циклу, починаючи від обстеження об'єкта, закінчуючи розробкою програмних компонентів та впровадженням системи до реальної експлуатації.

Докладно розглянуто склад та зміст технологічних операцій щодо створення ІС на різних рівнях проектування та реалізації, головні підходи до формалізації процесу проектування, методи управління проектуванням ІС тощо.

Матеріал посібника організовано у чотири розділи:

– головні поняття дисципліни. архітектура інформаційних систем;
– життєвий цикл інформаційної системи;
– методологія і технологія розробки інформаційних систем;
– організація проектування та нормативно-методична підтримка життєвого циклу інформаційних систем.

У першому розділі посібника розглянуто головні поняття щодо інформаційних систем, подано їхню класифікацію, наведено відомості щодо функціональних складових інформаційної систем та їхньої архітектури**.**

Другий розділ присвячено питанням життєвого циклу ІС та головних процесів, що там відбуваються. У цьому розділі розглянуто найпоширеніші моделі життєвого циклу, особливості їхнього практичного використання, проаналізовано взаємозв'язок між процесами життєвого циклу тощо.

У третьому розділі розглянуто головні методології проектування ІС, складові технології проектування ІС, описано головні принципи та засади структурного та об'єктно-орієнтованого підходу до проектування ІС, наведено перелік та внутрішній зміст головних фаз проектування ІС. Також у цьому розділі описано такі методології проектування інформаційних систем, як RAD, RUP, методика CDM від фірми Oracle.

Четвертий розділ присвячено питанням стандартизації процесів розробки та супроводження ІС. У цьому разі описані різновиди та класифікація стандартів для створення ІС, головні міжнародні та вітчизняні стандарти проектування ІС, стандарти на процеси життєвого циклу, корпоративні стандарти проектування тощо.

Матеріал посібника рекомендовано студентам для вивчення дисципліни «Технології створення програмних продуктів та інформаційних систем» спеціальності 122 – Комп'ютерні науки та інформаційні технології, а також викладачам, що проводять відповідну підготовку студентів на рівні бакалавра.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Инюшкина О. Г. Проектирование информационных систем (на примере методов структурного системного анализа) : учеб. пособие / О. Г. Инюшкина. – Екатеринбург : «Форт-Диалог Исеть», 2014. – 240 с.

2. Инюшкина О. Г. Исследование систем управления при проектировании информационных систем : учеб. пособие / О. Г. Инюшкина, В. М. Кормышев. – Екатеринбург : «Форт-Диалог Исеть», 2013. – 370 с.

3. Гольдштейн С. Л. Практика использования информационных технологий и систем (на примерах управления организацией) : учеб. пособие / С. Л. Гольдштейн, О. Г. Инюшкина. – Екатеринбург : УрФУ, 2010. – 185 с.

4. Рудаков А. В. Технология разработки программных продуктов : учебник для студ. сред. проф. образования / А. В. Рудаков. — 7-е изд., стер. — М. : Издательский центр «Академия», 2012. – 208 с.

5. Черемных С. В. Структурный анализ систем : IDEF-технологии / С. В. Черемных, И. О. Семенов, В. С. Ручки. – М. : Финансы и статистика, 2003. – 208 с.

6. Новоженов Ю. В. Объектно-ориентированные технологии разработки сложных программных систем / Ю. В. Новоженов. – М. : «Лори», 1996. – 112 с.

7. Калянов Г. Н. CASE. Структурный системный анализ (автоматизация и применение) / Г. Н. Калянов. – М. : «Лори», 1996. – 236 с.

8. Николайчук Я. М. Проектування спеціалізованих комп’ютерних систем : навч. посібник / Я. М. Николайчук, Н. Я. Возна, І. Р. Пітух. – Тернопіль : ТзОВ «Терно-граф», 2010. – 392 с.

9. Шаховська Н. Б. Проектування інформаційних систем : навч. посібник / Н. Б. Шаховська, В. В. Литвин. – Львів : Магнолія-2006, 2011. – 380 с.

10. Разработка программных проектов на основе Rational Unified Process (RUP) / Г. Поллис, Л. Огастин, К. Лоу, Дж. Мадхар. – Бином-Пресс, 2011. – 256 с.

11. Грекул В. И. Проектирование информационных систем : учеб. пособие / В. И. Грекул, Г. Н. Денищенко, Н. Л. Коровкина. – М. : Интернет-Университет информационных технологий (ИНТУИТ.РУ) : БИНОМ. Лаборатория знаний, 2010. – 299 с.

Навчальне видання

КАРПЕНКО Микола Юрійович,
МАНАКОВА Наталія Олегівна,
ГАВРИЛЕНКО Ірина Олександрівна

ТЕХНОЛОГІЇ СТВОРЕННЯ ПРОГРАМНИХ ПРОДУКТІВ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

НАВЧАЛЬНИЙ ПОСІБНИК

Відповідальний за випуск *О. Б. Костенко*

Редактор В. І. Шалда

Комп'ютерний набір *М. Ю. Карпенко*

Комп'ютерне верстання *І. В. Волосожарової*

Дизайн обкладинки *Т. А. Лазуренко*

Підп. до друку 23.12.2016 Формат 60×84/16
Друк на ризографі Ум. друк. арк. 4,0
Зам. № 9958 Тираж 50 пр.

Видавець і виготовлювач:
Харківський національний університет
міського господарства імені О. М. Бекетова,
вул. Маршала Бажанова, 17, Харків, 61002
Електронна адреса: rectorat@kname.edu.ua
Свідоцтво суб'єкта видавничої справи:
ДК № 5328 від 11.04.2017