Методичні матеріали

за курсом

«МОДЕЛЮВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ»

ЗМІСТ

Перелік умовних позначень

Інформаційна система – ІС

Інформаційні технології – ІТ

Життєвий цикл – ЖЦ

Пакет прикладних програм – ППП

Програмне забезпечення – ПЗ

Реінжиніринг бізнес–процесів – РБП

Система автоматизованого проектування – САПР

Система управління базою даних – СУБД

Технічне завдання – ТЗ

Технологія проектування – ТП

Технологічний процес – ТПр

Технологія створення – ТС

Типове проектне рішення – ТПР

Управління проектом – УП

Capability Maturity Model – CMM

International Electrotechnical Commission – IEC

Integration Definition Metodology – IDEF

International Organization of Standardization – ISO

Rapid Application Development – RAD

Rational Unified Process – RUP

Software Engineering Institute – SEI

Structured Analysis and Design Technique – SADT

ВСТУП

Інформація в сучасному світі перетворилася в один із найбільш важливих ресурсів, а інформаційні системи (ІС) стали необхідним інструментом практично в усіх сферах діяльності. Різноманітність завдань, що вирішуються за допомогою ІС, призвела до появи множини різнотипних систем, які відрізняються принципами побудови і закладеними в них правилами обробки інформації.

Сучасні компанії та організації функціонують в умовах великого обсягу постійно змінюваної інформації, яку необхідно оперативно аналізувати та на її основі приймати правильні рішення. Бурхливо розвивається обчислювальна техніка та інформаційні технології (ІТ). Важко знайти компанію, яка не займається розвитком ІТ. Нині успішність і прибутковість компанії залежать, в тому числі, і від рівня розвитку ІТ, швидкості та якості обробки інформації, обґрунтованості та виваженості прийнятих рішень.

Потрібна постійна серйозна робота не тільки ІТ-фахівців, а й топ-менеджерів щодо синхронізації всіх зусиль зі стратегічного розвитку компанії та її інформаційних систем. Великою помилкою є позиція керівників, які запровадивши одного разу інформаційну систему (ІС), перестають нею займатися. Тому процес проектування ІС стає обов'язковим. Якщо цей процес не вперше здійснюється компанією, то термін «проектування» прирівнюється до поняття «розвиток ІС». Цим пояснюється бурхливий розвиток технологій проектування ІС в останні роки. Перш за все, створення CASE-технологій, які набагато скорочують час проектування ІС, дозволяють організувати одночасну колективну роботу, оперативно вносити зміни і швидко реагувати на зміни обставин. Компанії отримують конкурентні переваги, якщо рівень розвитку ІС відповідає рівню розвитку організації-підприємства: поліпшується її

інвестиційна привабливість, основні бізнес-процеси стають прозорими і зрозумілими для контролю та управління, виключаються помилки, брак і пов'язані з ним втрати часу, коштів (в кінцевому рахунку збільшується прибуток).

Актуальність і важливість дисципліни «Моделювання інформаційних систем» визначається необхідністю вивчення теоретичних положень, пов'язаних із нормативно-технічною документацією на розробку і проектування ІС, управління життєвим циклом ІС, архітектурою ІС, впровадженням і супроводом ІС, а також отримання практичних навичок розробки основних проектних документів, моделювання та аналізу бізнес-процесів, застосування сучасних CASE-засобів.

Навчальний курс знайомить студентів із сучасними підходами до проектування ІС. Науковою основою курсу є методології системного аналізу і моделювання, структурний та об'єктно-орієнтований підходи до проектування програмного забезпечення (ПЗ).

1. ОСНОВНІ ПОНЯТТЯ ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

1. Основні визначення. Системний підхід до проектування програмного забезпечення

1.1. Системний підхід до проектування програмного забезпечення (ПЗ). Методологічну основу проектування ПЗ становить системний підхід.

Під поняттям «*система*» розуміють сукупність «взаємодіючих компонентів і взаємозв'язків між ними; вона має властивості, відсутні у окремих її елементів». [3 с. 12]. Весь світ можна розглядати як складну взаємопов'язану сукупність природних і штучних систем (наприклад, планети у складі Сонячної системи, космічний корабель, системи молекулярних взаємодій в живих організмах).

Системний підхід – це «методологія дослідження об'єктів будь–якої природи як систем, орієнтована на: 1) розкриття цілісності об'єкта і механізмів, які її забезпечують; 2) виявлення різноманітних типів зв'язків об'єкта; 3) зведення цих зв'язків у єдину картину». [3 с. 12–13]. Системний підхід реалізує подання складного об'єкта у вигляді ієрархічної системи взаємозв'язаних моделей, що дозволяє фіксувати цілісні властивості об'єкта, його структуру і динаміку. Зменшення складності реалізації системи відбуватиметься за рахунок розбиття складних завдань на простіші. Це може привести до появи множини простих завдань, на підставі якої можна реалізувати будь–яку (навіть дуже складну) систему.

Існують «два принципи, які дозволяють оцінити взаємний вплив компонентів системи один на одного: слабка зв'язність (low coupling); сильна зв'язність (high cohesion)». [5 с. 69 –70].

1. Принцип *Слабкої зв'язності* сприяє розподілу функцій між компонентами системи таким чином, щоб міра зв'язності між ними залишалася низькою.

Міра зв'язаності – це міра взаємозалежності підсистем. Цей принцип пов'язаний з одним із основних принципів системного підходу, який вимагає мінімізації інформаційних потоків між підсистемами. Підсистема з низькою мірою зв'язаності (або із слабким зв'язуванням) має такі властивості: 1) малу кількість залежностей між підсистемами; 2) слабку залежність однієї підсистеми від змін в іншій; 3) високу ступінь повторного використання підсистем.

2. Принцип *Сильної зв'язності* задає властивість сильного зв’язку усередині підсистеми, в результаті чого формуються керовані підсистеми.

Зчеплення (функціональне зчеплення) – це міра зв'язаності та сфокусованості функцій підсистеми. Підсистема має високу міру зчеплення, якщо її функції тісно зв'язані між собою, і вона не виконує роботи великих об'ємів.

Підсистема із низькою мірою зчеплення виконує множину різних функцій, не пов'язаних між собою. Такі підсистеми створювати небажано, оскільки вони призводять до виникнення: 1) труднощів у розумінні, під час підтримки, 2) ненадійності у функціонуванні, постійної схильності до змін. Підсистеми з низькою мірою зчеплення не мають чіткого функціонального призначення і виконують різнопланові функції, які можна розподілити між іншими підсистемами.

Зв'язаність і зчеплення є загальносистемними характеристиками, які можна застосовувати при проектуванні будь–яких систем (при цьому *зв'язність* є характеристикою системи в цілому, а *зчеплення* характеризує окремо взяту підсистему).

Як система ПЗ є підсистемою деякої інформаційної системи (ІС). *Інформаційною системою* є сукупність 1) «функціональних та інформаційних процесів конкретної предметної області; 2) засобів і методів збору, зберігання, аналізу, обробки і передачі інформації, що залежать від специфіки галузі застосування; 3) методів управління процесами вирішення функціональних

завдань, а також інформаційними, матеріальними та грошовими потоками в предметній області». [3 с. 13]. *Інформаційні системи можуть відрізнятись* за:

1) видами інформаційних процесів конкретної предметної області;

2) засобами і методами збору, зберігання, аналізу, обробки та передачі інформації (що залежать від специфіки галузі застосування);

3) методами управління процесами, які реалізуються в предметній області та стосуються інформації, документопотоків, матеріалів і фінансів.

Можна виділити такі *види забезпечення ІС*: інформаційне; математичне; лінгвістичне; програмне; технічне, організаційне; нормативно–правове; забезпечення безпеки даних та інформаціїї. Розкриємо їхню сутність:

1. *Інформаційне забезпечення* – це сукупність форм документів, нормативної бази та реалізованих рішень щодо обсягу, розміщення та форм організації інформації, яка циркулює у конкретній ІС.

2. *Математичне забезпечення* – це сукупність математичних моделей та алгоритмів для вирішення питань обробки інформації із застосуванням вибраної ІТ, а також комплекс засобів і методів, які дають змогу будувати математичні моделі для конкретних задач. Розрізняють загальне математичне забезпечення (для організації обчислювального процесу на заданому комп’ютері) і спеціальне математичне забезпечення (для вирішення конкретних завдань).

3. *Лінгвістичне забезпечення* – сукупність мовних засобів проектування, моделювання, програмування та взаємодії користувачів із системою (в тому числі й мови введення–виведення). До мовних засобів відносяться безпосередньо мови та їх термінологія.

4. *Програмне забезпечення* – це сукупність методів, правил, описів, інструкцій, математичних моделей та алгоритмів вирішення завдань (в тому числі й управлінських), завдань обробки інформації та прийняття відповідних рішень, а також пов'язана з ними технічна документація, яка дозволяє використовувати комп’ютерні засоби для вирішення конкретних завдань і

реалізована на мові, зрозумілій комп'ютеру. Програмне забезпечення пов'язано з інформаційним забезпеченням, технологією обробки даних, комплексом технічних засобів, організаційним забезпеченням; воно розробляється на основі забезпечуючих його підсистем.

Програмне забезпечення – це набір комп'ютерних програм, процедур, пов'язаної з ними документації та даних. Воно складається з операційної системи, мов програмування та різних прикладних програм (прикладного ПЗ):

1) *операційна система* – це комплекс спеціальних програмних засобів, призначених для управління завантаженням, запуском і виконанням прикладних програм, введенням–виведенням даних, а також для планування та управління обчислювальними ресурсами комп'ютера;

2) *прикладне ПЗ* призначено для підтримки реалізації конкретних завдань предметної області.

5. *Технічне забезпечення* – це комплекс взаємопов'язаних технічних засобів, до якого входять засоби обчислювальної техніки, обладнання для організації локальних мереж і підключення до глобальних мереж, пристрої реєстрації, накопичення та відображення інформації, призначені для автоматизованого збору, накопичення, обробки, передачі, обміну та відображення інформації. До *основних компонентів технічного забезпечення* ІС відносяться: 1) засоби обчислювальної техніки; 2) периферійне обладнання (засоби накопичення та зберігання даних; засоби виведення результативної інформації); 3) засоби автоматичного зчитування даних; торговельне обладнання (комп'ютерні касові апарати, сканери); 4) засоби управління технологічними та виробничими процесами; 5) комунікаційне обладнання; 6) обладнання передачі та обміну даними, підтримки функціонування мереж.

6. *Організаційне забезпечення* – це комплекс адміністративно–технічних заходів (посадові інструкції, методики, схеми, регламенти), які регламентують взаємодію працівників між собою, з одного боку, і технічними засобами – з

іншого. Організаційне забезпечення зумовлює утворення інформаційних потоків в середині об'єкта інформатизації.

7. *Нормативно–правове забезпечення* – це сукупність правових норм, які визначають процедуру створення, юридичний статус і функціонування ІС, вони регламентують порядок одержання, перетворення і використання інформації. До складу правового забезпечення входять закони, укази, постанови державних органів влади, накази, інструкції та інші нормативні документи міністерств, відомств, організацій, місцевих органів влади. У правовому забезпеченні можна виділити загальну частину, яка регулює функціонування будь–якої ІС, і локальну частину, яка регулює функціонування конкретної системи.

8. *Забезпечення безпеки інформації та даних на рівні ІС* – це стан захищеності ІС, при якому забезпечуються конфіденційність, доступність, цілісність, підзвітність і правдивість її ресурсів.

Орієнтуючись на різні міжнародні стандарти, *ІС* можна визначити як сукупність таких *складових частин*: 1) «система баз даних: база даних (БД) разом із системою управління базами даних (СУБД); 2) прикладне ПЗ; 3) персонал; 4) організаційно–методичне (нормативне) забезпечення; 5) технічні засоби». [3 с. 13]. Така ІС функціонує: 1) на конкретному рівні світового господарства, в муніципальних, державних, недержавних та міжнародних організаціях різного призначення; 2) в органах управління, міністерствах, відомствах і підпорядкованих їм організаціях; 3) в економічних, банківських, податкових установах; 4) на підприємствах різної організаційно–правової форми; 5) в різних галузях господарства країни або регіону.

За визначенням Інституту управління проектами (Project Management Institute, PMI), *проект* – це тимчасовий захід, який здійснюють для створення унікального продукту (у тому числі й ІС) або послуги.

Під *проектуванням* розуміють уніфікований підхід, за допомогою якого шукають шляхи вирішення конкретної проблеми, забезпечуючи виконання

поставленого завдання. В контексті інженерного проектування можна визначити *мету проектування* як створення системи, яка:

1) задовольняє заданим функціональним специфікаціям;

2) узгоджена із обмеженнями, які накладаються обладнанням;

3) задовольняє вимогам до експлуатаційних якостей і ресурсів споживання;

4) задовольняє критеріям дизайну продукту;

5) задовольняє вимогам до процесу розробки (наприклад, тривалість і вартість), до залучення додаткових інструментальних засобів.

Проект є остаточним продуктом процесу проектування, який враховує усі суперечливі вимоги. Продуктами проектування є моделі, які дозволяють зрозуміти структуру майбутньої системи, збалансувати вимоги до неї та намітити схему її реалізації.

Під **проектом ПЗ** будемо розуміти сукупність специфікацій ПЗ (які включають моделі та проектну документацію), що забезпечують створення ПЗ в конкретному програмно–технічному середовищі.

Проектування ПЗ є процесом створення специфікацій ПЗ на основі вихідних вимог до нього. Проектування ПЗ призводить до послідовного уточнення його специфікацій на різних стадіях процесу створення ПЗ.

1.2. Основні особливості проектів сучасних систем ПЗ

Можна виділити три покоління процесів розробки ПЗ і визначити їх таким чином:

1. *Традиційний процес*: 60–70–і рр. XX ст., кустарне виробництво. Організації використовують кустарний інструментарій, куртарниі процеси і практично всі компоненти для замовника пишуться на примітивних мовах. Результат виконання проекту був легко передбачуваний в тому сенсі, що він практично ніколи не вкладався в заздалегідь задані вартість, терміни та якість.

2. *Перехідний*: 80–90–і рр. XX ст., програмна інженерія. Організації використовують відтворювані процеси і готові інструменти, а більшість

створюваних компонентів (>70%) пишеться на мовах високого рівня. Деякі компоненти (<30%) стають доступними в якості комерційного продукту, включаючи операційні системи, системи управлянні базами даних (СУБД), мережеве ПЗ і гріфічний інтерфейс користувача. У 1980–х рр. деякі організації починають досягати економії при великих масштабах, однак зі зростанням складності застосунків (особливо при переході до розподілених систем) існуючі мови, методи і технології виявилися недостатніми для того, щоб підтримувати необхідний рівень промислового створення систем.

3. *Сучасна практика*: починаючи з 2000 р., виробництво ПЗ. Цей період характеризує застосування керованих і вимірюваних процесів, інтегрованих середовищ автоматизації і здебільшого (70%) готових компонентів (можливо, всього лише 30% компонентів слід створювати на замовлення). Використовуючи переваги технології створення ПЗ та інтегрованих середовищ розробки, можна швидко створювати системи, побудовані з компонентів.

Технології, які дозволяють автоматизувати середовище розробки, зменшити розмір ПЗ і вдосконалити процес, не є незалежними. Для кожного нового періоду часу ключовим стає питання вдосконалення всіх технологій (наприклад, на основі нових технологій створення компонентів і підвищення ступеня автоматизації). Організації досягають економії при великих масштабах протягом технологічно успішних періодів в рамках великих проектів (системи систем), продуктів довготривалого використання і продуктових ліній, які включають в себе множину однотипних проектів.

Індустрія ПЗ розпочала зароджуватися в середині 1950–х років, проте майже до кінця 1960–х їй не приділялося серйозної уваги, оскільки її частка в комп'ютерному бізнесі була занадто мала. Виробництво ПЗ сьогодні є найбільшою галуззю світової економіки, в якій зайняті мільйони фахівців (програмісти, розробники ПЗ тощо).

Процес створення ПЗ є складною, трудомісткою і тривалою роботою, яка вимагає високої кваліфікації зайнятих в ній фахівців. Однак дотепер створення

таких систем нерідко виконується на інтуїтивному рівні із застосуванням неформалізованих методів, заснованих на мистецтві, практичному досвіді, експертних оцінках та експериментальних перевірках якості функціонування ПЗ. В процесі створення і функціонування ПЗ потреби користувачів постійно змінюються або уточнюються, що ускладнює розробку і супровід таких систем.

Щоб зрозуміти відмінності складного ПЗ від звичайної програми, розглянемо рис. 1.1.

Програма	Програмний комплекс (інтерфейси, системна інтеграція)
Програмний продукт (докумен–тування, тестування, супровід)	Комплексний програмний продукт

Рис. 1.1. Еволюція програмного продукту [2]

У його лівому верхньому кутку знаходиться *програма* – завершений продукт, придатний для запуску тільки її автором і в середовищі системи, де вона була розроблена. При переміщенні вниз через горизонтальну межу програма перетворюється у *програмний продукт* – програму, яку будь–яка людина може використовувати і, при наявності програмістської кваліфікації, супроводжувати, вносити в неї різні зміни. Таку програму можна використовувати в різних операційних системах і з різними даними; її потрібно ретельно протестувати, щоб бути впевненими в її надійності (для цього потрібно підготувати достатню кількість контрольних прикладів для перевірки діапазону допустимих значень вхідних даних, обробити ці приклади і зафіксувати результати). Розвиток програми в програмний продукт вимагає створення докладної документації, за допомогою якої кожен міг би її використовувати. Програмний продукт коштує дорожче, ніж налагоджена програма з такою ж функціональністю.

При перетині вертикальної межі програма стає *компонентом програмного комплексу* (у вигляді набору взаємодіючих програм, узгоджених за функціями і форматом даних), призначеного для вирішення великомасштабних завдань.

Щоб стати частиною цього програмного комплексу, вхідні–вихідні дані і повідомлення програми повинні задовольняти визначеним інтерфейсам. Програма–компонент комплексу повинна бути спроектована таким чином, щоб використовувати визначені ресурси (обсяг пам'яті, зовнішні пристрої, час процесора). Цю програму потрібно протестувати разом з іншими системними компонентами в усіх поєднаннях, які можуть зустрітися. Тестування може виявитися більшим за обсягом, оскільки кількість протестованих випадків зростає експоненційно. Воно займає багато часу, тому що при несподіваних взаємодіях компонентів, які налагоджуються, виявляються приховані помилки. Компонент програмного комплексу коштує дорожче, ніж автономна програма з тими ж функціями.

У правому нижньому кутку знаходиться *комплексний програмний продукт*. Від звичайної програми він відрізняється в усіх перерахованих вище відношеннях і коштує дорожче. Саме такий продукт у вигляді системи ПЗ є метою більшості програмних проектів.

У 1975 р. Фредерік Брукс, проаналізувавши свій досвід керівництва проектом розробки операційної системи OS/360, визначив *перелік основних властивостей* ПЗ: «складність, узгодженість, змінність і невидимість» [2].

1. **Складність**. Складність є причиною труднощів, які виникають в процесі спілкування між розробниками, і спричиняє виникнення помилок в продукті, перевищення вартості розробки, затягування процесу виконання графіків робіт. Складність структури ускладнює розвиток ПЗ і додавання нових функцій.

2. **Узгодженість.** Зазвичай нове ПЗ повинно узгоджуватися із вже існуючим (наприклад, із різними інтерфейсами).

3. **Змінність**. Програмне забезпечення постійно піддається змінам. При цьому мають місце такі два процеси:

1) як тільки виявляють користь програмного продукту, розпочинаються спроби застосування його на межі або за межами первісної області (вимога

щодо розширення функцій йде, зазвичай, від користувачів, які задоволені основним призначенням і знаходять для нього нове застосування);

2) вдалий програмний продукт живе довше за звичайний термін існування комп'ютера, для якого він спочатку був створений; розробляють нові комп'ютери, і програма повинна бути узгоджена з їх можливостями.

4. **Програмний продукт є невидимим**. Для матеріальних об'єктів геометричні абстракції є потужним інструментом. План будівлі допомагає архітектору і замовнику оцінити проект у просторі (стають очевидними протиріччя, можна помітити упущення). При спробі графічно зобразити структуру програми виявляється, що потрібно не один, а декілька неорієнтованих графів, накладених один на інший.

Серед **особливостей** сучасних великомасштабних проектів створення програмних систем / ПЗ зазвичай можна виділити такі [3 с. 20]:

1. *Характеристики об'єкта впровадження*:

1) структурна складність (багаторівнева ієрархічна структура організації) і територіальна розподіленість;

2) функціональна складність (багаторівнева ієрархія і велика кількість функцій, які виконуються організацією; складні взаємозв'язки між ними);

3) інформаційна складність (велика кількість джерел і споживачів інформації (міністерства і відомства, місцеві органи влади, організації–партнери), різноманітні форми та формати подання інформації, складна інформаційна модель об'єкта – велика кількість інформаційних сутностей і складні взаємозв'язки між ними), складна технологія проходження документів;

4) складна динаміка поведінки, обумовлена високою мінливістю зовнішнього середовища (зміни в законодавчих і нормативних актах, нестабільність економіки і політики) і внутрішнього середовища (структурні реорганізації, плинність кадрів).

2. *Технічні характеристики проектів*:

1) різна ступінь уніфікованості проектних рішень в рамках одного проекту;

2) висока технічна складність, обумовлена наявністю сукупності тісно взаємодіючих компонентів (підсистем), які мають свої локальні завдання і цілі функціонування (транзакційних застосунків, що пред'являють підвищені вимоги до надійності, безпеки і продуктивності, і застосунків аналітичної обробки, або систем підтримки прийняття рішень, що використовують нерегламентовані запити до даних великого обсягу);

3) відсутність аналогів, що обмежує можливість використання будь–яких типових проектних рішень і прикладних систем, висока частка ПЗ розробляється наново;

4) велика кількість і висока вартість успадкованих застосунків (існуючого прикладного ПЗ), які функціонують в різному середовищі (персональні комп'ютери, мінікомп'ютери, мейнфрейми), необхідність інтеграції успадкованих і розроблених нових застосунків;

5) велика кількість локальних об'єктів впровадження, територіально розподілене і неоднорідне середовище функціонування (СУБД, операційні системи, апаратні платформи);

6) велика кількість зовнішніх взаємодіючих систем організацій з різними форматами обміну інформацією (податкова служба, податкова поліція, Держстандарт, Держкомстат, Міністерство фінансів, МВС, місцева адміністрація).

3. *Організаційні характеристики проектів*:

1) різні форми організації та управління проектом (УП): централізовано керована розробка ПЗ, яке тиражується, експериментальні пілотні проекти, ініціативні розробки, проекти за участю як власних розробників, так і сторонніх компаній на контрактній основі;

2) велика кількість учасників проекту як з боку замовників (із різнорідними вимогами), так і з боку розробників (понад 100 осіб),

роз'єднаність і різнорідність окремих груп розробників за рівнем кваліфікації, традиціями, що склалися, досвіду використання конкретних інструментальних засобів;

3) значна тривалість життєвого циклу системи, в тому числі значна часова протяжність проекту, обумовлена масштабами організації–замовника, різним ступенем готовності окремих її підрозділів до впровадження ПЗ і нестабільністю фінансування проекту;

4) високі вимоги з боку замовника до рівня технологічної зрілості організацій–розробників (наявність сертифікації відповідно до міжнародних і вітчизняних стандартів).

1.3. Основні проблеми сучасних проектів ПЗ. Наприкінці 1960–х років у США було відмічено явище під назвою «криза ПЗ» («software crisis»), яке виражалося в тому, що «великі проекти стали виконуватися із відставанням від графіка або з перевищенням кошторису витрат, розроблений продукт не володів необхідними функціональними можливостями, продуктивність його була низькою, якість одержуваного ПЗ не влаштовувала споживачів» [3 с. 23].

Дослідження, які виконувалися провідними зарубіжними аналітиками, показували не дуже обнадійливі результати. Результати аналітичних досліджень, виконаних у 1995 р. компанією Standish Group (США), яка проаналізувала роботу 364 американських корпорацій і підсумки виконання понад 23 тис. проектів, пов'язаних з розробкою ПЗ, виглядали таким чином:

1) тільки 16,2% проектів завершилися в термін, не перевищили запланований бюджет і реалізували всі необхідні функції і можливості;

2) 52,7% проектів завершилися із запізненням, витрати перевищили запланований бюджет, необхідні функції не були реалізовані в повному обсязі;

3) 31,1% проектів були анульовані до завершення;

4) для двох останніх категорій проектів бюджет середнього проекту виявився перевищеним на 89%, а термін виконання – на 122%.

На думку розробників ПЗ, серед *причин можливих невдач* фігурували:

– «нечітке і неповне формулювання вимог до ПЗ;

– недостатнє залучення користувачів до роботи над проектом;

– відсутність необхідних ресурсів;

– незадовільне планування і відсутність грамотного управління проектом;

– часта зміна вимог і специфікацій;

– новизна і недосконалість використовуваної технології;

– недостатня підтримка з боку вищого керівництва;

– недостатньо висока кваліфікація розробників, відсутність необхідного досвіду» [3 с. 24].

Двома найбільш явними проблемами невдалих програмних проектів є 1) переробка програм, 2) виявлення непридатності проекту на його пізніх стадіях. Розробник проектує архітектуру на ранніх стадіях розробки ПЗ, але не має можливості відразу ж оцінити її якість. У нього відсутні основні принципи для доказу адекватності проекту. Тестування ПЗ поступово виявляє всі дефекти архітектури, але тільки на пізніх стадіях розробки, коли виправлення помилок стає дорогим і руйнівним для проекту.

Розробники проектують ПЗ. Кінцевий результат проектування, який має вигляд коду на мові високого рівня, є кресленням ПЗ. Компілятор і компонувальник механічно будують програмний продукт (двійковий виконуваний файл) з цього спроектованого коду.

1.4. Програмна інженерія. *Програмна інженерія* визначається як: 1) сукупність інженерних методів і засобів створення ПЗ, 2) дисципліна, яка вивчає застосування чіткого систематичного кількісного (тобто інженерного) підходу до розробки, експлуатації та супроводу ПЗ. Нині *програмна інженерія* є підобластю комп'ютерних наук, яка включає в себе різноманітні математичні, інженерні, економічні та управлінські аспекти.

В основі програмної інженерії лежить така *фундаментальна ідея*: проектування ПЗ є формальним процесом, який можна вивчати та удосконалювати. Освоєння і правильне застосування методів і засобів

створення ПЗ дозволяє підвищити його якість, забезпечити керованийність процесу проектування та збільшити термін життя ПЗ.

Спроби надмірної формалізації процесу проектування, а також запозичення ідей і методів з інших областей інженерної діяльності (будівництва, виробництва) призвели до серйозних проблем. Після марних очікувань підвищення продуктивності процесів створення ПЗ, покладених на нові методи і технології, фахівці в індустрії ПЗ прийшли до розуміння того, що ***фундаментальна проблема в цій галузі*** *– нездатність ефективного управління проектами створення* ПЗ. Неможливо досягти потрібних результатів від застосування навіть найдосконаліших технологій та інструментальних засобів, якщо їх застосовують безсистемно, розробники не володіють необхідною кваліфікацією для роботи з ними, і сам проект виконується та управляється хаотично, в режимі «гасіння пожежі».

Безсистемне застосування технологій створення ПЗ породжує розчарування у використовуваних методах і засобах (серед факторів, які впливають на ефективність створення ПЗ, використовуваним методам і засобам надається менше значення, ніж кваліфікації та досвіду розробників). Якщо для управління проектами та впровадження технологій не створюється необхідної інфраструктури і не забезпечується технічна підтримка, то на виконання проектів витрачається значно більше часу і ресурсів по відношенню до запланованих. Якщо в таких умовах окремі проекти завершуються успішно, то цей успіх досягається за рахунок героїчних зусиль фанатично налаштованого колективу розробників.

Успіх, який тримається виключно на здібностях окремих організаторів «витягувати» проекти з проривів, не дає гарантії стійкої продуктивності та якості при створенні ПЗ. Постійне підвищення якості створюваного ПЗ і зниження його вартості можна забезпечити тільки за умови досягнення організацією необхідної технологічної зрілості, створенні ефективної інфраструктури як в сфері розробки ПЗ, так і в управлінні проектами.

Відповідно до моделі CMM (Capability Maturity Model) Інституту програмної інженерії (Software Engineering Institute, SEI), в добре підготовленій (зрілій) організації персонал володіє технологією та інструментарієм оцінки якості процесів створення ПЗ на протязі всього життєвого циклу ПЗ і на рівні всієї організації. Процеси вибудовуються таким чином, щоб забезпечити реальні терміни створення ПЗ.

При вирішенні проблеми підвищення ефективності створення ПЗ зазвичай основна увага приділяється процесу розробки ПЗ, що викликано бажанням розробників і замовників економити кошти з самого початку проекту (особливо в умовах обмежених фінансових можливостей і високої конкуренції). У той же час більшість великомасштабних проектів створення ПЗ характеризується тривалим життєвим циклом (10 ÷ 15 років), в якому на стадію створення (розробки) припадають лише перші 3÷4 роки, а решту часу експлуатації створеної системи (стадія супроводження і розвитку) розподіляють, за оцінками Інституту програмної інженерії (Software Engineering Institute, SEI), приблизно порівну на такі стадії:

1) *початкову стадію супроводу*, пов'язану з усуненням помилок і доробками, які виникли через невраховані або новопосталі вимоги користувачів;

2) *стадію «зрілого» супроводу*, яка характеризується відносно повним задоволенням основних потреб користувачів, але в той же час пов'язана з накопиченням деяких проблем, а саме: в процесі розвитку і зміни ПЗ його первинна структура спотворюється та ускладнюється, що призводить до зростання складності модифікації ПЗ;

3) *стадію еволюції / заміни*, яка характеризується досягненням системою порогу свого супроводження в існуючому вигляді і неможливістю задовольнити нові вимоги без внесення принципових змін. Постійна зміна і збільшення функцій ПЗ призводить до того, що стає економічно вигідним припинити супровід і розробити нове ПЗ (або піддати систему *реінжинірингу*,

Додаток А), який полягає у її суттєвій переробці та/або перенесенні у нове середовище.

Під **супроводом** в загальному випадку розуміють внесення змін до експлуатованого програмного продукту з метою виправлення виявлених помилок (коригуючий супровід), підвищення продуктивності і поліпшення експлуатаційних характеристик системи (удосконалюючий супровід), а також адаптації до нового або зміненого середовища (адаптуючий супровід). При цьому більше 50% загального обсягу робіт по супроводу доводиться на удосконалюючий супровід. В наш час *супровід* перетворився в трудомісткий процес модифікації та розвитку різних версій великомасштабного ПЗ і його компонентів, які значно різняться функціями та якістю. В багатьох випадках для розвитку версій необхідна практично така ж кількість фахівців, яка розробила першу версію ПЗ.

Проектування методом «кодування і виправлення помилок» («code and fix») є однією з причин згаданого вище прагнення заощадити на стадії розробки, не витрачаючи час і кошти на впровадження технологічного процесу створення ПЗ. Сучасні технологічні процеси є «важкими» і мають такі особливості:

1) необхідність документувати кожну дію розробників;

2) наявність робочих продуктів (в першу чергу – документів), які створюються в бюрократичній атмосфері;

3) відсутність гнучкості;

4) детермінованість (довгострокове детальне планування і передбачува–ність всіх видів діяльності, а також розподіл людських ресурсів на тривалий термін часу).

Альтернативою «важкого» процесу розробки ПЗ є адаптивний (гнучкий) процес, заснований на принципах «швидкої розробки ПЗ».

1.5. Сучасні тенденції в програмній інженерії (принципи «швидкої розробки ПЗ»). На початку 2001 р. провідні фахівціві в області програмної

інженерії (Алістер Коберн, Мартін Фаулер, Джим Хайсмит, Кент Бек та ін.) сформували групу під назвою «Agile Alliance». Слово швидкий (agile) відображало їх підхід до розробки, котрий базується на їхньому багаторічному досвіді участі в різноманітних проектах. Цей підхід під назвою «Швидка розробка ПЗ» (Agile software development) базується на чотирьох ідеях, сформульованих ними в документі «Маніфест швидкої розробки ПЗ» і полягає в наступному: 1) індивідууми і взаємодія між ними цінуються вище процесів та інструментів; 2) працююче ПЗ цінується вище всеосяжної документації; 3) співпраця із замовниками цінується вище формальних договорів; 4) реагування на зміни цінується вище суворого дотримання плану.

Центральною ідеєю швидкої розробки ПЗ є орієнтація на людей і комунікацію.

Швидкість передбачає маневреність. Розповсюдження ПЗ в Інтернеті ще більше посилило конкуренцію між програмними продуктами: потрібно не тільки випускати програми на ринок і скорочувати кількість дефектів в них, а й постійно стежити за вимогами користувачів і ринку, що змінюються.

При такому підході технологія займає в процесі створення ПЗ конкретне місце, вона підвищує ефективність діяльності розробників при наявності будь–яких з таких умов: вона 1) дозволяє людям легше висловити свої думки (мова високого рівня дає можливість стисло висловлювати ідеї; деякі мови високого рівня дозволяють людині мислити в технологічному просторі, наближеному до проблемного простору, майже не відволікаючись на думки про обмеження реалізації); 2) виконує завдання, нездійсненні вручну (інструменти виміру та аналізу збирають дані, які інакше зібрати було б неможливо; програмісти говорять про них, як про основні інструменти); 3) автоматизує схильні до помилок дії: компілятори, електронні таблиці, засоби управління конфігурацією ПЗ є настільки важливими, що деякі програмісти навіть не згадують про них як про інструменти, а припускають їх присутність; 4) полегшує спілкування між

людьми: у середовищі розподіленої розробки ПЗ всі види інструментів обміну інформацією допомагають команді працювати.

2. Класифікація інформаційних систем

Інформаційні системи можна класифікувати за різними ознаками (наприклад, за тими, що визначають функціональні можливості та особливості побудови систем). Інформаційні системи розділяють на класи залежно від «обсягу вирішуваних завдань, використовуваних технічних засобів та організації функціонування» [8 с. 11].

1. **За типом даних, які зберігаються**, ІС класифікують на фактографічні та документальні. *Фактографічні системи* призначені для зберігання та обробки структурованих даних у вигляді чисел і текстів (над такими даними можна виконувати різні операції). У *документальних системах* інформація подана у вигляді документів, які складаються з найменувань, описів, рефератів і текстів. Пошук серед неструктурованих даних здійснюють із використанням семантичних ознак. Відібрані документи надаються користувачеві, а обробка даних в таких системах практично не проводиться.

2. **За ступенем автоматизації інформаційних процесів в системі управління фірмою,** ІС діляться на ручні, автоматичні та автоматизовані. *Ручні ІС* характеризуються відсутністю сучасних технічних засобів переробки інформації та виконанням всіх операцій людиною. В *автоматичних ІС* всі операції з переробки інформації виконуються без участі людини. *Автоматизовані ІС* припускають участь в процесі обробки інформації людини і технічних засобів, при цьому головна роль у виконанні рутинних операцій обробки даних відводиться комп'ютеру.

3. **Залежно від характеру обробки даних** ІС класифікують на інформаційно–пошукові та інформаційно–розв’язуючі. *Інформаційно–пошукові системи* виконують введення, систематизацію, зберігання, видачу інформації за запитом користувача без складних перетворень даних (наприклад, ІС

бібліотечного обслуговування, резервування і продажу квитків на транспорті, бронювання місць в готелях тощо). *Інформаційно–розв'язуючі системи* здійснюють, крім того, операції переробки інформації за певним алгоритмом. За характером використання вихідної інформації такі системи прийнято ділити на: 1) керуючі, 2) призначені для надання порад.

Результуюча інформація *керуючих ІС* безпосередньо трансформується в прийняті людиною рішення. Для цих систем характерні завдання розрахункового характеру та обробка великих обсягів даних (наприклад, ІС планування виробництва або замовлень, бухгалтерського обліку).

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

4. **Залежно від сфери застосування** розрізняють такі класи ІС.

Інформаційна система організаційного управління призначена для автоматизації функцій управлінського персоналу як промислових підприємств, так і непромислових об'єктів (готелів, банків, магазинів тощо). Основними функціями подібних систем є: оперативний контроль і регулювання, оперативний облік та аналіз, перспективне та оперативне планування, бухгалтерський облік, управління збутом, постачанням, інші економічні та організаційні завдання.

Інформаційна система управління технологічними процесами (ТПр) призначена для автоматизації функцій виробничого персоналу по контролю та управлінню виробничими операціями. У таких системах зазвичай передбачається наявність розвинених засобів вимірювання параметрів технологічних процесів (температури, тиску, хімічного складу тощо), процедур контролю допустимості значень параметрів і регулювання ТПр.

Інформаційна система автоматизованого проектування (САПР) призначена для автоматизації функцій інженерів–проектувальників,

конструкторів, архітекторів, дизайнерів при створенні нової техніки або технології. Основними функціями подібних систем є: інженерні розрахунки, створення графічної документації (креслень, схем, планів), створення проектної документації, моделювання проектованих об'єктів.

Інтегровані (корпоративні) ІС використовують для автоматизації всіх функцій фірми та охоплюють весь цикл робіт від планування діяльності до збуту продукції. Вони включають в себе ряд модулів (підсистем), які працюють в єдиному інформаційному просторі та виконують функції підтримки відповідних напрямів діяльності. Типові завдання, які вирішуються модулями корпоративної системи, наведені в табл. 1.1.

Аналіз сучасного стану ринку ІС показує стійку тенденцію зростання попиту на ІС організаційного управління. При цьому попит продовжує рости саме на інтегровані системи управління. Для багатьох підприємств автоматизація окремої функції, наприклад, бухгалтерського обліку або збуту готової продукції, вважається вже пройденим етапом.

5. **Залежно від рівня управління, на якому система використовується**, ІС класифікують на ІС: оперативного рівня, рівня фахівців, рівня менеджменту, стратегічного рівня.

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

Інформаційні системи рівня фахівців підтримують роботу з даними і знаннями, підвищують продуктивність роботи інженерів і проектувальників. Завданням подібних ІС є інтеграція нових відомостей в організації, допомога в обробці паперових документів.

Таблиця 1.1

Функціональне призначення модулів корпоративної ІС

Підсистема маркетин–гу	Виробничі підсистеми	Фінансові та облікові підсистеми	Підсистема кадрів (людських ресурсів)	Інші підсистеми, (наприклад, управління)
Дослідження ринку та прогнозуван–ня продажів	Планування обсягів робіт і розробка календарних планів	Управління портфелем замовлень	Аналіз і прогнозування потреби в трудових ресурсах	Контроль за діяльністю фірми
Управління продажами	Оперативний контроль і управління виробництвом	Управління кредитною політикою	Ведення архівів записів про персонал	Виявлення оперативних проблем
Рекомендації з виробництва нової продукції	Аналіз роботи обладнання	Розробка фінансового плану	Аналіз і планування підготовки кадрів	Аналіз управлінських і стратегічних ситуацій
Аналіз і встановлен–ня ціни	Участь у формуванні замовлень постачальниками	Фінансовий аналіз і прогнозування		Забезпечення процесу вироблення стратегічних рішень
Облік замовлень	Управління запасами	Контроль бюджету, бухгалтерський облік та розрахунок зарплати		

Інформаційні системи рівня менеджменту використовують працівники середньої управлінської ланки для моніторингу, контролю, прийняття рішень та адміністрування.

Основні функції цих ІС: 1) порівняння поточних показників із минулими; 2) складання періодичних звітів за певний час (а не видача звітів щодо поточних подій, як на оперативному рівні); 3) забезпечення доступу до архівної інформації тощо.

Стратегічна ІС забезпечує підтримку прийняття рішень по реалізації стратегічних перспективних цілей розвитку організації. *Інформаційні системи*

стратегічного рівня допомагають вищій ланці управлінців вирішувати неструктуровані завдання, здійснювати довгострокове планування. Основним завданням ІС є порівняння змін, що відбуваються в зовнішньому оточенні, до існуючого потенціалу фірми. Вони покликані створити загальне середовище комп'ютерної телекомунікаційної підтримки рішень при виникненні несподіваних ситуацій. Ці системи здатні у будь–який момент надати інформацію з багатьох джерел. Деякі стратегічні системи мають обмежені аналітичні можливості.

3. Процес створення інформаційних систем

Можна визначити такі *три покоління процесів розробки* ПЗ:

1. *Традиційний процес*: 1960–1970-і рр., кустарне виробництво. Організації використовують кустарний інструментарій, куртарні процеси і практично всі компоненти для замовника пишуться на примітивних мовах. Результат виконання проекту був легко передбачуваний в тому сенсі, що він практично ніколи не вкладався в заздалегідь задані вартість, терміни та якість.

2. *Перехідний*: 1980–1990–і рр., програмна інженерія. Організації використовують відтворювані процеси і готові інструменти, а більшість створюваних компонентів (>70%) пишеться на мовах високого рівня. Деякі компоненти (<30%) стають доступними в якості комерційного продукту, включаючи ОС, системи управління базами даних (СУБД), мережеве ПЗ і графічний інтерфейс користувача. У 80–і рр. XX ст. деякі організації починають досягати економії при великих масштабах, однак зі зростанням складності додатків (особливо при переході до розподілених систем) існуючі мови, методи і технології виявилися недостатніми, щоб підтримувати необхідний рівень промислового створення.

3. *Сучасна практика*: починаючи з 2000 р., виробництво ПЗ. Цей період характеризує застосування керованих і вимірюваних процесів, інтегрованих середовищ автоматизації і здебільшого (на 70%) готових компонентів

(можливо, лише 30% компонентів слід створювати на замовлення). Використовуючи переваги технології створення ПЗ та інтегрованих середовищ розробки, можна швидко створювати системи, побудовані з компонентів.

Технології, які дозволяють автоматизувати середовище розробки, зменшити розмір ПЗ і вдосконалити процес, не є незалежними. Для кожного нового періоду часу ключовим стає деяке вдосконалення всіх технологій: наприклад, переваги нового процесу не можуть бути успішно використаними без 1) нових технологій створення компонентів, 2) підвищення ступеня автоматизації. Організації досягають більшої економії при великих масштабах протягом технологічно успішних періодів – в рамках великих проектів (системи систем), продуктів довготривалого використання і продуктових ліній, які включають в себе множину однотипних проектів.

Проектування ІС охоплює такі *три основні області*:

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

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

3) облік конкретного середовища або технології, а саме: топології мережі, конфігурації апаратних засобів, використовуваної архітектури (файл–сервер або клієнт–сервер), паралельної та розподіленої обробки даних тощо.

Проектування ІС завжди розпочинається із визначення *мети проекту*, яку у загальному вигляді можна визначити як вирішення ряду взаємопов'язаних завдань, що включають в себе забезпечення запуску системи, її експлуатацію протягом певного часу. Серед цих завдань можна виділити такі: визначення

1) необхідної функціональності системи та рівня її адаптивності до постійно змінюваних умов функціонування;

2) необхідної пропускної спроможності системи;

3) необхідного часу реакції системи на запит;

4) безвідмовної роботи системи;

5) необхідного рівня безпеки;

6) простоти експлуатації та підтримки системи.

Відповідно до сучасної методології, **процес створення ІС** є процесом побудови і послідовного перетворення ряду узгоджених моделей на всіх етапах життєвого циклу (ЖЦ) ІС. На кожному етапі ЖЦ створюються специфічні для нього моделі (організації, вимог до ІС, проекту ІС, вимог до застосунків тощо). Моделі формуються робочими групами команди проекту, зберігаються і накопичуються в репозитарії проекту. Створення моделей, їх контроль, перетворення і надання в колективне користування здійснюється із використанням спеціальних програмних інструментів – CASE–засобів.

Етапи створення ІС. *Процес створення* ІС охоплює ряд етапів, обмежених деякими часовими рамками і закінчується випуском конкретного продукту (моделей, програмних продуктів, документації тощо). Зазвичай виділяють такі *етапи створення* ІС: формування вимог до системи, проектування, реалізація, тестування, введення в дію, експлуатація та супровід.

1. *Початковим етапом процесу створення* ІС є моделювання бізнес–процесів, які мають місце в організації/на підприємстві, реалізують її цілі та завдання. Модель організації, описана в термінах бізнес–процесів і бізнес–функцій, дозволяє сформулювати основні вимоги до ІС. Множина моделей опису вимог до ІС потім перетворюється в систему моделей, які описують концептуальний проект ІС. Формуються моделі архітектури ІС, вимог до ПЗ та інформаційного забезпечення (ІЗ). Потім формується архітектура ПЗ та ІЗ, виділяються корпоративні БД та окремі додатки, формуються моделі вимог до додатків і проводиться їх розробка, тестування та інтеграція.

Метою початкових етапів створення ІС, виконуваних на стадії аналізу діяльності організації, є формування вимог до ІС, які відображають цілі та завдання організації–замовника. Щоб специфікувати процес створення ІС, яка відповідає потребам організації, потрібно з'ясувати і чітко сформулювати сутність цих потреб. Для цього необхідно визначити вимоги замовників до ІС і відобразити їх на мові моделей вимог до розробки проекту ІС так, щоб

забезпечити відповідність цілям і задачам організації. Завдання формування вимог до ІС є одним із найвідповідальніших, важко формалізованих, найдорожчих і важких для виправлення в разі помилки.

Сучасні інструментальні засоби і програмні продукти дозволяють швидко створювати ІС відповідно готовим вимогам. Але найчастіше ці системи не задовольняють замовників, вимагають численних доробок, що призводить до різкого подорожчання вартості ІС. Основною причиною такого становища є неправильне, неточне або неповне визначення вимог до ІС на етапі аналізу.

2. На *етапі проектування* перш за все формуються моделі даних. Проектувальники в якості вихідної інформації отримують результати аналізу. Побудова логічної і фізичної моделей даних є основною частиною проектування бази даних. Отримана в процесі аналізу інформаційна модель спочатку перетвориться в логічну, а потім у фізичну модель даних.

Паралельно із проектуванням схеми бази даних виконується проектування процесів, щоб отримати специфікації (опис) всіх модулів ІС. Обидва ці процесу проектування тісно пов'язані, оскільки частина бізнес–логіки зазвичай реалізується в базі даних (обмеження, тригери, збережені процедури).

Головна мета проектування процесів полягає у відображенні функцій, отриманих на етапі аналізу, в модулі ІС. При проектуванні модулів визначають інтерфейси програм: розмічають меню, вигляд вікон, гарячі клавіші і пов'язані з ними виклики.

Кінцевими продуктами етапу проектування є: 1) схема бази даних (на підставі моделі, розробленої на етапі аналізу); 2) набір специфікацій модулів системи (вони будуються на базі моделей функцій).

На етапі проектування здійснюють також розробку *архітектури* ІС, яка включає в себе вибір платформи і операційної системи. У неоднорідній ІС можуть працювати кілька комп'ютерів на різних апаратних платформах і під управлінням різних операційних систем. Крім вибору платформи на етапі проектування визначаються такі *характеристики архітектури*:

1) чи буде це архітектура «файл–сервер» або «клієнт–сервер»;

2) чи буде це трирівнева архітектура з такими шарами: сервер, ПЗ проміжного шару (сервер додатків), клієнтське ПЗ;

3) чи буде база даних централізованою або розподіленою: якщо база даних буде розподіленою, то які механізми підтримки узгодженості та актуальності даних будуть використані;

4) чи буде база даних однорідною, тобто, чи будуть всі сервери баз даних продуктами одного і того ж виробника (наприклад, всі сервери тільки Oracle). Якщо база даних не буде однорідною, то яке ПЗ буде використано для обміну даними між СУБД різних виробників (вже існуюче або розроблене спеціально як частина проекту);

5) чи будуть для досягнення належної продуктивності використовуватися паралельні сервери БД (наприклад, Oracle Parallel Server, DB2 UDB тощо).

Етап проектування завершується розробкою *технічного проекту* ІС.

3. *На етапі реалізації* здійснюється створення ПЗ системи, встановлення технічних засобів, розробка експлуатаційної документації.

4. *Етап тестування* зазвичай виявляється розподіленим в часі. Після завершення розробки окремого модуля системи виконують автономний тест, який переслідує таку мету: 1) виявлення відмов модуля (жорстких збоїв); 2) відповідність модуля специфікації (наявність всіх необхідних функцій, відсутність зайвих функцій).

Після того як *автономний тест* успішно пройдено, модуль включають до складу розробленої частини системи і група згенерованих модулів проходить тестування зв'язків, які повинні відстежити їх взаємний вплив.

Далі група *модулів тестується на надійність роботи*, тобто проходять тести імітації відмов системи і напрацювання на відмову. *Перша група тестів* показує, наскільки добре система відновлюється після збоїв ПЗ, відмов апаратного забезпечення. *Друга група тестів* визначає ступінь стійкості системи при штатній роботі і дозволяє оцінити час безвідмовної роботи

системи. У комплект тестів стійкості повинні входити тести, які імітують пікове навантаження на систему.

Потім весь комплект модулів проходить *системний тест* – тест внутрішнього приймання товару, який показує рівень його якості. Сюди входять тести функціональності і надійності системи.

Останній тест ІС – *приймально–здавальні випробування*, який передбачає показ ІС замовникові, повинен містити групу тестів, що моделюють реальні бізнес–процеси, щоб показати відповідність реалізації вимогам замовника.

Необхідність контролювати процес створення ІС, гарантувати досягнення цілей розробки і дотримання різних обмежень (бюджетних, часових тощо) привела до широкого використання у цій сфері методів і засобів програмної інженерії: структурного аналізу, об'єктно–орієнтованого моделювання, CASE–систем.

Підсумок. 1. *Методологічну основу проектування* ПЗ складає системний підхід, під час якого реалізують подання складного об'єкта у вигляді ієрархічної системи взаємопов'язаних моделей (останні дозволяють фіксувати цілісні властивості об'єкта, його структуру і динаміку).

2. *Проектування* ПЗ має вигляд процесу створення специфікацій ПЗ на основі вихідних вимог до нього і зводиться до послідовного уточнення його специфікацій на різних стадіях процесу створення ПЗ.

3. *Невід'ємними властивостями* ПЗ є складність, узгодженість, змінність і невидимість.

4. Об'єктивна потреба контролювати процес розробки складних систем ПЗ, прогнозувати і гарантувати вартість розробки, терміни та якість результатів призвела до необхідності переходу від кустарних до індустріальних способів створення ПЗ, появи сукупності інженерних методів і засобів створення ПЗ, об'єднаних загальною назвою «програмна інженерія».

Запитання для самоконтролю

1. У чому полягає сутність системного підходу до проектування ПЗ. Назвіть два принципи, які дозволяють оцінити взаємний вплив компонентів системи один на одного.

2. Визначіть поняття ІС, її складові. Назвіть основні властивості ІС та види забезпечення ІС

3. У чому полягає мета проектування ПЗ?

4. Назвіть основні ознаки класифікації ІС (різновиди ІС).

5. Назвіть особливості та проблеми сучасних великих проектів програмних систем. В чому полягають причини невдач проектів, і яким є вихід з цього положення?

2. ЖИТТЄВИЙ ЦИКЛ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

1. Нормативно–методичне забезпечення процесу створення програмного забезпечення

Розробка великих проектів пов'язана з роботою колективів людей з декількох організацій, тому вона вимагає сукупність нормативно–методичних документів, які регламентують різні аспекти діяльності розробників. Комплекс таких документів називають *нормативно–методичним забезпеченням* (НМЗ). Ці документи «регламентують: 1) порядок розробки, впровадження та супроводу ПЗ; 2) загальні вимоги до складу ПЗ і зв'язків між його компонентами, а також до його якості; 3) види, склад і зміст проектної та програмної документації [3 с. 44]. Дотримання вимог НМЗ дозволяє створювати ПЗ високої якості, яке відповідає вимогам міжнародних стандартів в області інформаційних технологій (ІТ).

До складу НМЗ входять стандарти та керівні документи, методики виконання складних операцій, шаблони проектних і програмних документів. Всі документи, які входять до складу НМЗ, класифікуються за такими ознаками: 1) «за видом регламентації (стандарт, керівний документ, положення, інструкція тощо); 2) за статусом, який регламентує документ (міжнародний, галузевий, підприємства); 3) за областю дії документа (замовник, підрядник, проект); 4) за об'єктом регламентації або методичного забезпечення» [3 с. 44].

Нормативною базою НМЗ є міжнародні та вітчизняні стандарти в області ІТ і перш за все: 1) міжнародні стандарти ISO / IEC (ISO – International Organization of Standardization, Міжнародна організація із стандартизації, IEC – International Electrotechnical Commission, Міжнародна комісія з електротехніки); 2) стандарти організації–замовника. У вітчизняних розробках доцільно використовувати сучасні міжнародні стандарти.

Поняття життєвого циклу (ЖЦ) ПЗ є одним із базових понять програмної інженерії. *Життєвий цикл* ІС – це модель створення та використання ІС, яка

відображає її різні стани, починаючи з моменту виникнення необхідності в даному комплексі засобів і закінчуючи моментом повного виходу системи з періоду використання користувачами. Умовно можна виділити такі основні етапи ЖЦ ІС:

1) аналіз – визначення того, що повинна робити система;

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

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

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

5) впровадження – встановлення і введення системи в дію;

6) супровід – забезпечення штатного процесу експлуатації системи на підприємстві замовника.

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

2. Деякі стандарти, які регламентують життєвий цикл програмного забезпечення

Існують різні стандарти, які регламентують ЖЦ ПЗ, а в деяких випадках і процеси розробки. Серед відомих стандартів проектування ІС можна виділити такі:

І. **Стандарт ISO 12207: 1995. Процеси життєвого циклу програмних засобів.** ISO 12207:1995 «Базовий стандарт процесів ЖЦ ПЗ, орієнтований на різні види ПЗ і типи проектів автоматизованих систем або інформаційних систем, в які ПЗ входить як частина». Стандарт визначає стратегію і загальний порядок при створенні та експлуатації ПЗ. Він охоплює ЖЦ ПЗ від концептуалізації ідей до завершення ЖЦ.

Користь стандарту: він визначає загальні набори завдань, характеристики якості, критерії оцінки та інші характеристики, необхідні для всебічного охоплення проектних ситуацій. Наприклад, при визначенні вимог до системи необхідно передбачити, щоб: 1) була розглянута область застосування системи; 2) специфікація вимог до системи описувала: функції і можливості системи, бізнес, організаційні вимоги і вимоги користувача, безпеку, захищеність, людські фактори, ергономіку, зв'язки, операції і вимоги супроводу; проектні обмеження та кваліфікаційні вимоги; 3) кваліфікаційні вимоги системи були задокументовані.

При виконанні аналізу вимог щодо ПЗ передбачено *одинадцять класів характеристик якості*, які використовуються пізніше при гарантуванні якості. При цьому розробник повинен встановити і документувати такі *вимоги до ПЗ*:

1) функціональні та інші можливі специфікації (включаючи виконання, фізичні характеристики та умови середовища експлуатації), при яких повинна бути створена одиниця програмного забезпечення;

2) зовнішні зв'язки (інтерфейси) з одиницею ПЗ;

3) специфікації надійності, пов'язані з методами функціонування і супроводу ПЗ, включаючи специфікації, пов'язані з впливом навколишнього середовища і ймовірністю травм персоналу;

4) специфікації захищеності, включаючи специфікації, пов'язані із спотворенням точності інформації;

5) людські фактори специфікацій з інженерної психології (ергономіки), включаючи фактори пов'язані з ручним керуванням, взаємодією людини та обладнання, обмеженнями щодо персоналу, а також фактори, що потребують концентрованої людської уваги;

6) типи даних і вимоги до бази даних;

7) вимоги щодо встановлення та приймання поставленого програмного продукту в місцях його функціонування і супроводу (експлуатації);

8) вимоги до документації користувача;

9) вимоги до роботи користувача і до виконання завдань на основі ПЗ;

10) вимоги до сервісу користувача.

II. Стандарт ISO/IEC/IEEE 15288: 2015(E) «Системи та ПЗ – процеси ЖЦ системи» («Systems and software engineering — System life cycle processes»). Він є базовим стандартом системної інженерії, який визначає поняття системи і ЖЦ, розрізняє цільову і забезпечуючу системи, вводить поняття «практики ЖЦ». Тут сформульовано перелік практик, які повинні виконуватися системними інженерами і менеджерами на всіх стадіях життя системи (мова йде не тільки про специфічні технічні практики, а й про такі практики, як підрядні відносини, управління проектами і портфелями, ризик–менеджмент, управління людськими ресурсами). Стандарт встановлює загальну основу для опису ЖЦ систем, визначає сукупність процесів та асоційовану термінологію з інженерної точки зору. Він є керівництвом для визначення, контролю та вдосконалення організації або процесу ЖЦ системи.

3. Життєвий цикл програмного забезпечення

Поняття життєвого циклу ПЗ. Методологія проектування ІС описує процес створення і супроводу систем у вигляді життєвого циклу ІС, подаючи його як деяку послідовність стадій і виконуваних на них процесів. Для кожного етапу визначаються склад і послідовність виконуваних робіт, одержувані результати, методи і засоби, необхідні для виконання робіт, ролі і відповідальність учасників тощо. Такий опис ЖЦ ІС дозволяє спланувати та організувати процес колективної розробки та забезпечити управління цим процесом.

Життєвий цикл ПЗ визначається як період часу, який розпочинається з моменту прийняття рішення про необхідність створення ПЗ і закінчується в момент його повного вилучення з експлуатації (табл. 2.1 «Зміст основних процесів ЖЦ ПЗ»).

Таблиця 2.1

Зміст основних процесів ЖЦ ПЗ (ISO/IEC 12207)

Процес (вико–навець проце–су)	Дії	Вхід	Результат
Прид–бання (замов–ник)	– ініціювання – підготовка заявоч–них пропозицій – підготовка угоди – контроль діяль–ності постачальника – приймання ІС	– рішенння Про початок робіт по впровадженню ІС – результати обстеження діяльності замовника – результати аналізу ринку ІС / тендеру – план поставки / розробки – комплексний тест ІС	– технікоекономічне об–грунтування впровад–ження ІС – технічне завдання на ІС – договір на поставку/ розробку – акти приймання етапів роботи – акт приймально–зда–вальних випробувань
Поста–чання (роз–робник ІС)	– ініціювання – відповідь на за–явочні пропозиції – підготовка догово–ру – планування вико–нання – контроль виконан–ня – постачання	– технічне завдання на ІС – рішення керівництва про участь в розробці – результати тендеру – план УП – розроблена ІС і до–кументація	– рішення про участь в розробці – комерційні пропозиції / конкурсна заявка – договір на поставку / розролення – план УП – реалізація / коригування – акт приймально–здаваль–них випробувань
Розроб–лення (роз–робник ІС)	– підготовка – аналіз вимог до ІС – проектування архі–тектури ІС – розробка вимог до ПЗ – проектування архі–тектури ПЗ – детальне проекту–вання ПЗ – кодування і тес–тування ПЗ – інтеграція ПЗ і кваліфікаційне тестування ПЗ – інтеграція ІС і кваліфікаційне тестування ІС	– технічне завдання на ІС – технічне завдання на ІС, модель ЖЦ – технічне завдання на ІС – підсистеми ІС – специфікації вимоги до компонентів ПЗ – архітектура ПЗ – матеріали детального проектування ПЗ – план інтеграції ПЗ, тести – архітектура ІС, ПЗ, документація на ІС, тести	– використовувана модель ЖЦ, стандарти розробки – план робіт – склад підсистем, компо–ненти обладнання – специфікації вимоги до компонентів ПЗ – склад компонентів ПЗ, інтерфейси з БД, план інтеграції ПЗ – проект БД, специфікації інтерфейсів між компо–нентами ПЗ, вимоги до тестів – тексти модулів ПЗ, акти автономного тестування – оцінка відповідності 1)комплексу ПЗ вимогам ТЗ; 2) ПЗ, БД, технічного комплексу та комплекту документації

Основним нормативним документом, який регламентує склад процесів ЖЦ ПЗ, є міжнародний стандарт ДСТУ ISO/IEC/IEEE 12207:2018 Інженерія систем і програмних засобів. Процеси ЖЦ програмних засобів (ISO/IEC/IEEE 12207: 2017, IDT). Цей стандарт встановлює загальну основу для процесів ЖЦ ПЗ з чітко визначеною термінологією.

Життєвий цикл передбачає залучення зацікавлених сторін для досягнення задоволення потреб клієнтів. Він містить процеси, дії та завдання, які мають застосовуватися під час придбання, постачання, розробки, експлуатації, обслуговування та утилізації програмної системи. При цьому **процес** визначають як «сукупність взаємопов'язаних дій, які перетворюють деякі вхідні дані у вихідні. Кожен процес: 1) характеризується конкретними задачами та методами їх вирішення, вихідними даними, отриманими від інших процесів, і результатами; 2) складається із набору дій, кожна дія – із набору завдань. Кожен процес, дія або завдання при необхідності ініціюється та виконується іншим процесом, причому не існує заздалегідь визначених послідовностей їх виконання». [3 с. 46].

Стандарт ISO/IEC/IEEE 12207:2018 поширюється на придбання програмних систем (продуктів чи послуг), на поставку, розробку, експлуатацію, технічне обслуговування та утилізацію програмних продуктів, частин ПЗ будь–якої системи (незалежно від того, здійснюються вони самою організацією або зовні).

Відповідно до стандарту ISO/IEC серії 15288 в структуру ЖЦ слід включати такі групи процесів:

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

2. *Процеси підприємства:* управління навколишнім середовищем підприємства; інвестиційне управління; управління ЖЦ 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; супровід; утилізація.

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

Стадії створення системи, передбачені в стандарті ISO/IEC 15288, дещо відрізняються від розглянутих вище. Перелік стадій та основні результати, які повинні бути досягнуті до моменту їх завершення, наведені в табл. 2.2.

Таблиця 2.2

Стадії створення систем (ISO/IEC 15288)

№ з\п	Стадія	Опис
1	Формування	Аналіз потреб, вибір концепції та проектних
2	Розробка	Проектування системи
3	Реалізація	Виготовлення системи
4	Експлуатація	Введення в експлуатацію і використання
5	Підтримка	Забезпечення функціонування системи
6	Зняття з експлуатації	Припинення використання, демонтаж, архівування системи

Під *процесом вдосконалення* в стандарті розуміють удосконалення процесів придбання, розробки, гарантування якості та інших процесів, здійснюваних в організації.

Взаємозв'язок між процесами ЖЦ ПЗ. Процеси ЖЦ ПЗ, які регламентовано стандартом ISO/IEC 12207, можуть використовуватися різними організаціями на конкретних проектах по різному. Стандарт пропонує базовий набір взаємозв'язків між процесами з різних точок зору (або в різних аспектах). Такими аспектами є:

1. *Договірнй аспект* (замовник–постачальник). У договірному аспекті замовник і постачальник вступають в договірні відносини і реалізують відповідно процеси придбання та постачання.

2. *Аспект управління* (менеджер). В аспекті управління замовник, постачальник, розробник, оператор, служба супроводження та інші сторони беруть участь в ЖЦ ПЗ, керують виконанням своїх процесів.

3. *Аспект експлуатації* (користувач). В аспекті експлуатації оператор, який експлуатує систему, надає необхідні послуги користувачам.

4. *Інженерний аспект*: розробка, супровід (розробник, служба супроводу). В інженерному аспекті розробник або служба супроводу вирішують відповідні технічних завдання, розробляючи або модифікуючи програмні продукти.

5. *Аспект підтримки* (виконавець допоміжних процесів). В аспекті підтримки задіяно служби, які реалізують допоміжні процеси і надають необхідні послуги всім іншим учасникам робіт. В рамках аспекту підтримки можна виділити аспект управління якістю ПЗ, який включає п'ять процесів: забезпечення якості, верифікацію, атестацію, спільну оцінку та аудит.

Взаємозв'язки між процесами, описані в стандарті, носять статичний характер. Більш важливі динамічні зв'язки між процесами та сторонами, які їх реалізують, встановлюються в реальних проектах. Співвідношення процесів ЖЦ ПЗ і стадій ЖЦ, що характеризують часовий аспект ЖЦ системи, розглядається в рамках моделі ЖЦ ПЗ.

Стандарт ISO/IEC 12207 формує підхід до вибору та оцінювання усіх сучасних технологій, процесів створення та супроводження ПЗ. На вибір конкретної технології в проекті впливає низка факторів, але принципи реалізації та склад процесів ЖЦ ПЗ залишаються незмінними.

Процеси, які входять до складу ЖЦ ПЗ. Відповідно до базових міжнародних стандартів ISO / IEC 12207 *усі процеси* ЖЦ ПЗ розділяють на такі три групи: основні, допоміжні, організаційні.

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

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

3. *Організаційні процеси* охоплюють процеси створення інфраструктури; управління; навчання; удосконалення. До них примикає процес адаптації, який визначає основні дії, необхідні для адаптації стандарту до умов конкретного проекту. *Організаційні процеси* визначають дії і завдання, що виконуються замовником та розробником проекту для управління їхніми процесами.

Організаційні процеси виконуються на корпоративному рівні або на рівні всієї організації в цілому, створюючи базу для реалізації та постійного вдосконалення інших процесів ЖЦ ПЗ. Процеси та організації, які їх реалізують, функціонально пов'язані між собою, при цьому внутрішня структура і статус організацій ніяк не регламентуються. Одна і та ж організація може виконувати різні ролі: постачальника, розробника та інші, і навпаки, одна й та сама роль може виконуватися декількома організаціями.

Розглянемо ці групи процесів більш детально. [3 с.48–64].

Основні процеси ЖЦ ПЗ:

1. **Процес придбання** складається з дій і завдань замовника, який купує ПЗ. Цей процес охоплює такі дії: 1) ініціювання придбання; 2) підготовку заявочних пропозицій; 3) підготовку та коригування договору; 4) нагляд за діяльністю постачальника; 5) приймання і завершення робіт.

Ініціювання придбання включає такі завдання: 1) визначення замовником своїх потреб у придбанні, розробці або вдосконаленні системи; 2) аналіз вимог до системи; 3) прийняття рішення щодо придбання, розробки або удосконалення існуючого ПЗ; 4) перевірка наявності необхідної документації, гарантій, сертифікатів, ліцензій і підтримка в разі придбання програмного

продукту; 5) підготовка та затвердження плану придбання, який включає вимоги до системи, тип договору, відповідальність сторін тощо.

Заявочні пропозиції повинні містити: вимоги до системи; перелік програмних продуктів; умови та угоди; технічні обмеження (наприклад, середовище функціонування системи). Їх направляють обраному постачальнику (або декільком постачальникам у разі проведення тендера). *Постачальник* – це організація, яка укладає договір із замовником на поставку системи, ПЗ або програмної послуги на умовах, обумовлених в договорі.

Підготовка та коригування договору включають такі завдання: 1) визначення замовником процедури вибору постачальника, яка включає критерії оцінки пропозицій можливих постачальників; 2) вибір конкретного постачальника на основі аналізу пропозицій; 3) підготовлення та укладення договору з постачальником; 4) внесення змін (при необхідності) в договір в процесі його виконання.

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

У процесі *приймання* готуються і виконуються необхідні тести. В разі задоволення усіх умов приймання здійснюється *завершення робіт* за договором.

2. **Процес постачання** охоплює дії та завдання, які виконуються постачальником, що постачає замовнику програмний продукт. Даний процес включає такі дії: 1) ініціювання поставки; 2) підготовку відповіді на заявочні пропозиції; 3) підготовку договору; 4) планування; 5) виконання та контроль; 6) перевірку та оцінку; 7) постачання і завершення робіт.

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

Планування включає такі завдання: 1) прийняття рішення постачальником щодо виконання робіт своїми силами або із залученням субпідрядника;

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

3. **Процес розробки** передбачає дії і завдання, які виконуються розробником, і охоплює роботи зі створення ПЗ і його компонентів відповідно до заданих вимог, включаючи оформлення проектної та експлуатаційної документації, підготовку матеріалів, необхідних для перевірки працездатності та відповідної якості програмних продуктів, матеріалів, необхідних для організації навчання персоналу тощо. Процес розробки включає такі дії: 1) підготовчу роботу; 2) аналіз вимог до системи; 3) проектування архітектури системи; 4) аналіз вимог до ПЗ; 5) проектування архітектури ПЗ; 6) детальне проектування ПЗ; 7) кодування і тестування ПЗ; 8) інтеграцію ПЗ; 9) кваліфікаційне тестування ПЗ; 10) інтеграцію системи; 11) кваліфікаційне тестування системи; 12) встановлення ПЗ; 13) приймання ПЗ.

Підготовча робота розпочинається з вибору моделі ЖЦ ПЗ, яка відповідає масштабу, значущості та складності проекту. Дії і завдання процесу розробки повинні відповідати обраній моделі. Розробник повинен вибрати, адаптувати до умов проекту і виконати узгоджені із замовником стандарти, методи і засоби розробки, а також скласти план виконання робіт.

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

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

Аналіз вимог до ПЗ передбачає визначення таких характеристик для кожного компонента ПЗ: 1) функціональних можливостей, включаючи характеристики продуктивності та середовища функціонування компонента; 2) зовнішніх інтерфейсів; 3) специфікацій надійності і безпеки; 4) ергономічних вимог; 5) вимог до використовуваних даних; 6) вимог до встановлення і приймання; 7) вимог до користувальницької документації; 8) вимог до експлуатації і супроводу. Вимоги до ПЗ оцінюються виходячи з критеріїв відповідності вимогам системи, можливостей системи бути реалізованою та її перевірки при тестуванні.

Проектування архітектури ПЗ передбачає такі завдання (для кожного компонента ПЗ): 1) трансформацію вимог до ПЗ в архітектуру, яка визначає на високому рівні структуру ПЗ і склад його компонентів; 2) розробку і документування програмних інтерфейсів ПЗ і баз даних; 3) розробку попередньої версії користувальницької документації; 4) розробку і документування попередніх вимог до тестів і плану інтеграції ПЗ.

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

Детальне проектування ПЗ передбачає такі завдання: 1) опис компонентів ПЗ та інтерфейсів між ними на більш низькому рівні, достатньому для їх подальшого самостійного кодування і тестування; 2) розробка і документування детального проекту бази даних; 3) оновлення (при необхідності) користувальницької документації; 4) розробка і документування вимог до тестів і плану тестування компонентів ПЗ; 5) оновлення плану інтеграції ПЗ.

Кодування і тестування ПЗ охоплює такі завдання: 1) розробка (кодування) і документування кожного компонента ПЗ і бази даних, а також сукупності тестових процедур і даних для їх тестування; 2) тестування кожного компонента ПЗ і бази даних на відповідність запропонованим до них вимогам (результати тестування компонентів повинні бути задокументовані);

3) оновлення (при необхідності) користувальницької документації;
4) оновлення плану інтеграції ПЗ.

Інтеграція ПЗ передбачає об'єднання розроблених компонентів ПЗ відповідно до плану інтеграції та тестування агрегованих компонентів. Для кожного з агрегованих компонентів розробляються набори тестів і тестові процедури, призначені для перевірки кожного з кваліфікаційних вимог при подальшому кваліфікаційному тестуванні.

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

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

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

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

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

4. **Процес експлуатації** охоплює дії і завдання оператора – організації, яка експлуатує систему. Даний процес включає такі дії: 1) підготовчу роботу; 2) експлуатаційне тестування; 3) експлуатацію системи; 4) підтримку користувачів. *Підготовча робота* включає проведення оператором таких завдань: 1) планування дій і робіт, які виконуються в процесі експлуатації, і встановлення експлуатаційних стандартів; 2) визначення процедур локалізації та ліквідації проблем, які виникають в процесі експлуатації. *Експлуатаційне тестування* здійснюється для кожної чергової редакції програмного продукту, після чого вона передається в експлуатацію. *Експлуатація системи* виконується в призначеному для цього середовищі відповідно до користувальницької документації. *Підтримка користувачів* полягає в наданні допомоги і консультацій при виявленні помилок в процесі експлуатації ПЗ.

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

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

Зміни, які вносяться до існуючого ПЗ, не повинні порушувати його цілісність. Процес супроводу включає перенесення ПЗ в інше середовище (міграцію) і закінчується зняттям ПЗ з експлуатації. Процес супроводу охоплює такі дії: 1) підготовчу роботу; 2) аналіз проблем і запитів на модифікацію ПЗ; 3) модифікацію ПЗ; 4) перевірку і приймання; 5) перенесення ПЗ в інше середовище; 6) зняття ПЗ з експлуатації.

Підготовча робота служби супроводу передбачає такі завдання: 1) планування дій і робіт, які виконуються в процесі супроводу; 2) визначення процедур локалізації та ліквідації проблем, які виникають в процесі супроводу.

Аналіз проблем і запитів на модифікацію ПЗ, що виконується службою супроводу, включає такі завдання:

1) аналіз повідомлення про проблему, яка виникла, або запит на модифікацію ПЗ щодо його впливу на організацію, існуючу систему та інтерфейси з іншими системами; при цьому визначаються такі характеристики можливої модифікації:

– тип модифікації (коригуюча, поліпшуюча, профілактична або адаптуюча до нового середовища);

– масштаб (розміри модифікації, вартість і час її реалізації);

– критичність (вплив на продуктивність, надійність або безпеку);

2) оцінка доцільності проведення модифікації та можливих варіантів її проведення;

3) затвердження обраного варіанту модифікації.

Модифікація ПЗ передбачає визначення компонентів ПЗ (їхніх версій і документації), які підлягають модифікації, внесення необхідних змін відповідно до правил процесу розробки. Підготовлені зміни тестуються і перевіряються за критеріями, визначеними в документації. При підтвердженні правильності змін в програмах виконується коригування документації.

Перевірка і приймання полягають у перевірці цілісності системи, яка модифікується, і у затвердженні внесених змін.

При перенесенні ПЗ в інше середовище, використовуються наявні або розробляються нові засоби перенесення, потім виконується конвертування програм і даних у нове середовище. Щоб полегшити перенос ПЗ в інше середовище, протягом деякого періоду передбачається паралельна експлуатація ПЗ в старому і новому середовищах (коли проводиться необхідне навчання користувачів роботі в новому середовищі).

Зняття ПЗ з експлуатації здійснюється за рішенням замовника за участю експлуатуючої організації, служби супроводу і користувачів. При цьому програмні продукти і відповідна документація підлягає архівуванню відповідно

до договору. Протягом деякого періоду передбачена паралельна експлуатація старого і нового ПЗ (виконується необхідне навчання користувачів роботі з новою системою).

Допоміжні процеси життєвого циклу ПЗ

1. **Процес документування** передбачає формалізований опис інформації, створеної протягом ЖЦ ПЗ. Цей процес складається з множини дій, за допомогою яких планують, проектують, розробляють, випускають, редагують, поширюють і сопроводжують документи, необхідні для всіх зацікавлених осіб, таких, як керівництво, технічні фахівці і користувачі системи. Процес документування включає такі дії: 1) підготовчу роботу; 2) проектування і розробку; 3) випуск документації; 4) супровід.

2. **Процес управління конфігурацією** передбачає застосування адміністративних і технічних процедур на всьому протязі ЖЦ ПЗ для визначення стану компонентів ПЗ в системі, управління модифікаціями ПЗ, опису і підготовки звітів про стан компонентів ПЗ і запитів на модифікацію, забезпечення повноти, сумісності і коректності компонентів ПЗ, управління зберіганням і постачанням ПЗ.

Під **конфігурацією ПЗ** розуміють «сукупністьність його функціональних і фізичних характеристик, встановлених в тихнічній документації і реалізованих в ПЗ. Управління конфігурацією дозволяє організувати, систематично враховувати і контролювати внесення змін до ПЗ на всіх стадіях ЖЦ. Процес управління конфігурацією включає такі дії: 1) підготовчу роботу; 2) ідентифікацію конфігурації; 3) контроль за конфігурацією; 4) облік стану конфігурації; 5) оцінку зміни; 6) управління випуском і постачання». [3 с. 57].

Підготовча робота полягає в плануванні управління конфігурацією.

Ідентифікація конфігурації встановлює правила, за допомогою яких можна ідентифікувати і розрізняти компоненти ПЗ та їх версії. Крім того, кожному компоненту і його версіями відповідає комплект документації, який однозначно ідентифікується. В результаті створюється база для вибору і

маніпулювання версіями компонентів ПЗ, яка використовує обмежену і впорядковану систему символів, ідентифікуючих різні версії ПЗ.

Контроль за конфігурацією призначений для оцінювання передбачуваних модифікацій ПЗ і координованої їх реалізації з урахуванням ефективності кожної модифікації і витрат на її виконання. Він забезпечує контроль за станом і розвитком компонентів ПЗ, їх версій, адекватність змінених компонентів та їх документації.

Облік стану конфігурації має вигляд реєстрації стану компонентів ПЗ, підготовки звітів про всі реалізовані і відкинуті версії модифікацій компонентів ПЗ. Сукупність звітів забезпечує однозначне відображення поточного стану системи та її компонентів, а також ведення історії модифікацій.

Процес оцінювання конфігурації полягає в оцінюванні функціональної повноти компонентів ПЗ, а також відповідності їх фізичного стану поточному технічному опису.

Управління випуском і постачання охоплюють виготовлення еталонних копій програм і документації, їх зберігання та постачання користувачам відповідно до порядку, прийнятому в організації.

3. **Процес забезпечення якості** забезпечує відповідні гарантії того, що ПЗ і процеси його ЖЦ відповідають заданим вимогам і затвердженим планам. Під *якістю* ПЗ розуміють сукупність властивостей, які характеризують здатність ПЗ задовольняти заданим вимогам.

Для отримання достовірних оцінок створюваного ПЗ процес забезпечення його якості повинен відбуватися незалежно від суб'єктів, безпосередньо пов'язаних з розробкою ПЗ. При цьому можуть використовуватися результати інших допоміжних процесів, таких, як верифікація, атестація, сумісна оцінка, аудит та вирішення проблем. Процес забезпечення якості включає такі дії: 1) підготовча робота; 2) забезпечення якості продукту; 3) забезпечення якості процесу; 4) забезпечення інших показників якості системи.

Підготовча робота полягає у координації з іншими допоміжними процесами, у плануванні самого процесу забезпечення якості з урахуванням використовуваних стандартів, методів, процедур і засобів.

Забезпечення якості продукту передбачає гарантування повної відповідності програмних продуктів та їх документації вимогам замовника, передбаченим в договорі.

Забезпечення якості процесу передбачає гарантування відповідності процесів ЖЦ ПЗ, методів розробки, середовища розробки і кваліфікації персоналу умовам договору, встановленим стандартам і процедурам.

Забезпечення інших показників якості системи здійснюється відповідно з умовами договору і стандартом якості ISO 9001.

4. **Процес верифікації** полягає у визначенні того, що програмні продукти, які є результатами певної дії, повністю задовольняють вимогам або умовам, обумовленим попередніми діями (верифікація у вузькому сенсі означає формальний доказ правильності ПЗ). Для підвищення ефективності верифікація повинна якомога раніше інтегруватися з процесами, які її використовують (такими, як постачання, розробка, експлуатація або супровід). Цей процес може включати аналіз, оцінку і тестування.

Верифікація може виконуватися із різним ступенем незалежності. Ступінь незалежності може змінюватися від виконання верифікації самим виконавцем (або іншим фахівцем організації) до її виконання фахівцем іншої організації з різними варіаціями. Якщо процес верифікації здійснюється організацією, яка не залежить від постачальника, розробника, оператора або служби супроводу, то він називається *процес незалежної верифікації*. Процес верифікації включає такі дії: 1) підготовча робота; 2) верифікація.

У процесі верифікації перевіряються такі умови: 1) несуперечність вимог до системи та ступеня врахування потреб користувачів; 2) можливості постачальника виконати задані вимоги; 3) відповідність обраних процесів ЖЦ ПЗ умовам договору; 4) адекватність стандартів, процедур і середовища

розробки процесам ЖЦ ПЗ; 5) відповідність проектних специфікацій ПЗ заданим вимогам; 6) коректність опису в проектних специфікаціях вхідних і вихідних даних, послідовності подій, інтерфейсів, логіки тощо; 7) відповідність коду проектним специфікаціям і вимогам; 8) можливість тестувати та перевіряти на коректність код, його відповідність прийнятим стандартам кодування; 9) коректність інтеграції компонентів ПЗ в систему; 10) адекватність, повнота і несуперечність документації.

5. **Процес атестації** передбачає визначення повноти відповідності заданих вимог до створеної системи її конкретному функціональному призначенню.

Під *атестацією* розуміють підтвердження та оцінку достовірності проведеного тестування ПЗ. Атестація повинна гарантувати повну відповідність ПЗ специфікаціям, вимогам і документації, а також можливість його безпечного і надійного застосування користувачем. Атестацію рекомендують виконувати шляхом тестування в усіх можливих ситуаціях і використовувати при цьому незалежних фахівців. Її можна виконувати на початкових стадіях ЖЦ ПЗ або як частина роботи з приймання ПЗ.

Атестація, як і процес перевірки, може здійснюватися з різним ступенем незалежності. Якщо процес атестації виконується організацією, яка не залежить від постачальника, розробника, оператора або служби супроводу, його називають *процесом незалежної атестації*. Процес атестації включає такі дії: 1) підготовча робота; 2) атестація.

6. **Процес спільного оцінювання** призначений для оцінювання стану робіт по проекту і ПЗ, яке створюється при виконанні цих робіт (дій). Він зосереджений в основному на контролі планування та управління ресурсами, персоналом, апаратурою та інструментальними засобами проекту. Оцінювання проводять як на рівні УП, так і на рівні технічної реалізації проекту (вона проводиться протягом усього терміну дії договору). Цей процес може виконуватися двома будь–якими сторонами, які беруть участь в договорі (при

цьому одна сторона перевіряє іншу). Процес спільного оцінювання включає такі дії: 1) підготовча робота; 2) оцінювання УП; 3) технічне оцінювання.

7. **Процес аудиту** передбачає визначення відповідності вимогам, планам та умовам договору. Аудит може виконуватися двома будь–якими сторонами, які беруть участь в договорі, коли одна сторона перевіряє іншу.

Аудит – це ревізія (перевірка), яка проводиться компетентним органом (особою) для забезпечення незалежної оцінки ступеня відповідності ПЗ (або процесів) встановленим вимогам. Аудит використовують для встановлення відповідності реальних робіт і звітів вимогам, планам і контракту. Аудитори (ревізори) не повинні мати прямої залежності від розробників ПЗ. Вони визначають стан робіт, використання ресурсів, відповідність документації специфікаціям і стандартам, коректність тестування. Процес аудиту включає такі дії: 1) підготовча робота; 2) аудит.

8. **Процес вирішення проблем** передбачає аналіз і вирішення проблем, незалежно від їх походження або джерела, виявлені невідповідності під час розроблення, експлуатації, супроводу або інших процесів. Кожна виявлена проблема повинна бути ідентифікована, описана, проаналізована та вирішена. Процес вирішення проблем включає такі дії: 1) підготовча робота; 2) розв'язання проблем.

Організаційні процеси життєвого циклу ПЗ

1. **Процес управління** складається з дій і завдань, які можуть виконуватися будь–якою стороною, що управляє своїми процесами. Ця сторона (*менеджер*) відповідає за управління випуском продукту, УП та визначення завдань відповідних процесів (таких, як придбання, постачання, розробка, експлуатація, супровід тощо). Процес управління включає такі дії: 1) ініціювання та визначення галузі управління; 2) планування; 3) виконання та контроль; 4) перевірка та оцінювання; 5) завершення.

При *ініціюванні* менеджер повинен переконатися, що в його розпорядженні в достатній кількості є необхідні для управління ресурси

(персонал, обладнання та технологія). *Планування* передбачає виконання таких завдань: 1) складання графіків виконання робіт; 2) оцінювання витрат; 3) виділення необхідних ресурсів; 4) розподіл відповідальності; 5) оцінювання ризиків, пов'язаних з конкретними завданнями; 6) створення інфраструктури управління.

2. **Процес створення інфраструктури** охоплює вибір і підтримку (супровід) технології, стандартів та інструментальних засобів, вибір та встановлення апаратних і програмних засобів, які використовуються для розробки, експлуатації або супроводу ПЗ. Інфраструктура повинна модифікуватися і супроводжуватися відповідно до змін вимог до відповідних процесів. Інфраструктура є одним із об'єктів управління конфігурацією. Процес створення інфраструктури включає такі дії: 1) підготовча робота; 2) створення інфраструктури; 3) супровід інфраструктури.

3. **Процес удосконалення** передбачає оцінювання, вимірювання, контроль та удосконалення процесів ЖЦ ПЗ. Цей процес включає такі дії: 1) створення процесу; 2) оцінювання процесу; 3) удосконалення процесу.

Удосконалення процесів ЖЦ ПЗ спрямовано на підвищення продуктивності праці всіх фахівців, які беруть участь в розробці, за рахунок використання технології та методів управління, вибору інструментальних засобів і навчання персоналу. Удосконалення базується на аналізі переваг і недоліків кожного процесу, чому сприяє накопичення історичної, технічної, економічної та іншої інформації про реалізовані проекти в організації.

4. **Процес навчання** охоплює початкове навчання і подальше постійне підвищення кваліфікації персоналу. Придбання, постачання, розробка, експлуатація та супровід ПЗ в значній мірі залежать від рівня знань і кваліфікації персоналу. Зміст навчання визначається вимогами до проекту, він повинен враховувати необхідні ресурси і технічні засоби навчання. Повинні бути розроблені і подані методичні матеріали, необхідні для навчання користувачів відповідно до навчаньних планів. Процес навчання включає такі

дії: 1) підготовча робота; 2) розробка навчальних матеріалів; 3) реалізація плану навчання.

4. Моделі життєвого циклу програмного забезпечення

Модель ЖЦ ІС – це структура, що містить процеси, дії і завдання, які здійснюються під час розробки, функціонування та супроводу програмного продукту протягом усього життя системи (від визначення вимог до завершення її використання).

Під *моделлю життєвого циклу* ПЗ розуміють «структуру, яка визначає послідовність виконання і взаємозв'язки процесів, дій і задач протягом ЖЦ. Модель ЖЦ залежить від 1) специфіки, масштабу та складності проекту, 2) специфіки умов, в яких система створюється і функціонує». [3 с. 67]. Загальними для будь–яких моделей ЖЦ створення ПЗ є положення стандарту ДСТУ ISO/IEC 12207–99, який описує структуру процесів ЖЦ ПЗ, не конкретизуючи в деталях як реалізувати або виконати дії і завдання, включені в ці процеси.

Модель ЖЦ ПЗ включає в себе: 1) стадії; 2) результати виконання робіт на кожній стадії; 3) ключові події (або точки завершення робіт і прийняття рішень). Модель ЖЦ будь–якого конкретного ПЗ визначає характер процесу його створення, який має вигляд сукупності впорядкованих у часі, взаємопов'язаних та об'єднаних у стадії робіт, виконання яких необхідно і досить для створення ПЗ, що відповідає заданим вимогам.

Під *стадією* розуміють «частину процесу створення ПЗ, обмежену певними часовими рамками і таку, що закінчується випуском конкуруючого продукту (моделей, програмних компонентів, документації), який визначається заданими для даної стадії вимогами» [3 с. 67]. Стадії виділяють з міркувань раціонального планування та організації робіт, що закінчуються визначеними результатами.

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

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

1. **Каскадна модель** (рис. 2.1) передбачає послідовне виконання всіх етапів проекту в чітко фіксованому порядку; припускає перехід на наступний етап після повного завершення робіт попереднього етапу.

2. **Поетапна ітераційна модель з проміжним контролем** (рис. 2.2) передбачає розробку ІС ітераціями з циклами зворотного зв'язку між етапами. Коригування між етапами забезпечують велику гнучкість і меншу трудомісткість порівняно з каскадною моделлю. Час життя кожного з етапів розтягується на весь період розробки.

3. **Спіральна модель** (рис. 2.3) робить наголос на початкових етапах ЖЦ (аналіз, проектування), а реалізація технічних рішень перевіряється та обґрунтовується за допомогою створення прототипів. Кожен виток спіралі (ітерація) відповідає поетапній моделі створення фрагмента або версії системи, на ньому уточнюються цілі й характеристики проекту, визначається його якість, плануються роботи наступного витка спіралі.

Розглянемо ці моделі більш детально. [3 с. 70–79].

1. Каскадна модель ЖЦ. У 1970 р. експерт в галузі ПЗ Уїнстон Ройс опублікував статтю, в якій виклав думки про методику «моделі водоспаду» (waterfall model), або «каскадну модель» (рис. 2.1). Згодом ця модель була регламентована множиною нормативних документів, зокрема, стандартом Міністерства оборони США Dod–STD–2167A.

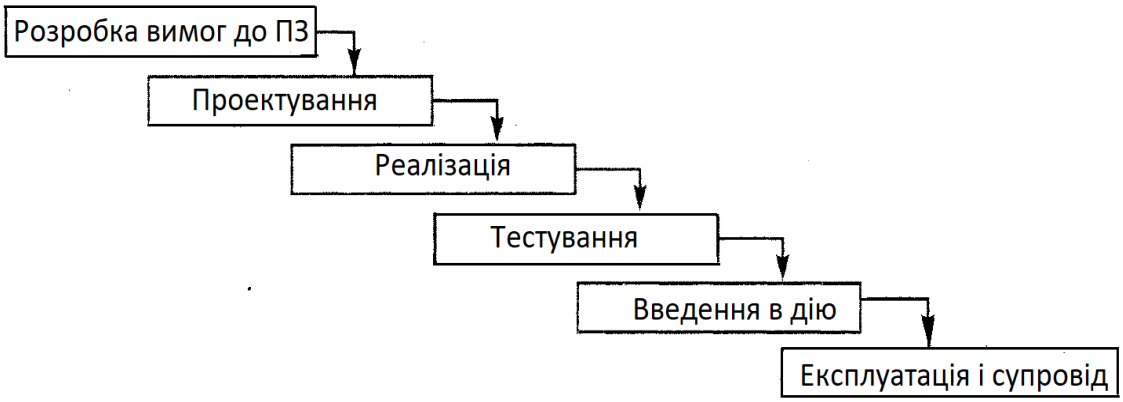


Рис. 2.1. Каскадна модель ЖЦ ПЗ

Серед принципових властивостей «чистої» каскадної моделі можна виділити такі: 1) фіксація вимог до системи до часу її здачі замовнику; 2)перехід на чергову стадію проекту відбувається тільки після того, як повністю завершена робота на поточній стадії, без повернень на пройдені стадії.

Кожна стадія закінчується отриманням результатів, які служать вихідними даними для наступної стадії. Кожна стадія завершується випуском комплекту документації, достатнього для того, щоб розробку можна було продовжити іншою командою розробників.

Вимоги до ПЗ, яке розробляється, визначені на стадії формування вимог, чітко документуються у вигляді технічного завдання і фіксуються на весь час розробки проекту. Узгодження отриманих результатів з користувачами проводиться тільки в точках, запланованих після завершення кожної стадії (при цьому можливе корегування результатів по зауваженнях користувачів, якщо вони не зачіпають вимоги, викладені в технічному завданні). Таким чином, користувачі можуть внести суттєві зауваження тільки після того, як робота над системою буде повністю завершена. У разі неточного викладу вимог або їх зміни протягом тривалого періоду створення ПЗ користувачі отримують систему, яка не задовольняє їх потребам. В результаті доводиться розпочинати новий проект, який може спіткати така ж доля.

Серед *переваг застосування каскадної моделі* можна виділити такі: 1) на кожній стадії формується закінчений набір проектної документації, який

відповідає критеріям повноти та узгодженості; 2) виконувані в логічній послідовності стадії робіт дозволяють планувати час завершення всіх робіт і відповідні витрати.

Каскадну модель можна використати при створенні ПЗ, для якого на початку розробки можна точно і повно сформулювати всі вимоги, щоб надати розробникам можливість якнайкраще реалізувати їх технічно. До цієї групи належать складні системи з великою кількістю завдань обчислювального характеру, системи управління виробничими процесами підвищеної небезпеки.

Серед *недоліків* застосування *каскадного підходу* можна виділити такі: пізнє виявлення проблем; вихід з календарного графіка, запізнення з отриманням результатів; надмірна кількість документації; неможливість розбити систему на частини (весь продукт розробляється за один раз); високий ризик створення системи, що не задовольняє зміненим потребам користувачів.

Реальний процес створення ПЗ ніколи повністю не укладається в чітку схему. Процес створення ПЗ, зазвичай, носить ітераційний характер: результати чергової стадії часто викликають зміни в проектних рішеннях, вироблених на більш ранніх стадіях. Отже, постійно виникає потреба в поверненні до попередніх стадій, в уточненні або перегляді раніше прийнятих рішень. В результаті реальний процес створення ПЗ має вигляд, зображений на рис. 2.2.

На початковій стадії проекту повністю і точно сформулювати всі вимоги до майбутньої системи не вдається, що пояснюється двома причинами: 1) користувачі не в змозі відразу викласти всі свої вимоги та передбачити, як вони зміняться під час розробки; 2) за час розробки можуть відбутися зміни у зовнішньому середовищі, які вплинуть на вимоги до системи.

Першоджерело проблем, пов'язаних з каскадним підходом, полягає у поглядах, що розробка ПЗ має багато спільного з будівельними або інженерними проектами.

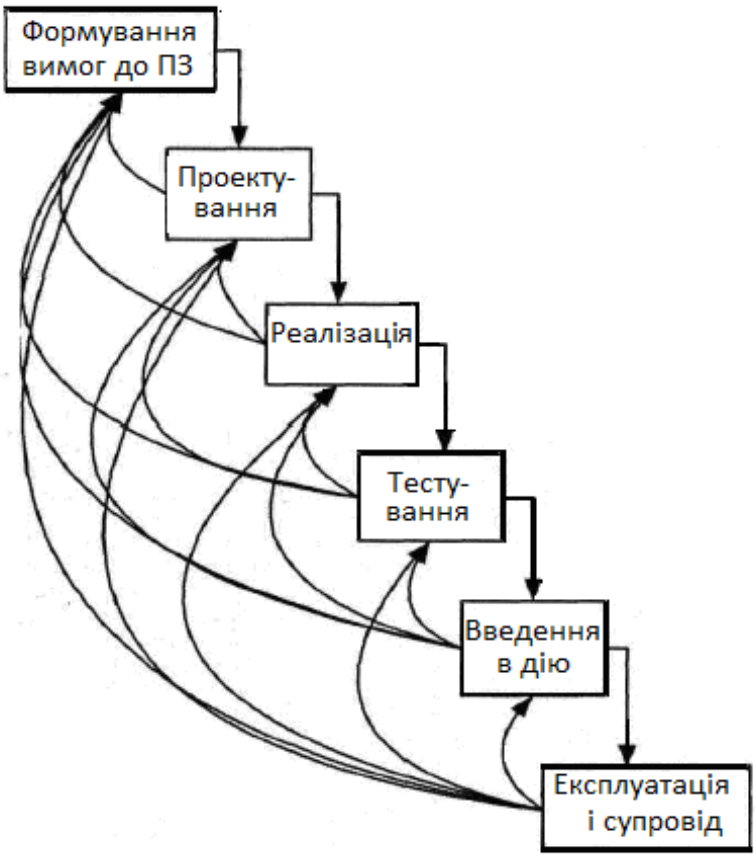


Рис. 2.2. Реальний процес розробки ПЗ

У будівельних проектах завдання виконуються чітко послідовно (наприклад, не можна закласти опору моста, поки не вирито для них ями; покладання настилу моста не може розпочатися, поки не завершено будівництво опор). У ранніх підходах до процесу розробки ПЗ використовувилися такі ж принципи: 1) група аналітиків збирала і документувала вимоги; 2) коли вимоги були затверджені, розпочиналося проектування; 3) після затвердження проекту розпочиналося написання коду; 4) кожен рядок коду підлягав перевірці. Якщо його затверджували, то його дозволяли інтегрувати в продукт. У цьому полягає сутність *«чистого» каскадного підходу*.

На практиці виявилося, що «будівельний» підхід привів до невдач при розробці ПЗ на основі каскадного підходу. Спроби перетворити дії по розробці ПЗ в послідовну форму призводять до одного з двох можливих результатів: 1) рання невдача, 2) пізня невдача. Успішні результати бувають дуже рідко.

Ранню невдачу терплять проекти, які «чітко виконуються». При цьому етап складання специфікації вимог або специфікацій проекту ніколи не буде виконаний: при кожному перегляді документів виникають нові проблеми і сумніви, виявляються недоліки, ставлять запитання, на які не можна дати відповіді. *Пізня невдача* має місце на завершальному етапі, коли виявляється, що споживачі не задоволені створеним продуктом.

2. Ітераційна модель ЖЦ. Джерела концепції ітераційної розробки простежуються до 1930–х років у роботах експерта з проблем якості продукції Уолтера Шеварта, який запропонував орієнтовану на підвищення якості методику, що складається з серії коротких циклів кроків з планування, реалізації, вивчення та дії (plan–do–study–act, PDSA). Пізніше PDSA була досліджена стосовно розробки ПЗ: в середині 1980–х років Баррі Боем запропонував свій варіант ітераційної моделі під назвою «*спіральна модель*» (рис. 2.3). Порівняння каскадної та ітераційної моделей виконано у Додатку Б.

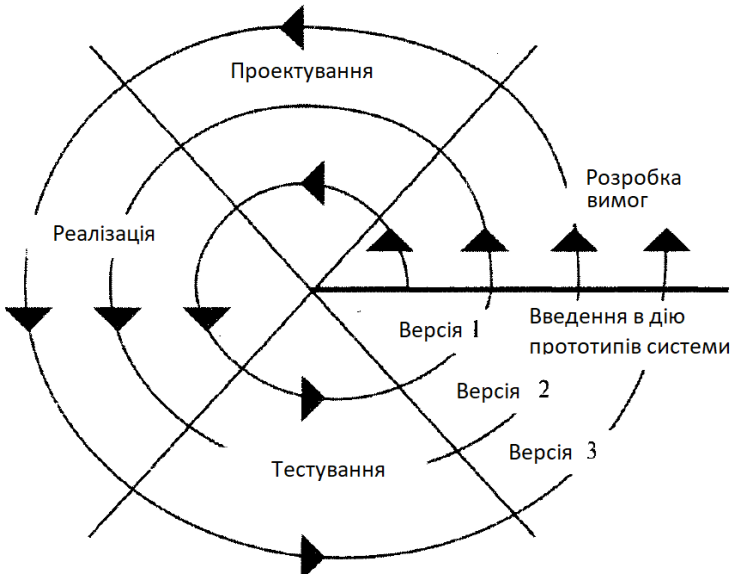


Рис. 2.3. Спіральна модель ЖЦ ПЗ

Серед *особливостей спіральної моделі* можна виділити такі: 1) відмова від фіксації вимог і призначення пріоритетів вимогам користувачів; 2) розроблення послідовності прототипів, починаючи з вимог найвищого пріоритету; 3) ідентифікація та аналіз ризику на кожній ітерації; 4) використання каскадної

моделі для реалізації остаточного прототипу; 5) оцінювання результатів по завершенні кожної ітерації і планування наступної ітерації.

При використанні спіральної моделі ПЗ створюється за кілька ітерацій (витків спіралі) методом прототипування (реалізації технічних рішень, коли ступінь задоволення потреб замовника перевіряється шляхом створення прототипів). Під **прототипом** розуміють чинний програмний компонент, який реалізує окремі функції та зовнішні інтерфейси ПЗ.

Створення прототипів (фрагменту або версії системи) здійснюється в кілька ітерацій. Кожна ітерація відповідає створенню фрагмента або версії ПЗ, на ній уточнюються цілі і характеристики проекту, оцінюється якість отриманих результатів і плануються роботи наступної ітерації. На кожній ітерації виконують ретельну оцінку ризику перевищення термінів і вартості проекту, щоб визначити необхідність виконання ще однієї ітерації, ступінь повноти і точності розуміння вимог до системи, а також доцільність припинення проекту.

Спіральна модель позбавляє користувачів і розробників ПЗ від необхідності повного і точного формулювання вимог до системи на початковій стадії, оскільки вони уточнюються на кожній ітерації. Таким чином, поглиблюються і послідовно конкретизуються деталі проекту, і в результаті вибирається обгрунтований варіант, який доводиться до реалізації.

Розробка ітераціями відображає об'єктивно існуючий спіральний цикл створення системи. Неповне завершення робіт на кожній стадії дозволяє переходити на наступну стадію, не чекаючи повного завершення работи на поточній (при цьому відсутню роботу можна виконати на наступній ітерації). Головним завданням є якомога швидше показати користувачам системи працездатний продукт, тим самим активуючи процес уточнення і доповнення вимог.

Серед *переваг спіральної моделі* можна виділити такі: 1) прискорення розробки (раннє отримання результату за рахунок прототипування); 2) постійна

участь замовника в процесі розробки; 3) розбиття великого обсягу робіт на невеликі частини; 4) зниження ризику (підвищення ймовірності передбачуваної поведінки системи). Спіральна модель не виключає використання каскадного підходу на завершальних стадіях проекту, коли вимоги до системи є повністю визначеними. До *недоліків* можна віднести: 1) складність планування (визначення кількості і тривалості ітерацій, оцінки витрат і ризиків); 2) складність застосування моделі з точки зору менеджерів і замовників (через звичку до чіткого і детального планування); 3) напружений режим роботи для розробників (при короткострокових ітераціях).

Основна проблема спірального циклу – визначення моменту переходу на наступну стадію. Для її вирішення необхідно ввести часові обмеження на кожну зі стадій життєвого циклу. Перехід здійснюється відповідно до плану, навіть якщо не вся запланована раніше робота закінчена. План складається на основі статистичних даних, отриманих в попередніх проектах, і особистого досвіду розробників.

При використанні ітераційної моделі існує ризик впасти в іншу (по відношенню до каскадної моделі) крайність. Розглянемо її на **прикладі *схеми «швидкого макетування»*.**

Розробники обговорюють вимоги до проекту із замовником. Потім протягом короткого проміжку часу, від чотирьох до шести тижнів, на основі розуміння цих вимог створюється прототип системи. Розробники разом із замовником аналізують його роботу. Замовник може виявити, що для задоволення реальних потреб прототип необхідно модифікувати. Виконавши оцінку прототипу, розробники отримують можливість уточнити вимоги, які пред'являються до системи, шляхом деталізації вхідних параметрів. Наприклад, замовник може сказати, що необхідно змінити інтерфейс або що звіти, створювані програмою, мають неправильний формат. На основі вхідних параметрів протягом кількох тижнів проводиться коригування прототипу, усуваються помилки і додаються певні функції. Отримане ПЗ знову

перевіряють разом із замовником. Процес триває, поки замовник не погодиться з тим, що продукт задовольняє вимогам.

Всі зусилля, спрямовані на задоволення потреб замовника, є позитивними. Така схема має такі додаткові переваги:

1. *Продуктивність праці колективу дуже висока*. Розробники витрачають час не на створення великої кількості специфікацій, а на розробку ПЗ.

2. *Взаємозв'язки із замовником є конструктивними*. Замовник може не мати чіткого уявлення про те, що треба, поки не розпочнеться процес розробки. Тому прихильники цього процесу можуть сказати: «Навіщо складати специфікації? Вимоги стануть зрозумілими в процесі роботи».

Хоча процес швидкого макетування має певні переваги, його застосованість обмежується такими *недоліками*.

1. *При швидкому макетування важко привести проект до завершальної фази*. Через наявність постійного зворотного зв'язку із замовником умова завершення проекту може ніколи не бути досягнутою. У розробників і замовника можуть виникати ідеї щодо поліпшення кожної з ітерацій. Постійне вдосконалення призводить до додавання або модифікації існуючих вимог, і процес може вийти з–під контролю. Замовник ніколи не буде задоволений повністю, і проект не завершиться. При цьому керівнику проекту потрібно докласти величезні зусилля для його завершення.

2. *Проект на основі методу швидкого макетування складно планувати і фінансувати*. Це положення пов'язано із попереднім: якщо проект важко завершити, то важко скласти графік його виконання та кошторис витрат (не можна передбачити, який час потрібен для завершення проекту).

3. *Метод швидкого макетування непридатний для розробки ПЗ великим колективом розробників*. Цей метод є гарним для невеликої групи розробників, які працюють тільки з одним замовником. Важче застосувати його до розробки великих систем із залученням сотен розробників.

4. *В результаті швидкого макетування можна нічого не отримати, окрім прототипу системи*. Метод швидкого макетування спрямований в основному на досягнення заданої функціональності. Код може володіти необхідними характеристиками та інтерфейсами, але він ніколи не стане придатним для широкого застосування.

Швидке макетування має багато спільного з підходом швидкого розроблення ПЗ, і так само має обмежене застосування.

Природний розвиток каскадної і спіральної моделей призвів до їх зближення і появи сучасного ітераційного підходу, який по суті є раціональним поєднанням цих моделей. Різні варіанти ітераційного підходу реалізовані в більшості сучасних технологій і методів: Rational Unified Process (RUP), Microsoft Solutions Framework (MSF), XP.

Невирішені питання й помилки, допущені на етапах аналізу і проектування ІС, породжують на подальших етапах складні, часто нерозв'язні проблеми і, нарешті, призводять до провалу всього проекту. Головна особливість сучасної індустрії замовлених ІС полягає в концентрації зусиль на двох початкових етапах її ЖЦ – аналізі та проектуванні при досить невисокій складності і трудовитратах на подальших етапах.

5. Створення інформаційної системи на основі каскадної моделі

Стадії та етапи створення ІС. [5 с. 46–54, 7 с. 33–45]. В стандарті ГОСТ 34.601–90 описано стадії та етапи проектування ІС на основі використання каскадної моделі життєвого циклу ІС. Залежно від складності об'єкта автоматизації і набору завдань, які потребують вирішення при створенні конкретної ІС, стадії та етапи робіт можуть мати різну трудомісткість. На будь–якій стадії проекту допускається: 1) об'єднувати послідовні етапи і вилучати деякі з них; 2) розпочинати виконання робіт наступної стадії до закінчення попередньої.

Стадії та етапи створення ІС, які виконуються організаціями–учасниками, прописуються в договорах і технічних завданнях на виконання робіт:

Стадія 1. Формування вимог до ІС. На початковій стадії проектування виділяють такі етапи робіт: 1) обстеження об'єкта інформатизації та обґрунтування необхідності створення ІС; 2) формування вимог користувачів до ІС; 3) оформлення звіту про виконану роботу і тактико–технічне завдання на розробку.

Стадія 2. Розробка концепції ІС. На цій стадії виділяють такі етапи робіт: 1) вивчення об'єкта автоматизації; 2) проведення необхідних науково–дослідних робіт; 3) розробка варіантів концепції ІС, які відповідають вимогам користувачів; 4) оформлення звіту та затвердження концепції.

Стадія 3. Технічне завдання. На цій стадії виділяють такі етапи робіт: розробка та затвердження технічного завдання на створення ІС.

Стадія 4. Ескізний проект. На цій стадії виділяють такі етапи робіт: 1) розробка попередніх проектних рішень по системі та по її частинам; 2) розробка ескізної документації на ІС та її частини.

Стадія 5. Технічний проект. На цій стадії виділяють такі етапи робіт: 1) розробка проектних рішень по системі та її частинам; 2) розробка документації на ІС та її частини; 3) розробка та оформлення документації на поставку комплектуючих виробів; 4) розробка завдань на проектування в суміжних частинах проекту.

Стадія 6. Робоча документація. На цій стадії виділяють такі етапи робіт: 1) розробка робочої документації на ІС та її частини; 2) розробка та адаптація програм.

Стадія 7. Введення в дію. На цій стадії виділяють такі етапи робіт: 1)підготовка об'єкта автоматизації; 2) підготовка персоналу; 3) комплектація ІС виконується програмними і технічними засобами, програмно–технічними комплексами, інформаційними виробами; 4) будівельно–монтажні роботи;

5)пуско–налагоджувальні роботи; 6) проведення попередніх випробувань та дослідної експлуатації; 7) проведення приймальних випробувань.

Стадія 8. Супровід ІС. На цій стадії виділяють такі етапи робіт: 1) виконання робіт відповідно до гарантійних зобов'язань; 2) післягарантійне обслуговування.

Обстеження об'єкта інформатизації (на прикладі організаційної структури підприємства). *Обстеження* охоплює вивчення і діагностичний аналіз організаційної структури підприємства, його діяльності та існуючої системи обробки інформації. Матеріали, отримані в результаті обстеження, використовуються для: 1) обґрунтування розробки та поетапного впровадження систем; 2) складання технічного завдання на розробку систем; 3) розробки технічного і робочого проектів систем. На етапі обстеження доцільно виділити дві складові: 1) етап визначення стратегії впровадження ІС, 2) етап детального аналізу діяльності організації.

1. *Основним завданням першого етапу обстеження* – етапу визначення стратегії – є оцінювання реального обсягу проекту, його цілей і завдань на основі виявлених функцій та інформаційних елементів об'єкта, що автоматизується. Ці завдання можуть бути реалізовані або замовником ІС самостійно, або із залученням консалтингових організацій. Етап передбачає взаємодію із основними потенційними користувачами системи та бізнес–експертами. Основним завданням взаємодії є отримати повне та однозначне розуміння вимог замовника (потрібну інформацію можна отримати в результаті інтерв'ю, бесід або семінарів із керівництвом, експертами і користувачами). По завершенні цієї стадії обстеження з'являється можливість визначити ймовірні технічні підходи до створення системи та оцінити витрати на її реалізацію (витрати на апаратне забезпечення, яке закуповується, ПЗ та розробку нового ПЗ).

Результатом етапу визначення стратегії є документ (*техніко–економічне обґрунтування проекту*), де чітко сформульовано, що отримає замовник, якщо

погодиться фінансувати проект, коли він отримає готовий продукт (графік виконання робіт) та скільки це буде коштувати (для великих проектів повинен бути складений графік фінансування на різних етапах робіт). У документі бажано відобразити не тільки витрати, але і вигоду проекту, наприклад час окупності проекту, очікуваний економічний ефект (якщо його вдається оцінити). Орієнтовний зміст цього документа:

– обмеження, ризики, критичні фактори, які можуть вплинути на успішність проекту;

– сукупність умов, при яких передбачається експлуатувати майбутню систему: 1) архітектура системи, апаратні і програмні ресурси, умови функціонування, 2) обслуговуючий персонал і користувачі системи;

– терміни завершення окремих етапів, форма приймання / здачі робіт, які залучаються ресурси, заходи щодо захисту інформації;

– опис виконуваних системою функцій;

– можливості розвитку системи;

– інформаційні об'єкти системи;

– інтерфейси і розподіл функцій між людиною і системою;

– вимоги до програмних та інформаційних компонентів ПЗ, вимоги до СУБД;

– що не буде реалізовано в рамках проекту.

2. *На етапі детального аналізу діяльності організації* вивчаються завдання, які забезпечують реалізацію функцій управління, організаційна структура, штат співробітників і зміст робіт по управлінню підприємством, а також характер підпорядкованості вищим органам управління. На цьому етапі повинні бути виявлені: 1) інструктивно–методичні та директивні матеріали, на підставі яких визначаються склад підсистем і перелік завдань; 2) можливості застосування нових методів вирішення завдань.

Аналітики збирають і фіксують інформацію у двох взаємопов'язаних формах: 1) функції – інформація про події та процеси, які відбуваються в

бізнесі; 2) сутності – інформація про поняття, які мають значення для організації і про які щось відомо.

При вивченні кожного функціонального завдання управління визначаються такі фактори:

– найменування завдання; терміни і періодичність його вирішення;

– ступінь формалізуємості завдання;

– джерела інформації, необхідні для вирішення завдання;

– показники та їх кількісні характеристики;

– порядок коригування інформації;

– діючі алгоритми розрахунку показників і можливі методи контролю;

– ефективні засоби збору, передачі та обробки інформації;

– ефективні засоби зв'язку;

– прийнята точність розв’язання задачі;

– трудомісткість розв’язання задачі;

– діючі форми подання вихідних даних і результатів їх обробки у вигляді документів;

– споживачі отриманих результатів по завданню.

Однією з найбільш трудомістких, хоча і добре формалізованих задач цього етапу, є *опис документообігу організації*. При обстеженні документообігу складається схема маршруту руху документів. За результатами обстеження встановлюється перелік завдань управління, вирішення яких доцільно автоматизувати, і черговість їх розробки.

На етапі обстеження слід класифікувати заплановані функції системи за ступенем важливості. Один із можливих форматів подання такої класифікації – *MuSCoW*: необхідні функції – Must have; бажані функції – Should have; можливі функції – Could have; відсутні функції – Will not have. Функції першої категорії забезпечують можливості для успішної роботи системи. Реалізація функцій другої і третьої категорій обмежується часовими і фінансовими рамками (розробляється те, що необхідно, і максимально

можливе). Четверта категорія функцій фіксує межу проекту і набір функцій, які будуть відсутні в системі.

Моделі діяльності організації створюються у вигляді: 1) моделі «як є», яка відображає існуючі в організації бізнес–процеси; 2) моделі «як повинно бути», яка відображає необхідні зміни бізнес–процесів з урахуванням впровадження ІС.

На етапі аналізу необхідно залучати до роботи групи тестування для вирішення таких завдань: 1) отримання порівняльних характеристик, передбачуваних до використання апаратних платформ, операційних систем, СУБД, іншого оточення; 2) розроблення плану робіт по забезпеченню надійності ІС та її тестування. Час на тестування системи і на виправлення виявлених помилок слід передбачати не тільки на етапі розробки, але і на етапі проектування. Для автоматизації тестування слід використовувати системи стеження за недоліками. Це дозволяє мати єдине сховище помилок, відстежувати їх повторну появу, контролювати швидкість та ефективність виправлення помилок, бачити найбільш нестабільні компоненти системи, а також підтримувати зв'язок між групою розробників і групою тестування. Чим більше проект, тим сильніше потреба у стеженні за недоліками.

Технічне завдання на ІС. Результати обстеження є об'єктивною основою для формування технічного завдання на ІС. *Технічне завдання* на ІС – це документ, який визначає цілі, вимоги та основні вихідні дані, необхідні для розробки системи. При його розробці необхідно вирішити такі завдання: 1) встановити спільну мету створення ІС, визначити склад підсистем і функціональних завдань; 2) розробити та обґрунтувати вимоги, які пред'являють до підсистем; 3) розробити та обґрунтувати вимоги до інформаційної бази, математичного та програмного забезпечення, комплексу технічних засобів (включаючи засоби зв'язку та передачі даних); 4) встановити загальні вимоги до проектної системи; 5)визначити перелік завдань створення системи і виконавців; 6) визначити етапи створення системи і терміни їх

виконання; 7) провести попередній розрахунок витрат на створення системи і визначити рівень економічної ефективності її впровадження. Типові вимоги до складу та змісту технічного завдання наведені в табл. 2.3.

Таблиця 2.3

Склад і зміст технічного завдання (ГОСТ 34.602–89)

№ з\п	Розділ	Зміст
1	Загальні відомості	– повна назва системи та її умовне позначення; – шифр теми або шифр (номер) договору; – найменування підприємств розробника і замовника системи, їх реквізити; – перелік документів, на підставі яких створюється ІС; – планові терміни початку і закінчення робіт; – відомості про джерела і порядок фінансування робіт; – порядок оформлення і подання замовнику результатів робіт зі створення системи, її частин та окремих засобів
2	Призначення і цілі створення (розвитку) системи	– вид діяльності, яка автоматизується; – перелік об'єктів, на яких передбачається використання системи; – найменування і необхідні значення технічних, технологічних, виробничо–економічних та інших показників об'єкта, які повинні бути досягнуті при впровадженні ІС
3	Характерис–тика об'єктів автоматиза–ції	– короткі відомості про об'єкт автоматизації; – відомості про умови експлуатації і характеристики навколишнього середовища;
4	Вимоги до системи	Вимоги до системи в цілому: – вимоги до структури і функціонування системи (перелік підсистем, рівні ієрархії, ступінь централізації, способи інформаційного обміну, режими функціонування, взаємодія із суміжними системами, перспективи розвитку системи); – вимоги до персоналу (чисельність користувачів, кваліфікація, режим роботи, порядок підготовки); – показники призначення (ступінь пристосованості системи до змін процесів управління і значень параметрів). Вимоги до функцій (по підсистемах): – перелік завдань, які підлягають автоматизації; – часовий регламент реалізації кожної функції; – вимоги до якості реалізації кожної функції, до форми подання вихідної інформації, характеристики точності, достовірності видачі результатів; – перелік і критерії відмов.

4	Вимоги до системи	Вимоги до видів забезпечення: – математичного (склад і область застосування мат. моделей і методів, типових і розроблюваних алгоритмів); – інформаційного (склад, структура та організація даних, обмін даними між компонентами системи, інформаційна сумісність із суміжними системами, які використовуються класифікатори, СУБД, контроль даних і ведення інформаційних масивів, процедури надання юридичної сили вихідних документів); – лінгвістичного (мови програмування, мови взаємодії користувачів із системою, системи кодування, мови введення–виведення); – програмного (незалежність програмних засобів від платформи, якість програмних засобів і способи його контролю, використання фондів алгоритмів і програм); – технічного, метрологічного, – організаційного (структура і функції підрозділів, які експлуатують, захист від помилкових дій персоналу), тодичного (склад нормативно–технічної документації)
5	Склад і зміст робіт зі створення системи	– перелік стадій та етапів робіт; терміни виконання; – склад організацій – виконавців робіт; – вид і порядок експертизи технічної документації; – програма забезпечення надійності; – програма метрологічного забезпечення
6	Порядок контролю і приймання ІС	– види, склад, обсяг і методи випробувань системи; – загальні вимоги до приймання робіт за стадіями; – статус приймальної комісії;
7	Вимоги до складу та змісту робіт з підготовки об'єкта авто–матизації до введення системи у дію	– перетворення вхідної інформації до машино–читаємого вигляду; – зміни в об'єкті автоматизації; – терміни і порядок комплектування, навчання персоналу
8	Вимоги до документу–вання	– перелік документів, які підлягають розробці; – перелік документів на машинних носіях
9	Джерела розробки	документи та інформаційні матеріали, на підставі яких розробляється ТЗ і система

Ескізний проект. Ескізний проект передбачає розробку попередніх проектних рішень щодо системи та її частин. Виконання ескізного проектування не є обов'язковим: якщо основні проектні рішення визначені

раніше або є очевидними для конкретної ІС та об'єкта автоматизації, то ця стадія може бути виключена із загальної послідовності робіт.

Зміст ескізного проекту задається в ТЗ. Зазвичай, на етапі ескізного проектування визначають: 1) функції ІС та її підсистем, їх цілі та очікуваний ефект від впровадження; 2) склад комплексів задач та окремих завдань; 3) концепцію інформаційної бази та її структуру; 4) функції СУБД; 5) склад обчислювальної системи та інших технічних засобів; 6) функції і параметри основних ПЗ. За результатами виконаної роботи оформляють, узгоджують і затверджують документацію в обсязі, необхідному для опису прийнятих проектних рішень і достатньому для подальшого виконання робіт зі створення ІС.

Технічний проект ІС. На основі технічного завдання (та ескізного проекту) розробляється технічний проект ІС. *Технічний проект* має вигляд технічної документації, яка містить загальносистемні проектні рішення, алгоритми розв’язання задач, а також оцінку економічної ефективності автоматизованої системи управління і перелік заходів з підготовки об'єкта до впровадження. На цьому етапі здійснюється комплекс науково–дослідних та експериментальних робіт для вибору основних проектних рішень та розрахунок економічної ефективності системи. Склад і зміст технічного проекту наведені в табл. 2.4.

Під час завершення стадії технічного проектування виконуються такі дії: 1) розроблення документації на поставку виробів для комплектування ІС, які серійно виготовляються, 2) визначення технічних вимог; складання ТЗ на розробку виробів, які не виготовлялися серійно.

На стадії «*робоча документація*» здійснюється створення програмного продукту і розробка всієї супровідної документації. Документація повинна містити всі необхідні і достатні відомості для забезпечення виконання робіт по введенню ІС в дію та її експлуатації, а також для підтримки рівня експлуатаційних характеристик (якості) системи.

Таблиця 2.4

Зміст технічного проекту

№ з/п	Розділ	Зміст
1	Пояснювальна записка	– підстава для розробки сістеми, – перелік організацій–розробників, – коротка характеристика об'єкта із зазначенням основних техніко–економічних показників його функціонування і зв'язків з іншими об'єктами, – короткі відомості про основні проектні рішення по функціональній і забезпечуючій частинам системи
2	Функціональна та організаційна структура системи	– обгрунтування виділених підсистем, їх перелік та призначення, – перелік завдань, які вирішуються в кожній підсистемі, з короткою характеристикою їх змісту; – схема інформаційних зв'язків між підсистемами і між завданнями в рамках кожної підсистеми
3	Постановка задач та алгоритми рішень	– організаційно–економічна сутність задачі (назва, мета рішення, короткий зміст методу, періодичність і час виконання завдання, способи збору і передачі даних, зв'язок задачі з іншими задачами, характер використання результатів рішення), – економіко–математична модель завдання (структурна і розгорнута форма подання), – вхідна оперативна інформація (характеристика показників, діапазон зміни, форми подання), – нормативно–довідкова інформація (зміст і форми подання); інформація, яка зберігається для зв'язку з іншими завданнями, – інформація, яка накопичується для наступних варіантів розв'язання цієї задачі, – інформація щодо внесення змін (система внесення змін і перелік інформації, яка піддається змінам); – алгоритм вирішення задачі (послідовність етапів розрахунку, схема, розрахункові формули); – контрольний приклад (набір заповнених даними форм вхідних документів, умовні документи з накопичуваною і збереженою інформацією, форми вихідних документів, заповнені за результатами вирішення економіко–технічної задачі і відповідно до розробленого алгоритму розрахунку)

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

Документація повинна містити всі необхідні і достатні відомості для забезпечення виконання робіт по введенню ІС в дію та її експлуатації, а також для підтримки рівня експлуатаційних характеристик (якості) системи. Розроблена документація повинна бути відповідним чином оформлена, узгоджена і затверджена.

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

Залежно від взаємозв'язків частин ІС та об'єкта автоматизації *випробування* можуть бути автономні або комплексні. Автономні випробування охоплюють частини системи. Їх проводять у міру готовності частин системи до здачі в дослідну експлуатацію. Комплексні випробування проводять для груп взаємопов'язаних частин або для системи в цілому. Для планування проведення всіх видів випробувань розробляється документ «Програма та методика випробувань». Розробник документа встановлюється в договорі або ТЗ. Як додаток до документа можуть включатися тести або контрольні приклади.

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

Дослідну експлуатацію ІС проводять для визначення: 1) фактичних значень кількісних та якісних характеристик системи і готовності персоналу до роботи в умовах її функціонування, 2) ефективності системи і коригування, при необхідності, документації.

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

6. Управління життєвим циклом програмних продуктів

Існують різні підходи до управління ЖЦ ІТ (в тому числі і ПЗ). Спочатку визначаються три основні фази життєвого циклу у вигляді таких видів діяльності: етап 1) планування, 2) розробки, 3) експлуатації.

На етапі планування. На цьому етапі визначається пріоритет ІТ порівняно з іншими можливостями для бізнес–інвестицій, визначаються нові ІТ–проекти та пріоритети між окремими ІТ проектами. Також аналізуються дані про поточний стан операційних систем, що є важливим вкладом у цю діяльність. Як добре виконують операційні системи свої функції, оцінюється якість та розраховується оціночний ефект від поліпшень На цьому етапі визначаються моделі ЖЦ та особливості їхнього застосування.

На етапі розробки реалізуються пріоритетні пропозиції, будується і апробується ІТ. Розроблення означає реалізацію списку пріоритетних пропозицій. Для цього потрібні певні ресурси розробників, які досконало володіють будь–якими технологіями та добре орієнтуються в бізнес–сфері користувачів (останні визначають, що їм потрібно). Для відстеження витрат і переваг розробляємих ІС слід керувати такими заходами:

1. Управління витратами шляхом активного управління ресурсами.

2. Управління вигодами через активне управління функціональністю ІТ, які розробляються.

3. Управління графіком часу.

На цьому етапі згодом розробляються пріоритетні пропозиції щодо побудови, тестування та впровадження.

На етапі експлуатації інформаційні системи знаходяться в стадії функціонування та підтримуються виробником.

Усі три етапи мають свої типові проблеми з витратами та вигодами для того, щоб налагодити їх ефективне управління.

Управління ЖЦ програмного продукту може бути спрямовано на реалізацію різних переваг, зокрема: 1) *зниження ризику*: зменшення обсягу непотрібної та неактуальної інформації, створення інформації, якою легше керувати та виявляти; 2) *економія витрат*: витрати на зберігання та правове утримання можна зменшити за рахунок кращого управління інформацією; 3)*більш ефективне управління*: бізнес може запровадити чіткість управління та контроль.

Стандарт COBIT 4.1. Для управління ЖЦ IT використовують стандарт COBIT (Control Objectives for Information and Related Technology – «Контрольні цілі для інформаційних та суміжних технологій»). COBIT – відкритий IT–стандарт, який містить документи зі стандартами щодо оптимізації управління IT: аудитом IT та IT–безпекою; підхід до управління IT, створений в 1992 р. Асоціацією контролю та аудиту систем (Information Systems Audit and Control Association – ISACA) та Інститутом управління IT (IT Governance Institute, ITGI). Він надає менеджерам, аудиторам і користувачам IT набір затверджених метрик, процесів і найкращих практик щоб допомогти їм в отриманні максимальної вигоди від використання IT, для розробки відповідного керівництва і контролю IT в компанії.

Перша редакція COBIT побачила світ в 1996 р., а версія COBIT 4.1 – у 2007 р. В основі COBIT 4.1 лежить процесний підхід, система збалансованих показників BSC, модель зрілості SEI CMM / CMMI, PMBoK (методологія проектного управління), а також підходи стандартів PRINCE2, TickIT, ITIL®.

Основними напрямками управління ЖЦ IT є пов’язані з його етапами процеси:

1) *відповідна стратегія* робить акцент на зв'язках між планами бізнесу та IT: відкриття, підтримка та контроль за пропозиціями щодо ціни IT; відповідність IT і бізнес–операцій;

2) *корисність*: передбачає реалізацію цінових пропозицій, контроль за тим, щоб ІТ забезпечив конкретні стратегічні переваги ресурсів, зосередження на оптимізації витрат і підтвердження об'єктивної цінності ІТ;

3) *управління ресурсами*: пов'язує питання, пов'язані з управлінням критичними ІТ ресурсами, а саме, оптимізацією інвестицій та довгостроким керівництвом додатками, інформацією, інфраструктурою та персоналом (ключові питання задаються оптимізацією значень та інфраструктури);

4) *управління ризиками*: вимагає обізнаності вищого керівництва в області ризиків, розуміння при цьому корпоративного підходу, необхідності вимагати прозорості у виокремленні важливих ризиків, включаючи функції та системи управління;

5) *оцінювання ефективності*: передбачає контроль за реалізацією стратегій, результатами проектів, використаними ресурсами, ефективними процесами та сервісним обслуговуванням (для цього використовують, наприклад, систему збалансованих показників).

Концепція COBIT запропонувала побудовані механізми управління ЖЦ ІТ, виходячи з того, яка інформація необхідна для досягнення бізнес–целей (при цьому інформація розглядається як результат використання ІТ–ресурсів, управління якими виконується в рамках ІТ–процесів).

ІТ–ресурси містять у собі застосунки, інформацію (дані в будь–якій формі), інфраструктуру, персонал.

Інформація про бізнес повинна задовільняти визначеним критериям, які в стандартах COBIT називають бізнес–вимоги до інформації. Виявляють такі *бізнес–вимоги до інформації* або інформаційні критерии: ефективність, раціональність, конфіденційність, цінність, доступність, відповідність нормам і надійність інформації. Механізми управління включають в себе політичні, організаційні структури, процедури та регламенти. Задача управління ЖЦ ІТ включає формулювання бажаних результатів або цілей, які повинні бути

досягнуті шляхом реалізації механізмів управління в рамках конкретного IT–процесу.

Концептуальний стандарт COBIT 4.1 сформовано із 34 високорівневих процесів (які охоплюють порядка 200 цілей контролю), згруповані у 4 домени, які містять інформацію про діяльність:

Планування та організація: включає в себе стратегію та тактику, а також визначає засоби найефективнішого використання IT для досягнення бізнес–цілей. Регламентовані процеси:

PO1 Розробка стратегічного плану

PO2 Визначення IT архітектури

PO3 Визначення напрямків розвитку технологій

PO4 Формалізація IT–процесів, організації та взаємозв'язку з бізнесом

PO5 Управління інвестиціями в IT

PO6 Узгоджене управління цілями та завданнями

PO7 Управління IT персоналом

PO8 Управління якістю

PO9 Оцінка та управління ризиками IT

PO10 Управління проектами

Придбання та впровадження: для реалізації IT–стратегій необхідно ідентифікувати, розробити або приєднати відповідні IT–рішення, які повинні бути внесеними та інтегрованими у бізнес–процеси, а також подані в інформаційних системах. Регламентовані процеси:

AI1 Ідентифікація та вибір рішення щодо автоматизації

AI2 Проектування та розробка запропонованих пропозицій

AI3 Проектування та підтримка технічної інфраструктури

AI4 Забезпечення роботи та використання ІС

AI5 Закупка IT ресурсів

AI6 Управління змінами

AI7 Встановлення та підтвердження рішень та змін

Подання та підтримка: передбачає наявність необхідних інформаційних служб, які забезпечують безпеку та постійне ведення бізнесу, навчання, а також обробку даних прикладними системами. Регламентовані процеси:

DS1 Визначення та управління рівнями обслуговування

DS2 Управління сервісами підрядчиків

DS3 Управління продуктивністю та потужністю

DS4 Забезпечення безперервності сервісів

DS5 Забезпечення безпеки системи

DS6 Визначення і розподіл IT витрат

DS7 Нвчання користувачів

DS8 Управління службами підтримки та інцидентами

DS9 Управління конфігурацією

DS10 Управління проблемами

DS11 Управління даними

DS12 Управління фізичним обладнанням

DS13 Управління експлуатацією

Моніторинг та оцінка: якість та відповідність IT–процесів, необхідних для контролю, повинні оцінюватися на регулярній основі. Цей домен включає в себе нагляд керівництва з управління процесами в організаціях, а також незалежний контроль внутрішніми і зовнішніми аудиторами. Регламентовані процеси:

ME1 Відстеження та оцінювання процесу виробництва IT

ME2 Відстеження та оцінювання внутрішнього контролю

ME3 Гарантування відповідності регулюючих вимог

ME4 Забезпечення управління ЖЦ IT

Домени співпадають з традиційними сферами відповідальності IT: планування, впровадження, експлуатація та моніторинг. Така структура охоплює усі аспекти управління та використання IT. Виконання всіх 34

високоякісних процесів може гарантувати власникам бізнес–процесів, що система управління IT є адекватною задачею бізнесу.

В стандарті COBIT детально описані цілі та принципи управління, об'єкти управління, чітко визначені всі IT–процеси (для кожного процесу визначено вхід–вихід, виконавці та відповідальні, об'єкти контролю та метрики), вимоги до них, описані можливі інструменти їх реалізація. У описі IT–процесів також подано практичні рекомендації з управління IT безпекою. COBIT застосовується для контролю та аудиту існуючої системи управління IT, організації оперативного та стратегічного керівництві IT, аналізу витрат на IT–проекти та підтримку відповідної інфраструктури, відповідність вимогам необхідних стандартних та регулюючих організацій (таких як SOX та COSO).

За допомогою використання стандартів COBIT керівники IT підрозділів переробляють завдання бізнесу в чіткі та зрозумілі плани розвитку IT. Основною перевагою стандарту COBIT є його повна та практична рекомендація, інструменти, за допомогою яких можна побудувати систему управління корпорацією, яка працює на основі IT. Таким чином, при використанні методології COBIT інформаційна система, яка створюється виходить з вимог бізнесу та використовує ефективні економічні ресурси, а також ефективні стратегії використання цих ресурсів. Іншими словами, стандарт COBIT описує бізнес–орієнтований підхід для створення інформаційного середовища: IT розглядають у вигляді інструментального бізнесу, а стандарт визначає принципи побудови та організації роботи IT–департаменту.

Підсумок. 1. Розробка великих проектів неможлива без нормативно–методичних документів, які регламентують різні аспекти процесів діяльності розробників.

2. Одним з базових понять програмної інженерії є поняття ЖЦ ПЗ. *Життєвий цикл ПЗ* визначають як період часу, який розпочинається з моменту

прийняття рішення про необхідність створення ПЗ і закінчується в момент його повного вилучення з експлуатації.

3. Під *моделлю* ЖЦ ПЗ розуміють структуру, яка визначає послідовність виконання та взаємозв'язки процесів, дій і завдань на протязі ЖЦ. Найбільш поширеними моделями є каскадна та ітераційна.

4. Зрілість процесів ЖЦ ПЗ – це ступінь їх керованості, контрольованості та ефективності. Підвищення технологічної зрілості означає можливість зростання стійкості процесів, вказує на ступінь ефективності та узгодженості використання різних процесів на основі ПЗ в рамках всієї організації.

Запитання для самоконтролю

1. Що таке життєвий цикл програмного забезпечення?

2. Чим регламентується ЖЦ ПЗ?

3. Назвіть групи процесів, які входять до складу ЖЦ ПЗ, які процеси входять до складу кожної групи? Які з процесів, на вашу думку, найчастіше використовують в реальних проектах, які в меншій мірі і чому?

4. Які стадії входять в процес створення ПЗ?

5. Яке співвідношення між стадіями і процесами ЖЦ ПЗ?

6. Назвіть: 1) стадії та етапи процесу проектування ІС на основі каскадної моделі; 2) принципові особливості каскадної моделі. У чому полягають переваги та недоліки каскадної моделі?

7. Які принципові особливості ітераційної моделі? У чому полягають переваги та недоліки ітераційної моделі?

8. Яким чином можна домогтися підвищення рівня зрілості процесів створення ПЗ?

9. Яку роль у підвищенні рівня зрілості грають процеси управління вимогами та управління конфігурацією ПЗ?

10. Назвіть складові технічного завдання на ІС.

3. АРХІТЕКТУРА ІНФОРМАЦІЙНОЇ СИСТЕМИ

1. Поняття «архітектура інформаційної системи»

Рівень розвитку сучасних технологій настільки високий, що дозволяє побудувати ІС будь–якої складності та функціональності. Проте, враховуючи вимоги бізнесу, грунтовані на показниках різних бізнес–оцінок, виникають задачі, розв’язання яких зводиться до забезпечення раціонального підходу до процесу проектування, реалізації та подальшої експлуатації ІС. Вибрану архітектуру можна вважати одним із основних показників ефективності створюваної ІС.

Напочатку термін архітектура застосовувався у проектуванні та при побудові різних споруд. Він визначав їх структуру, взаємозв'язки між складовими частинами, базові принципи організації і подальшого розвитку.

Нині існує різноманіття трактувань терміну «архітектура». Деякі вчені під **архітектурою** розуміють: 1) набір основних правил, які визначають організацію системи; 2) сукупність структурних елементів системи і зв'язків між ними; 3) поведінку елементів системи в процесі їх взаємодії; 4) ієрархію підсистем, яка об'єднує структурні елементи; 5) архітектурний стиль (використовувані методи і засоби опису архітектури, а також архітектурні зразки); 6) формальний опис системи, або детальний план системи на рівні компонентів і методології їх реалізації. Визначити поняття «архітектура інформаційної системи» можна різними способами, що пов'язано із відсутністю загальноприйнятого визначення поняття «ІС».

Відповідно до стандарту ANSI/IEEE 1471–2000, загальноприйнятим є таке визначення *архітектури системи* – це опис організації системи в термінах компонентів, їх взаємозв'язків між собою і з довкіллям, принципи управління їх розробкою і розвитком. *Архітектуру* ІС можна описати як концепцію, яка визначає модель, структуру, виконувані функції та взаємозв'язок компонентів.

Архітектура є багатовимірною, оскільки різні фахівці працюють з різними її аспектами. Наприклад, при будівництві будівлі використовують різні типи креслень, які зображують різні аспекти архітектури: плани поверхів; вертикальні проекції; плани монтажу кабельної проводки; креслення водопроводів, системи центрального опалення та вентиляції; вигляд будівлі на фоні місцевості (ескізи).

Архітектура ПЗ також передбачає різні подання, які служать різним цілям: 1) зображенню функціональних можливостей системи; 2) відображенню логічної організації системи; 3) опису фізичної структури програмних компонентів в середовищі реалізації; 4) відображенню структури потоків управління та аспектів паралельної роботи; 5) опису фізичного розміщення програмних компонентів на базовій платформі.

Зображення архітектури – це спрощений опис (абстракція) системи з конкретної точки зору, який охоплює певне коло інтересів та опускає об'єкти, несуттєві із заданої точки зору. Воно концентрує увагу лише на елементах, які є значущими з точки зору архітектури.

Архітектурно–значущий елемент – це елемент, який має значний вплив на структуру системи та її продуктивність, надійність і можливість розвитку. Наприклад, до складу архітектурно–значущих елементів об'єктно–орієнтованої архітектури входять основні класи предметної області, підсистеми та їх інтерфейси, основні процеси або потоки управління.

Розробка моделі архітектури системи ПЗ необхідна на стадії, що передує його реалізації або оновленню (у випадку розробки нової системи або при адаптації типових продуктів).

В ринкових умовах процедура вибору архітектури для проектованої ІС зводиться до визначення вартості володіння нею.

Вартість володіння ІС складається з: 1) планових витрат, 2) вартості ризиків.

Планові витрати складаються із вартості технічного обслуговування, модернізації, зарплати обслуговуючого персоналу тощо.

Сукупна вартість ризиків складається із вартості усіх типів ризиків, їх ймовірностей і матриці відповідності між ризиками (остання визначається вибраною архітектурою ІС). Можна виділити найбільш *важливі типи ризиків*:

– проектні ризики (ризики при створенні системи);

– ризики розробки (помилки, недостатня оптимізація);

– технічні ризики (простої, відмови, втрата даних);

– бізнес–ризики (виникають із–за технічних ризиків і пов'язані з експлу–атацією системи);

– невизначеності (пов'язані із зміною бізнес–процесів і складаються з необхідності внесення змін до системи, неоптимальною процедурою функціонування);

– операційні (передбачають невиконання набору операцій, можуть виникати із–за технічних ризиків та є ініціаторами бізнес–ризиків).

Концепція архітектури ІС повинна формуватися на етапі техніко–економічного обгрунтування і вибиратися такою, щоб вартість володіння нею була мінімальною.

Щоб визначити архітектуру ІС, необхідно відповісти на низку запитань: «Що робить система? На які складові частини вона розділена? Яким чином відбувається взаємодія цих частин? Як і де ці частини розміщені?» [5].

Таким чином, архітектуру ІС можна вважати моделлю, яка визначає вартість володіння через наявну в цій системі інфраструктуру.

Складність програмних систем постійно збільшується, що обумовлено зростанням об'єму інформації, яку передають або обробляють, ускладненням самих завдань по обробці інформації та збільшенням кількості таких завдань. Без застосування архітектурного підходу при побудові складних систем їх створення, обслуговування і модифікація, врешті–решт, стануть нерентабельними для бізнесу. Для вирішення цієї проблеми можна використати

методи абстракції, декомпозиції та інкапсуляції. Наприклад, при розробці програмної системи, яка входить до складу інфраструктури великої організації, вона подається у вигляді множини модулів, кожен з яких виконує конкретну функцію, а всі разом вони виконують функції самої системи.

2. Типові архітектури інформаційних систем

З точки зору програмно–апаратної реалізації можна виділити такі **типові архітектури ІС** [5 с. 65]:

1. *Традиційні архітектурні рішення*, засновані на використанні виділених файлів–серверів або серверів баз даних.

2. *Архітектури корпоративних ІС*, які базуються на технології Internet (Intranet–додатки).

3. *Архітектури ІС, які ґрунтуються на концепції «сховища даних»* (DataWarehouse) – інтегрованого інформаційного середовища, що складається із різнорідних інформаційних ресурсів.

4. Для побудови глобальних розподілених інформаційних додатків використовується *архітектура інтеграції інформаційно–обчислювальних компонентів на основі об'єктно–орієнтованого підходу*.

Індустрія розробки автоматизованих ІС управління зародилася в 1950 – 1960–х роках і до кінця XX ст. набула закінченої форми. У процесі розвитку підходів до проектування ПЗ виділимо такі етапи:

1. *На першому етапі* основним підходом у проектуванні ІС був метод «знизу–вгору», коли система створювалася як набір застосунків, найбільш важливих для підтримки діяльності підприємства. Основною метою цих проектів було не створення тиражованих продуктів, а обслуговування поточних потреб конкретної установи. В рамках «клаптикової автоматизації» забезпечується підтримка окремих функцій, але відсутня стратегія розвитку комплексної системи автоматизації (об'єднання функціональних підсистем перетворюється в самостійну складну проблему).

2. *Другий етап* пов'язаний із усвідомленням того факту, що існує потреба в стандартних програмних засобах автоматизації діяльності різних установ та підприємств. З усієї множини проблем розробники виділили найбільш помітні, а саме: автоматизацію ведення бухгалтерського аналітичного обліку та технологічних процесів. Системи почали проектуватися «зверху–вниз», тобто з припущення, що одна програма повинна задовольняти потреби багатьох користувачів.

Ідея використання універсальної програми накладає обмеження на можливості розробників по формуванню структури бази даних та екранних форм, вибору алгоритмів розрахунку. Закладені «зверху» жорсткі рамки не дають можливості адаптувати систему до специфіки діяльності конкретного підприємства, а саме: врахувати необхідну глибину аналітичного і виробничо–технологічного обліку, включити до її складу необхідні процедури обробки даних, забезпечити інтерфейс кожного робочого місця з урахуванням функцій і технології роботи конкретного користувача. Вирішення цих завдань вимагає серйозних доопрацювань системи. Таким чином, матеріальні і часові витрати на впровадження системи та її доведення під вимоги замовника зазвичай значно перевищують заплановані показники. У той же час замовники ІС стали висувати все більше вимог, спрямованих на забезпечення можливості *комплексного використання корпоративних даних в управлінні та плануванні їхньої діяльності*.

3. *Третій етап.* Виникла потреба у формуванні нової методології побудови ІС. Мета цієї методології полягає в регламентації процесу проектування ІС і забезпеченні управління цим процесом, щоб гарантувати виконання вимог як до самої ІС, так і до характеристик процесу розробки. Основними завданнями, вирішенню яких має сприяти *методологія проектування корпоративних* ІС, є такі:

1) «забезпечувати створення корпоративних ІС, які відповідають цілям і задачам організації, вимогам до автоматизації ділових процесів замовника;

2) гарантувати створення системи із заданою якістю та в задані терміни, в рамках встановленого бюджету проекту;

3) підтримувати зручну дисципліну супроводу, модифікації та нарощування системи;

4) забезпечувати спадкоємність розробки, тобто використання в ІС, яка розробляється, існуючої інформаційної інфраструктури організації». [5].

Впровадження методології повинно призводити до зниження складності процесу створення ІС за рахунок: 1) повного і точного опису цього процесу, 2) застосування сучасних методів і технологій створення ІС на всьому життєвому циклі ІС (від задуму до реалізації).

Розглядаючи архітектуру великих організацій, прийнято використовувати поняття «корпоративна архітектура» як сукупність декількох типів архітектур (рис. 3.1): бізнес архітектури; ІТ–архітектури; архітектури даних; програмної архітектури; технічної архітектури.

Розглянемо більш детально складові моделі корпоративної архітектури.

1. *Технічна архітектура* є першим рівнем архітектури ІС. Вона описує усі апаратні засоби, які використовуються при виконанні заявленого набору функцій, а також включає засоби забезпечення мережевої взаємодії і надійності.

Рис. 3.1. Модель корпоративної архітектури

У технічній архітектурі вказуються периферійні пристрої, мережеві комутатори і маршрутизатори, жорсткі диски, оперативна пам'ять, процесори, сполучні кабелі, джерела безперебійного живлення тощо.

2. *Програмна архітектура* є сукупністю комп'ютерних програм, призначених для вирішення конкретних завдань. Цей тип архітектури потрібний для опису додатків, що входять до складу ІС. На цьому рівні описують програмні інтерфейси, компоненти та їхню поведінку.

3. *Архітектура даних* об'єднує в собі фізичні сховища даних і засоби управління даними. Крім того, до неї входять логічні сховища даних, а при необхідності може бути виділений окремий рівень – архітектура знань. На цьому рівні описуються логічні та фізичні моделі даних, визначаються правила цілісності, складаються обмеження для даних.

4. *Рівень IT–архітектури*, який є єднальним. На ньому формується базовий набір сервісів, які використовуються як на рівні програмної архітектури, так і на рівні архітектури даних. Якщо для цих двох рівнів не була передбачена конкретна особливість функціонування, то зростає ймовірність збоїв в роботі, а, отже, втрат для бізнесу. В деяких випадках неможливо розділити IT–архітектуру і архітектуру окремого застосунку. Таке можливе при високому ступені інтеграції додатків.

Прикладом IT–архітектури може служити SharePoint від компанії Microsoft. Цей продукт надає сервіси для спільної роботи і зберігання інформації, що є важливим аспектом функціонування будь–якої компанії. Його базові системні модулі відносяться до IT–архітектури, а призначені для користувача – до програмної архітектури. Основною функцією IT–архітектури є забезпечення функціонування важливих бізнес–застосунків для досягнення визначених бізнес–цілей. Якщо деяка функція потрібна в декількох застосунках, то її слід перенести на рівень IT–архітектури (тим самим підвищити інтеграцію системи та понизити складність архітектури додатків).

5. *Рівень бізнес–архітектури* або архітектури бізнес–процесів. На цьому рівні визначаються стратегії ведення бізнесу, способи управління, принципи організації і ключові процеси, які є важливими для бізнесу.

3. Архітектурний підхід до проектування ІС

Процес проектування ІС тісно пов'язаний із її архітектурним описом. Можна виділити *п'ять різних підходів до проектування* [5 с. 71–74]: 1) календарний підхід; 2) підхід, за основу якого узято процес управління вимогами; 3) підхід, грунтований на процесі розробки документації; 4) підхід, в основі якого лежить система управління якістю; 5) архітектурний підхід. Розглянемо їх детально:

1. *Календарний підхід* передбачає складання графіку майбутніх робіт з їх поетапним виконанням (при цьому ключові рішення приймаються на підставі локальних завдань і цілей кожного конкретного етапу розробки). Цей підхід не приділяє часу розробці документації, формуванню архітектури і процесам внесення змін (у перспективі через це зростає вартість володіння розробленою на основі цього підходу системою). Він вважається застарілим, проте в деяких компаніях досі застосовується.

2. *Підхід, за основу якого взято процес управління вимогами.* Велику частину часу процесу розробки виділяють на функціональні характеристики системи, а нефункціональні (наприклад, масштабованість) практично не розглядають.

В ході проекту усі рішення формуються виходячи з локальних цілей по реалізації конкретного функціонала. Такий підхід може бути ефективний, якщо вимоги до системи, що розробляється, визначені заздалегідь і не змінюються в процесі проектування. До *недоліків* можна віднести невідповідність стандарту якості ISO 9126 і нестабільність архітектури системи, що розробляється, оскільки кожна функція, яку реалізують, зв'язується з одним або декількома компонентами. У зв'язку із цим при додаванні до подібних систем додаткових

функцій трудомісткість зростає. Застосування цього підходу в перспективі є нераціональним, проте дозволить успішно управляти вимогами в рамках необхідної функціональності.

3. *Підхід, грунтований на процесі розробки документації.* Велика кількість часу витрачається на формування пакету документів, який зазвичай не використовує ані замовник, ані користувач. Крім того, із–за нестачі часу страждає якість системи, яку розобляють. Цей підхід використовують в урядових організаціях і великих компаніях.

4. *Підхід, за основу якого взято систему управління якістю*, включає велику кількість різнопланових заходів для відстежування параметрів, найбільш значимих для функціонування системи. Вибрані параметри спостерігаються на всіх стадіях розробки системи (в деяких випадках на шкоду іншим). Іноді набір таких параметрів може бути складений неправильно, і оптимізація системи буде проведена помилково. У цьому полягає *основний недолік* підходу. Окрім цього, при появі нових вимог дуже важко змінювати функціональність, і подальший розвиток системи може бути неможливий у тому числі з причин архітектурних прорахунків. Такий підхід вважається консервативним, а його застосування доцільно при необхідності створити систему з екстремальними характеристиками.

5. *Архітектурний підхід* до проектування ІС можна вважати найбільш зрілим. Його ключовим аспектом є створення фреймворка, тобто каркаса, адаптацію якого під потреби конкретної системи легко здійснити. Відповідно до цього, завдання проектування розбивається на дві підзадачі: 1) розробка багаторазово використовуваного каркаса, 2) створення системи на його основі. Ці підзадачі можуть вирішуватися різними групами фахівців. При використанні каркасів з'являється можливість швидко змінювати функціональність системи за рахунок ітераційного характеру процесу проектування.

З усіх розглянутих стилів проектування не можливо однозначно визначити найкращий, оскільки кожна конкретна проектна група та кожна

спроектована система має свої особливості, на підставі яких слід вибрати той підхід, застосування якого дозволить вирішити поставлені завдання у встановлені терміни та у рамках виділеного бюджету.

4. Характеристики якості програмного забезпечення в інформаційних системах

Користувачі сучасних ІС зазвичай взаємодіють з ними за допомогою спеціальних програмних модулів, від показників якості яких залежить рівень якості усієї ІС цілком. Для створення правильної і надійної архітектури, розробки та інтеграції програмних систем існує велика кількість стандартів. Кожна існуюча у світі технологія має свій стандарт, описаний у відповідних нормативних документах. Застосування цих стандартів збільшує шанси на успішне створення системи та її подальше безвідмовне функціонування, але складність системи при їх інтеграції може істотно зрости.

Якістю ПЗ є «сукупність його характеристик, які відносяться до можливості задовольняти висловлені або передбачені потреби усіх зацікавлених осіб. Це визначення включено у стандарт ISO 9126, в якому також визначені і самі характеристики» [5]. Можна виділити такі три *аспекти якості*: 1) «внутрішня якість (характеристики ПЗ), 2) зовнішня якість (поведінкові характеристики ПЗ), 3) контекстна якість (відчуття користувачів при різних контекстах використання). [5 с. 75]. Керуючись цими аспектами, стандарт ISO 9126 виділяє такі шість *характеристик якості* ПЗ: 1) функціональність, 2) надійність, 3) продуктивність, 4) зручність використання, 5) зручність супроводу, 6) переносимість. Розглянемо їх детально [5].

1. **Функціональність** передбачає здатність ПЗ вирішувати завдання у визначених умовах і розділяється на такі підхарактеристики: 1) *функціональна придатність* – здатність вирішувати потрібний набір завдань; 2) *точність* – здатність отримувати необхідні результати; 3) *здатність до взаємодії* – здатність взаємодії з необхідним набором інших систем; 4) *захищеність* –

здатність запобігати неавторизованому доступу до даних і програм; 5) *відповідність стандартам і правилам* – відповідність ПЗ різним регламентуючим нормам.

2. **Надійність** характеризується здатністю ПЗ утримувати функціональність в заданих рамках за певних умов і розділяється на такі підхарактеристики: 1) *зрілість* – величина, зворотна частоті відмов ПЗ; 2) *стійкість до відмов* – здатність утримувати конкретний рівень працездатності при різних відмовах і порушеннях правил взаємодії із оточенням; 3) *здатність до відновлення* – здатність відновлювати необхідний рівень працездатності після відмови; 4) *відповідність стандартам надійності.*

3. **Продуктивність** визначається здатністю ПЗ за певних умов гарантувати необхідну працездатність по відношенню до ресурсів, які виділяються для цього (є відношенням отримуваних результатів до витрачених ресурсів). Цю характеристику розділяють на такі підхарактеристики:

1) *тимчасова ефективність* – здатність ПЗ отримувати необхідні результати і забезпечувати передачу необхідного об'єму даних за певний час;

2) *ефективність використання ресурсів* – здатність ПЗ вирішувати необхідні завдання і використовувати задані об'єми визначених видів ресурсів;

3) *відповідність стандартам продуктивності*.

4. **Зручність використання** характеризується привабливістю для користувачів, зручністю в навчанні і використанні ПЗ. У своєму складі вона має ряд таких підхарактеристик:

1) *зрозумілість* – величина зворотна зусиллям, витраченим користу–вачами на усвідомлення здатності ПЗ вирішувати необхідні завдання;

2) *зручність роботи* – величина зворотна зусиллям, витраченим користу–вачами для вирішення необхідних завдань за допомогою ПЗ;

3) *зручність навчання* – величина зворотна зусиллям, витраченим корис–тувачами, на процес навчання роботі з ПЗ;

4) *привабливість* – здатність ПЗ бути приємним для користувачів;

5) *відповідність стандартам зручності використання*.

5. **Зручність супроводу** характеризується зручністю супроводу ПЗ. Ця характеристика включає ряд таких підхарактеристик: 1) *зручність до аналізу*: характеризується зручністю проведення аналізу помилок, дефектів, недоліків, необхідності внесення змін та їхні можливі наслідки; 2) *зручність внесення змін* – це величина зворотна трудовитратам на виконання необхідних змін; 3) *стабільність* – величина зворотна ризику появи непередбачених наслідків при внесенні необхідних змін; 4) *зручність перевірки* – величина зворотна необхідним трудовитратам на тестування та інші види перевірок досягнення передбачених результатів при внесенні змін; 5) *відповідність стандартам зручності супроводу*.

6. **Переносимість** характеризується здатністю ПЗ зберігати працездатність при зміні організаційних, апаратних і програмних аспектів оточення. Для цієї характеристики виділяють такі підхарактеристики:

1) *здатність до адаптації* – здатність ПЗ без здійснення непередбачених дій пристосовуватися до змін оточення;

2) *зручність встановлення* – здатність ПЗ встановлюватися у заздалегідь визначене оточення;

3) *здатність до співіснування* – здатність ПЗ функціонувати в загальному оточенні з іншими програмами, розділяючи з ними ресурси;

4) *зручність заміни* – можливість застосування нового ПЗ замість старого (вже використовуваного) для вирішення тих же завдань, у тому ж оточенні;

5) *відповідність стандартам переносимості.*

Усі наведені характеристики описують внутрішню і зовнішню якість ПЗ. Для опису контекстної якості існує інший набір характеристик:

1) *ефективність* – здатність ПЗ вирішувати призначені для користувача завдання із заданою точністю та в заданому контексті;

2) *продуктивність* – здатність ПЗ отримувати необхідні результати при використанні заздалегідь визначеної кількості ресурсів;

3) *безпека* – здатність ПЗ підтримувати необхідний низький рівень ризику нанесення збитку людям, бізнесу і *довкіллю;*

4) *задоволеність користувачів* – здатність ПЗ при використанні в певному контексті приносити задоволення користувачам.

Керуючись розглянутими показниками можна збільшити якість програмних модулів, а, отже, й усієї ІС в цілому.

5. Функціональні компоненти інформаційної системи

Враховуючи принцип декомпозиції, прийнято проектувати ІС із розділенням функціонального призначення їх компонентів, тобто створювати багаторівневе подання. Можна виділити такі три основні функціональні групи, призначені для вирішення різних по змісту завдань: 1) взаємодія з користувачами, 2) бізнес–логіка, 3) управління ресурсами. Реалізація такого функціонала відбувається за допомогою створення відповідної програмної системи, яка також має багаторівневе подання компонентів (рис. 3.2).

Компонент подання служить для забезпечення взаємодії користувачів з програмою (Додаток В), тобто обробляє процеси натиснення клавіш, руху в різних контроллерах, здійснює виведення інформації, надає призначений для користувача інтерфейс.

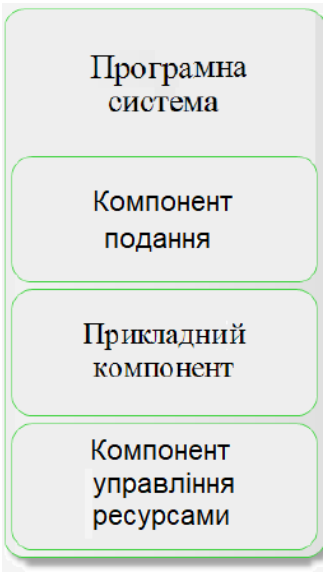


Рис. 3.2. Компоненти програмної системи

Прикладний компонент є набором правил та алгоритмів реалізації функцій системи, реакцій на дії користувачів або внутрішніх подій обробки даних. **Компонент управління ресурсами** відповідає за зберігання, модифікацію, вибірку і видалення даних, пов'язаних із вирішуваною прикладною задачею. Одним із найважливіших етапів проектування архітектури ІС є розподіл цих функціональних компонентів відповідно до вибраної платформеної архітектури.

6. Платформена архітектура інформаційних систем

Можна виділити такі три напрями розвитку платформеної архітектури: автономні, централізовані, розподілені [5 с. 82–84]:

1. **Автономна архітектура** передбачає наявність усіх функціональних компонентів системи на одному фізичному пристрої (наприклад, комп'ютері) і не повинна мати зв'язків із зовнішнім середовищем. Прикладом таких систем можуть служити системні утиліти, текстові редактори та корпоративні програми. В процесі побудови корпоративної ІС, зазвичай, не повинні формуватися не зв'язані з нею вузли або модулі (їх поява може бути обумовлена певними вимогами до безпеки або надійності).

2. **Централізована архітектура** (рис. 3.3) передбачає виконання усіх необхідних завдань на спеціально відведеному вузлі, потужності якого вистачає, щоб задовольнити потреби усіх користувачів. Такий тип архітектури був популярний у 70–х роках XX ст., проте, і зараз є затребуваним. Компоненти системи в цьоному випадку розподіляються між обчислювальним вузлом, який називається мейнфрейм, і термінальною станцією, за якою працює користувач. Термінал містить компонент подання, а мейнфрейм – прикладний компонент і компонент управління ресурсами. Термінал має вигляд виключно пристрою введення–виведення.

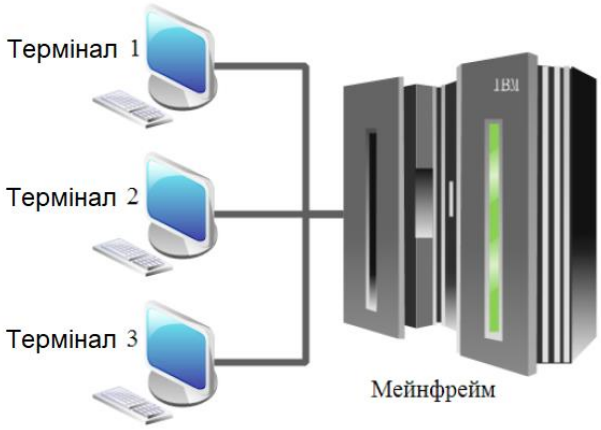


Рис. 3.3. Зображення централізованої архітектури

До *переваг* такої архітектури можна віднести: 1) відсутність необхідності адміністрування робочих місць; 2) легкість обслуговування та експлуатації системи, оскільки усі ресурси зосереджено в одному місці. *Недоліками* подібної архітектури є: 1) функціонування усієї системи повністю залежить від головного вузла (мейнфрейма); 2) всі ресурси та програмні засоби є колективними і не можуть бути змінені під потреби конкретних користувачів.

Щоб позбавитися від останнього недоліку, в сучасних ІС застосовують технології віртуалізації, завдяки яким кожному користувачеві можна виділити необхідну кількість ресурсів і встановити необхідне ПЗ.

3. **Розподілена архітектура**. У цьому типі архітектури функціональні компоненти ІС розподіляються по наявних вузлах залежно від поставленнях цілей і завдань. Можна виділити такі *основні характеристики систем розподіленої архітектури*:

1) спільне використання ресурсів (як апаратних, так і програмних);

2) відкритість – можливість збільшення типів і кількості ресурсів;

3) паралельність – можливість виконання декількох процесів на різних вузлах системи (при цьому вони можуть взаємодіяти);

4) масштабованість – можливість додавати нові властивості і методи;

5) відмовостійкість – здатність системи підтримувати часткову функці–ональність за рахунок можливості дублювання інформації, апаратної і програмної складової.

До *недоліків розподілених систем* слід віднести: 1) структурну складність; 2) складність у забезпеченні достатнього рівня безпеки; 3) велику кількість зусиль на управління системою; 4) непередбачувану реакцію на зміни. Усі ці недоліки пов'язано зі складністю структури ІС та системи розподілу прав доступу, різноплановим устаткуванням. Необхідно враховувати усі з них, інакше розроблена ІС не зможе функціонувати у межах очікуваних параметрів.

Існують такі *види архітектур розподілених систем* [5 с. 85]: архітектура «файл–сервер»; архітектура «клієнт–сервер»; архітектура Web–додатків.

1. **Архітектура «файл–сервер»** (рис. 3.4) передбачає наявність виділеного мережевого ресурсу для зберігання даних. Такий ресурс називають «файловим сервером». При такій архітектурі усі функціональні компоненти системи розташовані на призначеному для користувача комп'ютері, який називають «клієнтом», а самі дані знаходяться на сервері. Ця організація системи має такі переваги: 1) режим роботи з даними, які зберігаються на сервері, розрахований на багато користувачів; 2) централізоване управління правами доступу до загальних даних; 3) мала вартість розробки; 4) висока швидкість розробки. До недоліків цієї архітектури можна віднести такі: послідовний доступ до загальних даних і відсутність гарантії їх цілісності; продуктивність (яка залежить від продуктивності мережі, клієнта і сервера).

2. **Архітектура «клієнт–сервер»** є мережевою інфраструктурою, в якій сервери є постачальниками певних сервісів (послуг), а комп'ютери клієнтів виступають їх споживачами.

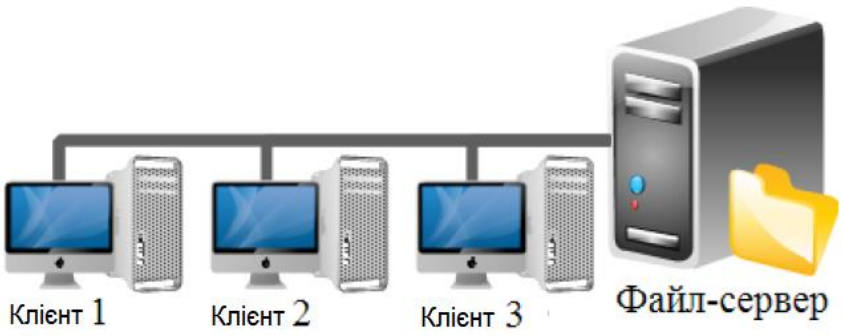


Рис. 3.4. Зображення архітектури «файл–сервер»

Класичне подання клієнт–серверної архітектури передбачає наявність у мережі сервера і декількох підключених до нього клієнтів. У таких системах сервер, в основному, грає роль постачальника послуг із використання бази даних. Ця архітектурна модель називається дворівневою (рис. 3.5). Серед *переваг* цієї архітектури можна виділити такі: підтримка розрахованої на багато користувачів роботи; гарантія цілісності даних; наявність механізмів управління правами доступу до ресурсів сервера; можливість розподілу функцій між вузлами мережі. До *недоліків* цієї архітектури можна віднести такі: 1) вихід з ладу сервера може спричинити непрацездатність усієї системи; 2) необхідність мати технічний персонал високого рівня; 3) висока вартість устаткування.

При збільшенні масштабів системи може знадобитися заміна апаратної частини сервера і клієнтських машин. Проте, при збільшенні числа користувачів виникає необхідність синхронізації версій великої кількості додатків. Для вирішення цієї проблеми використовують *багатоланкову архітектуру* (три і більше рівнів), в якій частина загальних застосувань переноситься на спеціально виділений сервер, тим самим знижуються вимоги до продуктивності клієнтських машин. Клієнтів з низькою обчислювальною потужністю називають «тонкими клієнтами», а з високою продуктивністю – «товстими клієнтами». При багатоланковій архітектурі з виділеним сервером додатків існує можливість використання портативних пристроїв. Багатоланкову архітектуру зображено на рис. 3.6.

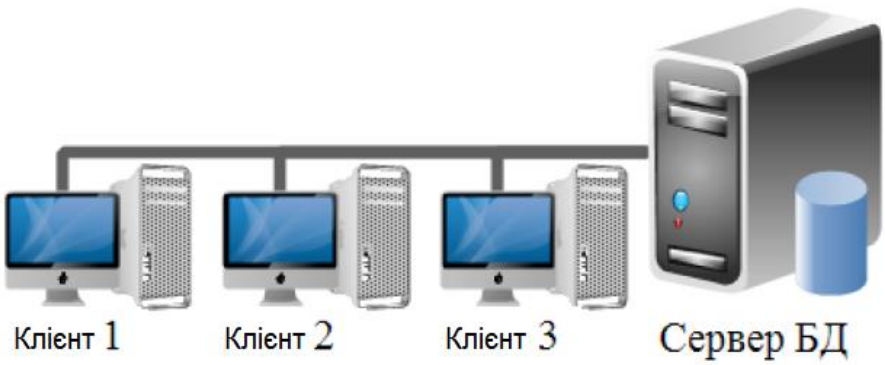


Рис. 3.5. Дворівнева клієнт–серверна архітектура

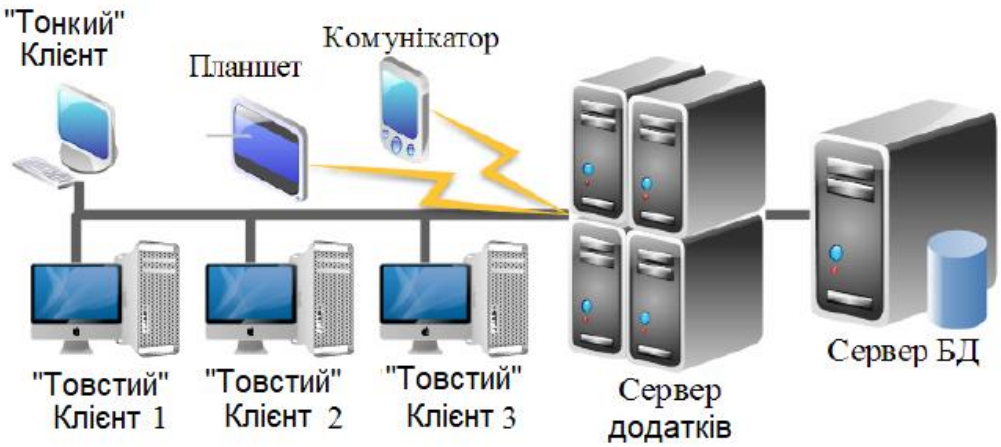


Рис. 3.6. Багатоланкова клієнт–серверна архітектура

Використання такого типу архітектури обумовлене високими вимогами додатка до ресурсів. У цьому випадку необхідно винести його на окремий сервер і, тим самим, знизити вимоги до продуктивності робочих станцій. Грамотний підбір характеристик сервера додатків, сервера баз даних і клієнтських робочих станцій дозволить створити ІС з прийнятною вартістю володіння.

3. **Архітектура Web–застосунків** або архітектура Web–сервісів передбачає надання деякого сервісу, доступного в мережі Internet, через спеціальне застосування. Основою для надання цих послуг служать такі відкриті стандарти і протоколи [5 с. 89]:

1. «SOAP (Simple Object Access Protocol) визначає формат запитів до Web–сервісі.

2. WSDL (Web Service Description Language) застосовують для опису інтерфейсу сервісу, який надається. Перед розгортанням web–застосунку вимагається скласти його опис, вказати адресу, список підтримуваних протоколів, перелік допустимих операцій, а також формати запитів і відповідей.

3. UDDI (Universal Description, Discovery and Integration) є протоколом пошуку Web–сервісів в мережі Internet. Пошук здійснюється за їхніми описами, розташованими у спеціальному реєстрі. Архітектура таких сервісів схожа на

багатоланкову клієнт–серверну, проте, сервери додатків і баз даних розташовуються в мережі Internet».

Для побудови розподіленої архітектури Web–сервісу можна виділити такі технології [5 с. 91]: EJB (Enterprise JavaBeans), DCOM (Distributed Component Object Model), CORBA (The Common Object Request Broker Architecture).

1. EJB – інфраструктура, в якій легко додавати і видаляти компоненти, при цьому змінюючи функціональність сервера. EJB дозволяє розробникам складати власні застосунки із заздалегідь створених модулів. При цьому можлива їх зміна, що робить процес розробки гнучким і набагато швидшим. Ця технологія сумісна з CORBA і Java API. Взаємодія між клієнтами і сервером в цьому випадку подається як взаємодія EJB–об'єкту, генерованого спеціальним генератором, і EJB–компонента, написаного розробником. При необхідності викликати метод у EJB–компонента, розташованого на сервері, викликається одноіменний метод EJB–об'єкту, розташованого на стороні клієнта, який зв'язується із необхідним компонентом і викликає відповідний метод. До переваг EJB можна віднести такі: просте і швидке створення; Java–оптимізація; кросплатформеність; вбудована безпека. До недоліків слід віднести: складність інтеграції з додатками; погана масштабованість; низька продуктивність; відсутність міжнародної стандартизації.

2. DCOM є розподіленою програмною архітектурою від компанії Microsoft. З її допомогою програмний компонент одного комп'ютера може передавати повідомлення програмного компонента іншого комп'ютера, причому з'єднання встановлюється автоматично. Для надійної роботи вимагається забезпечити захищене з'єднання між зв'язаними компонентами, а також створити систему перенаправлення трафіку. *Переваги* DCOM: незалежність від мови; динамічне знаходження об'єктів; масштабованість; відкритий стандарт. *Недоліки* DCOM: складність реалізації; залежність від платформи; пошук через службу Active Directory; відсутність іменування сервісів через URL.

3. Технологія CORBA розглядає усі застосунки в розподіленій системі як набір об'єктів. Об'єкти можуть одночасно виступати в ролі клієнта і сервера, викликаючи методи інших об'єктів і відповідаючи на їх виклики. Застосування цієї технології дозволяє будувати системи, які перевершують за складністю і гнучкістю системи з дворівневою або трирівневою архітектурою клієнт–сервер. *Переваги* CORBA: незалежність від платформи та мови; динамічні виклики; динамічне виявлення об'єктів; масштабованість; індустріальна підтримка. *Недоліки*: відсутність іменування по URL; практично повна відсутність реалізації CORBA–сервісів.

При побудові архітектури ІС може знадобитися використання відразу декількох з розглянутих технологій. Кожна з них реалізовуватиме конкретний функціонал, який може вимагати декількох типів платформеної архітектури. У одній системі можуть працювати декілька файл–серверів, декілька серверів додатків і декілька серверів баз даних. Таким чином можна розподіляти навантаження в системі або групувати набори сервісів за виконуваними функціями.

Велика частина процесів по проектуванню ІС передбачає використання досвіду реалізації схожих проектів. Складно подати систему, для реалізації якої не можна було б застосувати готові рішення або досвід, отриманий при їх створенні. Не існує стандартних мов опису архітектури.

7. Фреймворки

Термін *фреймворк* (каркас) можна визначити як «загальноприйняті архітектурно–структурні рішення та підходи до проектування. Фреймворк є загальним рішенням складної задачі» [5 с. 102]. Фреймворки класифікують за такими ознаками:

1. За місцем використання. *Інфраструктурні фреймворки* (System Infrastructure Frameworks) спрощують процес розробки інфраструктурних елементів, застосовуються усередині організації і не продаються. *Фреймворки рівня проміжного ПЗ* (Middleware Frameworks) застосовують для під’єднання

додатків. *Фреймворки, орієнтовані на додатки*, використовуються для підтримки систем, орієнтованих на роботу з кінцевими користувачами в конкретній предметній області. *Архітектурний фреймворк* визначається як «сукупність угод, принципів і практик, використовуваних для опису архітектури та прийнятих стосовно деякого предметного домена і (або) в співтоваристві фахівців (зацікавлених осіб)». [5 с. 103]. Він включає опис зацікавлених осіб, типові проблеми предметної області, архітектурні точки зору і методи їх інтеграції.

2. За способом використання. *Фреймворки, використовувані за принципом білого ящика*: механізмами формування основних елементів додатка тут виступають методи спадковості та динамічного зв'язування; такі фреймворки визначаються через інтерфейси об'єктів, які додаються в систему; для роботи з ними потрібна детальна інформація про класи, які потрібно розширити. *Фреймворки, які функціонують за принципом чорного ящика* (або фреймворки, керовані даними): основними механізмами формування додатків тут виступають композиція і параметризація, при цьому функціональність забезпечується додаванням додаткових компонентів. На практиці застосовують підхід *сірого ящика*, що є комбінацією обох підходів.

3. За маштабом використання. *Фреймворки рівня застосунків* подають функціонал по реалізації типових застосувань (GUI, бази даних тощо). *Фреймворки рівня домена* застосовуються для створення додатків в певній предметній області. *Допоміжні фреймворки* (Support Frameworks) застосовуються для вирішення конкретних завдань.

У світі існує велика кількість різних фреймворків. В якості прикладів нрозглянемо такі два найбільш відомих: фреймворки Захмана, TOGAF.

1. **Фреймворк Захмана** був створений співробітником компанії IBM Джоном Захманом (John Zachman), який заклав в основу свого фреймворка таку класифікацію (таксономію) артефактів системи: дані, функціональність, моделі,

специфікації, документи. Цей фреймворк можна вважати онтологією верхнього рівня, яка описує конкретну систему (табл. 3.2).

Таблиця 3.2

Матриця Захмана

	Дані, які використовують (що?)	Процеси і функції (як?)	Місця виконання процесів (де?)	Організації і персоналії (хто?)	Керуючі події (коли?)	Цілі та обмеження	
Контекст	Список основних сутностей	Основні бізнес-процеси	Територіальне розміщення організації	Важливі зовнішні організації	Список подій	Бізнес-стратегія	**Аналітики**
Бізнес-модель	Відношення між сутностями	Детальний опис бізнес-процесів	Система логістики	Модель потоків даних	Базовий графік робіт	Дерево цілей, Бізнес-план	**Топ-менеджери**
Системна модель	Концептуальні моделі	Архітектура застосунків	Архітектура розподіленої системи	Інтерфейси користувача	Модель роботи з подіями	Бізнес-правила	**Архітектори**
Технологічна модель	Фізична модель даних	Програмно-апаратна архітектура	Технологічна архітектура	Архітектура подання	Алгоритм обробки подій	Правила обробки подій	**Розробники**
Детальний опис	Специфікації форматів даних	Код, який використовується	Архітектура мережі	Ролі і права користувачів	Обробка подій за допомогою переривання	Алгоритми роботи системи	**Адміністратори**
Функціонуюча організація	Дані	Функціональність, яку реалізують	Функціонуюча мережна архітектура	Організаційна структура організації	Історія функціонування системи	Стратегії, які реалізуються	**Користувачі**

Для побудови таксономії Захманом запропоновано відповісти на шість питань про функціонування організації: що, як, де, хто, чому. Ці питання відносяться до таких аспектів системи: 1) використовувані дані (що?); 2) процеси і функції (як?); 3) місця виконання процесів (де?); 4) організації і персоналії (хто?); 5) події (коли?), які управляють; 6) цілі та обмеження, які визначають роботу системи (чому?).

Відповідати на ці питання необхідно з різною мірою деталізації. Описано шість рівнів: рівень контексту; рівень бізнес–описів; системний рівень; технологічний рівень; технічний рівень; рівень реальної системи. Для кожного рівня деталізації існує своя зацікавлена особа (точка зору): 1) аналітики (рівень контексту); 2) топ–менеджери (рівень бізнес–описів); 3) архітектори (системний рівень); 4) розробники (технологічний рівень); 5) адміністратори (технічний рівень); 6) користувачі (рівень реальної системи). За результатами виконаних дій формується матриця розміром 6×6, в кожній комірці якої розташовуються артефакти. Для заповнення комірок введені такі правила:

1. Колонки можна міняти місцями, але не можна їх додавати і видаляти.

2. Кожному стовпчику відповідає власна модель. Кожна відповідна модель повинна бути унікальною.

3. Кожен рівень (рядок) є описом системи з точки зору групи користувачів (подає окремий різновид).

4. Кожна комірка є унікальною та містить опис аспекту реалізації системи у вигляді моделі і текстового документу. Заповнення комірок виконується послідовно зверху вниз.

Перший рядок матриці визначає контекст усіх інших та є загальним поглядом на організацію. Другий рядок описує функціонування організації у бізнес–термінах. Третій рядок описує бізнес–процеси в термінах ІС. Четвертий рядок дозволяється розподілити ці бізнес–процеси і виконувані над ними операції між конкретними апаратними і програмними платформами. П'ятий рядок описує конкретні моделі устаткування, мережеві топології і програмний код. Шостий рядок описує готову систему у вигляді керівництва користувачів, довідкових баз даних тощо.

Стовпчик *використовуваних даних*, відповідно до рівнів, містить у своїх комірках такі артефакти:

1 рівень – список основних сутностей; 2 рівень – семантична модель;

3 рівень – нормалізована модель;

4 рівень – фізична модель даних або ієрархія класів;

5 рівень – опис моделі на мові управління даними;

6 рівень – фактичні набори даних, статистики тощо.

Стовпчик *процеси і функції* описує порядок дій для деталізації процесу переходу від місії організації до опису конкретних операцій:

1 рівень – перерахування ключових бізнес–процесів;

2 рівень – опис бізнес–процесів;

3 рівень – опис операцій над даними та архітектури додатків;

4 рівень – методи класів;

5 рівень – програмний код; 6 рівень – виконувані модулі.

З четвертого рівня розгляд ведеться у межах окремих застосувань.

Стовпчик *місця виконання процесів* визначає розташування компонент системи і мережеву інфраструктуру:

1 рівень – розташування основних об'єктів;

2 рівень – модель взаємодії об'єктів;

3 рівень – розподіл компонентів ІС по вузлах мережі;

4 рівень – фізична реалізація на апаратних і програмних платформах;

5 рівень – використовувані протоколи і специфікації каналів зв'язку;

6 рівень – опис функціонування мережі.

Стовпчик *організації і персоналії* визначає учасників процесу функціонування:

1 рівень – список партнерів, підрозділів організації, виконувані функції;

2 рівень – повна організаційна діаграма (можуть бути присутніми вимоги до інформаційної безпеки);

3 рівень – учасники бізнес–процесів та їх ролі;

4 рівень – вимоги до призначених для користувача інтерфейсів;

5 рівень – правила доступу до об'єктів;

6 рівень – фізична реалізація в коді.

Стовпчик *подія*, яка управляє, визначає часові параметри системи і бізнес–процесів:

1 рівень – список значимих для системи подій;

2 рівень – базовий графік робіт; 3 рівень – моделі роботи з подіями;

4 рівень – алгоритми обробки подій; 5 рівень – програмна реалізація;

6 рівень – історія функціонування системи.

Стовпчик *цілі та обмеження* вказує послідовність дій для переходу від завдань бізнесу до вимог для елементів системи:

1 рівень – бізнес–стратегія; 2 рівень – дерево цілей і бізнес–план;

3 рівень – правила та обмеження для бізнес–процесів;

4 рівень – додатки, які включаються до складу системи;

5 рівень – алгоритми роботи додатків;

6 рівень – фізична реалізація у коді.

За допомогою фреймворка Захмана не можна описати динаміку поведінки системи (її розвиток), крім того, в ньому не існує механізму контролю за змінами. Цей фреймворк розповсюджують на комерційній основі.

2. **Фреймворк TOGAF** (The Open Group Architecture Framework) є набором засобів для розробки архітектури різного призначення. З його допомогою ІС описується як сукупність модулів. В рамках TOGAF поняття «архітектура» визначається як «формальний опис системи, або детальний план системи на рівні компонентів і методології їх реалізації» [5 с. 111]. TOGAF складається з чотирьох архітектурних доменів: 1) бізнес–архітектура (описує ключові бізнес–процеси, стратегію розвитку бізнесу і принципи управління); 2) архітектура рівня додатків (описує інтерфейси додатків і способи їх застосування в термінах бізнес–сервісів); 3) архітектура рівня даних (визначає логічну і фізичну структуру даних в організації); 4) технологічна архітектура (визначає програмну, апаратну і мережеву інфраструктури).

Головними складовими TOGAF є:

1. ADM–методика (Architecture Development Method), яка описує процес розробки архітектури.

2. Керівництво і методики проектування для ADM.

3. Фреймворк архітектурного опису (Architecture Content Framework), який є моделлю результатів розробки.

4. Архітектурний контінуум організації (Enterprise Continuum) у вигляді репозиторія архітектурних артефактів і реалізацій.

5. Еталонні моделі TOGAF (TOGAF Reference Models): 1) TRM (Technical Reference Model) – технічна еталонна модель; 2) III–RM (The Integrated Information Infrastructure Model) – інтегрована модель інформаційної інфраструктури.

6. Фреймворк, який описує структуру організації, її персонал, необхідні ролі і рівні відповідальності.

Відповідно до ADM–методики архітектурний процес можна розбити на дев'ять фаз. ADM – це ітераційний процес, що відбувається на двох рівнях: 1) на верхньому рівні кожної ітерації повторюються загальні для кожної з фаз дії; 2) нижній рівень описує ітерації усередині кожної фази. Рішення приймаються на підставі існуючих вимог бізнесу та рішень. Схему фаз TOGAF наведено на рис. 3.7, опис кожної фази – у табл. 3.3. При розробці TOGAF можна використовувати один фреймворк та сукупність інших (зокрема, в нього входить спеціальний документ, що визначає відповідності між поняттями TOGAF і фреймворком Захмана).

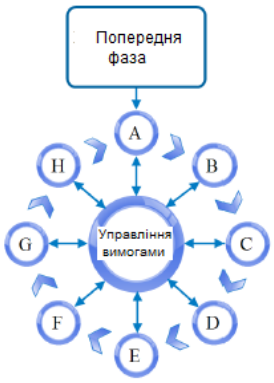


Рис. 3.7. Стадії TOGAF

Таблиця 3.3

Опис фаз TOGAF

Стадія процесу	Опис
Попередня фаза (Preliminary Phase)	Визначення способів управління, меж розробки, принципів реалізації, уточнення моделі відносно специфіки.
Фаза A: розробка загального подання (Architecture Vision)	Створення загального уявлення про архітектуру, визначення меж проекту, затвердження плану робіт і завдань для виконавців
Фаза B: розробка бізнес–архітектури (Business Architecture)	Виявлення принципів функціонування організації, принципів управління проектом
Фаза C: розробка інформаційної архітектури (Information Systems Architecture)	Розробка архітектури даних і додатків
Фаза D: розробка технологічної архітектури (Technology Architecture)	Підбір апаратних засобів, засобів мережевої інфраструктури, механізмів їх взаємодії
Фаза E: можливості та рішення (Opportunities and Solutions)	Реалізація розробленої архітектури (купити готове рішення або зробити самим)
Фаза F: планування переходу до нової архітектури (Migration Planning)	Оцінка ризиків, аналіз деталей переходу
Фаза G: формування системи управління реалізацією (Implementation Governance)	Моніторинг процесу впровадження і специфікація виникаючих проблем
Фаза H: управління зміною архітектури (Architecture Change Management)	Налаштування процесу управління змінами розробленої архітектури

8. Інтеграція інформаційних систем

Під терміном «*інтеграція*» можна розуміти об'єднання ІС, застосунків, різних компаній або людей. Виділяють зовнішню та внутрішню інтеграцію: *внутрішня* передбачає об'єднання різних корпоративних застосувань в одній організації, *зовнішня* – об'єднання ІС різних організацій.

Існують такі основні типові інтеграційні підходи: 1) «інтеграція на рівні даних; 2) інтеграція на рівні бізнес–функцій і бізнес–об'єктів; 3) інтеграція на рівні бізнес–процесів; 4) портали». [5 с. 125].

1. *Інтеграція на рівні даних* передбачає наявність в системах баз даних, для роботи з якими необхідно розробити єдиний програмний інтерфейс. До основних технологічних рішень цього підходу відносять: системи реплікації даних; обласні бази даних; використання API для доступу до EPR– систем.

Реплікація є процесом синхронізації даних між різними джерелами. Необхідність в цьому виникає у момент зміни блоку інформації в розподілених системах зберігання, щоб гарантувати коректність і несуперечність даних, використовуваних в усіх модулях або додатках інформаційної системи. Зазвичай функції реплікації покладають на проміжне ПЗ.

Обласні (федеративні) бази даних надають єдиний інтерфейс до розподілених даних. Це забезпечує інтеграцію множини автономних даних, які можуть бути фізично розташовані на різних пристроях в мережі. Такі бази даних прийнято називати *віртуальними*.

Використання API для доступу до ERP–систем покликано спростити механізми обміну інформацією між призначеними для користувача застосунками і програмним забезпеченням, призначеним для управління функціонуванням виробничих ІС (ERP).

2. *Інтеграція на рівні бізнес–функцій і бізнес об'єктів* передбачає реалізацію спільно використовуваних служб (сервісів). Служба може бути набором функцій, який використовується в декількох застосунках. Цей набір служб і буде бізнес–функціями. При використанні сервісно–орієнтованої архітектури бізнес–функції можна розглядати як бізнес–сервіси, а при компонентному підході – як бізнес–об'єкти (бізнес–компоненти).

3. *Інтеграція на рівні бізнес–процесів* розрізняється залежно від рівня інтеграції. При внутрішній інтеграції взаємодіє велика кількість сервісів, а при зовнішній інтеграції – в основному два. Бізнес–процеси функціонують над виділеними службами, для управління якими існує спеціальна мова, що інтерпретується.

4. *Портали* можна вважати графічними інтерфейсами бізнес–процесів, оскільки вони призначені для персоніфікованого доступу до інформації і консолідації даних з декількох джерел.

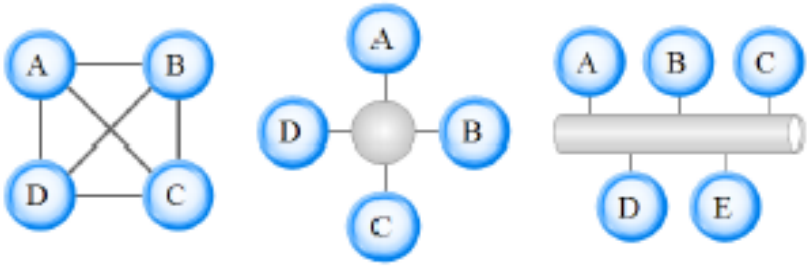
Головним призначенням процесу інтеграції є об'єднання функцій додатків або модулів для надання нової функціональності. При інтеграції додатків можна виділити два основні типи завдань: завдання інтеграції 1) корпоративних застосунків; 2) додатків з різних ІС. Для вирішення завдань першого типу застосовують системи EAI, які іноді називаються A2A (Application – to – Application Integration), а для вирішення завдань другого типу застосовуються системи B2B (Business – to – Business Integration). У деяких випадках складно визначити різницю між інтеграцією A2A і B2B, оскільки складність деяких рішень усередині ІС може перевищувати складність рішень для їх спільного функціонування.

Існують три альтернативні топології інтеграції: 1) точка–точка (Point – to – Point); 2) шлюз (hub – and – spoke); 3) шина (Bus).

У топології «точка–точка» усі об'єкти мають прямі зв'язки один з одним (рис. 3.8, *а*). Кожен зв'язок можна реалізувати будь–яким способом. Варіанти реалізації залежать від вимог і характеристик взаємодії між об'єктами. До *недоліків топології* можна віднести такі характеристики: недостатня гнучкість; складність підтримки численних з'єднань «точка–точка»; зміни одного об'єкту впливають на об'єкти, що залишилися; логіка маршрутизації часто програмується в коді об'єктів; відсутність загальної моделі безпеки; використання різних API; низька надійність; складність створення фреймворків і підтримки асинхронної взаємодії.

Для скорочення числа використовуваних інтерфейсів слід використати топологію із загальним шлюзом (рис. 3.8, *б*) або топологію із загальною шиною (рис. 3.8, *в*). Такі моделі інтеграції реалізуються на рівні проміжного ПЗ.

Наступним кроком в розробці інтеграційної архітектури можна вважати появу корпоративної сервісної шини (Enterprise Service Bus – ESB).



а) *б)* *в)*

Рис. 3.8. Топології інтеграції

Ряд авторів розглядає *системи* ESB, як наступний ступінь розвитку EAI. Проте є такі відмінності: EAI – централізована архітектура з обміном інформації через хаб (брокер), а ESB – шинна архітектура, яка може бути реалізована у вигляді декількох розподілених систем; вона орієнтована на використання відкритих стандартів. Ці дві відмінності демонструють можливість використання ESB як інтеграційної платформи, що дозволяє використати різні механізми.

ESB дозволяє проводити як внутрішню, так і зовнішню інтеграції, і є шиною, працюючою як слабко–зв'язна система, керована подіями. Концепції сервісно–орієнтованої архітектури (СОА) і ESB сильно пов'язані. ESB підтримує принцип реалізації СОА: розділення служби подання та її реалізації.

Функції ESB: надання інтерфейсів взаємодії; відправка і маршрутизація повідомлень; перетворення даних; реакція на події; управління політиками; віртуалізація.

Функції ESB: надання інтерфейсів взаємодії; відправка і маршрутизація повідомлень; перетворення даних; реакція на події; управління політиками; віртуалізація. На підставі функцій ESB можна сформувати типовий *список вимог, які пред'являють користувачі*: велика пропускна спроможність; підтримка декількох стилів інтеграції; забезпечення можливості додаткам працювати з сервісами як безпосередньо, так і через адаптери.

ESB є, по суті, логічним компонентом архітектури, що приводить інтеграційну інфраструктуру у відповідність принципу СОА. Архітектурою,

побудованою за принципом ESB, складніше управляти, але вона гнучкіша і масштабована (впровадження СОА не потребує змін в усіх елементах системи, внаслідок чого зможе відбуватися поетапно). Можна подати ESB у вигляді п'ятирівневої структури: 1) рівень сполучення (адаптери та інтерфейси); 2) транспортна підсистема; 3) рівень реалізації бізнес–логіки; 4) рівень управління бізнес–процесами; 5) рівень бізнес–управління.

Рівень сполучення покликаний вирішувати проблему використання різних інтерфейсів. На цьому рівні функціонують адаптери, які відстежують події в додатках і в інтеграційній підсистемі, і забезпечують перетворення передаваних об'єктів при взаємодії з транспортною підсистемою. Окрім заздалегідь створених адаптерів інтеграційної платформи існує можливість використати адаптери, створені самостійно. *Адаптери* можна розділити на дві категорії: 1) технологічні (застосовуються для інтеграції технологічних компонент, за відсутності у них API), 2) адаптери для додатків (застосовуються для інтеграції з конкретним застосунком).

Транспортна підсистема надає можливість асинхронної взаємодії інтегрованим застосункам. Цей рівень відповідає також за управління і безпеку інформації, може виконувати маршрутизацію повідомлень та їх обробку.

Рівень реалізації бізнес–логіки надає функції для трансформації і маршрутизації повідомлень. На цьому рівні функціонують брокери повідомлень, які обмінюються повідомленнями через транспортну підсистему. Брокер повідомлень може виконувати такі функції:

1. Прийняття повідомлень та їх відправка по вказаних адресах.
2. Перетворення форматів повідомлень.
3. Агрегація і фрагментація повідомлень.
4. Взаємодія з репозиторіями.
5. Вибірка даних через виклики Web–служб.
6. Обробка помилок і подій.
7. Маршрутизація повідомлень за адресою, змістом, темою.

Управління бізнес–процесами на одноіменному рівні здійснюється за допомогою мови управління бізнес–процесами (Business Process Execution Language) на основі Web–сервісів. *Рівень бізнес–управління* є надбудовою над попереднім рівнем і призначений для управління бізнес–процесами в термінах відповідної предметної області. Підхід ESB має ряд переваг і дозволяє будувати інтеграційну архітектуру будь–якої складності. Типова структура інтеграційної системи наведена на рис. 3.9.

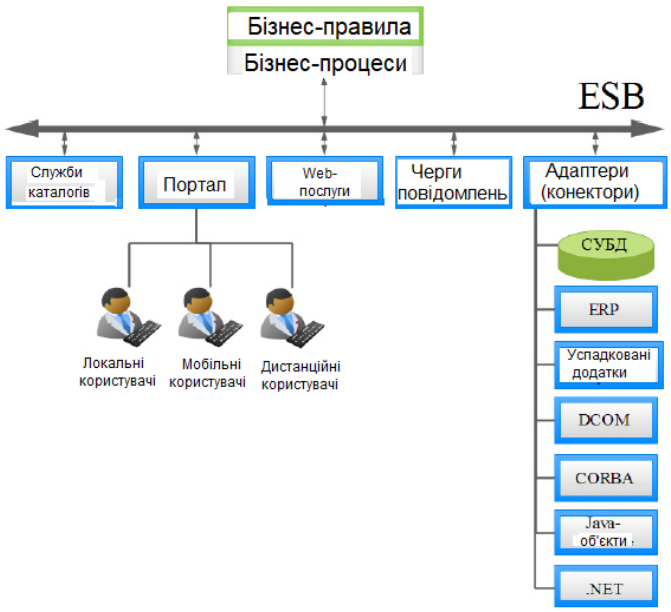


Рис. 3.9. Типова структура інтеграційної системи

Запитання для самоконтролю

1. Що таке архітектура інформаційних систем? Назвіть основні підходи до проектування ІС.

2. Назвіть типи архітектур та їх моделі.

3. Назвіть основні напрямки розвитку платформених архітектур.

4. Назвіть складові багатошарової архітектури, N–рівневої/3–рівневої архітектури.

5. Назвіть 1) різновиди розподілених архітектур; 2) технології, які можна використовувати для побудови розподіленої архітектури Web–сервісу; 3) основні інтеграційні підходи до розробки ІС.

4. ТЕХНОЛОГІЯ ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

1. Поняття «Технологія проектування інформаційних систем»

Масштаби розроблюваних систем визначають склад і кількість учасників процесу проектування. При великому обсязі і жорстких термінах виконання проектних робіт в розробці системи може брати участь кілька проектних колективів (організацій–розробників). У цьому випадку виділяється головна організація, яка координує діяльність усіх організацій–співвиконавців.

Проектування ІС передбачає використання проектувальниками певної технології проектування, яка відповідає масштабу та особливостям розроблюваного проекту.

Технологією проектування (ТП) ІС є «сукупність методології та засобів проектування ІС, а також методів і засобів організації проектування (управління процесом створення та модернізації проекту ІС)». [11].

В основі ТП ІС лежить технологічний процес, який визначає дії, їх послідовність, склад виконавців, кошти і ресурси, необхідні для виконання цих дій. Технологія проектування задається регламентованою послідовністю технологічних операцій, що виконуються в процесі створення проекту на основі конкретного методу, в результаті чого стало б ясно, не тільки ЩО повинно бути зроблено для створення проекту, але і ЯК, КОМУ і в ЯКІЙ ПОСЛІДОВНОСТІ це повинно бути зроблено.

До *основних вимог*, які пред'являють до обраної ТП ІС, відносяться такі [11]:

1) створений за допомогою цієї технології проект повинен відповідати вимогам замовника в частині функціональної повноти, достовірності, оперативності;

2) технологія повинна:

– максимально відображати всі етапи ЖЦ проекту; забезпечувати адаптівність проектних рішень в процесі експлуатації ІС;

– забезпечувати: мінімальні трудові та вартісні витрати на проектування і супровід проекту; розробку проекту у встановлені терміни; надійність процесу проектування ІС та експлуатації проекту;

– сприяти зростанню продуктивності праці проектувальника;

– бути основою зв'язку між проектуванням і супроводом проекту;

– сприяти простому веденню проектної документації, забезпечувати економічну ефективність проектної діяльності (витрати на розробки повінні окупатися за рахунок доходів від реализації проекту).

2. Історичні аспекти розвитку технологій проектування інформаційних систем

В середині XX–го ст. розробники національних і великомасштабних ІС стали першими усвідомлювати необхідність створення спеціальних засобів проектування і моделювання бізнес–процесів, які дозволяють зробити їх роботу більш ефективною, скоротити час створення ІС та мінімізувати помилки (помилки та неточності зустрічаються постійно, чим раніше вони діагностуються і локалізуються, тим менше вартість переробки).

Вартість виявлення та усунення помилки на стадії проектування ІС обходиться у два рази дорожче, на стадії тестування – в десять разів, а на стадії експлуатації – в сто раз, ніж на стадії аналізу бізнес–процесів і розробки технічного завдання. При створенні складних ІС зазвичай важко зрозуміти вимоги персоналу замовника. Вони можуть бути сформульовані некоректно, а в процесі аналізу конкретних бізнес–процесів навіть змінитися. Тому поява *методологій сучасного проектування і моделювання ІС* була нагальним завданням, над яким працювали фахівці різних країн.

Поява автоматизованих систем управління в 60–х роках XX ст. визначалася отриманням початкових знань, досвіду їх розробки і впровадження: аналізувалися успіхи та невдачі створення перших АСУ, але

безперечним було скорочення часу обробки інформації, виробничих та управлінських витрат і як наслідок персоналу.

Пізніше прийшло розуміння, що інформація – стратегічний ресурс будь–якої компанії, який необхідно використовувати грамотно, а головними споживачами інформації є керівники. Тому з'являються інші АСУ: управління виробництвом, логістикою, взаємовідносинами з клієнтами та постачальниками; розробляються ІС управління всією компанією, що дозволяють повністю перейти до електронного документообігу та автоматизувати всі сфери діяльності фірми.

Вісімдесяті роки XX ст. характеризуються появою спеціалізованих методологій проектування ІС і CASE–засобів. На їх основі розробляються перші програмні засоби, на базі персональних комп'ютерів створюють *децентралізовані* ІС. Різні персональні комп'ютери об'єднуються в локальну мережу. Цей період характеризується інтеграцією ІС і появою різних концепцій управління ними на єдиній методологічній основі. З появою персональних комп'ютерів відбувається децентралізація процесів управління, все частіше впроваджуються модулі з розподіленими системами обробки інформації. У компаніях та організаціях створюються інформаційні відділи та служби, обчислювальні центри і лабораторії.

Дев'яності роки XX ст. стали тріумфом персональних комп' ютерів. У цей час [5 с. 11]: 1) розробляються корпоративні ІС, які реалізують принципи розподіленої обробки даних; 2) стає можливим автоматизація всіх відділів і служб компаній; 3) з'являються системи електронного документообігу, в тому числі для підприємств з розвиненою мережею філій в різних містах і регіонах; 4) скорочуються терміни обробки даних і формування виробничих, складських та інших управлінських звітів.

У 1970–1980-х роках ІС інтегрувалися у комплексні АСУ, які вирішують завдання автоматизованого проектування нових виробів, технологічної підготовки виробництва та автоматизації організаційного управління

підприємством. Комплексні АСУ створювалися науково–дослідними інститутами.

У США Дуглас Т. Рос (SoftTech, Inc) розробив мову АПТ для роботи верстатів з ЧПУ, яка поклала початок розробкам графічних мов моделювання. А в 1969 р. він запропонував методологію SADT (IDEF0) для моделювання ІС середньої складності, у рамках програми ICAM (Інтеграція комп'ютерних і промислових технологій), яка проводилася департаментом Військово – Повітряних Сил США у рамках великого аерокосмічного проекту.

До кінця XX ст. було розроблено декілька десятків методів моделювання складних систем. Усі вони були різні за функціональними можливостями, але мали схожі підходи до аналізу та опису предметної області. Виникла необхідність об'єднати вдалі рішення в одну методику, яка влаштовувала велику частину розробників ІС. В результаті цих процесів була розроблена мова UML. У багатьох провідних компаніях, таких як Rational Software, Oracle Corporation, IBM, Microsoft, Hewlett–Packard, i–Logix існувала упевненість в необхідності створення подібної мови програмування. З цією метою був створений консорціум UML Partners. Провідну роль у створенні уніфікованої мови моделювання (UML) зіграли Гради Буч, Айвар Джекобсон і Джеймс Рамбо, працюючі в компанії Rational Software. Ними розроблені такі методи об'єктного моделювання складних ІС:

1. Метод об'єктного моделювання ПЗ складних ІС (метод Буча)
2. Метод аналізу вимог до бізнес–додатків (метод Джекобсона).
3. Метод аналізу обробки даних в складних ІС (метод Рамбо).

Попередня версія 0.8 уніфікованого методу програмування була випущена в жовтні 1995 р. Перша версія UML 0.9 вийшла в червні 1996 р. та отримала підтримку групи OMG, яка займалася розробкою єдиних стандартів у сфері web–технологій і включала декілька сотень компаній, працюючих в області IT– технологій та у виробництві комп'ютерної техніки.

У 2003 р. випустили версію UML 1.5, яка була прийнята як міжнародний стандарт ISO/IEC 19501–2005. В 2011 р. вийшла версія UML 2.4.1, оформлена у вигляді міжнародних стандартів ISO/IEC 19505–1, 19505–2. Для неї розроблені інструментальні засоби (наприклад, програма Rational Rose) підтримки та візуального програмування, що здійснюють пряму генерацію коду з моделей UML, в основному за допомогою мов програмування C++ і Java. Нині методології та засоби моделювання бізнес–процесів, процесно–орієнтованих методів аналізу і проектування ІС широко подані у більшості країн світу.

3. Методологія та методи проектування інформаційних систем

Основу технології проектування ІС становить методологія, яка визначає сутність, основні відмінні технологічні особливості. Методологія проектування передбачає наявність конкретної концепції, принципів проектування, які реалізуються множиною методів проектування, та повинні підтримуватися деякими засобами проектування. В процесі створення проекту ІС організація проектування передбачає визначення методів взаємодії проектувальників між собою та із замовником.

Методи проектування ІС можна класифікувати за ступенем: використання засобів автоматизації, використання типових проектних рішень, адаптивності до передбачуваних змін [11].

1. *За ступенем автоматизації* розрізняють такі методи:

- *ручного проектування*, при якому проектування компонентів ІС здійснюється без використання спеціальних інструментальних програмних засобів, а програмування виконується на алгоритмічних мовах;

- *комп'ютерного проектування*, яке передбачає генерацію або налаштування проектних рішень на основі використання спеціальних інструментальних програмних засобів.

2. *За ступенем використання типових проектних рішень* розрізняють такі методи: 1) *оригінального* (індивідуального) проектування, коли проектні

рішення розробляються «з нуля» відповідно до вимог ІС; 2) *типового* проектування (Додаток Г), який передбачає конфігурацію (налаштування) ІС з готових типових проектних рішень.

Оригінальне проектування ІС характеризується тим, що всі види проектних робіт орієнтовані на створення індивідуальних для кожного об'єкта проектів, які в максимальній мірі відображають всі його особливості. *Типове проектування* виконується на основі досвіду, отриманого при розробці індивідуальних проектів. Типові проекти (як узагальнення досвіду для деяких груп організаційно–економічних систем або видів робіт) в конкретному випадку пов'язані з специфічними особливостями та різняться за ступенем охоплення функцій управління, виконуваних робіт і розробляємої проектної документації.

3. *За ступенем адаптивності проектних рішень* розрізняють такі методи: 1) *реконструкції*, коли адаптація проектних рішень виконується шляхом переробки відповідних компонентів (перепрограмування програмних модулів); 2) *параметризації*, коли проектні рішення налаштовуються (перегенеруються) відповідно до змін параметрів; 3) *реструктуризації моделі*, коли змінюється модель проблемної області, на основі якої автоматично перегенеруються проектні рішення.

Поєднання різних ознак класифікації методів проектування обумовлює характер використовуваної ТП ІС, серед яких виділяються два основні класи: канонічна та індустріальна технології.

Для конкретних видів ТП властиво застосування засобів розробки ІС, які підтримують виконання як окремих проектних робіт, етапів, так і їх сукупностей. Тому перед розробниками ІС, зазвичай, стоїть завдання вибору засобів проектування, які за своїми характеристиками найбільшою мірою відповідають вимогам конкретного підприємства/організації.

4. Засоби проектування та їх класифікація

Засобами проектування ІС повинні бути такими: 1) інваріантними у своєму класі до об'єкта проектування; 2) охоплювати в сукупності всі етапи життєвого циклу ІС; 3) технічно, програмно та інформаційно сумісними; 4)простими в освоєнні і застосуванні; 5) економічно доцільними.

Засоби проектування ІС можна розділити на два класи: без використання ЕОМ, із використанням ЕОМ (автоматизовані) [11].

1. *Засоби проектування без використання* ЕОМ застосовують на усіх стадіях та етапах проектування ІС. Зазвичай, це є засоби організаційно–методичного забезпечення операцій проектування, які підтримують різні стандарти, що регламентують процес проектування систем. Сюди ж відносяться єдина система класифікації та кодування інформації, уніфікована система документації, моделі опису та аналізу потоків інформації тощо.

2. *Автоматизовані засоби проектування* можуть застосовуватися на окремих і на усіх стадіях та етапах процесу проектування ІС. Вони підтримують розробку елементів проекту, розділів проекту та проекту системи в цілому.

Засоби автоматизованого проектування ділять на чотири підкласи:

1. *До першого підкласу* відносяться операційні засоби, які підтримують проектування операцій обробки інформації. До даного підкласу засобів відносяться алгоритмічні мови, бібліотеки стандартних підпрограм і класів об'єктів, макрогенератори, генератори програм типових операцій обробки даних тощо, а також засоби розширення функцій операційних систем (утиліти). В цей клас включають також такі інструментальні засоби проектування, як засоби для тестування і налагодження програм, підтримки процесу документування проекту тощо. Особливість останніх програм полягає в тому, що з їх допомогою підвищується продуктивність праці проектувальників, але не розробляється закінчене проектне рішення.

2. *До другого підкласу* відносять засоби, що підтримують проектування окремих компонентів проекту ІС. До даного підкласу відносяться засоби загальносистемного призначення:

– системи управління базами даних (СУБД);

– методо–орієнтовані пакети прикладних програм (розв'язання задач дискретного програмування, математичної статистики тощо);

– табличні процесори;

– статистичні пакети прикладних програм (ППП);

– оболонки експертних систем;

– графічні редактори;

– текстові редактори;

– інтегровані ППП (інтерактивне середовище з вбудованими діалоговими можливостями, що дозволяє інтегрувати перераховані вище програмні засоби).

Для перерахованих засобів проектування характерним є їх використання для розробки технологічних підсистем ІС введення інформації та організації: 1) зберігання й доступу до даних, 2) обчислень, аналізу, відображення даних, 3) прийняття рішень.

3. *До третього підкласу* відносяться засоби, що підтримують проектування розділів проекту ІС. У цьому підкласі виділяють функціональні засоби проектування. *Функціональні засоби* спрямовані на розробку автоматизованих систем, що реалізують функції, комплекси завдань і завдання управління. Різноманітність предметних областей породжує різноманіття засобів цього підкласу, орієнтованих на: 1) тип організаційної системи (наприклад, промислова, непромислова сфери), 2) рівень управління (наприклад, підприємство, цех, відділ, ділянка, робоче місце), 3) функцію управління (наприклад, планування, облік тощо).

До функціональних засобів автоматизованого проектування систем обробки інформації відносяться типові проектні рішення, функціональні ППП, типові проекти.

4. *До четвертого підкласу* засобів проектування ІС відносяться засоби, які підтримують розробку проекту на стадіях та етапах процесу проектування. До даного класу відноситься підклас засобів автоматизації проектування ІС (CASE–засобу). Сучасні CASE–засоби класифікують в основному за двома ознаками: 1) за охопленими етапами процесу розробки ІС; 2) за ступенем інтегрованості: окремі локальні засоби (tools); набір неінтегрованих засобів, які охоплюють більшість етапів розробки ІС (toolkit); повністю інтегровані засоби, пов'язані загальною базою проектних даних – репозиторієм.

5. Методології моделювання предметної області

І. Структурна модель предметної області [8 с. 93–98]. В основі проектування ІС лежить моделювання предметної області. Щоб отримати адекватний предметній області проект ІС у вигляді системи правильно працюючих програм, необхідно мати цілісне, системне уявлення про модель, яке відображає усі аспекти функціонування майбутньої ІС. При цьому під *моделлю предметної області* розуміють деяку систему, яка імітує структуру або функціонування досліджуваної предметної області та відповідає основній вимозі – бути їй адекватною.

Попереднє моделювання предметної області дозволяє скоротити час і терміни проведення проектних робіт, отримати більш ефективний та якісний проект. Без проведення моделювання предметної області велика ймовірність допущення помилок у вирішенні стратегічних питань, що призводять до економічних втрат і високих витрат при подальшому перепроектуванні системи. Внаслідок цього усі сучасні технології проектування ІС ґрунтуються на використанні *методології моделювання предметної області.*

До моделей предметних областей висувають такі *вимоги*:

1) формалізація, яка забезпечує однозначний опис структури предметної області;

2) зрозумілість для замовників і розробників на основі застосування

графічних засобів відображення моделі;

3) реалізація, що передбачає наявність засобів фізичної реалізації моделі предметної області в ІС;

4) забезпечення оцінювання ефективності реалізації моделі предметної області на основі визначених методів та обчислюваних показників.

Для реалізації перерахованих вимог, зазвичай, будується система моделей, яка відображає структурний та оцінний аспекти функціонування предметної області у вигляді підприємства / організації.

1. ***Структурний аспект моделювання предметної області*** передбачає побудову:

1) *об'єктної структури*, яка відображує склад взаємодіючих у процесах матеріальних та інформаційних об'єктів предметної області;

2) *функціональної структури*, яка відображає взаємозв'язок функцій (дій) щодо перетворення об'єктів в процесах;

3) *структури управління*, яка відображає події та бізнес–правила, що впливають на виконання процесів;

4) *організаційної структури*, яка відображає взаємодію організаційних одиниць підприємства і персоналу в процесах;

5) *технічної структури*, яка описує топологію розташування і способи комунікації комплексу технічних засобів.

Для відображення структурного аспекту моделей предметних областей в основному використовують *графічні методи документування*, які повинні гарантувати подання інформації про компоненти системи. Графічні методи повинні забезпечувати можливість структурної декомпозиції специфікацій системи із максимальним ступенем деталізації і погоджень щодо описів на суміжних рівнях декомпозиції.

З моделюванням безпосередньо пов'язана проблема вибору мови зображення проектних рішень, що дозволяє якомога більше залучати майбутніх користувачів системи до її розробки.

Мова моделювання – це нотація, зазвичай графічна, яка використовується для опису проектів. Нотація є синтаксисом мови моделювання і має сукупність графічних об'єктів, які використовуються в моделі. Мова моделювання повинна 1) робити рішення проектувальників зрозумілими користувачеві, 2) надавати проектувальникам засоби формалізованого та однозначного визначення проектних рішень, які підлягають реалізації у вигляді програмних комплексів, що утворюють цілісну систему ПЗ.

Графічне зображення нерідко виявляється найбільш ємною формою подання інформації. При цьому проектувальники повинні враховувати, що графічні методи документування не можуть повністю забезпечити декомпозицію проектних рішень від постановки задачі проектування до реалізації програм на комп'ютері. Труднощі виникають при переході від етапу аналізу системи до етапу проектування та особливо до програмування.

Головний *критерій адекватності структурної моделі* предметної області полягає у функціональній повноті ІС, яка розробляється.

2***. Оціночні аспекти моделювання предметної області*** пов'язані із показниками ефективності автоматизованих процесів, що розробляються. До них відносять такі: 1) час вирішення завдань; 2) вартісні витрати на обробку даних; 3) надійність процесів; 4) непрямі показники ефективності (обсяги виробництва, продуктивність праці, оборотність капіталу, рентабельність тощо). Для розрахунку показників ефективності, зазвичай, використовують статичні методи функціонально–вартісного аналізу і динамічні методи імітаційного моделювання.

В основі різних методологій моделювання предметної області ІС лежать принципи послідовної деталізації абстрактних категорій. Зазвичай моделі будуються на трьох рівнях: 1) на зовнішньому рівні (визначення вимог), 2) на концептуальному рівні (специфікація вимог), 3) на внутрішньому рівні (реалізація вимог).

На *зовнішньому рівні* модель відповідає на запитання, що повинна робити

система? (тобто визначається склад основних компонентів системи: об'єктів, функцій, подій, організаційних одиниць, технічних засобів).

На *концептуальному рівні* модель відповідає на запитання «як повинна функціонувати система ? (визначається характер взаємодії компонентів системи одного і різних типів)».

На *внутрішньому рівні* модель відповідає на запитання «за допомогою яких програмно–технічних засобів реалізуються вимоги до системи ?».

З позиції життєвого циклу ІС описані рівні моделей будуються відповідно на етапах аналізу вимог, логічного (технічного) та фізичного (робочого) проектування.

Розглянемо *особливості побудови моделей предметної області щодо діяльності підприємства на різних рівнях деталізації.*

1. **Об'єктна структура.** Об'єкт – це сутність, яку використовують при виконанні деякої функції або операції (перетворення, обробка, формування тощо). Об'єкти можуть мати динамічну або статичну природу: динамічні об'єкти використовують в одному циклі відтворення, наприклад замовлення на продукцію, рахунки на оплату, платежі. *Статичні об'єкти* використовують у багатьох циклах відтворення, наприклад, обладнання, персонал, запаси матеріалів.

На зовнішньому рівні деталізації моделі виділяють:

– основні види матеріальних об'єктів (наприклад, сировина і матеріали, напівфабрикати, готові вироби, послуги),

– основні види інформаційних об'єктів або документів (наприклад, замовлення, накладні, рахунки тощо).

На концептуальному рівні побудови моделі предметної області уточнюється склад класів об'єктів, визначаються їх атрибути та взаємозв'язки. Таким чином будується узагальнене уявлення структури предметної області.

На внутрішньому рівні концептуальна модель відображається у вигляді файлів бази даних, вхідних і вихідних документів ІС (при цьому динамічні

об'єкти, наприклад економічної ІС, зображують одиницями змінної інформації або документами, а статичні об'єкти – одиницями умовно–постійної інформації у вигляді списків, номенклатур, цінників, довідників, класифікаторів).

Модель бази даних як постійно підтримуваний інформаційний ресурс відображає зберігання умовно–постійної і накопичуваної змінної інформації, яку використовують в повторюваних інформаційних процесах.

2. **Функціональна структура.** Функція (операція) має вигляд деякого перетворювача вхідних об'єктів у вихідні. Послідовність взаємопов'язаних по входах і виходах функцій становить *бізнес–процес*. Функція бізнес–процесу може породжувати об'єкти будь–якої природи (матеріальні, фінансові, інформаційні). Причому бізнес–процеси та інформаційні процеси, зазвичай, нерозривні, тобто функції матеріального процесу не можуть здійснюватися без інформаційної підтримки. Наприклад, відвантаження готової продукції здійснюється на основі документа «Замовлення», який, в свою чергу, породжує документ «Накладна», який супроводжує партію відвантаженого товару.

Функцію подають однією дією або сукупністю дій. В останньому випадку кожній функції може відповідати визначений процес, в якому можуть існувати свої підпроцеси, поки кожна з підфункцій не становитиме деяку послідовність дій, над якою не можна виконати декомпозицію.

На зовнішньому рівні моделювання визначається список основних бізнес–функцій або видів бізнес–процесів. На *концептуальному рівні* виділені функції декомпозують і будуються ієрархії взаємозалежних функцій. *На внутрішньому рівні* відображається структура інформаційного процесу в комп'ютері: визначаються ієрархічні структури програмних модулів, які реалізують функції, що підлягають автоматизації.

3. **Структура управління.** В залежності від різних умов перебігу бізнес–процесу у сукупності функцій цього процесу можливі альтернативні або циклічні послідовності. Ці умови пов'язані з подіями, які відбуваються у зовнішньому середовищі або у самих процесах і з виникненням конкретних

станів об'єктів (наприклад, замовлення прийняте, відкинуте, відправлене на коригування). Події викликають виконання функцій, які, в свою чергу, змінюють стан об'єктів і формують нові події, поки не буде завершено певний бізнес–процес. Тоді послідовність подій становить конкретну реалізацію бізнес–процесу.

Кожна подія описується з двох точок зору: інформаційної та процедурної.

1. *Інформаційна подія* відбивається у вигляді деякого повідомлення, яке фіксує факт виконання деякої функції зміни стану або появи нової.

2. *Процедурна подія* викликає виконання нової функції, і тому для кожного стану об'єкта повинні бути задані описи цих викликів.

Таким чином, події виступають в сполучній (зв'язуючій) ролі для виконання функцій бізнес–процесів.

На зовнішньому рівні визначаються: 1) список зовнішніх подій, які викликаються взаємодією підприємства із зовнішнім середовищем (платежі податків, відсотки за кредитами, поставки за контрактами тощо), 2) список цільових установок, яким повинні відповідати бізнес–процеси (регламент виконання процесів, підтримка рівня матеріальних запасів, рівень якості продукції тощо).

На концептуальному рівні встановлюються бізнес–правила, які визначають умови виконання функцій при виникненні подій і досягненні станів об'єктів.

На внутрішньому рівні виконується формалізація бізнес–правил у вигляді викликів програмних модулів.

4. **Організаційна структура.** Організаційна структура має вигляд сукупності організаційних одиниць, зазвичай, пов'язаних ієрархічними і процесними відношеннями. *Організаційна одиниця* – це підрозділ, який зображує собою об'єднання людей (персоналу) для виконання сукупності загальних функцій або бізнес–процесів. У функціонально–орієнтованій організаційній структурі організаційна одиниця виконує набір функцій, які

відносяться до однієї функції управління і входять до різних процесів. У процесно–орієнтованій структурі організаційна одиниця виконує набір функцій, які входять в один тип процесу і відносяться до різних функцій управління.

На зовнішньому рівні будується структурна модель підприємства у вигляді ієрархії підпорядкування організаційних одиниць або списків взаємодіючих підрозділів. *На концептуальному рівні* для кожного підрозділу задається організаційно–штатна структура посад (ролей персоналу). *На внутрішньому рівні* визначаються вимоги до прав доступу персоналу до функцій ІС, які підлягають автоматизації.

5. **Технічна структура.** Топологія визначає територіальне розміщення технічних засобів за структурними підрозділами підприємства, а комунікація – технічний спосіб реалізації взаємодії структурних підрозділів.

На зовнішньому рівні моделі визначаються типи технічних засобів обробки даних та їх розміщення по структурним підрозділам. *На концептуальному рівні* визначаються способи комунікацій між технічними комплексами структурних підрозділів: фізичне переміщення документів, машинних носіїв, обмін інформацією по каналах зв'язку тощо. *На внутрішньому рівні* будується модель «клієнт–серверної» архітектури обчислювальної мережі.

Описані моделі предметної області націлені на проектування окремих компонентів ІС: даних, функціональних програмних модулів, керуючих програмних модулів, програмних модулів інтерфейсів користувачів, структури технічного комплексу. Для більш якісного проектування зазначених компонентів потрібна побудова моделей, яка погоджує різні компоненти ІС між собою. У найпростішому випадку в якості таких моделей взаємодії можуть використовуватися матриці перехресних посилань: «функції – об'єкти», «функції – події», «організаційні одиниці – функції», «організаційні одиниці – об'єкти», «організаційні одиниці – технічні засоби» тощо. Такі матриці не

наочні і не відображають особливості реалізації взаємодій.

Для правильного відображення взаємодій компонентів ІС важливо здійснювати спільне моделювання таких компонентів, особливо із змістовної точки зору об'єктів і функцій. Методологія структурного системного аналізу істотно допомагає у вирішенні таких задач.

6. Функціонально– та об'єктно–орієнтовані методології структурного моделювання

Аналіз. Аналіз є першим етапом створення ІС, на якому вимоги замовника уточнюються, формалізуються й документуються. На цьому етапі дається відповідь на питання: «Що повинна робити майбутня система?».

Метою аналізу є перетворення загальних розпливчатих знань про початкову наочну сферу (предметну область) в точні визначення і специфікації, а також генерація функціонального опису системи. На цьому етапі специфікуються: зовнішні умови роботи системи; функціональна структура системи; розподіл функцій між людиною і системою, інтерфейси; вимоги до технічних, інформаційних і програмних компонентів системи; умови експлуатації.

Структурним аналізом прийнято називати метод дослідження системи, який розпочинається з її загального огляду, а потім деталізується, набуваючи вигляду ієрархічної структури з дедалі більшою кількістю рівнів. Для таких методів характерно: розбиття на рівні абстракції з обмеженим числом елементів (від 3 до 7); обмежений контекст, який включає тільки істотні деталі кожного рівня; використання чітких формальних правил запису; послідовне наближення до результату.

Структурний аналіз засновано на таких двох базових принципах: 1) декомпозиції, 2) ієрархічної впорядкованості. Вирішення важких проблем шляхом їх розбиття на множину менших незалежних завдань («чорних ящиків») та організація цих завдань в деревоподібні ієрархічні структури значно підвищують розуміння складних систем.

Для моделювання систем та об’єктів взагалі, зокрема для використання структурного аналізу, використовуються три групи засобів, що визначають такі компоненти: 1) функції, які система повинна виконувати; 2) відношення між даними; 3) поводження системи залежно від часу (поведінковий аспект).

Серед засобів вирішення цих завдань у методологіях структурного аналізу найбільш часто застосовують такі: 1) діаграми потоків даних: DFD (Data Flow Diagrams); 2) діаграми «сутність – зв'язок»: ERD (Entity–Relationship Diagrams); 3) діаграми станів переходів: STD (State Transition Diagrams).

Логічна DFD виконує такі функції:

– показує зовнішні (стосовно системи) джерела та витоки даних;

– ідентифікує логічні функції (процеси) і групи елементів даних;

– з’єднує одну функцію з іншими (за допомогою потоків);

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

Структури потоків даних та їх компонентів зберігаються й аналізуються у словнику даних. Зміст кожного сховища також зберігають у словнику даних. Модель даних сховища розкривається за допомогою ERD. У випадку наявності реального часу засоби DFD доповнюються засобами опису поводження системи залежного від часу, який визначається на основі діаграм станів переходів STD (рис. 4.1).

Визначимо ключові поняття структурного аналізу:

1. Операція – це елементарна (неподільна) дія, яка виконується на одному робочому місці.

2. Функція – це сукупність операцій, згрупованих за певною ознакою.

3. Бізнес–процес – це зв'язана сукупність функцій, під час виконання якої споживаються певні ресурси і створюється продукт (предмет, послуга, наукове відкриття, ідея), який є цінним для споживача.

4. Підпроцес – це бізнес–процес, який є структурним елементом конкретного бізнес–процесу і має цінність для споживача.

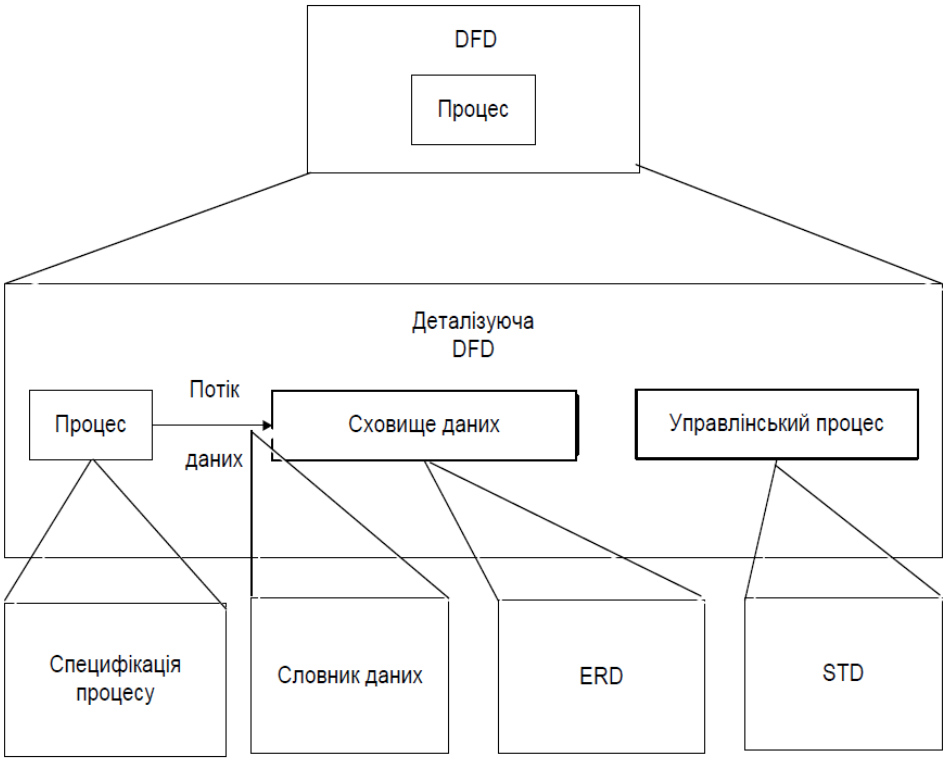


Рис. 4.1. Схема взаємодії діаграм у структурному аналізі

Бізнес–модель – це структурований графічний опис мережі процесів та операцій, пов'язаних з даними, документами, організаційними одиницями та іншими об'єктами, що відображають існуючу (або передбачувану) діяльність конкретного підприємства / організації, яке моделюють.

Існують різні *методології структурного моделювання предметної області*, серед яких слід виділити функціонально–орієнтовані та об'єктно–орієнтовані *методології опису предметної області* [8].

Процес бізнес–моделювання можна реалізувати в рамках різних методик, які різняться насамперед своїм підходом до того, що являє собою підприємство / організація. У відповідності з різними уявленнями про підприємство / організацію, методики прийнято ділити на об'єктні та функціональні (структурні).

Об'єктні методики розглядають організацію, яку моделюють, як набір взаємодіючих об'єктів – виробничих одиниць. Об'єкт визначається як

реальність – предмет (або явище), який має чітко визначену поведінку. Метою застосування даної методики є виділення об'єктів, які формують підприємство / організацію, і розподіл між ними відповідальностей за виконувані дії.

Функціональні методики, найвідомішою з яких є методика Integration Definition Metodology або IDEF (Додаток Д. Методології моделювання, засновані на графічному зображенні систем), розглядають організацію як набір функцій, що перетворює у вихідний потік потік інформації, який надходить (при цьому процес перетворення інформації споживає певні ресурси). Основна *відмінність функціональної методики від об'єктної* полягає у відділенні функцій (методів обробки даних) від самих даних.

З точки зору бізнес–моделювання кожен з наведених підходів має свої *переваги*: 1) *об'єктний підхід* дозволяє побудувати більш стійку до змін систему, яка найкраще відповідає існуючим структурам підприємства/ організації; 2) *функціональне моделювання* добре показує себе, коли організаційна структура знаходиться в процесі зміни або слабко оформлена.

Принципова *відмінність між функціональним та об'єктним підходом* полягає в способі декомпозиції системи. Об'єктно–орієнтований підхід використовує об'єктну декомпозицію, при цьому статична структура описується в термінах об'єктів і зв'язків між ними, а поведінка системи описується в термінах обміну повідомленнями між об'єктами. Метою методики є побудова бізнес–моделі суб'єкта господарювання, що дозволяє перейти від моделі сценаріїв використання до моделі, яка визначає окремі об'єкти, які беруть участь в реалізації бізнес–функцій.

Концептуальною основою об'єктно–орієнтованого підходу є об'єктна модель, яка будується з урахуванням таких принципів: абстрагування; інкапсуляція; модульність; ієрархія; типізація; паралелізм; стійкість. Основними поняттями об'єктно–орієнтованого підходу є об'єкт і клас.

1. *Об'єкт* – це предмет (або явище), який має чітко визначену поведінку і володіє станом та індивідуальністю. Структура й поведінка подібних об'єктів

визначають загальний для них клас.

2. *Клас* – це множина об'єктів, пов'язаних спільністю структури й поведінки.

До групи важливих понять об'єктного підходу належать успадкування і поліморфізм: 1) поняття *поліморфізм* можна інтерпретувати як здатність класу належати більш ніж одному типу; 2) *успадкування* означає побудову нових класів на основі існуючих з можливістю додавання або перевизначення даних і методів. За об'єктними моделями можна простежити відображення реальних сутностей модельованої предметної області (підприємства/організації) в об'єкти і класи ІС. Більшість існуючих методів об'єктно–орієнтованого підходу включають мову моделювання та опис процесу моделювання.

Процес – це опис кроків, які необхідно виконати при розробці проекту. В якості мови моделювання об'єктного підходу використовується *уніфікована мова моделювання UML*, яка містить стандартний набір діаграм для моделювання. *Діаграма* – це графічне подання множини елементів. Найчастіше вона зображується у вигляді зв'язного графа з вершинами (сутностями) і ребрами (відносинами) та формує деяку проекцію системи.

Об'єктно–орієнтований підхід має такі *переваги:*

1. *Об'єктна декомпозиція* дає можливість створювати моделі меншого розміру шляхом використання загальних механізмів, які забезпечують необхідну економію засобів виразу. Використання об'єктного підходу підвищує рівень уніфікації розробки і придатність для повторного використання, що веде до створення середовища розробки і переходу до створення моделей шляхом їх збирання.

2. *Об'єктна декомпозиція* дозволяє уникнути створення складних моделей, тому що передбачає еволюційний шлях розвитку моделі на основі відносно невеликих підсистем.

3. *Об'єктна модель* є природною, оскільки орієнтована на людське сприйняття світу.

До *недоліків* об'єктно–орієнтованого підходу відносяться високі початкові витрати (цей підхід не дає негайної віддачі). Ефект від його застосування з'являється після розробки двох–трьох проектів і накопичення повторно використовуваних компонентів. Діаграми, які відображають специфіку об'єктного підходу, менш наочні.

Серед причин, які призвели до успішного завершення об'єктно–орієнтованих проектів можна виділити такі.

1. Об'єктно–орієнтовані мови програмування дозволили отримати виграш 1) у продуктивності праці при створенні ПЗ та 2) у якості ПЗ.

2. *Надаючи більш формалізовані нотації для відображення і візуалізації абстракцій ПЗ, об'єктно–орієнтована технологія впливає на зменшення загального розміру того, що потрібно розробити*.

3. «*Об'єктно–орієнтована модель та її реалізація передбачають наявність загального словника у кінцевих користувачів системи та у її розробників, що призводить до розуміння всіма розв'язуваної проблеми*.

4. *Використання розтягнутої у часі інтеграції створює можливість раннього визначення ризику та внесення необхідних правок без дестабілізації процесу розробки*. Цей аспект об'єктно–орієнтованої технології надає можливість використовувати процес попереджувальної розробки архітектури, при якому інтеграція є ранньою і розтягнутою у часі процедурою ЖЦ.

5. *Об'єктно–орієнтована архітектура передбачає чіткий поділ елементів системи, створюючи надійний захист, який дозволяє при внесенні змін в одну частину системи зберегти незмінною архітектуру в цілому*». [Буч – Booch, 1996].

Порівняння існуючих методик. У функціональних моделях (DEF–діаграмах потоків даних, DEF0–діаграмах) головними структурними компонентами є функції (операції, дії, роботи), які на діаграмах зв'язуються між собою потоками об'єктів. *Перевагою функціональних моделей* є реалізація структурного підходу до проектування ІС за принципом «зверху–вниз», коли

кожен функціональний блок можна декомпозувати на множину підфункцій, виконуючи, таким чином, модульне проектування ІС. Для функціональних моделей характерні процедурна чіткість декомпозиції ІС і наочність подання.

При функціональному підході об'єктні моделі даних із застосуванням «природної мови» у вигляді діаграм «об'єкт□властивість□зв'язок» розробляються окремо. Для перевірки коректності моделювання предметної області між функціональними та об'єктними моделями встановлюють взаємно–однозначні зв'язки.

Головний недолік функціональних моделей полягає в тому, що процеси та дані існують окремо один від одного (крім функціональної декомпозиції існує структура даних, яка перебуває на другому плані). Не відомо умови виконання процесів обробки інформації, які можуть динамічно змінюватися.

Переваги об'єктно–орієнтованого підходу. Перераховані недоліки функціональних моделей зникають в об'єктно–орієнтованих моделях, в яких головним структурно–утворюючим компонентом виступає клас об'єктів з набором функцій, які можуть звертатися до атрибутів цього класу. Для класів об'єктів характерна ієрархія узагальнення, що дозволяє здійснювати спадкування не тільки атрибутів (властивостей) об'єктів від вищого класу до нижчого, але і функцій (методів). У разі спадкування функцій можна абстрагуватися від конкретної реалізації процедур (абстрактні типи даних). Це надає можливість звертатися до подібних програмних модулів за загальними іменами (поліморфізм) і здійснювати повторне використання програмного коду при модифікації ПЗ. Таким чином, адаптивність об'єктно–орієнтованих систем до змін предметної області в порівнянні з функціональним підходом значно вище.

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

Для об'єктно–орієнтованого підходу розроблено графічні методи моделювання предметної області, узагальнені в мові уніфікованого моделювання UML. Однак за наочністю подання моделі користувачеві–замовнику об'єктно–орієнтовані моделі поступаються функціональним моделям.

При виборі методики моделювання предметної області зазвичай в якості критерію виступає ступінь її динамічності. Для більш регламентованих завдань більше підходять функціональні моделі, для більш адаптивних бізнес–процесів (управління робочими потоками, реалізація динамічних запитів до інформаційних сховищ) – об'єктно–орієнтовані моделі. Однак в рамках однієї і тієї ж ІС для різних класів задач можуть вимагатися різні види моделей, які описують одну проблемну область. В такому випадку повинні використовуватися комбіновані моделі предметної області.

Синергетична методика. Кожна з розглянутих методик дозволяє вирішити задачу побудови формального опису робочих процедур досліджуваної системи. Всі методики дозволяють побудувати модель «як є» і «як повинно бути». З іншого боку, кожна з цих методик має недоліки.

Функціональні методики в цілому краще дають уявлення про існуючі функції в підприємстві / організації, про методи їх реалізації (при цьому чим вище ступінь деталізації досліджуваного процесу, тим краще вони дають можливість окреслити систему). Під кращим описом в даному випадку розуміється найменша помилка при спробі за отриманою моделлю передбачити поведінку реальної системи. На рівні окремих робочих процедур їх опис практично збігається з фактичною реалізацією в потоці робіт.

На рівні загального опису системи функціональні методики допускають значний ступінь свободи у виборі загальних інтерфейсів системи, її механізмів, тобто у визначенні меж системи.

Добре описати систему на цьому рівні дозволяє *об'єктний підхід*, заснований на понятті сценарію використання. При цьому під сценарієм

використання розуміють сеанс взаємодії дійової особи із системою, в результаті якої дійова особа отримує те, що має для неї цінність. Використання критерію цінності для користувача дає можливість відкинути деталі потоків робіт, які не мають значення, і зосередитися на тих функціях системи, які виправдовують її існування. Однак і в цьому випадку завдання визначення меж системи, виділення зовнішніх користувачів є складним.

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

Найкращим способом подолання недоліків розглянутих методик є формування *синергетичної методики*, яка об'єднує різні етапи окремих методик. При цьому з кожної методики необхідно взяти частину методології, найбільш повно і формально викладену, і забезпечити можливість обміну результатами на різних етапах застосування синергетичної методики. У бінес–моделюванні відбувається формування подібної синергетичної методики. Ідея синтетичної методики полягає в послідовному застосуванні функціонального та об'єктного підходу з урахуванням можливості *реінжинірингу* існуючої ситуації (Додаток А).

7. Основні методології проектування інформаційних систем

Серед основних методологій проектування ІС виділяють такі [5 с. 38–45]: методологія

1) функціонального моделювання робіт SADT (Structured Analysis and Design Technique – структурного аналізу і проектування),

2) швидкої розробки застосунків RAD (Rapid Application Development);

3) RUP (Rational Unified Process).

1. ***Методологія SADT.*** Методологія SADT, розроблена Дугласом Т. Росом в 1969–1973 рр., базується на структурному аналізі систем і графічному зображенні організації у вигляді системи функцій, які мають три класи структурних моделей: 1) функціональна модель, 2) інформаційна модель, 3)динамічна модель. Процес моделювання за методологією SADT складається з таких етапів:

1. Збирання інформації та її аналіз про предметну область.

2. Документування отриманої інформації.

3. Моделювання (IDEF0).

4. Корегування моделі в процесі ітеративного рецензування.

Методологія, відома як нотація IDEF0, використовує формалізований процес моделювання ІС і має такі стадії: аналіз, проектування, реалізація, об'єднання та тестування, встановлення (інсталяція), функціонування. Проектування ІС за стандартом IDEF0 зводиться до декомпозиції основних функцій організації на окремі бізнес–процеси, роботи або дії. В результаті розробляється ієрархічна модель аналізованої організації, при цьому декомпозицію можна проводити багаторазово, прямуючи до чіткого та детального опису усіх процесів.

Діаграми IDEF0 верхнього рівня прийнято називати батьківськими, а нижнього рівня – дочірніми. Приклад діаграми IDEF0 верхнього рівня наведений на рис. 4.2. Аналізований процес подають у вигляді прямокутника: ліворуч зображають вхідні дані; справа – вихідні; дані, розташовані зверху – дії, які управляють або регламентуючі, а знизу – об'єкти управління. У діаграмі IDEF0 спочатку описуються усі зовнішні зв'язки досліджуваного процесу. Після цього здійснюється декомпозиція цього процесу, і відбувається опис внутрішніх підпроцесів із позначенням усіх зв'язків. При цьому раніше позначені стрілками зовнішні зв'язки не повинні загубитися. Вони переносяться на діаграму декомпозиції у відповідні підпроцеси. Приклад декомпозиції діаграми IDEF0 (дочірньої діаграми) наведено на рис. 4.3.

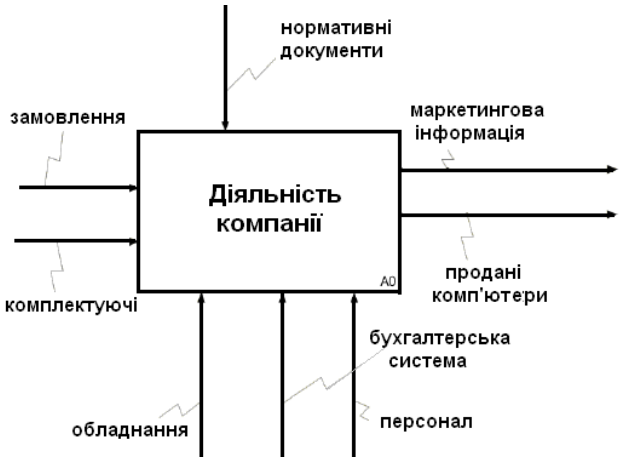


Рис. 4.2. Діаграма IDEF0 верхнього рівня

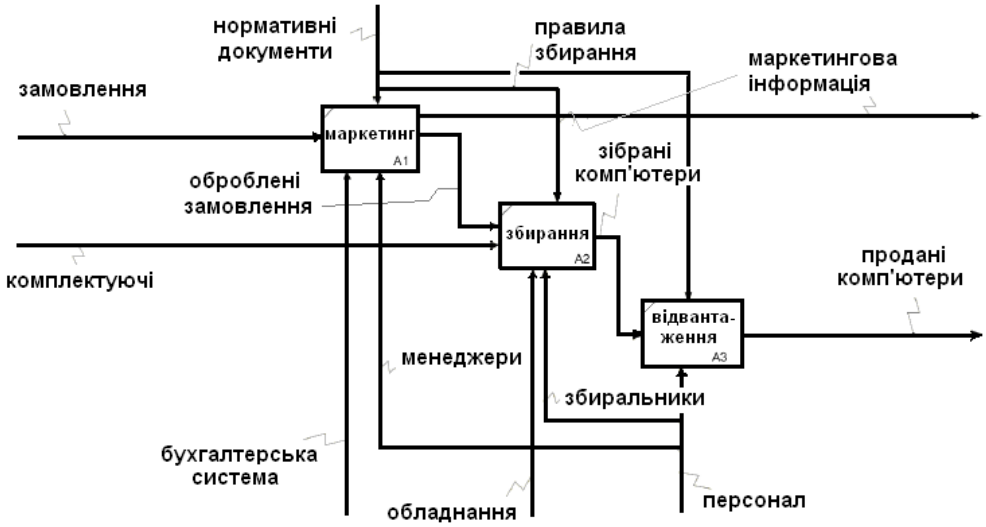


Рис. 4.3. Дочірня діаграма IDEF0 (декомпозиція)

Далі кожен підпроцес теж можна декомпозувати і детально описати усі зв'язки до необхідної межі.

Основною перевагою цієї методології є простота і наочність. В якості недоліку можна вважати неможливість описати реакцію описуваного процесу на зовнішні чинники, які змінюються. Для цих цілей використовують інші методології.

2. ***Методологія швидкої розробки застосунків*** **RAD**. Принципи RAD були сформульовані у 1980 р. співробітником компанії IBM Джеймсом

Мартіном і базувалися на ідеях Скотта Шульца і Барри Бойма. Підхід RAD передбачає наявність трьох складових:

1. *Невеликі групи розробників* (від 3 до 7 осіб), які виконують роботи з проектування окремих підсистем ПЗ (це обумовлено вимогою максимальної керованості колективу). Команда розробників повинна бути групою професіоналів, які мають досвід в проектуванні, програмуванні та тестуванні ПЗ, здатних добре взаємодіяти з кінцевими користувачами і трансформувати їх пропозиції в робочі прототипи.

2. *Короткий виробничий графік*, який ретельно пропрацювали (до трьох місяців).

3. *Використання інкрементного прототипування* – повторюваного циклу, при якому розробники в міру того, як додаток починає набувати форму, реалізують у продукті вимоги, отримані в результаті взаємодії із замовником.

Нині методологія RAD стала загальноприйнятою схемою для проектування і розробки ІС. Вона дозволяє на ранній стадії проектування ІС продемонструвати замовникові діючу інтерактивну модель системи–прототипу, уточнити проектні рішення та вимоги замовника, оцінити експлуатаційні характеристики.

Серед *основних принципів підходу* RAD можна виділити такі: 1) розробка додатків ітераціями; 2) необов'язковість повного завершення робіт на кожній стадії життєвого циклу ПЗ; 3) обов'язковість залучення користувачів у процес розробки; 4) застосування засобів управління конфігурацією, які полегшують внесення змін до проекту і супровід готової системи; 5) використання прототипування, що дозволяє повніше з'ясувати і задовольнити потреби користувачів; 6) тестування і розвиток проекту, здійснювані одночасно з розробкою; 7) ведення розробки нечисленною добре керованою командою професіоналів; 8) грамотне керівництво розробкою системи, чітке планування і контроль виконання робіт. Засоби розробки, грунтовані на RAD, є популярними

за рахунок використання таких програмних середовищ розробки: IBM Lotus Domino Designer, Borland C++ Builder, Microsoft Visual Studio тощо.

В методології RAD швидка розробка додатків досягається за рахунок використання компонентно–орієнтованого конструювання і застосовується, якщо: 1) бюджет проектованої ІС обмежений, 2) нечітко визначені вимоги до інформаційної системи, 3) потрібно реалізувати проект ІС в мінімальні терміни, 4) інтерфейс користувача можна продемонструвати в прототипі, 5) за функціональним призначенням проект можна розділити на складові елементи.

Методологія RAD охоплює такі *стадії*:

1. Моделювання інформаційних потоків між бізнес–функціями.

2. Моделювання даних.

3. Перетворення об'єктів даних, які забезпечують реалізацію бізнес–функцій.

4. Генерація застосунків.

5. Тестування та об'єднання.

Серед *недоліків методології RAD* можна виділити такі: вона 1) вимагає великий колектив розробників для великих ІС, 2) використовується для ІС, які можна декомпозувати на окремі модулі і в яких продуктивність не є критичною величиною, 3) не використовується у разі застосування нових технологій.

Життєвий цикл ПЗ відповідно до підходу RAD складається з чотирьох стадій: аналіз і планування вимог; проектування; реалізація; впровадження. RAD найкраще підходить для відносно невеликих проектів, що розробляються для конкретного замовника, його не застосовують для побудови складних розрахункових програм, операційних систем або програм управління складними об'єктами в реальному масштабі часу, тобто програм, що містять великий обсяг (сотні тисяч рядків) унікального коду, а також систем, від яких залежить безпека людей (наприклад, керування літаком або атомною електростанцією).

3. ***Методологія RUP.*** Серед усіх фірм–виробників CASE–засобів компанія IBM Rational Software Corp. одна з перших усвідомила перспективність розвитку об'єктно–орієнтованих технологій аналізу і проектування програмних систем. Ця компанія виступила ініціатором уніфікації мови візуального моделювання у рамках консорціуму OMG, що привело до появи перших версій мови UML. Ця ж компанія першою розробила інструментальний об'єктно–орієнтований CASE–засіб, в якому була реалізована мова UML, як базова нотація візуального моделювання. Графічне подання методології RUP з Вікіпедії зображено на рис. 1.4.

Однією з найпопулярніших технологій є Rational Unified Process (RUP), розроблена компанією Rational Software (яка нині входить до складу IBM). Авторами UML вважаються співробітники фірми Rational Software: Гради Буч, Айвар Якобсон, Джемс Рамбо. RUP відповідає стандартам, які визначають проектні роботи в процесі ЖЦ ІС.

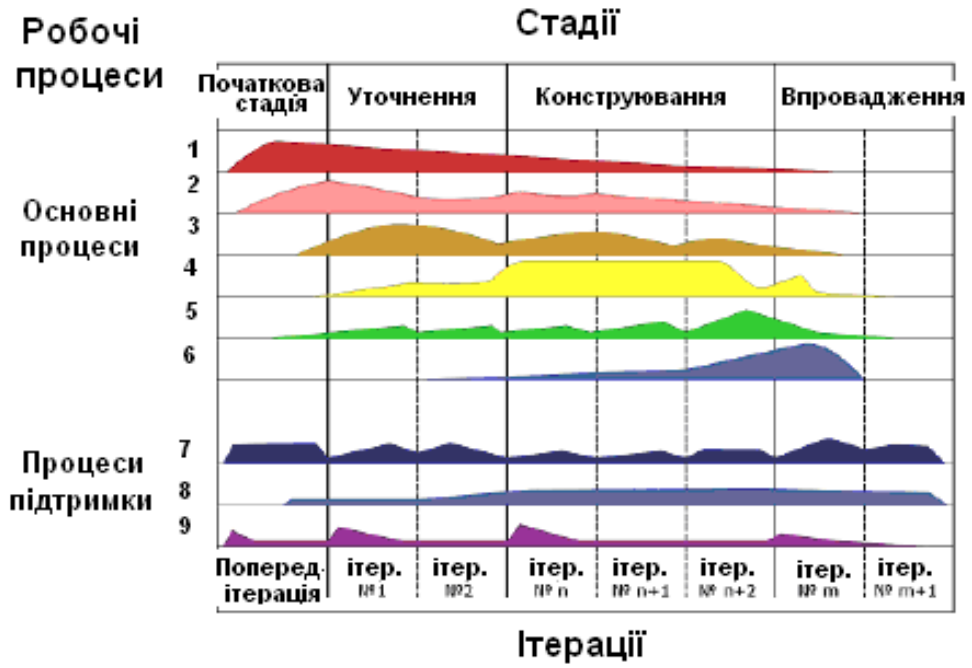


Рис. 1.4. Подання методології RUP:

1 – бізнес–моделювання, 2 – управління вимогами, 3 – аналіз і проектування, 4 – реалізація, 5 – тестування, 6 – розгортання, 7 – управління проектом, 8 – управління конфігурацією та змінами, 9 – створення інфраструктури (середовище розробки)

У методології RUP реалізуються такі підходи:

1. Ітераційний та інкрементний (нарощуваний).

2. Побудова системи на базі архітектури інформаційної системи.

3. Планування та управління проектом на основі функціональних вимог до ІС.

Розробка ІС виконується ітераціями: створюють окремі проекти, невеликі за об'ємом та змістом, які охоплюють свої власні етапи аналізу вимог, проектування, реалізації, тестування, інтеграції. Закінчуються ітерації створенням працюючої ІС.

Ітераційний цикл характеризується періодичним зворотним зв'язком і може адаптуватися до ядра системи, яка розробляється. Створювана ІС поступово росте та вдосконалюється.

Запитання для самоконтролю

1. Розкрийте сутність поняття «типове проектування» та назвіть методи типового проектування. Назвіть основні вимоги, які висувають до обраної ТП ІС. Назвіть ознаки, за якими класифікують методи проектування ІС.

2. Назвіть різновиди засобів автоматизованого проектування.

3. Назвіть вимоги, які висувають до моделей предметних областей.

4. Назвіть основні методології сучасного проектування ІС.

5. Що означає «швидка розробка ПЗ»?

5. ТЕХНОЛОГІЇ СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1. Поняття «технологія створення програмного забезпечення»

Технологія створення ПЗ – це «впорядкована сукупність взаємопов'язаних технологічних процесів в рамках ЖЦ ПЗ. *Технологічний процес* – це сукупність взаємопов'язаних технологічних операцій. *Технологічна операція* – це основна одиниця роботи, що виконується конкретною роллю, яка: 1) має чітко визначені обов’язки ролі (область її відповідальності); 2) дає чітко визначений результат (набір робочих продуктів), який базується на конкретних вихідних даних (іншому наборі робочих продуктів); 3) зображує собою одиницю роботи з чітко визначеними межами, які встановлюються при плануванні проекту» [3 с. 384].

Робочий продукт – це інформаційна або матеріальна сутність, яка створюється, модифікується або використовується в деякій технологічної операції (модель, документ, код, тест тощо). Робочий продукт визначає область відповідальності ролі та є об'єктом управління конфігурацією.

Роль – це визначення поведінки та обов'язків окремої особи або групи осіб в середовищі організації–розробника ПЗ, які здійснюють діяльність в рамках деякого технологічного процесу і відповідальні за визначені робочі продукти.

Керівництво – це практичний посібник з виконання однієї або сукупності технологічних операцій, який включає методичні матеріали, інструкції, нормативи, стандарти і критерії оцінки якості робочих продуктів.

Інструментальний засіб (CASE–засіб) – це програмний засіб, який забезпечує автоматизовану підтримку діяльності, що виконується в рамках технологічних операцій.

2. Загальні вимоги, які висувають до технології створення програмного забезпечення

Основною вимогою, яку пред'являють до сучасних технологій створення (ТС) ПЗ, є їх відповідність стандартам і нормативним документам, пов'язаним з

процесами ЖЦ ПЗ та оцінкою технологічної зрілості організацій–розробників (ISO 12207, ISO 9000, CMM та ін.). Згідно з цими нормативами ТС ПЗ *повинна підтримувати такі процеси*: управління вимогами; аналіз і проектування ПЗ; розробка ПЗ; експлуатація; супровід; документування; управління конфігурацією і змінами; тестування; управління проектом.

Повнота підтримки процесів ЖЦ ПЗ повинна підтримуватися комплексом інструментальних засобів (CASE–засобів).

Відповідність стандартам означає, зокрема, використання загальноприйнятих, стандартних нотацій та угод. Щоб проект міг виконуватися різними колективами розробників, необхідні (*вимоги*):

1. *Використання стандартних методів моделювання і нотацій, які повинні бути оформлені у вигляді нормативів до початку процесу проектування*. Невиконання проектних стандартів ставить розробників у залежність від фірми – виробника даного засобу, утруднює формальний контроль коректності проектних рішень, зменшує можливість залучення додаткових колективів розробників і зміни виконавців, оскільки кількість фахівців, знайомих із даним методом (нотацією), може бути обмеженою.

2. *Здатність до адаптації відповідно до умов використання*, що досягається за рахунок постачання технології в електронному вигляді разом з CASE–засобами та бібліотеками процесів, шаблонів, методів, моделей та інших компонентів, призначених для побудови ПЗ того класу систем, на який орієнтована технологія.

Електронні технології повинні включати засоби, які забезпечують їх адаптацію та розвиток за результатами виконання конкретних проектів. Процес адаптації полягає у видаленні непотрібних процесів і дій ЖЦ ПЗ, у зміні непридатних або додаванні власних процесів і дій, методик, стандартів і керівництв.

3. Деякі приклади технологій створення програмного забезпечення

На сьогоднішній день практично всі провідні компанії – розробники технологій і програмних продуктів (IBM, Microsoft, Oracle, Computer Associates, Sybase та ін.) мають у своєму розпорядженні розвинені технології створення ПЗ, створені власними силами і за рахунок придбання продуктів і технологій невеликих спеціалізованих компаній. Серед них можна виділити такі технології: RUP (Rational Unified Process), Computer Associates [3 с. 442].

3.1. Технолоія Rational Unified Process (RUP). Це – програмний продукт, розроблений компанією Rational Software (www.rational.com), яка входить до складу IBM. Технологія RUP в значній мірі відповідає стандартам і нормативним документам, пов'язаним з процесами ЖЦ ПЗ, з оцінкою технологічної зрілості організацій–розробників (ISO 12207, ISO 9000, CMM та ін.). Основними *принципами* цієї технології є:

1. Ітераційний та інкрементний (нарощуваний) підхід до створення ПЗ.

2. Планування та управління проектом на основі функціональних вимог до системи – варіантів використання.

3. Побудова системи на базі архітектури ПЗ.

Перший принцип є визначальним. Відповідно до нього розробка системи виконується у вигляді декількох короткострокових міні–проектів фіксованої тривалості (від 2 до 6 тижнів), названих ітераціями. Кожна ітерація включає власні етапи аналізу вимог, проектування, реалізації, тестування, інтеграції та завершується створенням працюючої системи. Ітераційний цикл ґрунтується на постійному розширенні та доповненні системи в процесі декількох ітерацій з періодичним зворотним зв'язком та адаптацією модулів, які додаються до існуючого ядра системи. Система постійно розростається крок за кроком, тому такий підхід називають ітераційним та інкрементним. При цьому підході виключена побудова моделей без зворотного зв'язку.

Динамічний аспект. Згідно технології RUP ЖЦ ПЗ розбивається на окремі цикли, в кожному з яких створюється нове покоління продукту. Кожен

цикл, в свою чергу, розбивається на чотири *послідовні стадії*: 1) початкова стадія (inception); 2) стадія розробки (elaboration); 3) стадія конструювання (construction); 4) стадія введення в дію (transition). Кожна стадія завершується в чітко визначеній контрольній точці. У цей момент часу повинні досягатися важливі результати і прийматися критично важливі рішення про подальшу розробку.

1. *Початкова стадія проекту* та *стадія розробки*. Початкова стадія може приймати різні форми. Для великих проектів початкова стадія передбачає всебічне вивчення усіх можливостей реалізації проекту, що займе місяці. Під час її перебігу: 1) виробляється бізнес–план проекту: визначається скільки приблизно він буде коштувати та який дохід принесе; 2) визначається межа проекту, і виконується початковий аналіз для оцінки розмірів проекту.

Щоб виконати таку роботу, необхідно ідентифікувати всі зовнішні сутності (дійові особи), з якими система буде взаємодіяти, і визначити в найзагальнішому вигляді природу цієї взаємодії. При цьому мають на увазі ідентифікацію всіх варіантів використання та опис найбільш важливих з них. Бізнес–план включає критерії успіху, оцінку ризику, оцінку необхідних ресурсів і загальний план по стадіях, який включає дати основних контрольних точок.

Результатом початкової стадії є:

– загальний опис системи: основні вимоги до проекту, його характеристики та обмеження;

– початкова модель варіантів використання (ступінь готовності – 10–20%);

– початковий проектний глосарій (словник термінів);

– початковий бізнес–план;

– план проекту, який відображає стадії та ітерації;

– один або кілька прототипів.

На стадії розробки більш детально визначають вимоги до системи, виконується високорівневий аналіз предметної області і проектування для побудови базової архітектури системи, створюється план конструювання та усуваються найбільш ризиковані елементи проекту.

Результатом стадії розробки є:

– модель варіантів використання (завершена принаймні на 80%), яка визначає функціональні вимоги до системи;

– перелік додаткових вимог, включаючи вимоги нефункціонального характеру і вимоги, не пов'язані з конкретними варіантами використання;

– опис базової архітектури майбутньої системи;

– працюючий прототип;

– уточнений бізнес–план;

– план розробки всього проекту, який відображає ітерації та критерії оцінки для кожної ітерації.

Найважливішим результатом стадії розробки є опис базової архітектури майбутньої системи, яка є основою всієї подальшої розробки. Ця архітектура включає: 1) модель предметної області, яка відображає розуміння бізнесу і служить відправним пунктом для формування основних класів предметної області; 2) технологічну платформу, яка визначає основні елементи технології реалізації системи та їх взаємодію.

2. *Стадія розробки* займає близько п'ятої частини загальної тривалості проекту. Основними ознаками завершення стадії розробки є дві події:

1) розробники в змозі оцінити з достатньо високою точністю, скільки часу потрібно на реалізацію кожного варіанту використання;

2) ідентифіковано всі найбільш серйозні ризики, а ступінь розуміння найважливіших з них така, що відомо, як впоратися з ними.

Сутність планування полягає у визначенні послідовності ітерацій конструювання і варіантів використання, які реалізуються на кожній ітерації.

Планування завершується, коли визначено місце кожного варіанта використання на деякій ітерації та дата початку кожної ітерації.

3. Технологія RUP має вигляд ітераційного і покрокового процесу розробки, в якому ПЗ розробляється і реалізується вроздріб.

На стадії конструювання побудова системи виконується шляхом серії ітерацій. Кожна ітерація є своєрідним міні–проектом, під час реалізації якого для конкретних варіантів використання виконують аналіз, проектування, кодування, тестування та інтеграцію. Ітерація завершується демонстрацією результатів користувачам і виконанням системних тестів з метою контролю правильності реалізації варіантів використання. Призначенням цього процесу є зниження ступеня ризику. Причиною появи ризику часто є відкладання вирішення складних проблем на самий кінець проекту.

Тестування та *інтеграція* – це завдання, які завжди займають більше часу, ніж очікується. Чим пізніше виконувати тестування та інтеграцію, тим важчими завданнями вони стають і тим більше здатні дезорганізувати весь проект. При ітеративній розробці на кожній ітерації виконується весь процес, що дозволяє оперативно справлятися з усіма проблемами, які виникають.

Ітерації на стадії конструювання є одночасно інкрементними і повторюваними: 1) ітерації є інкрементними відповідно до тієї функцією, яку вони виконують; кожна ітерація додає чергові конструкції до варіантів використання, реалізованих під час попередніх ітерацій; 2) ітерації є повторюваними по відношенню до розробляємого коду; на кожній ітерації деяка частина існуючого коду переписується щоб зробити його більш гнучким.

Процес інтеграції повинен бути неперервним: в кінці кожної ітерації повинна виконуватися повна інтеграція. Додатки слід інтегрувати після виконання кожної значної частини роботи. Під час кожної інтеграції повинен виконуватися повний набір тестів.

Головна особливість ітераційної розробки полягає в тому, що вона обмежена часовими рамками, і неприпустиме перенесення термінів. Винятком

може бути перенесення реалізації будь–яких варіантів використання на більш пізню ітерацію за угодою із замовником. Сенсом таких обмежень є підтримувати сувору дисципліну розробки і не допускати перенесення термінів. При цьому якщо на більш пізній термін перенесено занадто багато варіантів використання, то необхідно коригувати план, переглянувши при цьому оцінку трудомісткості реалізації варіантів використання.

Крім конструювання, ітерації можуть бути присутніми в усіх стадіях, проте конструювання – це ключова стадія. Результатом стадії конструювання є продукт, готовий до передачі кінцевим користувачам. Як мінімум, він містить такі складові: 1) ПЗ, інтегроване на необхідних платформах; 2) керівництво користувача; 3) опис поточної реалізації.

Призначенням стадії введення в дію є передача готового продукту для розпорядження користувачам. Ця стадія включає:

– бета–тестування, яке дозволяє переконатися, що нова система відповідає очікуванням користувачів;

– паралельне функціонування з існуючою системою, яка підлягає поступовій заміні;

– конвертацію баз даних;

– оптимізацію продуктивності;

– навчання користувачів і фахівців служби супроводу.

Головна ідея ітераційної розробки – поставити весь процес розробки на регулярну основу, щоб команда розробників змогла отримати кінцевий продукт. Однак деякі речі не слід виконувати занадто рано, наприклад оптимізацію. Рання оптимізація ускладнить подальшу розробку, тому її слід виконувати в останню чергу.

4. На *стадії введення в дію* продукт не доповнюється ніякою новою функціональністю (крім мінімальної та абсолютно необхідної), тут тільки виловлюють помилки. Прикладом для стадії введення в дію може служити період часу між випуском бета–версії та остаточної версії продукту.

Статичний аспект. Статичний аспект RUP подано чотирма основними елементами: ролі; види діяльності; робочі продукти; дисципліни.

Поняття «роль» визначає поведінку та відповідальність особи або групи осіб, які формують команду проекту. Одна особа може грати в проекті різні ролі.

Під *діяльністю конкретного виконавця* розуміють одиницю виконуваної ним роботи. Різновид діяльності відповідає поняттю технологічної операції. Він має чітко визначену мету, яку зазвичай виражають в термінах отримання або модифікації деяких робочих продуктів (таких, як модель, елемент моделі, документ, вихідний код або план).

Кожен вид діяльності пов'язаний з конкретною роллю. Тривалість виду діяльності становить від декількох годин до декількох днів, вона зазвичай виконується одним виконавцем і породжує один робочий продукт (або їх невелику кількість). Будь–який вид діяльності повинен бути елементом процесу планування. Прикладами видів діяльності можуть бути такі: планування ітерації, визначення варіантів використання та дійових осіб, виконання тесту на продуктивність. Кожен вид діяльності супроводжується набором керівництв, які мають вигляд методики виконання технологічних операцій.

Дисципліна відповідає поняттю технологічного процесу і має вигляд послідовності дій, яка приводить до отримання відчутного результату.

В рамках RUP визначено:

1) *шість основних дисциплін*: побудова бізнес–моделей; визначення вимог; аналіз і проектування; реалізація; тестування; розгортання;

2) *три допоміжні*: управління конфігурацією та змінами; управління проектом; створення інфраструктури.

RUP як продукт входить до складу комплексу Rational Suite, причому кожна з перерахованих вище дисциплін підтримується конкретним інструментальним засобом комплексу.

Одним із основних інструментальних засобів комплексу Rational Rose є група об'єктно–орієнтованих CASE–засобів, призначених для автоматизації процесів аналізу і проектування ПЗ, для генерації кодів на різних мовах і випуску проектної документації.

Rational Rose реалізує процес об'єктно–орієнтованого аналізу і проектування ПЗ, описаний в RUP. В основі роботи Rational Rose лежить побудова діаграм і специфікацій UML, які визначають архітектуру системи, її статичні і динамічні аспекти. У складі Rational Rose можна виділити *шість основних структурних компонентів*: репозиторій, графічний інтерфейс користувача, засоби перегляду проекту (браузер), засоби контролю проекту, засоби збору статистики і генератор документів. До них додаються генератори кодів для кожної підтримуваної мови, склад яких змінюється від версії до версії.

Репозиторій є базою даних проекту. Браузер забезпечує «навігацію» за проектом, в тому числі переміщення по ієрархії класів і підсистем, переключення від одного виду діаграм до іншого тощо. Засоби контролю і збору статистики дають можливість знаходити та усувати помилки у міру розвитку проекту, а не після завершення його опису. Генератор звітів формує тексти вихідних документів на основі інформації, яка міститься в репозиторії.

Засоби автоматичної генерації коду, використовуючи інформацію, яку містять діаграми класів і компонентів, формують файли описів класів. Створюваний таким чином скелет програми можна уточнювати шляхом прямого програмування на відповідній мові (основні мови, підтримувані Rational Rose – C++ і Java).

В результаті розробки проекту за допомогою Rational Rose формуються такі документи: 1) діаграми UML, які в сукупності зображують собою модель програмної системи, яку розробляють; 2) специфікації класів, об'єктів, атрибутів та операцій; 3) заготовки текстів програм.

Тексти програм є заготовками для подальшої роботи програмістов. Склад інформації, яку включають в програмні файли, визначається або за замовчуванням, або за бажанням користувача. Надалі ці вихідні тексти розвиваються програмістами в повноцінні програми.

Інструментальний засіб Rational XDE є розвитком можливостей Rational Rose в частині синхронізації моделі і коду (яка виключає необхідність прямої і зворотної генерації коду). Rational XDE забезпечує:

– синхронізацію між кодом і моделлю;

– відображення елементів коду Java і C # в UML;

– автоматичну синхронізацію з налаштованим врегулюванням конфліктів;

– одночасне відображення коду і моделі;

– постійну синхронізацію моделі UML як частини проекту Java або C #.

3.2. Технологія Computer Associates. Компанія Computer Associates (www.ca.com) пропонує такі комплекси інструментальних засобів підтримки різних процесів ЖЦ ПЗ:

I. AllFusion Modeling Suite – інтегрований комплекс CASE–засобів, який включає такі продукти:

AllFusion Process Modeler (BPwin) – функціональне моделювання;

AllFusion ERwin Data Modeler (ERwin) – моделювання даних;

AllFusion Component Modeler (Paradigm Plus) – об'єктно–орієнтований аналіз і проектування з використанням UML і можливістю генерації коду;

AllFusion Model Manager (Model Mart) – організація спільної роботи команди розробників;

AllFusion Data Model Validator (ERwin Examiner) – перевірка структури та якості моделей даних.

II. AllFusion Change Management Suite – комплекс засобів управління конфігурацією і змінами.

III. AllFusion Process Management Suite – засоби управління процесами і проектами для різних типів додатків.

CASE–засобу ERwin і BPwin були розроблені фірмою Logic Works, яка в 1998р. увійшла до складу PLATINUM Technology, а потім Computer Associates.

BPwin – засіб моделювання бізнес–процесів, який реалізує метод IDEF0, а також підтримує діаграми потоків даних і IDEF3. У процесі моделювання BPwin дозволяє переключитися з нотації IDEF0 на будь–якій гілці моделі на нотацію IDEF3 або DFD і створити змішану модель. BPwin підтримує функціонально–вартісний аналіз (ABC).

Група продуктів ERwin є набором засобів концептуального моделювання даних, які використовують метод IDEF1X. ERwin реалізує проектування схеми БД, генерацію її опису на мові цільової СУБД (Oracle, Sybase, DB2, Microsoft SQL Server тощо) і реверсний інжиніринг існуючої БД. ERwin випускається в декількох конфігураціях, орієнтовані на найпоширеніші засоби розробки додатків.

Для керування груповою розробкою використовується засіб Model Mart, що забезпечує багатокористувацький доступ до моделей, створених за допомогою ERwin і BPwin. Моделі зберігаються на центральному сервері і доступні для всіх учасників групи проектування.

Model Mart задовольняє ряду вимог, які пред'являються до засобів управління розробкою великих систем, а саме:

1. *Спільне моделювання*. Кожен учасник проекту має інструмент пошуку і доступу до моделі, яка цікавить його в будь–який час. При спільній роботі використовуються три режими: незахищений, захищений і режим перегляду (коли забороняється будь–яка зміна моделей).

2. *Створення бібліотек рішень*. Model Mart дозволяє 1) формувати бібліотеку стандартних рішень, які включають найбільш вдалі фрагменти реалізованих проектів, 2) накопичувати і використовувати типові моделі, об'єднуючи їх при необхідності під час «збирання» великих систем. На основі існуючих баз даних за допомогою ERwin можна відновлювати моделі

(реверсний інжиніринг), які в процесі аналізу придатності їх для нової системи можуть об'єднуватися з типовими моделями з бібліотек моделей.

3. *Управління доступом*. Для кожного учасника проекту визначаються права доступу, відповідно до яких він отримує можливість працювати тільки з конкретними моделями. Права доступу можуть бути визначені як для груп, так і для окремих учасників проекту. Роль спеціалістів, що беруть участь в різних проектах, може змінюватися, тому в Model Mart можна визначати та управляти правами доступу учасників проекту до бібліотек, моделей і навіть до специфічних областей моделі.

Підсумок. 1. *Технологія створення* ПЗ – це впорядкована сукупність взаємопов'язаних технологічних процесів в рамках ЖЦ ПЗ.

Технологічний процес – це сукупність взаємопов'язаних технологічних операцій.

Технологічна операція – це основна одиниця роботи, що виконується конкретною роллю, яка: 1) має чітко визначені обов’язки; 2) дає чітко визначений результат (набір робочих продуктів), який базується на конкретних вихідних даних (іншому наборі робочих продуктів); 3) має вигляд одиниці роботи з чітко визначеними межами, які встановлюються при плануванні проекту. *Роль* передбачає визначення поведінки та обов'язків окремої особи (або групи осіб) в середовищі організації–розробника ПЗ, яка здійснює діяльність в рамках деякого технологічного процесу і відповідальна за визначені робочі продукти.

2. Основною вимогою, яку пред'являють до сучасних ТС ПЗ, є їх відповідність стандартам і нормативним документам, пов'язаним з процесами ЖЦ ПЗ та із оцінкою технологічної зрілості організацій–розробників (ISO 12207, ISO 9000, CMM та ін.).

Запитання для самоконтролю

1. Визначіть поняття «технологія створення ПЗ», охарактеризуйте систему понять, які описують ТС ПЗ. Які поняття є найбільш важливими? Назвіть можливості ТС ПЗ.

2. Які вимоги пред'являють до сучасних ТС ПЗ і є найбільш важливими, чому?

3. Охарактеризуйте принципи і сферу застосування методики аналізу і проектування Rational Unified Process.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Брауде Э. Дж. Технология разработки программного обеспечения. – СПб: Питер, 2004. – 655 с.

2. Брукс Ф. Мифический человеко–месяц, или как создаются программные системы. – СПб.: Символ–Плюс, 2006. – 304 с.

3. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: учебник. – 2–е изд., перераб. и доп. – М.: Финансы и статистика, 2005. – 544 с: ил.

4. Илес П. Что такое архитектура программного обеспечения? – Ресурс: http://www.interface.ru/home.asp?artId=2116

5. Коцюба И.Ю., Чунаев А.В., Шиков А.Н. Основы проектирования информационных систем: уч. пособие. – СПб:Университет ИТМО, 2015. – 206с.

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

7. Пущин М.Н. Проектирование информационных систем: Учеб. пособие. – М: Изд–во МИЭТ, 2008. – 234 с.

8. Проектирование информационных систем: учеб. пособие / В.И. Грекул, Г. Н. Денищенко, Н. Л. Коровкина. – М. Интернет–Ун–т Информ. технологий, 2005. – 304 с.

9. Реінжиніринг бізнес–процесів. – Ресурс: https://library.if.ua/book/28/1899.html

10. Соммервилл И. Инженерия программного обеспечения, 6–е издание.: Пер. с англ. – М.: Вильямс, 2002. – 624 с.

11. Технологии проэктирования. – Ресурс: https://studfile.net/preview/3997729/page:5

12. Фаулер М. и др. Архитектура корпоративных программных приложений. – М.: Вильямс, 2016. – 548 с.

13. Фаулер М. UML в кратком изложении: Применение стандартного языка объектного моделирования / М. Фаулер, К. Скотт; Пер. с англ. – М.: Мир, 1999. – 191 с

ДОДАТКИ

Додаток А. Реінжиніринг бізнес–процесів

Одна з нових концепцій розвитку бізнесу та управління базується на системі *реінжинірингу бізнес–процесів* (РБП), створеній в 90–х роках XX ст. і прийнятої на озброєння багатьма провідними компаніями світу.

Реінжиніринг бізнес–процесів бере свій початок від двох статей, написаних у 1990 р. Хаммером (Hammer) та Давенпортом і Шортом (Davenport and Short).

У 1980–х роках у багатьох організаціях було впроваджено систему *Тотальної якості*, яка стала родоначальником ідеї управління процесами. Багато методів системи Тотальної якості змушують по–новому подивитися на роботу та на її мету, і ці методи дають цінну інформацію для управління бізнес–процесами (Business Process Management). До складу цих методів належать: 1) метод аналізу процесів (Method for Analyzing Processes), 2) внутрішньо-фірмова оцінка діяльності (In Department Evaluation of Activity), 3) аналіз сприйняття процесів (Process Perception Analysis, PPA), 4) управління якістю процесів (Process Quality Management). Усі вони відіграють важливу роль у будь–якому масштабному проекті з реінжинірингу бізнес–процесів.

Реінжиніринг – це фундаментальне переосмислення і радикальне перепроектування бізнес–процесів з метою досягнення істотного поліпшення якості функціонування.

Реінжиніринг бізнес–процесів – це сукупність методів і засобів, призначених для кардинального поліпшення основних показників діяльності підприємства шляхом моделювання, аналізу і перепроектування існуючих бізнес–процесів. Реінжиніринг бізнес–процесів застосовується за необхідності радикальних змін, яка передбачає створення нових, більш ефективних бізнес–процесів на підприємстві без урахування їх старої організації.

Визначення РБП містять в собі три ключові характеристики: «істотне поліпшення», «радикальне перепроектування» та «бізнес–процес». Саме розкриття змісту цих ключових слів приводить до розуміння суті реінжинірингу.

1. *Істотне поліпшення* – це не просто підвищення певного параметру підприємства або покращання роботи окремої ланки організації на визначений відсоток. Це якісний перехід до нового рівня ефективності бізнесу, здійснення прориву. Критерій реінжинірингу – разючий прорив в ефективності функціонування.

2. *Радикальне перетворення* означає звернення до першопричин речей. Тобто реінжиніринг – не поліпшення існуючого положення речей, не проведення косметичних заходів та часткових змін, не перетасування вже існуючих систем функціонування організації. Це відмова від того, що було раніше, новий винахід того, як робота повинна бути виконана.

3. *Бізнес–процес*. Під «бізнес–процес» розуміють групу взаємозалежних завдань, що спільно створюють цінність для споживача. Споживачу однаково, яким чином всередині підприємства організовано виконання тих або інших завдань/робіт, – для нього важлива тільки та цінність, яку він одержує у вигляді продукту і послуг підприємства. У традиційній організації процеси, проходячи через численні організаційні підрозділи, є фрагментовані, приховані і по суті некеровані. Проте процеси є першоосновою організації, засобом створення цінності для споживачів.

Отже, реінжиніринг побудовано на системі докорінних перетворень в організації. Реінжиніринг має коріння в інженерному підході до науки управління. Його сутність – це спочатку моделювання організації, а потім зміна цієї моделі під рішення конкретних поточних та перспективних завдань (найчастіше шляхом рішучого відрубування нераціональних ланок та функцій).

РБП втілюється стрибкоподібно, у великих масштабах, згори вниз по організаційній структурі. Цей підхід дає можливість радикального поновлення отримання результату за рахунок створення нових технологій бізнес–процесів.

Реінжиніринг базується на використанні сучасних інформаційних технологій для досягнення нових ділових цілей.

Проведення РБП є складним процесом, який потребує часу та концентрації зусиль всієї організації. Як і будь–який глобальний процес, який впливає на життєдіяльність підприємства та радикально змінює способи його функціонування, РБП підкоряється певним принципам проведення. Окрім принципів проведення, РБП має відповідати певним вимогам. Рушійною силою реінжинірингу є розуміння потреб клієнта підприємства, тобто погляд на підприємство з позиції клієнта. Такий підхід часто призводить до проектування нових процесів, які раніше не існували в організації.

Основною метою РБП вважається забезпечення виживання підприємства в екстремальній ситуації, різке прискорення його реакції на зміни у вимогах споживачів. Цілі РБП такі: 1) суттєве підвищення ступеня задоволення споживача, включаючи роботу з клієнтом у режимі 24 × 365, орієнтацію на його поточні і майбутні потреби; 2) радикальне скорочення тривалості виробничого циклу, докорінне зменшення кількості процесів та їх вартості, різке зниження витрат часу на виконання функцій; 3) значне поліпшення процесу управління якістю; 4) підвищення ролі рішень та ініціативи кожного окремого працівника, організація групової роботи; 5) різке зниження кількості працівників; 6) забезпечення прискореного впровадження нових технологій; 7) забезпечення адаптації підприємства до функціонування в умовах інформаційного суспільства і «суспільства знань».

Досягнення поставлених цілей забезпечує набір організаційних, методичних та інформаційних компонентів реінжинірингу:

– стратегія фірми, орієнтована на перспективні потреби клієнта;

– новий набір бізнес–правил або бізнес–процедур, який дозволить знизити витрати, зменшити час прийняття рішень;

– нова організаційна структура управління;

– нові умови праці персоналу, нові обсяги прав та ресурсів робітників;

– новий підхід щодо отримання інформації від споживачів;

– забезпечення функціонування всіх попередніх процедур та структур за допомогою інформаційних систем та новітніх інформаційних технологій.

Наслідки реінжинірингу бізнес–процесів полягають у такому:

– відбувається перехід від функціональної структури підрозділів до горизонтальної, яка дозволяє вирішити проблему неузгодженості або протиріч між цілями діяльності різних функціональних підрозділів;

– робота виконавця стає багатоплановою; відбувається ріст розмаїтості роботи виконавця, що само по собі може стати значним фактором мотивації його праці;

– замість контрольованого виконання завдань працівники самостійно приймають рішення і вибирають можливі варіанти досягнення цілей; виконавці повинні не чекати вказівок згори, а діяти за власною ініціативою в рамках своїх значно розширених повноважень;

– змінюються вимоги до підготовки працівників: від короткострокових курсів до професійної освіти; у зв'язку з багатоплановістю і змінюваністю робіт перебудованих процесів підприємству необхідно піклуватися не тільки про проведення курсів, ціль яких навчити, як виконувати деяку роботу або як управляти окремою ситуацією, але і про безперервну і широку освіту своїх працівників;

– змінюється оцінка ефективності роботи й оплати праці: від оцінки діяльності до оцінки результату; після проведення реінжинірингу команда процесу відповідає за його результати, і в цьому разі підприємство може

виміряти ефективність роботи команди й оплатити її відповідно до отриманого результату;

– змінюється критерій просування на посаді: від ефективності виконання роботи до здатності виконувати роботу; у нових умовах варто чітко розмежувати просування співробітника по службі й ефективність його роботи; просування по службі є функцією від здібностей співробітника, а не від ефективності його роботи; метою виконавця стає задоволення потреб клієнта; *реінжиніринг* вимагає від працівників зміни позиції: робота для клієнта, а не для начальника;

– змінюються функції менеджерів: від контролюючих до тренерських; ускладнення робіт виконавців приводить до того, що менеджер менше контролює хід виконання процесу; команда процесу цілком відповідає за його результати, а керуючий вплив на виконавців з боку менеджерів мінімізується; функції менеджера змінюються, вони складаються тепер не з видачі керуючих і контролюючих впливів, а з допомоги членам команди в рішенні проблем, що виникають у них у ході виконання процесу;

– організаційна структура нового підприємства стає більш горизонтальною, більш плоскою, що дає змогу орієнтувати організацію не на функції, а на процеси: усувається велика кількість рівнів управління;

– змінюються адміністративні функції; одним з наслідків реінжинірингу є зміна ролі вищого керівництва; зменшення рівнів управління наближає керівництво до безпосередніх виконавців і клієнтів; керівники в таких умовах повинні ставати лідерами, що словом і справою сприяють зміцненню переконань і цінностей виконавців

Додаток Б. Ітераційні та каскадні процеси

Ітераційні та каскадні (водоспадні) **процеси.** Істотна різниця між двома цими типами процесів проявляється в тому, яким чином проект ділиться на більш дрібні частини.

1. *При організації роботи на основі каскаду* проект ділиться на підставі виду робіт. Щоб створити ПЗ, необхідно зробити певні дії: проаналізувати вимоги, створити проект, виконати кодування та тестування. Наприклад, річний проект може включати двомісячну фазу аналізу, за якою слідує чотиримісячна фаза проектування, потім тримісячна фаза кодування і тримісячна фаза тестування.

2. *Ітеративний підхід розробки* передбачає поділ проекту за принципом функціональності продукту: можна взяти рік і розділити його на тримісячні ітерації. *Під час першої ітерації* береться чверть вимог і виконується повний цикл розробки ПЗ для цієї чверті: аналіз, дизайн, кодування і тестування. До кінця першої ітерації у вас є система, яка володіє чвертю необхідної функціональності. Потім ви приступаєте *до другої ітерації* і через шість місяців отримуєте систему, яка виконує половину того, що їй належить.

Наведений опис спрощено, але в цьому полягає сутність відмінності.

При розробці способом водоспаду після кожного етапу зазвичай в будь–якому вигляді виконується формальна здача, але має місце й повернення назад. У процесі кодування можуть з'ясуватися обставини, які змушують знову повернутися до етапів аналізу та проектування. При цьому на початку кодування не слід думати, що аналіз завершено, і рішення, прийняті на стадії аналізу та проектування, будуть переглядатися пізніше. Однак ці зворотні потоки формують виключення і повинні бути (по можливості) зведені до мінімуму.

При ітеративному процесі розробки перед початком реальної ітерації зазвичай спостерігається деяка дослідницька активність. Як мінімум на вимоги

буде кинутий поверхневий погляд, достатній для поділу вимог на ітерації для подальшого виконання. У процесі такого дослідження можуть бути прийняті деякі рішення з проектування самого вищого рівня. З іншого боку, незважаючи на те, що в результаті кожної ітерації має з'явитися інтегроване ПЗ, готове до постачання, часто буває, що воно ще не готове і потрібен деякий період для виправлення останніх помилок. Крім того, деякі види робіт, такі як тренування користувачів, залишаються на кінець.

Ви можете не передавати систему на реалізацію в кінці кожної ітерації, але вона повинна перебувати в стані виробничої готовності. Однак часто буває, що система здається на регулярній основі. Це добре, оскільки ви оцінюєте працездатність системи і отримуєте більш якісну зворотну реакцію. При цьому говорять про проект, який має кілька версій, кожна з яких ділиться на декілька ітерацій.

Ітераційну розробку називають по–різному: інкрементною, спіральною, еволюційною. Різні люди вкладають в ці терміни різний зміст, але ці відмінності не мають широкого визнання і не так важливі, як протистояння ітеративного методу і методу водоспаду.

Можливий і *змішаний підхід* (наприклад, Мак–Коннелла), відповідно до якого спочатку виконуються аналіз і проектування верхнього рівня в стилі водоспаду, а потім кодування і тестування, розділені на ітерації. В такому проекті може бути чотиримісячний етап аналізу і проектування, а потім чотири двомісячні ітерації побудови системи.

Більшість авторів публікацій з теми «процес створення ПЗ», особливо ті, що належать до об'єктно–орієнтованого співтовариства, відчувають неприязнь до каскадного підходу: при використанні методу водоспаду важко стверджувати, що розробка проекту дійсно йде в правильному напрямку. Легко оголосити перемогу на ранньому етапі і приховати помилки планування. Зазвичай єдиний спосіб, яким ви дійсно можете показати, що прямуєте по правильному шляху, полягає в тому, щоб отримати протестоване інтегроване

ПЗ. У разі ітеративного процесу це повторюється багато разів, і в результаті, якщо щось буде не так, як треба, ми своєчасно отримуємо відповідний сигнал.

З цієї причини рекомендують уникати метод водоспаду в чистому вигляді, а якщо неможливо використовувати ітеративний метод в повному обсязі, то застосовувати поетапну доставку.

Значна частина розробників вважають за краще ітеративну розробку. При цьому кожна ітерація повинна завершуватися створенням протестованого, інтегрованого програмного продукту, який мав би якість, як можна ближчу до якості серійної продукції. Найважче оцінити тестування й інтеграцію, тому наприкінці розробки проекту краще ці етапи не проводити. Процес тестування слід організувати так, щоб будь–яка ітерація, не оголошена в плані як здавальна, могла б бути переведена в такий статус без серйозних додаткових зусиль розробників.

Загальним прийомом при ітераційній розробці є упаковка за часом, коли ітерація займає фіксований проміжок часу. Якщо виявилося, що ви не в змозі виконати все, що планували зробити за час ітерації, то необхідно викинути деяку функціональність з цієї ітерації, але не слід змінювати дату виконання. У більшості проектів, заснованих на ітераційному процесі, протяжність ітерацій однакова, і це дозволяє вести розробку в постійному ритмі.

Регулярно практикуючи скорочення функціональності, люди вчаться робити осмислений вибір між змінною часу розробки і змінною функціональності. Скорочення функціональності в ході ітерації дозволяє навчитися розставляти пріоритети між вимогами до проекту.

Одним із найбільш загальних аспектів ітераційної розробки є питання переробки: розробка передбачає, що ви будете переробляти і видаляти існуючий код на останній ітерації проекту. При створенні ПЗ вигідніше переробити існуючий код, ніж латати код невдало спроектованої програми.

Щоб процес переробки був найбільш ефективним, деякі технічні прийоми здатні допомогти. Серед них можна виділити такі:

1. Автоматизовані регресивні тести дозволяють швидко знайти будь–які помилки, внесені в процесі змін.

2. Рефакторинг – це спосіб зміни існуючого ПЗ. Механізм рефакторингу заснований на застосуванні серії невеликих трансформацій основного коду, що зберігають поведінку (багато з цих трансформацій можуть бути автоматизовані).

3. Послідовна інтеграція зберігає синхронізацію дій розробників, об'єднаних в команду, що дозволяє уникнути болючих циклів інтеграції. В її основі лежить повністю автоматизований процес побудови, який може бути автоматично припинено, як тільки будь–який член команди почне записувати код в базу. Передбачається, що розробники записують код щодня, тому автоматичні збірки виконуються кілька разів в день. Процес побудови включає запуск великого блоку автоматичних регресійних тестів, що дозволяє виявити і виправити будь–яку неузгодженість.

Прогнозуюче та адаптивне планування. Одна з причин, чому метод водоспаду ще живий, полягає у бажанні забезпечити передбачуваність при створенні ПЗ. Ніщо так не дратує, як відсутність точної оцінки вартості створення програмного продукту і термінів його розробки.

Прогнозуючий підхід спрямований на виконання роботи на початковому етапі проекту (щоб краще зрозуміти, що потрібно робити в подальшому). Таким чином, настає момент, коли решту проекту можна оцінити з достатнім ступенем точності. В процесі прогнозу планування проект розділяється на дві стадії:

– на першій стадії складаються плани (і тут прогнозувати важко),

– друга стадія більш передбачувана, оскільки плани вже готові.

Однак все ще тривають дискусії про те, чи багато проектів можуть бути передбачуваними. Сутність цього питання полягає в аналізі вимог. Одна з найбільш істотних причин складності програмних проектів полягає в труднощі розуміння вимог до програмних систем. У більшості програмних проектів

вимоги піддаються істотному перегляду та зміні на пізній стадії виконання проекту. Такий перегляд вщент розбиває основу прогнозів.

Наслідки перегляду можна запобігти, заморозивши вимоги на ранній стадії проекту і не дозволяючи з'являтися змінам, але це призводить до ризику поставити клієнту систему, яка не задовольняє вимоги користувачів. Ця проблема призводить до двох різних варіантів дій: 1) направити більше зусиль на опрацювання вимог, 2) отримати більш визначену множину вимог, щоб скоротити можливі зміни.

Прихильники іншої школи стверджують, що перегляд вимог неминучий, що в багатьох проектах важко стабілізувати вимоги в такій мірі, щоб існувала можливість використовувати прогнозуюче планування. Це може бути або наслідком того, що важко уявити 1) що може робити програмний продукт, або 2) що умови ринку диктують непередбачувані зміни. Ця школа підтримує *адаптивне планування* відповідно до твердженням, що прогнозованість – це ілюзія.

Необхідно повернутися обличчям до реальності постійних змін і використовувати такий підхід у плануванні, при якому зміна в проекті вважається величиною постійною. Ця зміна контролюється таким чином, щоб в результаті виконання проекту поставлялося якомога краще ПЗ; але хоча проект і є контрольованим, передбачити його не можна.

При адаптивній розробці не можна сказати «відповідно до плану», оскільки план весь час змінюється. Це не означає, що адаптивні проекти не плануються; зазвичай планування займає чимало часу, але план трактується як основна лінія проведення послідовних змін, а не як прогноз майбутнього.

На основі прогнозуючого плану можна розробити контракт з фіксованою функціональністю за фіксованою ціною. В такому контракті точно вказується, що має бути створено, скільки це коштує і коли продукт буде поставлений.

На основі адаптивного плану таке фіксування неможливо. Ви можете визначити бюджет і терміни поставки, але ви не можете зафіксувати

функціональність поставленого товару. Адаптивний контракт передбачає, що користувачі будуть співпрацювати з командою розробників, щоб регулярно переглядати необхідну функціональність і переривати проект, якщо прогрес незначний (як такий процес адаптивного планування може визначати проект зі змінними межами функціональності за фіксованою ціною?).

Адаптивний підхід менш бажаний, оскільки всі вважають за краще більшу передбачуваність програмних проектів. Однак передбачуваність залежить від точності, коректності та стабільності множини вимог.

Не складайте прогнозуючий план, поки не отримаєте точні і коректні вимоги і не будете впевнені, що вони не піддадуться істотним змінам. Якщо не можна отримати точні, коректні і стабільні вимоги, то використовуйте метод адаптивного планування. Передбачуваність та адаптивність передбачають вибір життєвого циклу: 1) адаптивне планування має на увазі ітеративний процес, 2) прогнозуючий план можна реалізувати будь–яким з двох способів, хоча за його виконанням легше спостерігати в разі застосування методу водоспаду, або методу поетапної поставки.

Додаток В. Інтерфейс, призначений для користувача

Користувальницький інтерфейс – це сукупність таких складових:

1) інформаційної моделі проблемної області,

2) засобів і способів взаємодії користувача з інформаційною моделлю,

3) компонентів, які забезпечують формування інформаційної моделі в процесі роботи програмної системи.

Засоби і способи взаємодії з інформаційною моделлю визначаються: 1) складом апаратного і програмного забезпечення, наявного у розпорядженні користувача, 2) характером розв'язуваної задачі. Ефективність роботи користувача визначається функціональними можливостями наявних в його розпорядженні апаратних і програмних засобів.

Провідні фахівці в галузі людино–машинних комп'ютерних систем вже до середини 70–х років XX ст. усвідомили необхідність формування єдиних підходів до реалізації призначеного для користувача інтерфейсу. Однак в силу обмежених технічних можливостей обчислювальних систем багато з розглянутих вище принципів сприймалися програмістами як абстрактні побажання.

Тривалий час основною формою спілкування користувача з комп'ютером залишався діалог у формі «питання–відповідь». Але, можливо, саме тому, що комп'ютер виступав в ролі співрозмовника, виникла необхідність дослідження психологічних аспектів спілкування людини з комп'ютером. Результати цих досліджень змусили згадати про *ергономіку робочого місця*. На сьогодні жодна серйозна публікація, присвячена призначеному для користувача інтерфейсу, не обходиться без посилань на результати, отримані в таких галузях знань, як психологія, ергономіка, математична лінгвістика, кібернетика тощо.

В якості ілюстрації серйозного ставлення «законодавців моди» в області комп'ютерних технологій до проблем інтерфейсу можна відзначити той факт, що Американський Національний інститут стандартів (ANSI) має за даним

напрямком спеціальну консультативну групу – Комітет зі стандартів інтерфейсу «людина–комп'ютер» (The Human–Computer Interface Standard Committee). Подібні організації існують не тільки у США, але й у інших країнах. Існують також міжнародні дослідницькі групи, які працюють в цьому напрямку (наприклад, Міжнародний консультативний комітет з телеграфії та телефонії – International Telegraph and Telephone Consultation Committee, що вивчає особливості інтерактивних елементів інтерфейсу).

Багато із цих організацій або робочих груп свого часу підготували проекти документів по стандартизації призначених для користувача інтерфейсів, змісту, принципам їх проектування та реалізації. Так, в 1986 р. було опубліковано «Керівництво по розробці програмного інтерфейсу», що містить 944 принципів, які стосуються введення і відображення даних, підтримки користувача, захисту даних тощо. Однак жоден з цих проектів не отримав статусу офіційного документа, оскільки всі вони мали спільний недолік: в них не враховувалися технологічні можливості інструментальних засобів, які були в розпорядженні розробників ПЗ.

Ситуація докорінно змінилася у 1987 р., коли корпорація IBM оголосила про намір створити єдине середовище розробки додатків (Systems Application Architecture – SAA).

Програмні засоби інтерфейсу – це сукупність програмних засобів, які забезпечують діалог оператора з обчислювальними засобами та візуалізацію віртуальних об'єктів на екрані.

Технічні засоби інтерфейсу – це засоби відображення інформації та управління, які використовуються оператором під час діалогу з обчислювальними засобами.

Відповідно до наведених документів *при розробці інтерфейсу необхідно дотримуватися таких принципів*.

1. **Дружній інтерфейс** (принцип «пробачення» користувачу). Користувачі зазвичай вивчають особливості роботи з новим програмним продуктом

методом проб і помилок. Ефективний інтерфейс повинен враховувати це. На кожному етапі роботи він повинен дозволяти тільки відповідний набір дій і попереджати користувачів про ситуації, де вони можуть нашкодити системі або даним; найкраще надати користувачу можливість скасувати або виправити виконані дії.

Навіть при наявності добре спроектованого інтерфейсу користувачі можуть робити ті чи інші помилки. Ці помилки можуть бути як «фізичного» типу (випадковий вибір неправильної команди або даних), так і «логічного» (прийняття неправильного рішення щодо вибору команди або даних). Ефективний інтерфейс повинен 1) запобігати ситуаціям, які, ймовірно закінчаться помилками; 2) вміти адаптуватися до потенційних помилок користувача і полегшувати йому процес усунення наслідків таких помилок.

2. **Принцип «зворотного зв'язку».** Необхідно завжди забезпечувати зворотний зв'язок для дій користувача. Кожна дія користувача має отримувати візуальне, а іноді й звукове підтвердження того, що ПЗ сприйняло введену команду; при цьому різновид реакції, по можливості, повинен враховувати природу виконаної дії.

Зворотній зв'язок ефективний у випадку, якщо він реалізується вчасно, тобто якомога ближче до точки останньої взаємодії користувача із системою. Коли комп'ютер обробляє «завдання, яке надійшло», корисно надати користувачеві інформацію щодо стану процесу, а також при необхідності можливість перервати цей процес. Ніщо так не бентежить не досвідченого користувача, як заблокований екран, який не реагує на його дії. Звичайний користувач здатний витерпіти тільки кілька секунд очікування відповідної реакції від свого електронного «співрозмовника».

3. **Простота інтерфейсу.** Інтерфейс повинен бути простим. При цьому мають на увазі забезпечення легкості у його вивченні та у використанні. Крім того, він повинен надавати доступ до всього переліку функціональних можливостей, передбачених додатком. Реалізація доступу до широких

функціональних можливостей і забезпечення простоти роботи суперечать один одному. Розробка ефективного інтерфейсу покликана збалансувати ці цілі.

Один із можливих шляхів підтримки простоти – виведення на екрані інформації, мінімально необхідної для виконання користувачем чергового кроку завдання. Багатослівні командні імена або повідомлення, непродумані або надлишкові фрази ускладнюють процес розуміння суттєвої інформації.

Інший шлях до створення простого та ефективного інтерфейсу – розміщення на екрані елементів з урахуванням їх смислового значення та логічного взаємозв'язку (це дозволяє використовувати в процесі роботи асоціативне мислення).

Можна допомогти користувачам управляти складністю інформації, яка відображається, використовуючи послідовне розкриття діалогових вікон, розділів меню тощо. *Послідовне розкриття* передбачає таку організацію інформації, при якій в кожен момент часу на екрані знаходиться тільки та її частина, яка необхідна для виконання чергового кроку. Скорочуючи обсяг інформації, поданої користувачеві, зменшують обсяг інформації, що підлягає обробці. Прикладом такої організації є ієрархічне (каскадне) меню, кожен рівень якого відображає тільки ті пункти, які відповідають одному, обраному користувачем, пункту вищого рівня.

4. **Гнучкість інтерфейсу**. Гнучкість інтерфейсу – це його здатність враховувати рівень підготовки і продуктивність праці користувача. Вона передбачає можливість зміни структури діалогу і/або вхідних даних. Концепція гнучкого (адаптивного) інтерфейсу в даний час є однією із основних областей дослідження взаємодії людини та ЕОМ. Основна проблема полягає не в тому, як організувати зміни в діалозі, а в тому, які ознаки потрібно використовувати для визначення необхідності внесення змін у діалог та їх сутність.

5. **Естетична привабливість**. Проектування візуальних компонентів є найважливішою складовою частиною розробки програмного інтерфейсу. Коректне візуальне подання використовуваних об'єктів забезпечує передачу

важливої додаткової інформації про поведінку та взаємодію різних об'єктів (але слід пам'ятати, що кожен візуальний елемент, який з'являється на екрані, потенційно вимагає уваги користувача, яка не є безмежною). Слід формувати на екрані середовище, яке не тільки б сприяло розумінню користувачем поданої інформації, але і дозволяло б зосередитися на найбільш важливих її аспектах.

До теперішнього часу найбільших успіхів в проектуванні користувальницького інтерфейсу з перерахованими властивостями домоглися розробники комп'ютерних ігор.

Якість інтерфейсу складно оцінити кількісними характеристиками, однак більш–менш об'єктивну його оцінку можна отримати на основі таких показників:

1. *Час, необхідний конкретному користувачу для досягнення заданого рівня знань і навичок по роботі з додатком* (наприклад, непрофесійний користувач повинен освоїти команди роботи з файлами не більше ніж за чотири години).

2. *Збереження отриманих робочих навичок після закінчення деякого часу* (наприклад, після тижневої перерви користувач повинен виконати конкретну послідовність операцій за заданий час).

3. *Швидкіть розв'язання задачі* за допомогою конкретного додатка. При цьому має оцінюватися не швидкодія системи і не швидкість введення даних з клавіатури, а час, необхідний для досягнення мети розв'язуваної задачі. Виходячи з цього, критерій оцінювання за цим показником може бути сформульований, наприклад, так: користувач повинен обробити за годину не менше 20 документів з помилкою не більше 1%.

4. *Суб'єктивна задоволеність користувача при роботі з системою* (яку кількісно можна зобразити у відсотках або у вигляді оцінки за n–бальною шкалою).

Узагальнюючи викладене вище, можна коротко сформулювати *основні правила, дотримання яких дозволяє розраховувати на створення ефективного призначеного для користувача інтерфейсу*.

Правило 1. Інтерфейс користувача необхідно проектувати і розробляти як окремий компонент створюваного додатка.

Правило 2. Необхідно враховувати можливості та особливості апаратно–програмних засобів, на базі яких реалізується інтерфейс.

Правило 3. Доцільно враховувати особливості і традиції предметної області, до якої належить створюваний додаток.

Стратегія розробки інтерфейсу передбачає: 1) розробляти інтерфейс як окремий компонент системи, 2) враховувати можливості апаратних і програмних засобів, 3) використовувати прийняті принципи та послідовність дій в розробці інтерфейсу,

– розуміння завдання користувача і його залучення до процесу виконання.

Етапи проектування користувацького інтерфейсу. При проектуванні користувальницького інтерфейсу необхідно визначити: 1) структуру діалогу; 2)можливий сценарій розвитку діалогу; 3) зміст керуючих повідомлень і даних, якими можуть обмінюватися людина і додаток (семантику повідомлень); 4)візуальні атрибути, які відображаються (синтаксис повідомлень).

Вибір структури діалогу – це перший з етапів, який повинен бути виконаний при розробці інтерфейсу. Розглянемо чотири варіанти структури діалогу типу «питання – відповідь».

1. **Діалог типу «питання – відповідь».** Структура діалогу типу «питання–відповідь» (Q&A) базується на інтерв'ю. Система бере на себе роль інтерв'юера та отримує інформацію від користувача у вигляді відповідей на питання.

Всі діалоги, керовані комп'ютером, в тій чи іншій мірі складаються з питань, на які користувач відповідає. Однак в структурі «Q & A» цей процес

виражено явно. У кожній точці діалогу система виводить одне питання, на яке користувач дає одну відповідь. Залежно від отриманої відповіді система може вирішити, яке наступне питання задавати. Існують системи, відповіді в яких даються на природній мові, але найчастіше використовують відповіді з одного слова з обмеженою граматикою.

Структура «Q & A» має суттєвий недолік: навіть якщо процес введення відбувається досить швидко, для людини, що знає, які питання задає система і які відповіді потрібно на них давати, відповідати на всю серію питань досить втомливо. З появою графічного інтерфейсу структура «Q & A» дещо застаріла, проте у неї є такі *переваги*: ця структура може задовольняти вимоги різних користувачів (зокрема, така структура доречна при реалізації діалогу з множиною «відгалужень» тобто у випадках, коли на питання передбачається велика кількість відповідей, кожна з яких впливає на наступне питання).

2. **Діалог на основі меню**. Меню є, мабуть, найбільш популярним варіантом організації запитів на введення даних під час діалогу, керованого комп'ютером. Меню можна застосовувати для введення керуючих повідомлень і даних. Існують різні формати подання меню на екрані: у вигляді 1) списку об'єктів, які обирають прямою вказівкою або зазначенням номера (або мнемонічного коду); 2) блоків даних; 3) рядка даних; 4) піктограм.

Меню у вигляді рядка даних може з'являтися вгорі або внизу екрану і часто залишається в цій позиції протягом усього діалогу. Таким чином, за допомогою меню для введення зручно відображати можливі варіанти даних, доступних в будь–який час роботи із системою.

Якщо в системі має місце досить велике різноманіття варіантів дій, організовується ієрархічна структура з відповідних меню. Додаткові *меню у вигляді блоків даних* «спливають» на екрані в позиції, яка визначається поточним станом покажчика, або «випадають» безпосередньо з рядка меню верхнього рівня. Ці меню зникають після вибору варіанта.

Меню у вигляді піктограм є множиною об'єктів вибору, розкиданих на всьому екрані; часто об'єкти вибору містять графічне подання варіантів роботи.

Користувач діалогового меню може вибрати потрібний пункт, вводячи текстовий рядок, який ідентифікує цей пункт, вказуючи на нього безпосередньо або переглядаючи список і вибираючи з нього. Система може виводити пункти меню послідовно, при цьому користувач вибирає потрібний йому пункт натисканням потрібної клавіші.

3. **Діалог на основі екранних форм**. Структури типу «питання – відповідь» і типу меню припускають обробку на кожному кроці діалогу однієї відповіді. *Діалог на основі екранних форм* допускає обробку на одному кроці діалогу декількох відповідей.

На практиці форми використовують там, де облік будь–якої діяльності вимагає введення досить стандартного набору даних. Людина, яка заповнює форму, може вибирати послідовність відповідей, тимчасово пропускати деякі питання, повертатися назад для корекції попередньої відповіді і навіть «порвати бланк–форму» і почати заповнювати новий. Користувач працює з формою, поки не заповнить її повністю і не передасть системі.

Система може перевіряти кожну відповідь безпосередньо після введення або почекати і вивести список помилок тільки після заповнення всієї форми. У деяких системах інформація, що вводиться користувачем, стає доступною тільки після натискання клавіші «Enter» після закінчення заповнення форми.

Цю структуру доречно застосовувати там, де джерелом даних служать існуючі вхідні («паперові») форми документа. Не обов'язково, щоб зовнішній вигляд цих форм збігався (це навіть може погіршити сприйняття даних на екрані), але всі введені елементи даних повинні розташовуватися в тому ж відносному порядку і мати такий же формат, що і в оригінальному документі.

Зазичай всі необхідні одиниці введення не можна відобразити одночасно в межах одного екрану (або вікна), і їх необхідно розділити на групи, які

відображаються на послідовності екранів (вікон). Важливо, щоб це розбиття зберігало логічні зв'язки і не приводило до поділу пов'язаних частин документа.

Структура діалогу на основі екранної форми 1) забезпечує високий рівень підтримки користувача: для кожного питання форми можуть бути передбачені повідомлення про помилки і довідкова інформація (користувачеві можна також надати допомогу, включивши деякі елементи формату відповіді в питання або в поле відповіді); 2) дозволяє підвищити швидкість введення даних в порівнянні зі структурою типу «питання – відповідь» і маніпулювати ширшим діапазоном вхідних даних, ніж меню; 3) з нею можуть працювати користувачі будь–якої кваліфікації. Оскільки ця структура має послідовну, а не деревоподібну організацію, вона в меншій мірі підходить для роботи в режимі вибору варіантів.

4. **Діалог на основі командної мови**. Структура діалогу на основі командної мови часто використовується в операційних системах. Історично вона є першою із реалізованих структур діалогу. При такій організації діалогу програмна система не виводить нічого, крім постійної підказки (запрошення на введення команди), яка означає готовність системи до роботи. Кожну команду вводять з нового рядка і зазвичай закінчують натисканням клавіші «Enter». Відповідальність за правильність поставлених команд лежить на користувачеві. Система інформує про неможливість виконання неправильної команди, не пояснюючи, зазвичай, причин.

Діалог на базі команд зручний для введення керуючих повідомлень, проте він забезпечує більш широкі можливості вибору в будь–якій точці діалогу і не вимагає ієрархічної організації обслуговуючих його програм.

Система може підтримувати велику кількість команд, але на практиці слід обмежувати їх число, щоб не перевантажувати пам'ять користувача. Структура на базі командної мови не відрізняється гарною підтримкою користувача і придатна в основному для підготовлених фахівців. Перед використанням такої системи необхідно пройти курс навчання і в подальшому

вивчати особливості роботи по документації, а не на практиці. Більш того, оскільки системі невідомо, що має намір робити користувач, важко запропонувати якусь реальну допомогу в процесі роботи, крім надати довідку загального характеру.

Оскільки ця структура передбачає великий обсяг матеріалу, який запам'ятовується, імена команд слід вибирати так, щоб вони несли смислове навантаження і легко запам'ятовувалися.

Діалог повинен управляти даними. У інтерфейсах на основі мов команд це зазвичай досягається за допомогою складових командних рядків, де ключове слово для позначення команди (що робити) передує списку параметрів (вхідних даних). Параметри в списку можна задавати в одній з двох форм – в позиційній або у ключовій. У першому випадку призначення параметра визначається за його місцем в командному рядку. У разі ключових параметрів кожне значення передує певному ідентифікатору, який визначає його призначення.

Позиційні параметри зменшують обсяг інформації, яка вводиться, але їх істотним недоліком є те, що значення, які вводяться, повинні вказуватися в чітко визначеному порядку, порушення якого погано діагностується системою і може спричинити серйозні наслідки. Завдання позиційних параметрів ускладнюється, якщо їх список досить великий. Цей недолік прагнуть компенсувати, дозволяючи опускати незмінні параметри, вводячи два роздільники один за одним.

Ключові параметри зменшують навантаження на пам'ять користувача в тому відношенні, що відпадає необхідність в запам'ятовуванні порядку їх слідування; крім того, можна опускати необов'язкові параметри. З іншого боку, в цьому випадку користувачеві необхідно запам'ятати безліч ключових слів, а розробнику – підібрати для них «осмислені» імена. Цей підхід також вимагає більшого часу роботи системи, щоб розпізнати ключові слова, задані в довільному порядку.

Зазвичай командні мови підтримують макроси, які розширюють функціональні можливості діалогу без збільшення кількості команд. Макрос містить кілька окремих командних рядків. При зверненні до нього в процесі діалогу окремі рядки команд макросу виконуються одна за одною, як якщо б вони вводилися з клавіатури. Це скорочує сеанси діалогу.

Структура на основі мови команд за своїми можливостями найшвидша і гнучка з усіх структур діалогу. Більшість призначених для користувача інтерфейсів на базі «природного» мови реалізується за допомогою мов команд з великим набором ключових слів. Підготовлений користувач відчуває задоволення від відчуття того, що він управляє системою, а не навпаки. Однак ця структура не забезпечує користувача підтримкою, тому навіть підготовлені користувачі вважають, що дуже складно використовувати всі закладені в ній можливості. Більшість же користувачів добре знайомі тільки з вельми обмеженим набором засобів, з яким вони працюють регулярно.

Розробка сценарію діалогу. Розвиток діалогу в часі можна розглядати як послідовність переходів системи з одного стану в інший. Очевидно, що жодний з цих станів не повинен бути «глухим кутом», тобто користувач повинен мати можливість перейти з будь–якого поточного стану діалогу в потрібний (за один або кілька кроків). Для цього під час розробки інтерфейсу необхідно визначити всі можливі стани діалогу і шляхи переходу з одного стану в інший. Іншими словами, необхідно розробити *сценарій діалогу*.

Цілями розробки сценарію діалогу є: 1) виявлення та усунення можливих глухих кутів – ситуацій в ході розвитку діалогу; 2) вибір раціональних шляхів переходу з одного стану діалогу в інший (з поточного в потрібний); 3) виявлення неоднозначних ситуацій, які вимагають надання додаткової допомоги користувачу.

Складність розробки сценарію визначається в основному двома чинниками: функціональними можливостями створюваного додатка (тобто

числом і складністю реалізованих функцій обробки інформації) і ступенем невизначеності можливих дій користувача.

У свою чергу, ступінь невизначеності дій користувача залежить від обраної структури діалогу. Найбільшою детермінованістю володіє діалог на основі меню, найменшою – діалог типу «питання–відповідь», керований користувачем.

Отже, *сценарій діалогу* можна спростити, зменшивши ступінь невизначеності дій користувача. Можливими способами вирішення цього завдання є: 1) використання змішаної структури діалогу (застосування меню з метою «обмеження свободи» користувача там, де це можливо); 2) застосування вхідного контролю введеної інформації (команд і даних).

Додаткові можливості зі зменшення невизначеності дій користувача надає об'єктно–орієнтований підхід до розробки інтерфейсу, при якому для кожного об'єкта заздалегідь встановлюється перелік властивостей і допустимих операцій. Найбільш ефективний такий підхід при створенні графічного інтерфейсу. Скорочуючи число можливих станів діалогу, розробник разом з тим повинен пам'ятати про необхідність відображення в його сценарії роботи засобів підтримки користувача, що робить сценарій більш складним.

Спосіб опису сценарію діалогу залежить від ступеня його складності. Існуючі методи опису сценаріїв можна розділити на дві великі групи: неформальні і формальні методи.

Формальні методи дозволяють автоматизувати як проектування діалогу, так і його модифікацію (адаптацію) відповідно до характеристик користувача. Нині найбільш широко використовують формальні методи опису сценаріїв на основі мереж Петрі та їх розширень, а також на основі систем подання знань.

Додаток Г. Типове проектування інформаційних систем

(на прикладі ІС підприємства)

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

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

Прийнята класифікація ТПР заснована на рівні декомпозиції системи. Виділяються такі класи ТПР: 1) *елементні ТПР* – типові рішення по завданню або за окремим видом забезпечення завдання (інформаційному, програмному, технічному, математичному, організаційному); 2) *підсистемні ТПР* – як елементи типізації виступають окремі підсистеми, розроблені з урахуванням функціональної повноти і мінімізації зовнішніх інформаційних зв'язків; 3) *об'єктні ТПР* – типові галузеві проекти, які включають повний набір функціональних і забезпечуючих підсистем ІС. Кожне типове рішення передбачає наявність, крім функціональних елементів (програмних або апаратних), документації з детальним описом ТПР і процедур налаштування відповідно до вимог, які розробляються. Основні особливості різних класів ТПР наведені в табл. Б.1. Для реалізації типового проектування використовуються два підходи: параметрично–орієнтоване і модельно–орієнтоване проектування.

1. Параметрично–орієнтоване проектування включає в себе такі етапи: визначення критеріїв оцінки придатності пакетів прикладних програм (ППП) для вирішення поставлених завдань, аналіз та оцінка доступних ППП по сформульованим критеріям, вибір і закупівля найбільш підходящого пакета, налаштування параметрів (дороблення) закупленого ППП.

Таблиця Б.1.

Переваги та недоліки ТПР

Клас ТПР Реалізація ТПР	Переваги	Недоліки
Елементні ТПР. Бібліотеки методо–орієнтованих програм	- забезпечується застосування модульного підходу до проектування і документування ІС	- великі витрати часу; - на створення пари різнорідних елементів внаслідок інформаційної, програмної та технічної несумісності; – на доопрацювання ТПР окремих елементів
Підсистемні ТПР. Пакети прикладних програм	- досягається високий ступінь інтеграції елементів ІС; - дозволяють здійснювати: модульне проектування; параметричне налаштування програмних компонентів на різні об'єкти управління; - забезпечують: скорочення витрат на проектування і програмування взаємопов'язаних компонентів; гарне документування процесів обробки інформації, які відображаються.	- адаптивність ТПР недостатня з позиції безперервного інжинірингу ділових процесів; - виникають проблеми в об’єднанні різних функціональних підсистем, особливо в разі використання рішень кількох виробників ПЗ
Об'єктні ТПР. Галузеві проекти ІС	- інтегрування всіх компонентів ІС за рахунок методологічної єдності та інформаційної, програмної та технічної сумісності; - відкритість архітектури дозволяє встановлювати ТПР на різних програмно–технічних платформах; - масштабованість допускає конфігурацію ІС для змінної кількості робочих місць; - конфігурованість дозволяє вибирати необхідну підмножину компонентів	- проблеми прив'язування типового проекту до конкретного об'єкта управління, що викликає в деяких випадках необхідність зміни організаціонно–економічної структури об'єкта автоматизації

Критерії оцінювання ППП поділяються на такі групи: призначення і можливості пакета; відмінні ознаки і властивості пакета; вимоги до технічних і програмних засобів; документація пакета; фактори фінансового порядку; особливості установки пакета; особливості експлуатації пакета; допомога постачальника по впровадженню і підтримці пакета; оцінка якості пакету і досвід його використання; перспективи розвитку пакета.

Усередині кожної групи критеріїв виділяється деяка підмножина конкретних показників, які деталізують кожен з десяти виділених аспектів аналізу обираємих ППП.

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

Модельно–орієнтоване проектування полягає в адаптації складу і характеристик типової ІС відповідно до моделі об'єкта автоматизації.

Технологія проектування в цьому випадку повинна забезпечувати єдині засоби для роботи як із моделлю типової ІС, так із моделлю конкретного підприємства.

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

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

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

Типові моделі описують конфігурації ІС для певних галузей або типів виробництва.

Модель конкретного підприємства будується або шляхом вибору фрагментів основної або типової моделі у відповідності зі специфічними особливостями підприємства (BAAN Enterprise Modeler), або шляхом автоматизованої адаптації цих моделей в результаті експертного опитування (SAP Business Engineering Workbench). Побудована модель підприємства у вигляді мета–опису зберігається в репозиторії і при необхідності може бути відкорегована. На основі цієї моделі формується конфігурація та виконується налаштування ІС.

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

Модель бізнес–функцій є ієрархічною декомпозицію функціональної діяльності підприємства.

Модель бізнес–процесів відображає виконання робіт для функцій самого нижнього рівня моделі бізнес–функцій. Для відображення процесів використовується модель управління подіями (EPC – Event–driven Process Chain). Саме модель бізнес–процесів дозволяє виконати налаштування програмних модулів – додатків ІС відповідно до характерних особливостей конкретного підприємства.

Моделі бізнес–об'єктів використовують для інтеграції додатків, що підтримують виконання різних бізнес–процесів (тут передбачено застосування застосуванням UML).

Модель організаційної структури підприємства є традиційною ієрархічною структурою підпорядкування підрозділів і персоналу.

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

Реалізація типового проекту передбачає виконання таких операцій: встановлення глобальних параметрів системи; завдання структури об'єкта автоматизації; визначення структури основних даних; задання переліку реалізованих функцій і процесів; опис інтерфейсів; опис звітів; налаштування авторизації доступу; налаштування системи архівування.

Додаток Д. Методології моделювання в нотації IDEF

(*методології моделювання, засновані на графічному зображенні систем*)

IDEF (Об'єднання методологічних понять) – множина спільно використовуваних методів моделювання [5 с. 150–153]:

IDEF0 (Function Modeling): метод використовують для створення функціональної моделі, яка є структурованим відображенням функцій виробничої системи або середовища, а також інформації та об'єктів, що зв'язують ці функції.

IDEF1 (Information Modeling): метод застосовують для побудови інформаційної моделі, яка зображує структуровану інформацію, необхідну для підтримки функцій виробничої системи або середовища.

IDEF2 (Simulation Model Design): метод дозволяє побудувати динамічну модель змінної у часі поведінки функцій, інформації і ресурсів виробничої системи або середовища. Ця модель використовується рідко: в основному затребувана на підприємствах, де необхідно описати безперервну діяльність на конвеєрах або аналогічні функції.

IDEF3 (Process Description Capture): метод використовують для збору інформації про стан системи, що моделюється. Це – структурний метод, який показує причинно–наслідкові зв'язки та події, а також як організована робота і які користувачі працюють системою. IDEF3 складається з таких двох методів:

1. Process Flow Description: опис процесів з описом того, як організована робота між різними елементами системи, що моделюється.

2. Object State Transition Description: опис переходів станів об'єктів з описом того, які існують проміжні стани у об'єктів в системі, що моделюється.

IDEF4 (Object–Oriented Design): метод об'єктно–орієнтованого планування був розроблений для підтримки об'єктно–орієнтованої ідеології (детальніше – технологія UML).

IDEF5 (Ontology Description Capture): метод дозволяє розробляти, вивчати і підтримувати онтологію, яка моделюється. Термін «онтологія» включає в себе такі складові: 1) каталог термінів галузі знань; 2) правила, що пояснюють, як терміни можуть комбінуватися, створюючи при цьому коректні ситуації в галузі знань і узгоджені висновки, які використовуються в моделюється системі.

IDEF6 (Design Rational Capture Method): метод дозволяє використовувати раціональний досвід проектування.

IDEF7 (Information System Auditing): метод описує проведення методології аудиту інформаційної системи.

IDEF8 (User Interface Modeling): метод дозволяє розробляти необхідні моделі Графічного Інтерфейсу Користувача (Human–System Interaction Design). Метод призначено для проектування взаємодії людини і технічної системи.

IDEF9 (Business Constraint Discovery): модель призначена для аналізу наявних умов й обмежень (в тому числі фізичних, юридичних або будь–яких інших), їх вплив на рішення, які приймаються в процесі реінжинірингу.

IDEF10 – Впровадження архітектури моделювання.

IDEF11 – Інформаційне моделювання артефактів.

IDEF12 – Моделювання організації.

IDEF13 – Проектування трьох схем.

IDEF14 – Проектування мережі: метод дозволяє моделювати обчислювальні мережі. Модель призначена для зображення на графічній мові та аналізу даних при проектуванні обчислювальних мереж з описом конфігурацій, черг, мережевих компонентів, вимог до надійності.

Серед найбільш популярних методологій моделювання бізнес–процесів та їх аналізу можна виділити такі:

1. Моделювання бізнес–процесів в нотації IDEF0, призначене для функціонального опису бізнес–процесів. IDEF0 має вигляд системи, яка

складається з блоків і дуг (стрілок), що позначають зовнішні зв'язки бізнес–процесу і його декомпозицію.

2. Моделювання робочих потоків в нотації IDEF3, призначене для опису робочих процесів.

3. Моделювання потоків даних в нотації DFD, призначене для опису потоків інформації в процесі виконання робіт.