Міністерство освіти і наукиУкраїни
Рівненський державний гуманітарний університет

Г.О. Шліхта

ПРОЕКТУВАННЯ ТА РОЗРОБКА МУЛЬТИМЕДІЙНИХ ОСВІТНІХ ІНФОРМАЦІЙНИХ СИСТЕМ

НАВЧАЛЬНИЙ ПОСІБНИК

2019

УДК 004.032.6:378.091.315.7 (075.8)

Ш 69

Рецензенти:

Климюк Ю.Є., кандидат технічних наук, доцент завідувач кафедри інформаційних систем та обчислювальних методів Міжнародного економіко-гуманітарного університету імені академіка Степана Дем'янчука

Шахрайчук М.І., кандидат фізико-математичних наук, доцент, декан факультету математики та інформатики Рівненського державного гуманітарного університету

Затверджено Вченою радою Рівненського державного гуманітарного університету протокол № від 2019 р.

Проектування та розробка мультимедійних освітніх інформаційних систем: навчальний посібник/ за ред. Г.О.Шліхта. - РДГУ. – Рівне: РВВ РДГУ, 2019. – 183 с.

В навчальному посібнику викладено загально-теоретичні відомості про сучасні інформаційні системи, процес їх роброблки, методології та технології, а також розкрито питання основних інструментальних засобів для створення інформаційних систем. Описано новітні мультимедійні технології для використання в освітніх інформаційних системах. Посібник призначено для здобувачів вищої освіти, аспірантів, фахівців галузі професійної педагогіки.

ЗМІСТ

ВСТУП

Сучасні інформаційні системи створюються для обробки великих обсягів інформації при жорстких обмеженнях на час видачі результатів. Вони мають складну формалізацію процедур прийняття рішень для більшості задач, високий ступінь інтеграції елементів, які входять до складу системи, велику кількість зв'язків між елементами, характеризуються гнучкістю і можливістю модифікації.

Мета дисципліни «Проектування та розробка мультимедійних освітніх інформаційних систем» – дати основні теоретичні положення щодо створення інформаційних систем, зокрема мультимедійних освітніх (далі (МОІС)), ознайомити із сучасними підходами до даної проблеми, зі складом і змістом технологічних операцій створення МОІС на різних рівнях ієрархії, а також з засобами автоматизації проектних робіт, формалізації процесу проектування та методами управління проектуванням МОІС.

Завданнями вивчення навчальної дисципліни є:

- оволодіння методами та засобами розробки складного програмного забезпечення, що входить до складу МОІС;
- дослідження етапів життєвого циклу МОІС і стандартів та процесів програмної інженерії;
- засвоєння методів побудови архітектури та технологій проектування МОІС;
- оволодіння методами інженерії вимог та оцінювання якості МОІС, а також технологією керування якістю програмних продуктів;
- засвоєння методологій тестування, документування, супроводу та маркетингу МОІС відповідно до міжнародних та національних стандартів.

Дисципліна є з циклу професійної підготовки. Отримані знання повинні відповідати сучасним уявленням в області проектування інформаційних систем в освіті та визначати вміння студентів самостійно вирішувати завдання проектування освітнього процесу з використанням інструментальних засобів проектування та реалізації.

Тому **предметом дисципліни** є створення МОІС з метою автоматизованого отримання всіх показників, які необхідні для прийняття рішення з керування цільовим об'єктом. Вона має свої теоретичні основи та методологію і потребує попереднього вивчення циклу математичних, економічних, технічних та інших дисциплін.

Процес створення МОІС багато в чому ще не формалізовано. Вміння правильно створити систему чи окрему задачу, виявити і

коректно сформулювати критерії і обмеження приходять з досвідом. Існуючі стандарти, керівні документи і методичні матеріали визначають організаційні питання і регламентують склад і зміст проектної документації, але не містять рекомендацій і вказівок, які розкривають суть процесу створення МОІС. Це зумовило певні складнощі в ході підготовки навчального матеріалу, який складено з урахуванням окремих питань дисципліни, висвітлених у вітчизняній і зарубіжній літературі, а також досвіду щодо наукових основ створення МОІС, практичних розробок МОІС педагогічного призначення.

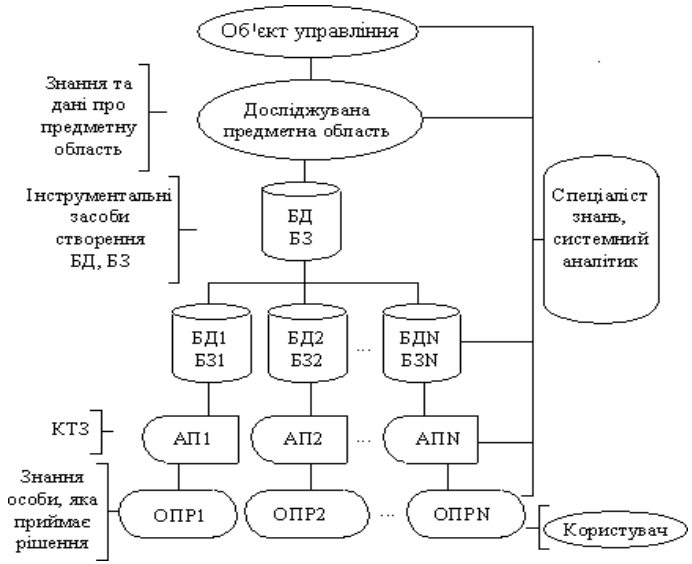
У перших розділах розглянуто загально-теоретичний підхід до створе процес та життєвий цикл інформаційних систем. Третій розділ присвячено методології та технологіям створення МОІС. В четвертому розділі розглянуто питання стандартизації проектування та створення МОІС. П’ятий розділ присвіяено питанням застосування мультимедійних технологій при розробці освітніх інформаційних систем.

РОЗДІЛ І. ЗАГАЛЬНО-ТЕОРЕТИЧНІ ПИТАННЯ ПРОЕКТУВАННЯ ТА СТВОРЕННЯ МУЛЬТИМЕДІЙНИХ ОСВІТНІХ ІНФОРМАЦІЙНИХ СИСТЕМ

1.1 Базові поняття інформаційних технологій і систем

Розглянемо основні поняття проаналізувавши схему взаємозв'язку при функціонуванні інформаційної системи (рис. 1.1).

Рис. 1.1. Схема взаємозв'язку при функціонуванні інформаційної



системи

Об'єкт управління – це люди, матеріальні цінності, уявні побудови, моделі, події чи факти, про які можуть бути зібрані дані.

Предметна область – це означена будь-якими ознаками сукупність об'єктів (усі елементи знання про процес, проблему, організацію, систему та ін.).

Об'єктами можуть бути обрані різні класи систем управління: технологічний процес, галузь, виробниче об'єднання, підприємство, цех, дільниця, робітник і т.п.

Інформаційні системи – це людино-машинні системи, які збирають, нагромаджують, зберігають, оброблюють і видають за запитом чи замовленням інформацію у вигляді даних і знань, необхідних для керування об'єктом управління.

Автоматизація установи – застосування системи оброблення

інформації в діловодстві та управлінській діяльності установи.

Система автоматизації установи – система оброблення інформації, яка використовується для інтеграції діяльності установи.

Особа, яка приймає рішення (ОПР) – це спеціаліст, що керує об’єктом управління.

Користувач ІС – особа, що бере участь у функціонуванні ІС, або має право використовувати і використовує результати її функціонування.

Комплект технічних засобів (КТЗ) - сукупність взаємопов'язаних єдиним управлінням автономних технічних засобів збирання, накопичення, обробки, передачі, ведення та подання інформації, пристроїв управління ними. Сюди ж належать засоби оргтехніки, призначені для організації тривалого збереження (накопичення) інформації і здійснення інформаційного обміну між різними технічними засобами.

Програмне забезпечення (ПЗ) - комп'ютерні програми та відповідна документація.

Розробляється за приватним замовленням або для продажу на ринку ПЗ.

Інженерія ПЗ - інженерна дисципліна, що охоплює всі аспекти розробки ПЗ.

Системотехніка (технологія створення обчислювальних систем) - дисципліна, яка охоплює всі аспекти створення і модернізації складних обчислювальних систем, де програмне забезпечення відіграє провідну роль.

Сюди можна віднести технології створення апаратних засобів, створення обчислювальних процесів, розгортання всієї системи, а також технологію створення безпосередньо ПЗ.

Процес створення ПЗ - сукупність процесів, що призводять до створення програмного продукту.

Фундаментальні процеси, властиві будь-яким проектом створення ПЗ:

- Розробка специфікації вимог на ПЗ (Визначають функціональні характеристики системи і обов'язкові для виконання).
- Створення програмного забезпечення (створення ПЗ згідно специфікації).
- Атестація ПЗ (Створене ПЗ повинно пройти атестацію для підтвердження відповідності вимогам замовника).
- Модернізація ПЗ (вдосконалення ПЗ згідно зміненим вимогам споживача).

Модель процесу створення ПЗ - послідовність етапів, необхідних для розробки створюваного ПЗ:

1.Каскадна модель

2.Еволюційна модель

3.Формальне перетворення

4.Складання програмних продуктів з раніше створених компонентів (модель збірки)

5.Ітераційна (спіральна) модель

Методи створення ПЗ являють собою структурний підхід до створення ПЗ, який сприяє виробництву ПЗ ефективним, з економічної точки зору, способом.

Всі методи засновані на використанні моделей системи в якості специфікації її структури:

- **Функціонально-орієнтовані** (структурний аналіз, JSD, 70- е роки) засновані на визначенні основних функціональних компонент системи.
- **Об'єктно-орієнтовані** (Booch, Rumbaugh) використовують підходи, засновані на використанні уніфікованої мови моделювання UML.
- **Computer-Aided Software Engineering** - автоматизована розробка ПЗ.

Базові процеси створення ПЗ:

- Розробка специфікації.
- Проектування і реалізація.
- Атестація.
- Еволюція.

Життєвий цикл ПЗ - сукупність процесів, що протікають від моменту прийняття рішення про створення ПЗ до його повного виведення з експлуатації.

1.2. Класифікація інформаційних систем

Інформаційні системи можуть значно різнитися за типами об’єктів управління, характером та обсягом розв’язуваних задач і рядом інших ознак, тому їх можна класифікувати за такими ознаками.

1. За рівнем або сферою діяльності - державні, територіальні (регіональні), галузеві, об’єднань, підприємств або установ, технологічних процесів. (стратегічні/ функціональні / операційні).

Державні ІС призначені для складання перспективних та поточних планів розвитку країни, обліку результатів та регулювання діяльності окремих ланцюгів народного господарства, розроблюють державний бюджет та контролюють його виконання і та ін. До них відносяться автоматизована система державної статистики (АСДС), автоматизована система планових розрахунків (АСПР), державна інформаційна система фінансових розрахунків (АСФР) при

Міністерстві фінансів України, система обробки інформації з цін (АСОІ цін), система управління національним банком (АСУ банк), система обробки науково-технічної інформації (АСО НТІ) і т.ін.

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

Галузеві ІС управління призначені для управління підвідомчими підприємствами та організаціями. Галузеві ІС діють у промисловості та в сільському господарстві, будівництві на транспорті і та ін. В них розв'язуються задачі інформаційного обслуговування апарату управління галузевих міністерств і їх підрозділів. Галузеві ІС відрізняються сферами застосування - промислова, непромислова, наукова.

Інформаційні системи управління підприємствами (ІСУП) або виробничими об'єднаннями (ІСУ ВО) — це системи із застосуванням сучасних засобів автоматизованої обробки даних, економіко-математичних та інших методів для регулярного розв'язування задач управління виробничо-господарського діяльністю підприємства.

Інформаційні системи управління технологічними процесами (ІСУ ТП) керують станом технологічних процесів (робота верстата, домни тощо). Перша й головна відмінність цих систем від розглянутих раніше полягає передусім у характері об'єкта управління: - для ІСУ ТП це різноманітні машини, прилади, обладнання, а для державних, територіальних та інших АСУ — це колективи людей. Друга відмінність полягає у формі передачі інформації: - для ІСУ ТП основною формою передачі інформації є сигнал, а в інших ІСУ — документи.

За рівнем автоматизації процесів управління - інформаційно-пошукові, інформаційно-довідкові, інформаційно-керівні, системи підтримки прийняття рішень (СППР), інтелектуальні ІС.

Залежно від мети функціонування та завдань, які покладені на ІС на етапах збору та змістової обробки даних, розрізняють такі типи ІС: інформаційно-пошукові, інформаційно-довідкові, інформаційно-управляючі (управлінські), інтелектуальні інформаційні системи та системи підтримки прийняття рішень.

Інформаційно-пошукові системи (ІСП) орієнтовані на розв'язування задач пошуку інформації. Змістова обробка інформації у

таких системах відсутня.

В інформаційно-довідкових системах (ІДС) за результатами пошуку обчислюють значення арифметичних функцій.

Інформаційно-управляючі, або управлінські, системи (відомі у вітчизняній літературі під назвою «автоматизовані системи організаційного управління») являють собою організаційно-технічні системи, які забезпечують вироблення рішення на основі автоматизації інформаційних процесів у сфері управління. Отже, ці системи призначені для автоматизованого розв'язування широкого кола задач управління.

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

СППР - це інтерактивна комп'ютерна система, яка призначена для підтримки різних видів діяльності при прийнятті рішень із слабо структурованих або неструктурованих проблем.

Інтерес до СППР, як перспективної галузі використання обчислювальної техніки та інструментарію підвищення ефективності праці у всіх сферах управління, постійно зростає. У багатьох країнах розробка та реалізація СППР перетворилася на дільницю бізнесу, що швидко розвивається.

Штучний інтелект - це штучні системи, створені людиною на базі ЕОМ, що імітують розв'язування людиною складних творчих задач. Створенню інтелектуальних інформаційних систем сприяла розробка в теорії штучного інтелекту логіко-лінгвістичних моделей. Ці моделі дають змогу формалізувати конкретні зміс- товні знання про об'єкти управління та процеси, що відбуваються в них, тобто ввести в ЕОМ логіко-лінгвістичні моделі поряд з математичними. Логіко-лінгвістичні моделі - це семантичні мережі, фрейми, продукувальні системи - іноді об'єднуються терміном «програмно-апаратні засоби в системах штучного інтелекту».

Розрізняють три види інтелектуальних ІС:

- інтелектуальні інформаційно-пошукові системи (системи типу «запитання-відповідь»), які у процесі діалогу забезпечують взаємодію кінцевих користувачів — нейропрограмістів з базами даних та знань професійними мовами користувачів, близьких до природних;
- розрахунково-логічні системи, які дають змогу кінцевим користувачам, що не є програмістами та спеціалістами в галузі прикладної математики, розв'язувати в режимі діалогу з ЕОМ свої задачі з використанням складних методів і відповідних прикладних програм;

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

3. За ступенем централізації обробки інформації - централізовані ІС, децентралізовані ІС, інформаційні системи колективного використання.

4. За ступенем інтеграції функцій - багаторівневі ІС з інтеграцією за рівнями управління (підприємство — об'єднання, об'єднання — галузь і т.ін.), багаторівневі ІС з інтеграцією за рівнями планування і т.ін.

5. За типом ІС розподіляються на фактографічні, документальні і документально-фактографічні ІС.

Документальна ІС — це система, в якій об'єктом зберігання і обробки є власне документи.

Фактографічна ІС — це система, в якій, об'єктом або сутністю є дещо, що являє для проблемної сфери багатосторонній інтерес (співробітник, договір, виріб тощо). Відомості про ці сутності можуть знаходитись у множині різних вхідних і вихідних повідомлень.

1.3.Мета, задачі та принципи створення інформаційних систем

Мета створення інформаційних систем – у гранично короткі терміни створити систему обробки даних, яка має задані споживчі якості. До них належать: функціональна повнота, своєчасність, функціональна надійність, адаптивна надійність, економічна ефективність.

Функціональна повнота – це властивість інформаційної системи, яка характеризує рівень автоматизації управлінських робіт.

Своєчасність – це властивість інформаційної системи, яка характеризує можливість отримання апаратом керівництва необхідної інформації.

Функціональна надійність – це властивість інформаційної системи виконувати свої функції з обробки даних. Це сукупність надійностей програмного, інформаційного та технічного забезпечення.

Адаптивна надійність – це властивість інформаційної системи виконувати свої функції, якщо вони змінюються в межах умов, зумовлених розвитком системи керування об'єкта впродовж заданого проміжку часу.

Економічна ефективність інформаційної системи виявляється в покращенні економічних результатів функціонування об'єкта в результаті впровадження інформаційної системи.

Створення інформаційної системи передбачає частковий чи повний перегляд методів і засобів функціонування інформаційної системи об'єкта управління і виконання таких завдань:

- виявлення його суттєвих характеристик;
- створення математичних і фізичних моделей досліджуваної системи та її елементів;
- встановлення умов взаємодії людини та комплексу технічних засобів;
- детальна розробка окремих проектних рішень;
- аналіз проектних рішень, практична апробація та впровадження.

Перше що потрібно зробити це вивчити питання доцільності створення інформаційної системи, що проходить декілька етапів показаних на рис.1.2.

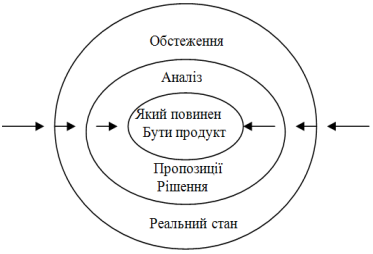


Рис. 1.2 Прийняття рішення про необхідність створення ІС

Інформаційну систему створюють у тих випадках, коли потрібно організувати нові обчислювальні центри, вдосконалити діючу методику й техніку розв'язання задач, впровадити нові задачі, а також організувати інформаційну систему.

Принципи створення інформаційної системи поділяють на дві частини: загальні та часткові.

Загальні принципи мають універсальний характер і визначають методологічний підхід до створення будь-яких об'єктів. Це такі принципи: науковості, нормативності, неперервності, розвитку, ефективності, послідовності, від загального до часткового, системний, комплексності, використання типових і керівних матеріалів.

Часткові принципи: систему управління потрібно розглядати як людино- машинну; чіткий поділ системи на складові, забезпечення сумісності й зв'язку між усіма видами забезпечення; забезпечення єдності обліку, типізація, уніфікація та стандартизація.

При створенні інформаційної системи треба керуватися

принципами, визначеними ДСТУ 3918-99 (ISO/IEC 12207:1995): системності, розвитку (відкритості), сумісності, стандартизації (уніфікації) та ефективності.

У теорії та практиці створення інформаційних систем виділяють три підходи: локальний, глобальний та системний.

Суть **локального** підходу полягає в тому, що інформаційні системи створюють послідовним нарощуванням задач, які розв'язуються в системі управління з допомогою ЕОМ. Він передбачає необмежений розвиток інформаційних систем, а тому кожну із них неможливо пізнати в цілому. Крім того, проект на предмет його повноти взагалі не розглядається та втрачається можливість науково обґрунтувати вибір і оцінити напрямки розвитку інформаційної системи, комплексу технічних засобів, а також побудувати її модель. До позитивних сторін цього підходу віднесемо: відносно швидку віддачу, наочність задач, можливість розробки невеликими «замкненими» групами, простоту керування створенням систем. Недоліки: надмірність інформації, неможливість забезпечення раціональної організації комплексів задач, негнучкість, дублювання, суперечливість, погана стандартизація програм, постійна перебудова програм та організації задач, що призводить до дискредитації самої ідеї створення інформаційної системи.

При **глобальному** підході спочатку розробляють проект немовби повної, завершеної системи, а потім її впроваджують. Як правило, цей підхід призводить до морального старіння проекту ще до його впровадження, оскільки час його розробки може перевищувати період оновлення технічних, програмних та інших засобів, використаних у ньому.

Системний підхід до створення інформаційної системи – це комплексне вивчення об'єкта управління як одного цілого з представленням частин його як цілеспрямованих систем і вивчення цих систем та взаємовідносин між ними. При системному підході об'єкт управління розглядається як сукупність взаємопов'язаних елементів однієї складної динамічної системи, яка перебуває в стані постійних змін під впливом багатьох внутрішніх і зовнішніх факторів, пов'язаних процесами перетворення вхідного набору ресурсів в інші вихідні ресурси.

Про системність великих і складних об'єктів свідчить те, що їх можна поділяти, оскільки вони мають структуру. Процеси декомпозиції й композиції є засобами отримання інформації для здійснення аналізу та синтезу систем.

Декомпозиція – це процес поділу систем на елементи, зручні для якихось операцій з нею, а саме поділ до елементів, які

приймаються за неподільні об’єкти.

Будь яка система по-своєму складна. Це означає що всю сукупність інформації, яка характеризує систему, і всю сукупність зв`язків між елементами системи, неможливо сприйняти в цілому і повністю, звідси, додержуючись методу декомпозиції, для швидкого впровадження ІС необхідно дотримуватися принципу добре структурованої системи”, і тому головна мета декомпозиції – поділ системи на простіші частини. Зменшуючи складність системи, ми забезпечуємо умови для аналізу та синтезу компонентів, для проектування, побудови, впровадження, експлуатації та вдосконалення систем управління. Поділ звичайно виконують у такий спосіб, щоб компоненти піддавались якій-небудь класифікації. Рекомендується зважати на природну декомпозицію, відбиту в існуючій структурі управління, обов’язках посадових осіб, діючого документообігу і т.п. Доцільно проводити багаторазову декомпозицію по кількох різних напрямках.

Загальна мета, критерії функціонування та основні обмеження на роботу системи звичайно формуються на початку створення системи.

Так, при декомпозиції можуть застосовуватись різні засоби, методи та ознаки поділу системи. Поділ може мати матеріальну, функціональну, алгоритмічну та іншу основу. Однак сам процес декомпозиції кінцевий, оскільки поділ відбувається до створення елементів, які приймаються за неподільні об’єкти.

Так, при поділі системи на компоненти можемо мати різні варіанти. **Компонент** — це частина ІС, яку після декомпозиції можемо розглядати як самостійне ціле (ГОСТ 34.003–90).

Інколи пропонують поділ системи здійснювати відповідно до адміністративного поділу системи керування об’єктом. При такій декомпозиції виділяють: керування технічною підготовкою виробництва, техніко-економічне планування, оперативне керування виробництвом і т. ін. Також систему можна поділяти за функціями, які виконуються (облік, контроль, планування і т.п.), і за ресурсами (матеріальні, трудові, основні засоби, готова продукція, грошові).

Наступним кроком декомпозиції є виділення в компоненті функціональних процесів (задач). Задача ІС, функція чи частина функції ІС, є формалізована сукупність автоматизованих дій, в результаті виконання яких здобувають результати заданого виду (ГОСТ 34.003–90). Може виявитися, що при одному й тому самому засобі декомпозиції системи на компоненти одна й та сама задача за змістом в різних проектах належить до різних компонентів. Однак неоднозначність закінчується, тільки- но процес декомпозиції

доводиться до рівня інформаційних показників; його можемо вважати неподільним елементом, оскільки поділ його на атрибути приводить до втрати практичної суті, й він уже не зможе відігравати роль змінної, яка характеризує стан об'єкта, котрий він описує. В інформаційному аспекті показник не є кінцевим елементом і може бути поділений на атрибути. У свою чергу в лінгвістичному аспекті атрибути також не є кінцевими елементами, оскільки можуть бути поділені на окремі слова та символи.

Отже, вибір основи та межі декомпозиції визначається суттю об'єкта, який досліджується, метою, предметною областю обстеження, запасом знань дослідника відносно об'єкта обстеження.

Проте при поділі системи на різні рівні ієрархії потрібно виконувати наступні вимоги:

- кожен рівень ієрархії повинен повністю оглядатися і бути зрозумілим без детального знання нижчих рівнів;
- зв'язки між елементами на одному рівні ієрархії мають бути мінімальними;
- не повинно бути зв`язків між елементами через один рівень ієрархії;
- елемент вищого рівня має викликати елемент наступного рівня і передаючи йому необхідну вхідну інформацію, повинен утворювати з ним єдине ціле;
- елемент наступного рівня після закінчення своєї роботи повертає управління елементу, що його викликав.

Надійність та ефективність є важливими компонентами процесів проектування та створення інформаційних систем. Якість створення інформаційних систем визначається її ефективністю та надійністю.

Основні положення та визначення наведено в ГОСТ 24.701–86 «Надежность АСУ», ГОСТ 24.702–85 «Эффективность АСУ».

Надійність інформаційної системи – це її властивість зберігати в часі в установлених межах значення всіх параметрів, які характеризують здатність системи виконувати потрібні функції в заданих режимах і умовах експлуатації.

Надійність інформаційної системи має властивості безвідмовності, ремонтопридатності, а інколи й довговічності.

Рівень надійності інформаційної системи залежить від таких факторів:

- складу та рівня надійності технічних засобів, їх взаємодії та надійної структури;
- складу та рівня надійності програмних засобів, їх можливостей і взаємозв'язку в структурі програмного

забезпечення інформаційної системи;

- раціонального розподілу задач, які розв'язуються системою, між технічними засобами, програмним забезпеченням і персоналом;
- рівня кваліфікації персоналу, організації робіт і рівня надійності дій персоналу інформаційної системи;
- режимів, параметрів і організаційних форм технічної експлуатації комплексу технічних засобів;
- ступеня використання різних видів резервування (структурного, інформаційного, часового, алгоритмічного, функціонального);
- ступеня використання методів і засобів технічної діагностики;
- реальних умов функціонування інформаційної системи.

Ефективність інформаційної системи визначається порівнянням результатів від функціонування інформаційної системи і затрат усіх видів ресурсів, необхідних для її створення, функціонування та розвитку.

До показників затрат ресурсів відносять матеріальні, людські, фінансові, часові та ін. Ефективність інформаційної системи оцінюють у таких випадках:

- при формуванні вимог, що висуваються до інформаційної системи;
- при аналізі інформаційних систем, які створюються чи функціонують, на відповідність заданим критеріям;
- при виборі найкращого варіанта створення, функціонування та розвитку інформаційної системи;
- при синтезі найдоцільнішого варіанта побудови інформаційної системи за критерієм «ефективність – затрати».

Питання до самоконтролю:

1. Яка мета створення інформаційної системи?
2. Назвати основні задачі, які потрібно розв'язувати в процесі створення інформаційної системи?
3. Які основні принципи створення інформаційної системи?
4. Які є підходи до створення інформаційної системи?
5. Які основні принципи системного підходу?
6. Що таке декомпозиція інформаційної системи?
7. Які можливі структури при аналізі й описуванні інформаційних систем?
8. Що таке надійність і ефективність інформаційної системи?

9.

РОЗДІЛ 2. ПРОЦЕС СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ. ЖИТТЄВИЙ ЦИКЛ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Етапи створення інформаційної системи

У процесі створення інформаційної системи (ІС) можна виділити 4 базових етапи/стадії (рис.2.1.):

- Специфікація – визначення основних вимог.
- Розроблення – створення ІС відповідно до специфікацій.
- Тестування – перевірка ІС на відповідність вимогам клієнта.
- Супровід/Модернізація – розвиток ІС відповідно до змін потреб замовника

Рисунок 2.1. – Схема життя ІС

Етап **«Специфікація»** являє собою визначення сервісів, якими буде володіти створювана ІС, а також обмежень, що накладаються на функціональні можливості і розробку ПЗ.

Результат процесу визначення вимог – документація, що формалізує вимоги до системи.

Два рівня деталізації:

- Вимоги, що пред’являються кінцевими користувачами;
- Системна специфікація для розробників.

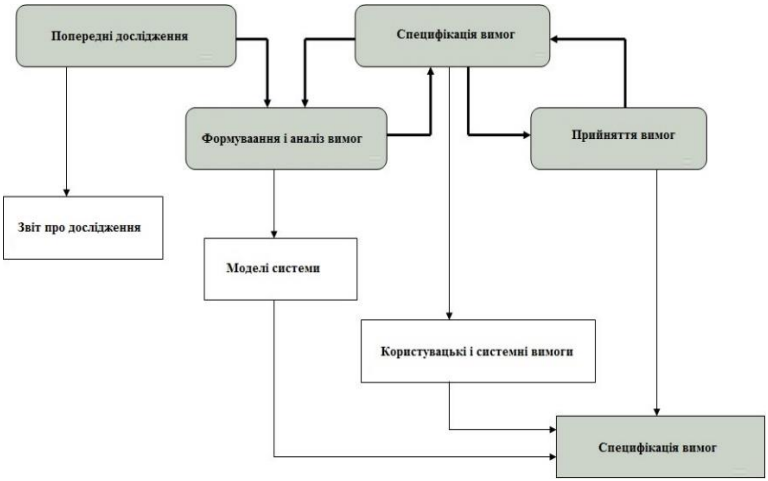


Рис.2.2. Процес специфікації вимог

Етап **«Розроблення»** ІС – процес перекладу системної специфікації в працездатну систему. Включає в себе процеси проектування та програмування.

Процес проектування – це визначення структури даних ІС, інтерфейсів взаємодії системних компонентів, алгоритмів для використання. Проектування передбачає послідовну формалізацію і деталізацію створюваної ІС.

Результат кожного етапу розроблення – специфікація, необхідна для виконання наступного етапу (рис. 2.3.).

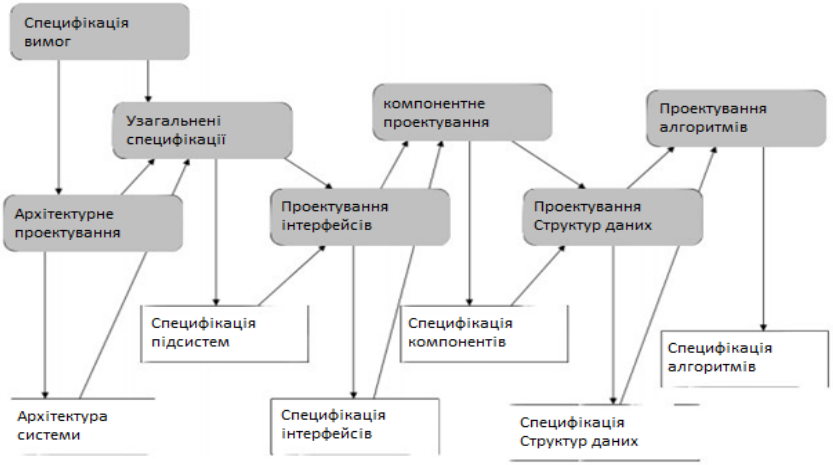


Рис.2.4.Процес формування специфікації вимог

Методи проектування – множина формалізованих нотацій і нормативних документів для проектування ПО.

Структурні методи підтримують моделі системи:

- Модель потоків даних;
- Модель «сутність-зв'язок»;
- Структурна модель;
- Об'єктно-орієнтовані ієрархічна модель системи, модель відносин між об'єктами, модель взаємодії об'єктів;
- Діаграми переходів або сценарії життя сутностей.

Етап **«Тестування»** - процес встановлення програмних помилок. Включає процес налагодження, тобто встановлення місця розташування помилок та їх усунення.

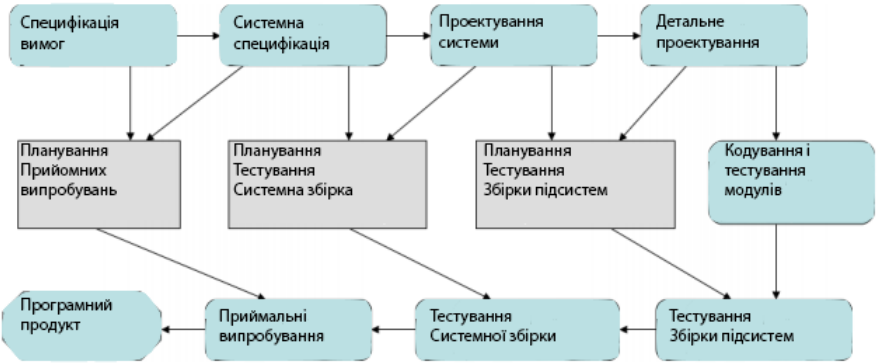


Рис. 2.5.Етапи тестування

Етап «Введення в дію» - це внесення змін до системи, яка знаходиться в експлуатації. На фазі введення системи до експлуатації проводяться: випробування окремих компонент та ІС в цілому, дослідна експлуатація системи в реальних умовах, переговори щодо результатів виконання проекту, можливих змін та нових контрактів.

Ключові роботи цього етапу:

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

досвіду, стану, визначення напрямів розвитку.

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

- помилки у визначенні інтересів замовника;
- концентрація на другорядних інтересах;
- хибна інтерпретація початкового завдання;
- невірне або недостатнє розуміння деталей;
- неповнота функціональних специфікацій;
- вади у визначенні необхідних ресурсів і термінів виконання робіт;
- недостатня перевірка на узгодженість етапів, відсутність контролю з боку замовника.

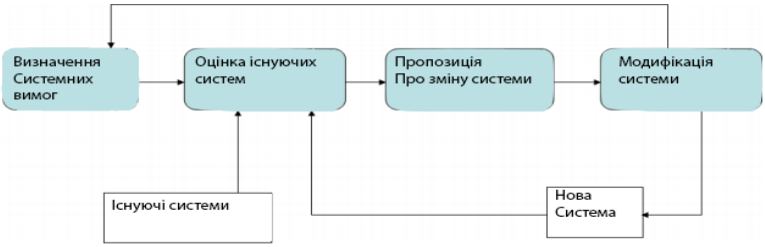


Рис. 2.6. Еволюція систем

2.2. Життєвий цикл інформаційної системи

Оскільки розроблення та супровід ІС фактично є проектною діяльністю, частина ключових понять управління проектів знайшла широке застосування у програмній інженерії. Таким є і поняття **життєвого циклу** проекту (Project Lifecycle Management, PLM), що в програмній інженерії трансформувалось у поняття життєвого циклу програмного забезпечення.

Життєвий цикл ІС - період часу, що починається з моменту прийняття рішення про необхідність створення інформаційної системи і закінчується в момент його повного вилучення з експлуатації.

ГОСТ 34.601-90 визначає життєвий цикл автоматизованої системи (АС) як сукупність взаємозв'язаних процесів створення та

послідовної зміни стану АС, від формування вхідних вимог до неї до закінчення експлуатації та утилізації комплексу засобів автоматизації.

Як і будь-що модель, модель ЖЦ є абстракцією реального процесу, в якій відсутні деталі, несуттєві з точки зору призначення моделі.

Поняття ЖЦ виникло під впливом потреби у систематизації робіт у процесі розроблення ІС. Систематизація була першим етапом на шляху до автоматизації процесу розроблення ІС. Наступними кроками переходу до автоматизації процесу розроблення ІС були такі: встановлення технологічних маршрутів діяльності розробників ІС, визначення можливості їх автоматизації та виявлення ризиків, розроблення інструментів для автоматизації.

Спочатку з'явилися інструменти підтримки розроблення програмного коду та налагодження програм (60-ті рр. XX ст.). Після усвідомлення недостатності таких засобів для істотного підвищення якості програм та створення інструментів керування процесом розроблення виникло поняття життєвого циклу ІС.

Виявлення закономірностей розвитку програмного забезпечення одразу показало нерозвиненість методик конструювання ІС та недостатність тестування для визначення якості програмних продуктів. Також на цьому етапі стало зрозуміло, що нечіткість завдання на створення ІС викликає більшість проблем розроблення та перевірки програм. У результаті виникли вимоги до постановки завдання, сформувалися підходи до управління вимогами на етапі аналізу та інструменти зв'язку вимог на етапах аналізу та реалізації.

Використання поняття життєвого циклу дозволяє обрати підходи, які найбільш ефективні для завдань певного етапу життя ІС. Залежно від особливостей процесів розроблення та супровіду програм існують різні моделі ЖЦ.

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

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

- **основні** – придбання, постачання, розроблення, експлуатація, супроводження;
- **допоміжні**, що забезпечують виконання головних процесів (документування, забезпечення якості, тестування, атестація, оцінка ефективності, аудит, ліцензування, сертифікація тощо);

- **управлінські** – управління проектами, створення інфраструктури проекту, навчання.

До основних процесів відносять:

1. *процес придбання*, що ініціює життєвий цикл ІС та визначає її покупця; передбачають виконання замовлення та постачання продукту замовнику;
2. *процес розроблення*, що визначає дії організації-розробника інформаційного продукту; передбачає дії, що виконуються розробником, і охоплює роботи зі створення ІС та його компонентів відповідно до вимог, включаючи оформлення проектної й експлуатаційної документації, підготовку матеріалів, необхідних для перевірки працездатності і відповідної якості програмних продуктів, матеріалів, потрібних для організації навчання персоналу;
3. *процес постачання*, що визначає дії під час передачі розробленого продукту покупцеві;
4. *процес експлуатації*, що означає дії з обслуговування системи під час її використання – консультації користувачів, вивчення їхніх побажань тощо;
5. *процес супроводження*, що означає дії з керування модифікаціями, підтримки актуального стану та функціональної придатності, інсталяції та вилучення версій систем у користувача.

Процес розроблення ІС має забезпечити шлях від усвідомлення потреб замовника до передачі йому готового продукту (рис. 2.7.). Він складається з таких *етапів*:

Рис 2.7. Процеси розроблення ІС

1. визначення вимог – збір та аналіз вимог замовника виконавцем та подання їх у нотації, що зрозуміла як замовнику, так і виконавцю;
2. проектування – перетворення вимог до розроблення у

послідовність проектних рішень щодо способів реалізації вимог: формування загальної архітектури програмної системи та принципів її прив'язки до конкретного середовища функціонування; визначення детального складу модулів кожної з архітектурних компонент;

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

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

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

Управління проектом щільно пов'язано з плануванням і організацією робіт, створенням колективу проектувальників, з контролем за термінами та якістю реалізації всіх етапів розробки ІС. Вирішення цих питань покладено на технічне й організаційне забезпечення проекту, яке передбачає:

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

Забезпечення якості проекту щільно пов'язане з проблемами верифікації3, перевірки й тестування компонентів ІС.

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

2.3 Моделі життєвого циклу інформаційної системи

Використання поняття життєвого циклу дозволяє обрати підходи, які найбільш ефективні для завдань певного етапу життя ІС. Залежно від особливостей процесів розроблення та супровіду програм існують різні моделі ЖЦ.

Використання певної моделі ЖЦ дозволяє визначитися з основними моментами процесу замовлення, розроблення та супроводу ІС навіть недосвідченому програмісту. Також використання моделей дозволяє чітко зрозуміти, в який період переходити від версії до версії, які дії з удосконалення виконувати, на якому етапі. Знання про закономірності розвитку програмного продукту, які відбиваються в обраній моделі ЖЦ, дозволяють отримати надійні орієнтири для планування процесу розроблення та супроводу ІС, економно витрачати ресурси та підвищувати якість управління усіма процесами.

Також моделі життєвого циклу є основою знань технологій програмування та інструментарію, що їх підтримує. Будь-що технологія базується на певних уявленнях про життєвий цикл та організує свої методи та інструменти навколо фаз та етапів ЖЦ.Розвиток методологій програмування у 70-х рр. XX ст. привів до формування потреби вивчення життєвого циклу ІС. До цього часу моделі ЖЦ розвиваються і модифікуються, уточнюючи та доповнюючи дві базові моделі – каскадну та ітеративну. Ці зміни обумовлені потребою організаційної та технологічної підтримки проектів з розроблення ІС.

Каскадна модель (waterflow model)

Каскадна модель ЖЦ ІС виникла для задоволення потреби у систематизації робіт ще на ранніх етапах розроблення програм. Згідно з цією моделлю програмні системи проходять в своєму розвитку дві фази:

- розроблення;
- супровід.

Фази розбиваються на ряд етапів (рис. 2.8).

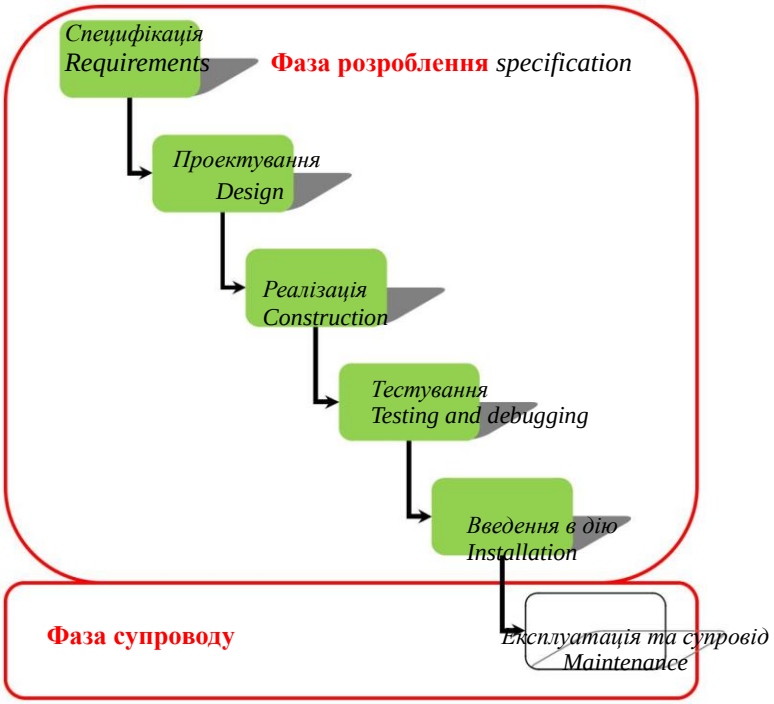


Рис. 2.8. - Каскадна модель створення ІС

Каскадна модель передбачає послідовне виконання всіх етапів проекту в строго фіксованому порядку. Перехід на наступний етап означає повне завершення робіт на попередньому етапі.

Розроблення починається з ідентифікації потреби в новому додатку, а закінчується передачею продукту розроблення в експлуатацію. Усі етапи розроблення програмного забезпечення регламентуються стандартами підприємства та державним стандартом ГОСТ 34.601-90 [7].

Першим етапом фази розроблення є **специфікація** (Requirements Specification) – постановка завдання і визначення вимог. На етапі постановки завдання замовник спільно з розробниками приймають рішення про створення системи. Визначення вимог включає опис загального контексту задачі, очікуваних функцій системи та її обмежень. Особливо важливим є цей етап для нетрадиційних додатків. У разі позитивного рішення починається аналіз системи відповідно до вимог. Розробники програмного забезпечення намагаються осмислити висунуті замовником вимоги і

зафіксувати їх у вигляді специфікацій системи. Призначення специфікацій – описувати зовнішню поведінку системи, а не її внутрішню організацію.

Перш ніж розпочати створювати проект за специфікаціями, вимоги повинні бути ретельно перевірені на відповідність вихідним цілям, повноту, сумісність (несуперечність) та однозначність. Завдання етапу аналізу полягає в тому, щоб вибудувати опис програми у вигляді логічної системи, зрозумілої як для замовника, майбутніх користувачів, так і для виконавців проекту. На етапі специфікації обов'язково формується технічне завдання на розроблення ПП [8].

Розроблення проектних рішень, що відповідають на питання, як повинна бути реалізована система, щоб вона могла задовольняти визначені вимоги, виконується на етапі **проектування** (Design). Оскільки складність системи в цілому може бути дуже великою, головним завданням цього етапу є послідовна декомпозиція системи до рівня очевидно реалізованих модулів або процедур. Результати виконання цього етапу оформлюютья як технічний проект, вимоги до документів якого встановлені стандартом ГОСТ 34.201-89 [9].

На наступному етапі **реалізації** (Construction), або кодування, кожен з цих модулів програмується на найбільш підходящій для даного застосування мові. З точки зору автоматизації цей етап традиційно є найбільш розвиненим.

У каскадній моделі фаза розроблення закінчується етапом **тестування** (Testing and debugging), автономного і комплексного, та передачею системи в експлуатацію (Installation).

Фаза експлуатації та супроводу, включає в себе всю діяльність щодо забезпечення нормального функціонування програмних систем, у тому числі фіксування розкритих під час виконання програм помилок, пошук їх причин та виправлення, підвищення експлуатаційних характеристик системи, адаптацію системи до довкілля, а також за необхідності і більш суттєві роботи з удосконалення системи. Фактично відбувається еволюція системи. У ряді випадків на дану фазу припадає більша частина коштів, що витрачаються в процесі життєвого циклу програмного забезпечення.

Зрозуміло, що увага програмістів до тих чи інших етапів розроблення залежить від конкретного проекту. Часто розробнику немає необхідності проходити через усі етапи, наприклад, якщо створюється невелика, добре зрозуміла програма із чітко поставленою метою.

Стисла характеристика:

1. фіксований набір стадій; кажна стадія закінчується документованим результатом;

2. наступна стадія починається лише після закінчення попередньої.

Недоліки:

1. негнучкість;
2. фаза повинна бути завершена до переходу до наступної;
3. набір фаз фіксований;
4. важко реагувати на зміни вимог.

Використання: там, де вимоги добре зрозумілі та стабільні.

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

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

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

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

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

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

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

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

технології проектування.

Спіральна модель життєвого циклу

Найбільш відомим і поширеним варіантом ітераційної моделі є спіральна модель (рис.2.9.), що була вперше сформульована Баррі Боемом (Barry Boehm) у 1986 році [12]. Відмінною особливістю цієї моделі є спеціальна увага ризикам, що впливає на організацію життєвого циклу.

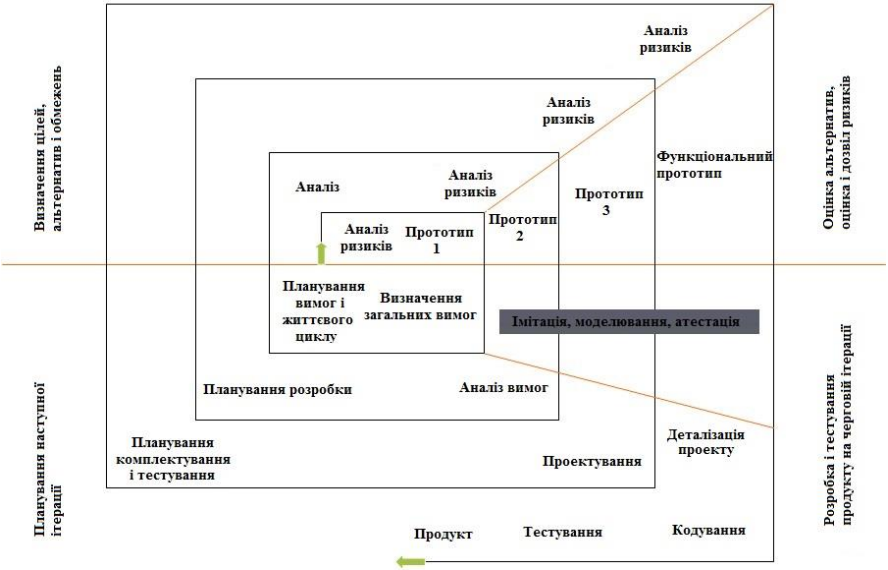


Рис. 2.9. Спіральна модель створення ІС

У спіральній моделі розроблення програми має вигляд серії послідовних ітерацій. На перших етапах уточнюються специфікації продукту, на наступних - додаються нові можливості і функції. Мета цієї моделі - по закінченні кожної ітерації заново здійснити оцінку ризиків продовження робіт.

На кожному витку спіралі виконується створення чергової версії продукту, уточнюються вимоги проекту, визначається його якість і плануються роботи наступного витка. Особлива увага приділяється початковим етапам розроблення – аналізу і проектуванню, де реалізованість тих чи інших технічних рішень перевіряється і обґрунтовується за допомогою створення прототипів (макетування).

Завдяки ітеративній природі спіральна модель допускає коригування у ході роботи, що сприяє поліпшенню продукту.

При великому числі ітерацій розроблення за цією моделлю потребує глибокої автоматизації усіх процесів, інакше вона стає

неефективною. На практиці у замовників і користувачів іноді виникає відчуття нестабільності продукту, оскільки вони не встигають стежити за швидкими змінами в ньому.

Спіральна модель розроблення ІС вимагає визначення ключових контрольних точок проекту - milestones. Виділяють такі основні контрольні точки [12]:

1. Concept of Operations (COO) – концепція використання системи;
2. Life Cycle Objectives (LCO) – цілі та зміст життєвого циклу;
3. Life Cycle Architecture (LCA) – архітектура життєвого циклу;
4. Initial Operational Capability (IOC) – перша версія ІС, що перевіряється у ході дослідницької експлуатації;
5. Final Operational Capability (FOC) – готова ІС, що експлуатується в реальних умовах.

Аналіз у ключових точках особливо актуальний для менеджерів і лідерів проектів, які відстежують хід виконання проекту і планують подальші роботи.

Стисла характеристика:

1. проект має контрольні точки – milestones;
2. на кожному витку спіралі створюється прототип
3. на виході – продукт, що відповідає вимогам користувачів.

Переваги:

1. ранній аналіз можливостей повторного використання;
2. наявні механізми досягнення параметрів якості;
3. модель дозволяє контролювати джерела проектних робіт і відповідних витрат;
4. модель дозволяє вирішувати інтегровані завдання системного розроблення, які охоплюють програмну і апаратну складові створюваного продукту.

Недоліки:

1. після уточнення вимог відкидається частина раніше виконаної роботи;
2. потрібні засоби для швидкого розроблення.

Використання: підходить для малих та середніх проектів.

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

Під час проектування за методологією RAD активно використовуються об’єктно-орієнтовані методи опису предметної

області. Методологія добре підходить для проектів із коротким та добре опрацьованим графіком робіт. Серед відмінностей RAD можна вказати такі:

– розроблення програм ітераціями;
– допустимість часткового завершення робіт на кожному з етапів ЖЦ;
– обов'язкове залучення користувачів до процесу розробки ІС;
– обов'язкове використання CASE-засобів для забезпечення цілісності проекту;
– використання засобів управління конфігурацією, що полегшують внесення змін до проекту й супровід готової системи;
– використання генераторів коду;
– прототипування, що дозволяє краще з'ясувати й задовольнити потреби кінцевого користувача;
– тестування і коригування проекту одночасно з його розробкою;
– ведення розробки невеликою, але добре керованою командою професіоналів (зазвичай від 2 до 10 осіб);
– професійне керівництво розробкою системи, чітке планування і контроль за виконанням робіт.

Життєвий цикл ПЗ за методологією RAD має чотири фази:

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

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

Ітераційний підхід до моделі життєвого циклу

Розвиток каскадної та спіральної моделей призвів до їхнього природнього зближення. Результатом такого зближення стала поява сучасного ітераційного підходу, який фактично становить раціональне поєднання цих двох моделей.

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

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

– Rational Rose – засіб візуального моделювання (аналізу і проектування), використовує мову UML;
– Rational XDE – засіб аналізу і проектування, інтегрується з платформами MS Visual Studio .NET, IBM WebSphere Studio Application Developer та ін.

2. MSF (Microsoft Framework, 1993 р.) схожа з RUP, так само включає чотири фази: аналіз, проектування, розробка, стабілізація, є ітераційною і передбачає використання об'єктно-орієнтованого моделювання. MSF, на відміну від, RUP більшою мірою орієнтована на розробку бізнес-додатків.
3. Методологія XP (Extreme Programming або Екстремальне програмування, 1996 р.). Розробка виглядає як ітеративний процес, де фази розбиваються на малі кроки. В основі методології – командна робота й ефективна комунікація між замовником та виконавцем.

2.4. Інженерія вимог до інформаційних систем

Стадія розробки (інженерії) вимог до ІС – це найважливіша стадія, оскільки вона визначає успіх усього проекту.

Потреби (needs) – відображають проблеми бізнесу, персоналій або процесу, що повинні співвідноситися з використанням або придбанням системи [13].

Розробка вимог – це процес, що включає заходи, необхідні для створення і затвердження документа, що містить специфікацію системних вимог.

Вимога (Requirement) – умова або можливість, що визначена користувачем для вирішення проблеми або досягнення мети, та якій повинна відповідати або якою повинна володіти система чи її компонент, щоб задовольняти умови контракту, стандарту, специфікації або іншого формально репрезентованого документа. Вимоги поділяються на:

- вимоги користувача (User Requirements) – описують цілі/задачі користувачів системи, які повинні досягатися/виконуватися користувачами за допомогою створюваної програмної системи [13];
- функціональні вимоги (Functional Requirements) – вимоги, що конкретизують функції, які система або її компонент повинен виконувати;
- програмні вимоги (Software Requirements) – вимоги до створюваної системи, зрозумілі користувачами (замовниками) і розробниками (виконавцями) стосовно того, що робитиме система і чого від неї не варто чекати.

Отже, розрізняють чотири **основні етапи процесу розробки вимог**:

- аналіз технічної здійсненності створення системи,
- формування і аналіз вимог,
- специфікування вимог та створення відповідної документації,
- атестація цих вимог (рис. 2.11.).

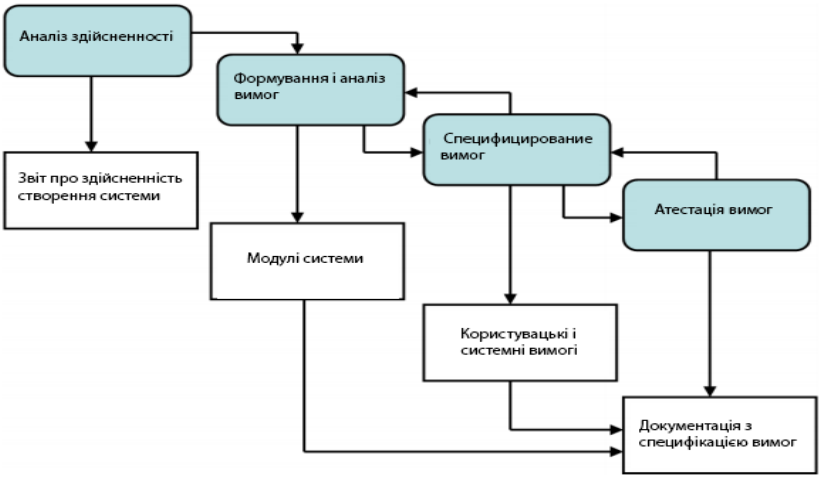


Рис. 2.11 Процес формування вимог

Аналіз технічної здійсненності створення системи повинен висвітлити такі питання:

- Чи відповідає система загальним і бізнес-цілям організації-замовника і організації-розробника?
- Чи можна реалізувати систему, використовуючи існуючі наразі технології і не виходячи за межі заданої вартості?
- Чи можна об'єднати систему з іншими системами, які вже експлуатуються?

Виконання аналізу здійсненності включає збір і аналіз інформації про майбутню систему, написання відповідного звіту. Наприклад, цю інформацію можна отримати, відповівши на наступні питання:

- Що станеться з організацією, якщо система не буде введена в експлуатацію?
- Які поточні проблеми існують в організації і як нова система допоможе їх вирішити?
- Яким чином система буде сприяти цілям бізнесу?
- Чи потребує розробка системи технології, яка до цього не використовувалася в організації?

Після обробки зібраної інформації готується звіт з аналізу здійсненності створення системи.

На **етапі формування і аналізу вимог** команда розробників ІС працює з замовником і кінцевими користувачами системи для з'ясування області застосування, опису системних сервісів, визначення режимів роботи системи та її характеристик виконання, апаратних обмежень і т.д. Тому, для якісного визначення вимог до ІС потрібно спочатку провести аналіз та сформувати їх специфікації.

Аналіз вимог (Requirements Analysis) – трансформація інформації, отриманої від користувачів (та інших зацікавлених осіб) у чітко та однозначно визначені програмні вимоги, що передаються інженерам для реалізації у програмному коді.

Аналіз вимог включає:

- виявлення і розв'язання конфліктів між вимогами;
- визначення меж задачі, що вирішується створюваним програмним забезпеченням; у загальному випадку – визначення меж (Scope) і змісту програмного проекту;
- деталізацію системних вимог для встановлення програмних вимог.

Процес формування і аналізу вимог досить складний по ряду причин. Особлтво, на вимоги до системи можуть впливати людські чинники, такі як:

- Особи, які беруть участь у формуванні вимог, висловлюють в цих вимогах власні точки зору, грунтуючись на особистому досвіді роботи.
- Особи беруть участь у формуванні вимог, мають різні переваги і можуть висловлювати їх різними способами.
- Розробники повинні визначити всі потенційні джерела вимог і виділити загальні і суперечливі вимоги.
- Економічна і бізнес-обстановка, в якій відбувається формування вимог, неминуче буде змінюватися в ході виконання цього процесу.
- Особи, які беруть участь у формуванні вимог, часто не знають конкретно, чого вони хочуть від комп'ютерної системи (рис.2.12).

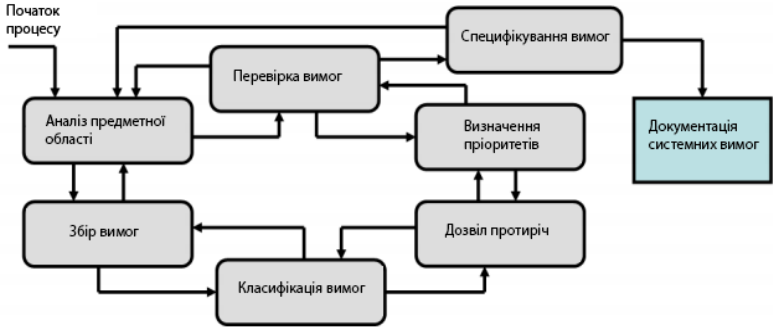


Рис. 2.12 Процес формування і аналізу вимог

Процес формування і аналізу вимог, в свою чергу, проходить через ряд етапів:

- Аналіз предметної області. Аналітики повинні вивчити предметну область, де буде експлуатуватися система.
- Збір вимог. Це процес взаємодії з особами, формують вимоги. Під час цього процесу триває аналіз предметної області.
- Класифікація вимог. На цьому етапі безформний набір вимог перетворюється в логічно пов'язані групи вимозі.
- Дозвіл протиріч. Без сумніву, вимоги численних осіб, зайнятих в процесі формування вимог, будуть суперечливими. На цьому етапі визначаються і вирішуються протиріччя такого роду.
- Призначення пріоритетів. У будь-якому наборі вимог одні з них будуть більш важливі, ніж інші. На цьому етапі спільно з особами, що формують вимоги, визначаються найбільш важливі вимоги.
- Перевірка вимог. На цьому етапі визначається їх повнота, послідовність і несуперечливість.

Поширені три підходи до формування вимог: метод, заснований на безлічі **опорних точок зору**, **сценарії і етнографічний метод**.

Інші підходи, які можуть використовуватися в процесі розробки вимог - це методи структурного аналізу і методи прототипіювання.

Не існує універсального підходу до формування та аналізу вимог. Зазвичай для розробки вимог одночасно використовується кілька підходів. Розглянемо перші три методи.

Метод опорних точок зору

Різні точки зору на проблему дозволяють побачити її з різних сторін. Однак ці погляди не є повністю незалежними і зазвичай перекривають один одного, а тому можуть служити основою загальних вимог.

Підхід з використанням різних опорних точок зору до розробці вимог визнає ці точки зору і використовує їх в якості основи побудови

та організації як процесу формування вимог, так і безпосередньо самих вимог.

Різні методи пропонують різні трактування виразу „точка зору”. Точки зору можна трактувати наступним чином:

- Як джерело інформації про системні даних. У цьому випадку на основі опорних точок зору будується модель створення і використання даних в системі. У процесі формування вимог відбираються всі такі точки зору, на їх основі визначаються дані, які будуть створені або використані при роботі системи, і способи обробки цих даних.
- Як одержувачі системних сервісів. В цьому випадку точки зору є зовнішніми (щодо системи) одержувачами системних сервісів. Точки зору допомагають визначити дані, необхідні для виконання системних сервісів або їх управління.
- Як одержувачі системних сервісів. В цьому випадку точки зору є зовнішніми (щодо системи) одержувачами системних сервісів. Точки зору допомагають визначити дані, необхідні для виконання системних сервісів або їх управління.

Найбільш ефективним підходом до аналізу інтерактивних систем є використання зовнішніх опорних точок зору. Ці точки зору взаємодіють з системою, отримуючи від неї сервіси і продукуючи дані і керуючі сигнали (рис. 2.13).

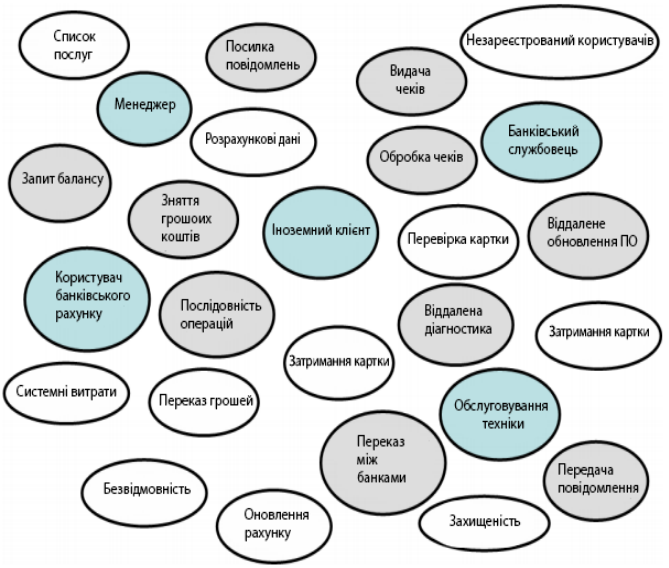


Рис. 2.13 Діаграма ідентифікації точок зору

Цей тип точок зору має ряд переваг:

- Точки зору, зовнішні до системи, - природний спосіб структурування процесу формування вимог.
- Порівняно просто вирішити, які точки зору слід залишити в якості опорних: вони повинні відображати будь або спосіб взаємодії з системою (табл. 1).

Таблиця 1. Сервіси, співвіднесені з точками зору

Власник рахунку	**Закордонний клієнт**	**Касир банку**
Список сервісів	Список сервісів	Список сервісів
Отримання готівки Запит балансу Отримання чеків Відправлення повідомлень Список транзакцій Сортування операцій Переказ готівки	Отримання готівки Запит балансу	Виконання діагностики Зарахування готівки Опрацювання рахунків Відправлення повідомлень

- Даний підхід корисний для створення нефункціональних вимог, з якими можна пов'язати будь-якій сервіс.

Інформація, витягнута з точок зору, використовується для заповнення форм шаблонів точок зору і організації точок зору в ієрархію спадкування. Сервіси, дані і керуюча інформація успадковуються підмножиною точок зору (рис. 2.14).

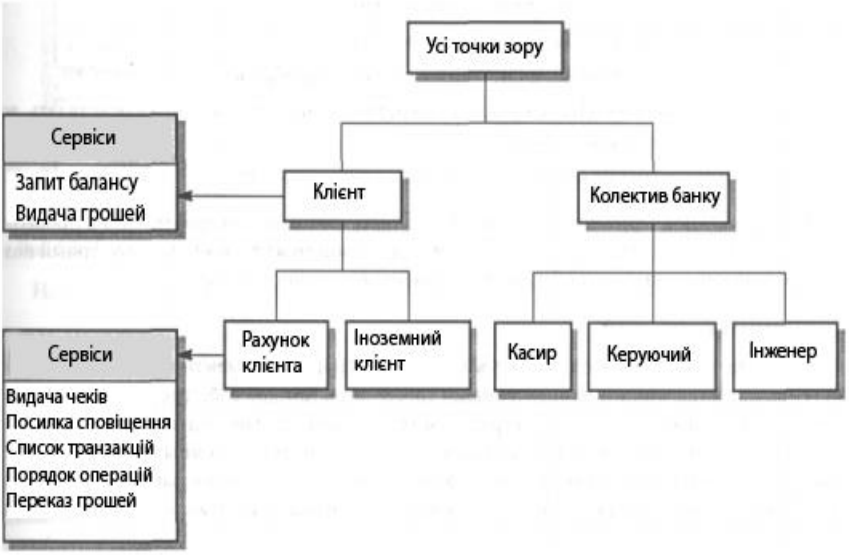


Рис. 2.14. Діаграма користувачів, співвіднесені з можливими сервісами

Сценарії подій - особливо корисні для деталізації вже сформульованих вимог, оскільки описують послідовність інтерактивної роботи користувача з системою. Кожен сценарій описує одне або кілька можливих взаємодій.

Сценарій починається з загального опису, потім поступово деталізується для створення повного опису взаємодії користувача з системою.

У більшості випадків сценарій включає наступне:

- Опис стану системи після завершення сценарію.
- Інформацію щодо інших дій, які можна здійснювати під час виконання сценарію.
- Опис виняткових ситуацій і способів їх обробки.
- Опис нормального протікання подій.
- Опис стану системи на початку сценарію.

Сценарії подій використовуються для документування поведінки системи, представленого певними подіями. Сценарії включають опис потоків даних, системних операцій і виняткових ситуацій, які можуть виникнути (рис. 2.15).

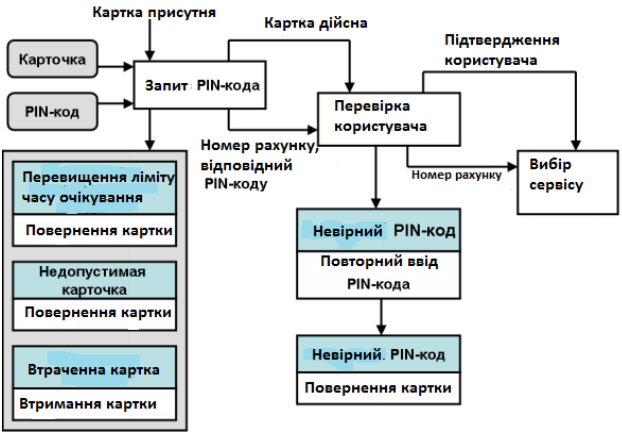


Рис.2.15. Діаграма сценаріїв

Умовні позначення:

1. Дані, що надходять в систему або виходять з неї, представлені в еліпс.

2. Керуюча інформація показана стрілками у верхній частині прямокутників.

3. Внутрішньосистемні дані показані праворуч від прямокутників.

4. Виняткові ситуації показані в нижній частині прямокутників.

5. Ім'я наступного події, очікуваного після завершення сценарію, наводиться в затіненому прямокутнику.

Варіанти використання (use-case) - це методика формування вимог, заснована на сценарії. Вони стали основою нотацій в мові моделювання UML при описі об'єктних моделей систем (рис. 2.16, рис. 2.17).

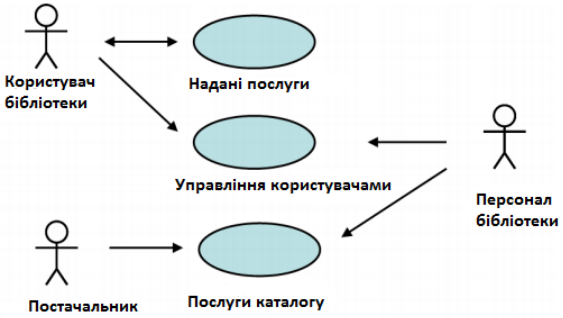


Рис. 2.16. Діаграма варіантів використання

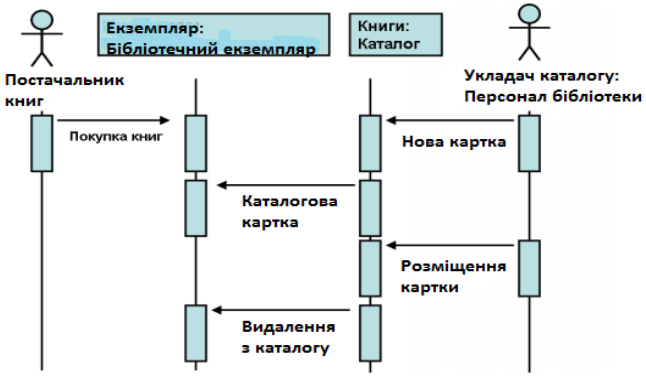


Рис.2.17. Діаграма послідовності

Етнографічний підхід

Етнографічний підхід до формування системних вимог використовується для розуміння і формування соціальних і організаційних аспектів експлуатації системи. Розробник вимог занурюється в робоче середовище, де буде використовуватися система. Його щоденна робота пов'язана з наглядом і протоколюванням реальних дій, які виконуються користувачами системи. Значення етнографічного підходу полягає в тому, що він допомагає виявити неявні вимоги до системи, які відображають реальні аспекти її

експлуатації, а не формальні умоглядні процеси.

Етнографічний підхід дозволяє деталізувати вимоги для критичних систем, чого не завжди можна домогтися іншими методами розробки вимог. Однак, оскільки цей метод орієнтований на кінцевого користувача, він не може охопити всі вимоги предметної області та вимоги організаційного характеру (рис. 2.17).

Рис. 2.17. Процес розробки вимог згідно до етнографічного підходу

Для якісного визначення вимог до ІС після етапу формування та проведення аналізу потрібно сформувати їх специфікації.

Специфікація (Specification) – документ, що в закінченій, точній і перевіреній формі описує вимоги, проект, поведінку або інші характеристики компонента або системи, а також процедури, спрямовані на визначення того, чи задовольняються описані характеристики.

Для опису комплексних проектів (у частині вимог) використовують три *основні специфікації:*

- визначення системи (System Definition), або специфікація вимог користувачів (User Requirements Specification);
- системних вимог (System Requirements);
- програмних вимог (Software Requirements).

Специфікація програмних вимог (Software Requirements Specification – SRS) встановлює основні угоди між користувачами (замовниками) і розробниками (виконавцями) стосовно того, що робитиме система і чого від неї не варто чекати. Цей документ може включати процедури перевірки створеного ПЗ на відповідність вимогам, що висуваються (у т.ч. плани тестування), описи характеристик стосовно якості та методів його оцінювання, питань безпеки тощо [13].

Специфікація вимог користувачів (User Requirements Specification) або концепція (concept <of operation>) визначає високорівневі вимоги, часто – стратегічні цілі, для досягнення яких створюється програмна система. Важливо, що цей документ описує

вимоги до системи з позицій предметної області – домену [13].

Специфікація системних вимог (System Requirements) – описує програмну систему в контексті системної інженерії. Зокрема високорівневі вимоги до програмного забезпечення, що містить кілька або багато взаємозв'язаних підсистем і застосувань. При цьому система може бути як цілком програмною, так і містити програмні та апаратні компоненти. У загальному випадку до складу системи може входити персонал, що виконує певні функції системи, наприклад, авторизацію виконання певних операцій з використанням програмно-апаратних підсистем.

При постановці завдання потрібно, щоб програмні вимоги були зрозумілі, зв'язки між ними прозорі, а зміст специфікації не допускав різночитань та інтерпретацій, через які програмний продукт не буде відповідати потребам зацікавлених осіб. Тому потрібні інструменти управління вимогами.

Управління вимогами (Requirements Management – діяльність, виконання якої забезпечує опис вимог, відстежування їх змін, перевірки на несуперечливість і на порушення наперед визначених правил.

Від вхідної інформації про майбутній програмний продукт залежить те, яку методологію буде обрано в проекті з розроблення ІС. Методологій багато: і дуже формалізованих, і тих, що дають творчу свободу програмістам. Вибір методології обумовлюється досвідом керівника групи розробників та умовами, які встановлюють замовники до документування етапів робіт.

4. Під час процесу **атестації** повинні бути виконані різні типи перевірок документації вимог:

1. Перевірка правильності вимог.
2. Перевірка на несуперечливість.
3. Перевірка на повноту.
4. Перевірка на здійсненність.

Існує ряд методів атестації вимог:

1. Огляд вимог.
2. Прототипування.
3. Генерація тестових сценаріїв.
4. Автоматизований аналіз несуперечності.

Управління вимогами - це процес управління змінами системних вимог. Процес управління вимогами виконується спільно з іншими процесами розробки вимог. Початок цього процесу планується на той же час, коли починається процес первинного формування вимог, безпосередньо процес управління вимогами повинен початися відразу після того, як чорнова версія специфікації вимог буде готова.

З точки зору розробки вимоги можна розділити на два класи: постійні вимоги та змінні вимоги.

Планування управління вимогами

1. Ідентифікація вимог.
2. Управління процесом внесення змін.
3. Стратегія оперативного контролю.

- Інформація про джерело вимоги
- Інформація про вимоги
- Інформація про структуру системи

4. Підтримка CASE-засобів.

Процес управління змінами складається з трьох основних етапів (Рис. 2.16.):

1. Аналіз проблем зміни специфікації.
2. Аналіз змін і розрахунок їх вартості.
3. Реалізація змін.

Рис. 2.16. – Процес управління вимогами

2.5. Розробка програмних продуктів як проектна діяльність

Діяльність під час розроблення ІС, як і будь-що, складається з виконання операцій і проектів. Ті й інші мають багато спільного, наприклад, виконуються людьми та на їх виконання виділяються обмежені ресурси.

Головна відмінність операцій від проектів полягає в тому, що операції виконуються постійно і повторюються, тоді як проект тимчасовий і унікальний. Виходячи з цього, проект визначається як тимчасове зусилля, розпочате для створення унікального продукту чи послуги. «Тимчасове» означає, що кожен проект має точно визначені дати початку та закінчення. Говорячи про унікальність продукту, ми маємо на увазі, що вони мають помітні відмінності від усіх аналогічних продуктів або послуг. Таким чином, розроблення ІС відповідає визначенню проекту і для організації цього процесу можна застосовувати методи та інструментарій управління проектами. Наприклад, розроблення веб-сайту є проектом, тоді як підтримка його впродовж тривалого часу – це операційна діяльність.

У кожного проекту є чітко визначені початок і кінець. Кінець

проекту настає разом із досягненням усіх його цілей або коли стає зрозумілим, що ці цілі не будуть або не можуть бути досягнуті. Тимчасовість не означає короткостроковість проекту – розроблення складної програмної системи може тривати кілька років, хоча, як правило проекти мають обмежені часові рамки для створення ІС, оскільки сприятлива для них ситуація на ринку складається на обмежений час. Крім того, проектна команда після його закінчення розпадається, а її члени переходять в інші проекти.

Проект дуже часто плутають із програмою, тобто координованим управлінням групою проектів всередині однієї організації. Управління відразу декількома проектами скоординоване для того, щоб отримати вигоду, яку неможливо одержати від окремого управління кожним із них.

Проект виконується для досягнення певного результату в певні терміни і за певні гроші. Це **тріо часу (time), бюджету, (cost) і обсягу робіт (scope)** часто називають проектним, або залізним, трикутником (рис. 2.17.), оскільки при внесенні змін в один із цих елементів змінюються інші. Хоча для проекту однаковою мірою важливі всі три елементи, як правило, тільки один із них залежно від пріоритетів має найбільший вплив на інші.

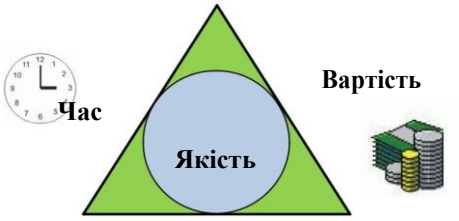


Рис.2.17. Проектний трикутник

Те, як зміни у плані впливають на інші сторони трикутника, залежить від обставин і специфіки проекту. У деяких випадках зменшення терміну збільшує вартість, а в інших - зменшує.

Якість (quality) – це четвертий елемент проектного трикутника. Вона знаходиться у центрі, і будь-що зміна сторін впливає на неї.

Наприклад, якщо керівникові вдалося знайти додатковий час у розкладі, то можна збільшити обсяг робіт, додавши завдання і збільшивши тривалість проекту. Ці додаткові завдання і час дозволяють домогтися більш високого рівня якості проекту і виробленого продукту.

Якщо ж потрібно знизити витрати, щоб вкластися в бюджет,

можливо, знадобиться зменшити обсяг робіт, прибравши деякі із завдань або зменшивши їх тривалість. Зі зменшеним обсягом робіт у проекті буде менше шансів вийти на необхідний рівень якості, тому зниження витрат може призвести до погіршення якості проекту.

План проекту складається для того, щоб визначити, за допомогою яких робіт буде досягатися результат проекту, які люди й устаткування потрібні для виконання цих робіт і в що час ці люди й устаткування будуть зайняті роботою в проекті. Тому проектний план містить **три основні елементи**: завдання, ресурси і призначення.

Завдання (Tasks)

Завданням є робота, здійснювана у рамках проекту для досягнення певного результату. Наприклад, у проекті створення програми для автоматизації формування документації підприємства завданням буде Створення шаблонів.

Оскільки, як правило, проект містить багато завдань, то для зручності відстеження плану їх об'єднують у групи, або фази. Сукупність фаз проекту називається його життєвим циклом.

Фази (Summary tasks)

Фаза проекту складається з одного або кількох завдань, у результаті виконання яких досягається один або кілька основних результатів проекту. Таким чином, результати, досягнуті завдяки виконанню кожної із задач, що входять у фазу, формують її результат. Фази можуть складатися як із завдань, так і з інших фаз.

Якщо для досягнення результатів завдання потрібно виконати тільки її, то для досягнення результату фази потрібно виконати групу інших завдань.

При плануванні робіт потрібно пам'ятати, що чим детальніше буде план проекту, тим точніше він буде відповідати реальній ситуації. У тих випадках, де це можливо, варто розбивати великі завдання на підзадачі (перетворювати завдання в фази). Формальними критеріями, які показують, що завдання можна розбити на підзадачі, є тривалість (завдання рідко тривають довше 2-3 днів) і велике число задіяних виконавців (як правило, якщо над вирішенням завдання працюють більше 2-3 осіб, то кожен вирішує свою власну задачу, яку можна окремо врахувати у плані проекту).

Завершальні завдання

Завдання, у результаті виконання яких досягаються проміжні цілі, називаються завершальними завданнями.

Тривалість (Duration) і трудовитрати (Work)

Тривалість завдання - це період робочого часу, що необхідний для того, щоб виконати її.

Тривалість може не відповідати трудовитратам співробітника, що займається завданням. Тривалість відповідає часу, через що буде отриманий результат задачі, а трудовитрати– часу, витраченому співробітниками на отримання результату.

Залежності (Dependencies) та зв'язки (Links)

Завдання у плані проекту взаємозв’язані. Наприклад, часто одна задача не може початися, поки не закінчена інша (тестування модуля не може початися раніше написання його коду).

На плані проекту залежності позначаються за допомогою зв'язків. Обидва ці терміни – залежність і зв’язок – використовуються з одним і тим самим змістом і позначають логіку, певну послідовність робіт у плані проекту.

Ролі (Roles) і ресурси (Resources)

Під ресурсами розуміють співробітників та обладнання, які необхідні для виконання проектних завдань.

Кожен співробітник, що бере участь у проекті, отримує певну роль відповідно до своєї кваліфікації, вимог проекту та регламентів, що діють в організації. Наприклад, в одному проекті співробітник може виступати в ролі архітектора додатків, а в іншому той самий працівник може бути задіяний у ролі програміста.

При складанні списку ресурсів часто використовується рольове планування. Наприклад, спочатку визначається, що для виконання робіт потрібні три програмісти і один менеджер, а потім, коли план проекту затверджений, вибираються конкретні співробітники для участі в цих ролях.

Призначення (Assignments)

Призначення - це зв’язок певної задачі і ресурсів, необхідних для її виконання. При цьому на одну задачу можуть бути призначені декілька ресурсів, як матеріальних, так і нематеріальних.

Призначення об’єднують у плані ресурси і завдання, роблячи план цілісним. Завдяки призначенням вирішується цілий ряд завдань планування. По-перше, визначаються відповідальні за виконання завдань. По-друге, коли визначені завдання, за які відповідає ресурс, можна розрахувати загальний обсяг часу, що витрачається ним на проект, а отже, його вартість для проекту. По-третє, визначивши вартість участі всіх ресурсів у проекті, можна підрахувати його загальну вартість. Нарешті, призначаючи ресурси на завдання, можна скорочувати термін виконання робіт, виділяючи на них більше ресурсів і тим самим скорочуючи загальну тривалість проекту.

2.6. Управління проектом розробки інформаційних систем

Планування робіт

Результатом фази оцінки здійсненності є детальна специфікація (технічне завдання), план роботи та оцінка вартості. Найбільш традиційними формами плану можна вважати *мережевий графік та діаграму Ганта (Gantt diagram).*

Діаграма Ганта дозволяє визначити, що частина робіт повинна бути виконана у кожен момент часу, тому її частіше використовують керівники проектів, тоді як технічні фахівці віддають перевагу мережевим графікам, оскільки вони містять інформацію про тривалість кожного виду роботи.

Мережевий графік подається у вигляді орієнтованого графа з двома виділеними вершинами - початок і кінець роботи. Вершинами графа є події, що відповідають пунктам плану, а передані в базу вхідних даних", "усі тести пропущені", "група оцінки якості дала позитивний висновок" і т.д. Ребра навантажуються оцінками тривалості робіт, наприклад, у днях або тижнях.

Переваги використання мережевого графіка при плануванні робіт полягають у такому:

- на ньому чітко видно залежність робіт одна від одної;
- на основі мережевого графіка можна обчислити тривалість усієї роботи;
- користуючись результатами цих розрахунків, можна оптимізувати тривалість і витрати на роботу.

Тривалість робіт обчислюється так: підсумовуємо тривалості робіт за всіма можливими шляхами в графі. Той шлях від початку до кінця, що є найдовшим, оголошується ***критичним***, оскільки затримка будь-якої роботи, що лежить на цьому шляху, приводить до затримки всієї роботи в цілому. Зрозуміло, що критичних шляхів може бути декілька.

Ще однією популярною формою графічного представлення плану робіт є діаграма Ганта. ***Діаграма Ганта*** являє собою прямокутник: зліва направо рівномірно відлічуються періоди часу (тижні, місяці), зверху вниз перераховуються роботи, причому кожна робота представляється у вигляді відрізка, початок і кінець якого розміщуються у відповідному періоді.

Переваги використання діаграм Ганта при плануванні робіт полягають у такому:

- чітко видно, що виконується кожного тижня;
- зручно позначати кількість виконавців роботи.

Технічні менеджери частіше використовують мережеві графіки, оскільки їм важливіша інформація про взаємну залежність робіт. Окрім цього за мережевим графіком можна перераховувати

критичні шляхи, що доводиться робити досить часто.

Керування та організація робіт

Особливості процесу керування визначаються в першу чергу набором методів та технологій, які будуть використовуватися під час розроблення ПП. Цей вибір виконує керівник проекту, і обрана ним методологія в першу чергу визначається його особистим досвідом, а потім вимогами замовника.

У роботі А.Коуберна [5] формулюються принципи вибору методології розроблення. *Перший принцип* визначає, що велика за розміром методологія потрібна великим командам розробників. Розмір методології визначається кількістю елементів керування у проекті, до яких належать види діяльності, застосовані стандарти, створювані артефакти, показники якості й інші характеристики. На рис.11 наведені складові методології. Коуберн так пояснює зміст цих складових:

1. ***Ролі*** *(Roles)* – визначення посади, що наводиться в оголошенні про роботу (наприклад, менеджер проекту, тестувальник, розробник інтерфейсу та ін.).

2. ***Уміння*** *(Skills)* – навички, які повинні мати претенденти на посаду.

2.7. Забезпечення якості інформаційних систем

Якість інформаційної системи (Software quality) асоціація IEEE визначає як ступінь відповідності ІС, а саме, програмного забезпечення встановленій комбінації властивостей. У Міжнародному стандарті якості ISO 9000:2007 це поняття визначається як сукупність характеристик ПП, які забезпечують встановлені та очікуваі вимоги.

До **характеристик якості ІС** відносять (рис.2.17):

- функціональність – виконання заявлених функцій, відповідність стандартам, функціональна сумісність, безпека, точність; надійність – стійкість до відмов, можливість відновлення, завершеність;
- ефективність – економія часу, ефективність використання; зручність у використанні - ергономічність, інтуїтивна зрозумілість, повна документація; зручність для супроводу – стабільність, придатність для контролю та внесення змін;
- портативність – зручність установки, можливість заміни, сумісність.

Якість ІС

Рис. 2.17 Характеристики якості ІС

Забезпечення якості (Quality assurance, QA) – сукупність заходів, що охоплюють усі технологічні етапи розроблення, випуску та експлуатації інформаційних систем на різних стадіях життєвого циклу для забезпечення їх якості.

При виконанні цих заходів для кожного продукту перевіряються:

- повнота та коректність документації;
- коректність процедур встановлення та запуску;
- ергономічність використання;
- повнота тестування.

У 80-х рр. XX ст. розмір проектів із створення програмних систем зріс настільки, що вручну стало неможливо тестувати ІС, тому активно стали розроблятися засоби автоматизації процесу тестування. Через дестиліття поняття якості ІС розширилося настільки, що було виділено окремий вид діяльності при створенні ІС – забезпечення якості (Quality assurance, QA). На цей час автоматизоване тестування значно поширене, а засоби автоматизованого тестування часто вбудовані у середовище програмування.

Для виконання заходів забезпечення якості ІС розробники часто використовують спеціальні групи контролю якості, які мають назву QA. Група QA всередині компанії фактично виконує роль вимогливого користувача. У деяких компаніях заборонено неформальне спілкування між групою розробників та групою тестувальників. Від відповідальності групи QA залежить успіх продукту. Якщо ІС не сподобалося, з будь-яких причин користувачам продукт назавжди втрачає репутацію і споживача.

Незнайдені на стадії розроблення помилки коштують дорого. Традиційна стратегія розроблення ІС підпорядковується фундаментальному правилу, що визначає, що впродовж роботи над проектом вартість внесення змін у створюване ПП збільшується за експонентою (рис.2.18) [19].

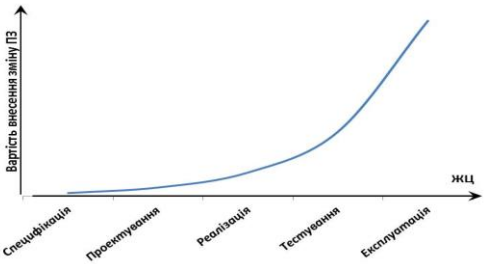
Рис. 2.18. Залежність вартості змін проектів від часу

Фактично від того, наскільки якісно виконані початкові етапи розроблення ПП залежить його вартість. З метою підвищення якості розроблення та уникнення ризиків, пов'язаних із нечіткими або постійно змінюваними вимогами, була створена методологія XP (eXtreme Programming).

Тестування *(Software Testing)* – діяльність, що виконується для оцінювання та поліпшення якості ПП. Ця діяльність базується на виявленні дефектів і проблем програмного забезпечення. Тестування ПП включає в себе діяльність із планування робіт (Test Management), проектування тестів (Test Design), виконнання тестування (Test Execution) та аналізу отриманих результатів (Test Analysis).

Потрібно відмітити, що програмування та тестування – різні за суттю види діяльності. Якщо програмування – процес синтезу, то тестування – процес аналізу. Тестування потребує від виконавця в пешу чергу уважності та педантичності. Тестувальник повинен шукати недоліки будь-де в системі. Якщо початкове тестування для налагодження коду виконує сам програміст, то наступні етапи перевірки повинні готувати і виконувати інші особи, щоб перевірка була повноцінною, а не тільки для очікуваних проблемних місць.

Для планування потрібно мати уявлення про потрібні обсяги тестування. У роботі [22] наведені важливі статистичні дані щодо тестування:



- на 1000 рядків коду програміст у середньому робить 100 помилок, 70% з яких усуваються на стадії налагодження коду;
- при системному тестуванні у відділі контролю якості на ліквідацію однієї помилки потрібно від чотирьох до шістнадцяти годин;

- для усунення помилки, що була виявлена у ході експлуатації, потрібно від 33 до 88 годин.

Ці дані показують, як важливо виявити та усунути проблеми ще на ранніх етапах розроблення. Цікавим підходом для підвищення якості програмування є парне програмування, що зменшує кількість помилок на 15 % порівняно із традиційним одиночним кодуванням [23].

Найдавнішим прийомом тестування є перевірка усіх вводів та виводів даних, передбачених та неприпустимих (complete testing). Програма повинна адекватно реагувати на помилкові ситуації, без втрати стійкості в роботі. Також при тестуванні потрібно перевірити виконання усіх передбачених функцій системи. Перелік таких тестів складається на ранніх етапах проектування паралельно із описом функцій програмного продукту.

Види тестування можна класифікувати за різними показниками [24]:

1. За рівнем знання системи:

- чорний ящик (Black-box testing) – без перегляду програмного коду;
- білий ящик (White-box testing) – тестування із вивченням коду програми;
- сірий ящик (Gray-box testing) – тестування із частковим вивченням коду програми.

2. За об'єктом тестування:

- функціональне (Functional testing) – перевірка виконання заданих функцій;
- тестування продуктивності (Performance testing) – перевірка стабільності роботи програми або її частини при заданому навантаженні (Load testing), при перевантаженні (Stress testing) та тривалому середньому навантаженні (Stability testing);
- юзабіліті-тестування (Usability testing) – перевірка ергономічності ПП;
- перевірка інтерфейсу користувача (UI testing);
- перевірка безпеки (Security testing) – тестування роботи системи з точки зору безпеки інформації;
- тестування сумісності (Compatibility testing) – нефункціональне тестування, метою якого є перевірка коректної роботи ПП у певному оточенні.

3. За часом виконання тестування:

- альфа-тестування (Alpha testing) – перевірка роботи ПП перед передачею в експлуатацію силами компанії-розробника;

- ✓ півгодинне тестування (Smoke testing) – коротка перевірка функцій системи, як правило, один тест для кожної функції;
- ✓ тестування нової функціональності (New feature testing);
- ✓ регресійне тестуванн (Regression testing) – перевірка роботи вже протестованих модулів;
- ✓ тестування під час передачі ПП (Acceptance testing);

- бета-тестування (Beta testing) – тестування ПП перед випуском сторонніми силами.

4. За ступенем ізольованості компонентів:

- компонентне тестування (Component / Unit testing) – перевірка коректності окремого модуля або компонента системи;
- інтеграційне тестування (Integration testing) – перевірка роботи модулів, об'єднаних у групу;
- системне тестування (System / end-to-end testing) – перевірка роботи зібраного ПП з метою встановлення відповідності очікуваним функціям.

5. За ступенем автоматизації:

- ручне тестування (Manual testing);
- автоматизоване тестуванняя (Automated testing);
- напівавтоаматизоване тестування (Semiautomated testing).

6. За ступенем підготовленості до тестування:

- формальне тестування (Formal testing) – тестування відповідно до плану, встановлених процедур;
- інтуїтивне тестування (Ad hoc testing) – тестування без попереднього плану дозволяє на раніх стадіях виявляти помилки.

7. За ознакою позитивності сценаріїв:

- перевірка позитивних сценаріїв (Positive testing);
- перевірка негативних сценаріїв (Negative testing).

Тестування, як і будь-який процес, повино мати методи оцінки якості тестування. Одним із способів є спосіб, якому в програму спеціально вставляється певна кількість помилок. Після проведення тестування оцінюється, скільки таких помилок було знайдено. Частка знайдених помилок показує якість тестування і допомагає оцінити, що обсяг робіт потрібно виконати для повного усунення проблем. Плануючи тестування, потрібно визначати очікувані результати тестування, які покажуть готовність створюваного продукту. Бажано затвердити їх із замовником для чіткого визначення показників завершення проекту.

Якщо проектувальники тестів повинні мати інтегральне уявлення про систему, то виконавці тестів не потребують високої

кваліфікації. Часто цю роль виконують новачки у компаніях, які не мають досвіду роботи.

Для підвищення рівня контролю якості кожного проекту повинна формуватися **база даних помилок**, в яку вноситься така інформація:

- хто знайшов помилку,коли;
- опис помилки;
- модуль, у якому була знайдена помилка;
- версія продукту; статус помилки:
  - ✓ open: знайдена;
  - ✓ fixed: виправлена;
  - ✓ can't reproduce: неможливо повторити; o by design: помилка проектування;
  - ✓ wont fix: не помилка;
  - ✓ postponed: зараз виправити важко, буде виправлена у наступній версії;
  - ✓ regression: виправлена помилка з'явилася знову;
- важливість (severity) помилки:
  - ✓ crash: повна втрата даних;
  - ✓ major problem: програма частково не працює, часткова втрата даних;
  - ✓ minor problem: програма працює не так, як очікувалось, але дані не втрачаються;
  - ✓ trivial: зараз не потрібно виправляти;
- пріоритет помилки:
  - ✓ highest: не можна поставити продукт із такою помилкою, не можна перейти до наступної версії;
  - ✓ high: поставити не можна, але можна перейти до наступної версії;
  - ✓ medium: помилка буде виправлена;
  - ✓ low: косметичні поліпшення – помилка залишиється до наступної версії.

Така база даних дозволяє проаналізувати якість роботи окремих програмістів, їх груп, оцінювати якість проекту та можливості використовуваних інструментальних засобів. Також керівники можуть відслідковувати ризики розроблення за помилками найвищих рівнів пріоритету та важливості.

Дані з бази даних помилок дозволяють приймати рішення про можливість випуску програмного продукту (помилки з важливістю

"crash" або з пріоритетом "highest/high" не дозволяють поставляти ПП). Якщо ПП обов'язково потрібно поставити замовнику, а часу на виправлення помилок немає, за домовленістю можна відкинути функції, які виконуються із помилками.

Питання для самоконтролю

1. Типові помилки проектування інформаційних систем.
2. Дайте визначення: що таке «життєвий цикл» інформаційної системи.
3. Перелічіть головні процеси життєвого циклу інформаційної системи.
4. Що таке допоміжні процеси життєвого циклу? Перелічіть їх та надайте стислу характеристику.
5. Склад робіт початкової стадії життєвого циклу
6. Які типи вимог до ПП можна виділити? Як виконується опис цих вимог?
7. Як пов'язані між собою строки проектних робіт, їх обсяг та вартість? Яким чином, керуючи цими показниками, можна забезпечити якість проекту?
8. Які елементи повинен містити проектний план?
9. Які форми планів проектних робіт вам відомі? У чому принципова різниця між ними?
10. Охарактеризуйте принципи керування проектом.
11. Які складові методології розроблення потрібно враховувати при її виборі?
12. З якими ризиками стикаються розробники ПП?
13. Перелічіть та коротко опишіть характеристики якості ПП.
14. Що таке Quality assurance, чим воно відрізняється від тестування?
15. Опишіть види тестування. Яку інформацію доцільно зберігати в базі даних помилок?

РОЗДІЛ 3. МЕТОДОЛОГІЇ І ТЕХНОЛОГІЇ СТВОРЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

3.1. Теоретичні основи методологій створення інформаційних систем

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

- проектування об'єктів даних, які будуть реалізовані в базі даних;
- проектування програм, екранних форм, звітів, які будуть забезпечувати виконання запитів до даних;
- проектування певного середовища або технології, тобто топології мережі, конфігурації апаратних засобів, використовуваної архітектури (файл- серверної або клієнт-серверної), паралельної обробки, розподіленої обробки даних тощо.

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

Основу проекту будь ІС складають такі компоненти:

- методологія проектування;
- технологія проектування;
- стандарти й методики проектування;
- інструментальні засоби проектування (або CASE-засоби).

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

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

процесу її розроблення.

Найважливіші завдання, які має виконувати методологія створення ІС:

- відповідність ІС, що створюється, меті та завданням підприємства, а також вимогам щодо автоматизації бізнес-процесів;
- гарантоване створення системи із заданими параметрами у межах виділеного часу та бюджету;
- підтримка дисципліни супроводження, модифікації та нарощування системи для забезпечення її адаптації до мінливих умов роботи підприємства;
- забезпечення відповідності ІС, що створюється, вимогам відкритості та здатності до масштабування;
- забезпечення спадкоємності розробки, тобто використання у системі, що розробляється, існуючої інформаційної інфраструктури організації.
- забезпечення можливості використання в новій системі елементів інформаційних існуючих технологій, а саме: програмного забезпечення, баз даних, апаратних засобів, телекомунікацій тощо.

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

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

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

Технологію проектування прийнято подавати як сукупність із трьох складових:

- послідовності виконання технологічних операцій, що передбачені процесом проектування;
- критеріїв щодо оцінювання результатів виконання технологічних операцій;
- нотацій (тобто графічних і текстових засобів) для описання системи, що проектується.

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

- дані, що отримані від попередньої операції;

- методичні матеріалами, інструкції, нормативи, стандарти;
- програмні та технічні засоби;
- виконавці (розробники).

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

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

- підтримувати повний життєвий цикл ІС;
- гарантувати досягнення цілей розробки із заданою якістю й у межах встановленого часу;
- забезпечувати можливість розподілу (декомпозиції) великих проектів на кілька складових частин (підсистем), що розробляються окремими групами виконавців із подальшою інтеграцією цих частин;
- надавати можливість проектування підсистем малими групами, що гарантує керованість колективом та підвищує продуктивність внаслідок мінімізації кількості зв'язків із зовнішнім середовищем;
- забезпечувати мінімальний час для отримання працездатної системи;
- давати можливість керувати конфігурацією проекту, вести облік та синхронізацію версій проекту та його складових;
- забезпечувати незалежність проектних рішень від специфіки засобів реалізації, систем управління базами даних, операційної системи, мов програмування.

Проект ІС – це проектно-конструкторська та технологічна документація, в якій подається опис рішень створення та експлуатації ІС у конкретному програмно-технічному середовищі.

В основі проектування будь-якого продукту лежить парадигма подолання складності завдання шляхом його декомпозиції на окремі компоненти.

Технологія проектування ІС – це поєднання методології та інструментальних засобів проектування ІС.

Методологія проектування передбачає наявність концепції, принципів проектування, засобів проектування. Метод проектування ПЗ – це організована сукупність процесів створення моделей, що описують різні аспекти ІС з використанням нотацій. Метод – це сукупність:

- концепцій і теоретичних основ (наприклад, структурний або об'єктно орієнтований підхід);

- нотацій, що використовуються для побудови моделей статичної структури і динаміки поведінки ІС (діаграми потоків даних і діаграми "сутність - зв'язок" для структурного підходу, діаграми варіантів використання, діаграми класів в об'єктно орієнтованому підході);
- процедур, що визначають практичне застосування методу (послідовність і правила побудови моделей, критерії для оцінювання результатів).

Технологія проектування ПЗ - це сукупність технологічних операцій проектування (рис. 3.1) у певній послідовності і взаємозв'язку. Апарат технологічних мереж проектування - це зручний інструмент формалізації технології проектування ІС. Основа його формалізації - визначення технологічної операції проектування у вигляді множини документів (описувач множини фактів), параметрів (описувач одного факту), програм (опис алгоритмів рішення задачі), універсальних множин (повна множина фактів одного типу), на яких задані перетворювачі, ресурси, засоби проектування на конкретному вході/виході.

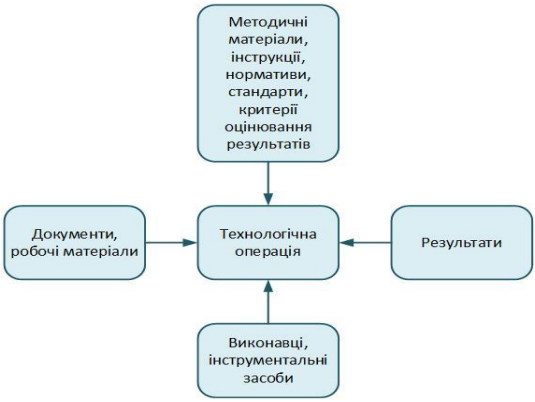


Рисунок 3.1. Контекст технологічної операції проектування

3.2 Методи проектування інформаційних систем

Структурний аналіз

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

Сутність структурного підходу до розробки ІС полягає в її декомпозиції (розбитті) на функції, що підлягають подальшої автоматизації, а саме: система розбивається на функціональні

підсистеми, ті зі свого боку розподіляються на підфункції, завдання тощо. Процес розбиття триває до рівня деталізації процедур. До того ж автоматизована система зберігає цілісне уявлення, коли всі складові компоненти взаємопов'язані та мають загальне підґрунтя.

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

Спершу розробляється функціональна модель, за допомогою якої визначаються, аналізуються і фіксуються вимоги до складу та структури функцій системи, тобто визначається, для яких цілей розробляється система, які функції вона виконуватиме. На цій же моделі вказуються вихідна інформація, проміжні та підсумкові результати роботи системи. На основі інформаційних потоків визначається склад і структура необхідних даних, що зберігаються в системі (будується інформаційна модель). Далі, з урахуванням

розроблених моделей, створюються процедури реалізації функції, тобто алгоритми обробки даних і поведінки елементів системи. На заключній стадії встановлюється розподіл функцій по підсистемах (компонентам), необхідне технічне забезпечення і будується модель їх розподілу по вузлах системи (рис. 3.2.).

Рис. 3.2. Схема застосування структурного підходу

Приведена на рисунку схема не означає, що побудова моделей має строго відповідати зазначеному порядку – одна за одною. Як правило, розробка кожної наступної моделі починається ще до повного завершення розробки попередньої, а іноді – паралельно.

На стадії проектування моделі розширюються , уточнюються і доповнюються діаграмами, що відбивають технологію використання системи, її архітектуру, екранні форми і т. п.

Усі найбільш поширені методології структурного підходу базуються на ряді загальних принципів. В якості двох базових принципів використовуються:

- принцип "розділяй і володарюй" – принцип вирішення

складних проблем шляхом їх розбиття на безліч менших незалежних завдань, легких для розуміння і вирішення;

- принцип ієрархічного впорядкування – принцип організації складових частин проблеми в ієрархічні деревоподібні структури з додаванням нових деталей на кожному рівні.

Виділення двох базових принципів не означає, що інші принципи є другорядними, оскільки ігнорування кожного з них може призвести до непередбачуваних наслідків (у тому числі і до провалу всього проекту). Основними з цих принципів є такі:

- принцип абстрагування – полягає у виділенні істотних аспектів системи і відволікання від несуттєвих;
- принцип формалізації – полягає в необхідності суворого методичного підходу до вирішення проблеми;
- принцип несуперечності – полягає в обґрунтованості та узгодженості елементів;
- принцип структурування даних – полягає в тому, що дані повинні бути структуровані і ієрархічно організовані.

Більшість методологій структурного аналізу і проектування ґрунтується на поданні моделей розроблюваних систем у вигляді діаграм. Найбільш популярні з них представлені в табл. 3.1.

Таблиця 3.1 – Методології структурного аналізу і проектування

Методологія	Тип моделі, що розробляється
SADT (Structured Analysis and Design Technique, методологія структурного аналізу і проектування)	Функціональна
DFD (Data Flow Diagrams,діаграми потоків даних)	Функціональна або компонентна
ERD (Entity-Relationship Diagrams,діаграми "сутність-зв'язок")	Інформаційна
BPMN (Business Process Model and Notation,модель і нотація бізнес-процесів)	Функціональна або поведінкова

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

що мають три класи структурних моделей:

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

Процес моделювання за методологією SADT складається з таких етапів:

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

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

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

Проектування інформаційних систем за стандартом IDEF0 полягає у послідовній декомпозиції основних функцій організації на окремі бізнес- процеси, роботи або дії з подальшим об'єднанням їх у єдину ієрархічну модель організації. Декомпозицію можна робити багаторазово, аж до чіткого й вичерпного опису всіх процесів.

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

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

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

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

методології.

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

Об'єктно-орієнтований аналіз

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

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

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

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

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

Перша відмінність цих підходів один від одного полягає в принципах декомпозиції та структурної організації елементів (компонентів, модулів) системи. Згідно з цими принципами система являє собою структуру, що складається з чітко виражених модулів, пов'язаних між собою певними відносинами. Другий вид

декомпозиції – об’єктно-орієнтована. В рамках цього підходу система розбивається на набір об’єктів, відповідних об’єктам реального світу, взаємодіючих між собою шляхом посилки повідомлень.

Другою відмінністю є об’єднання в об’єкті як атрибутивних даних (характеристики, властивості), так і поведінки (функції, методи). У функціонально-орієнтованих системах функції і дані зберігаються (існують) окремо.

Третя відмінність двох підходів полягає в структурній організації всередині модулів системи. У структурному підході модуль складається з функцій, ієрархічно пов’язаних між собою відношенням композиції, тобто функція складається з підфункцій, підфункція з підпідфункцій і т.д. В об’єктно-орієнтованому підході ієрархія вибудовується з використанням двох типів відношень: композиції і спадкування (англ. IS A – це є). При цьому в об’єктно-орієнтованому підході «об’єкт-частина» може включатися відразу в кілька «об’єктів-ціле». Таким чином, модуль в структурному підході представляється у вигляді дерева, а в об’єктно-орієнтованому підході – у вигляді орієнтованого графа, тобто за допомогою більш загальної структури.

Найбільш популярними методологіями, що підтримують об’єктно-орієнтований підхід, зараз є:

- уніфікований процес (Unified Process, UP);
- екстремальне програмування (eXtreme Programming, XP);
- гнучке моделювання (Agile Modeling, AM).

Базовим засобом фіксації (документування) результатів проектування систем за допомогою цих методологій є уніфікована мова моделювання (Unified Modeling Language, UML).

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

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

2) UML забезпечує підтримку всіх етапів життєвого циклу ІС і пропонує для цього відповідний набір діаграм: структурні

(Structure Diagrams); діаграми поведінки (Behavior Diagrams); діаграми взаємодії (Interaction Diagrams).

UML фокусується на трьох архітектурних інтерпретаціях ІС:

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

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

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

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

3.3 Сучасні методології проектування інформаційних систем

CASE–засоби та нотації моделювання програмних систем

На становлення технологій виробничого програмування найбільш помітний вплив мали методологія структурного програмування та об'єктно-орієнтоване програмування. Перша дозволила усвідомити обмеженість здібностей людини, необхідних для створення великих програм. Об'єктно-орієнтоване програмування дало поштовх до розроблення методів декомпозиції, пристосованих для подолання складності проектів, та привело до модернізації основних принципів проектування програм. Незважаючи на ці та інші впливи, стадія комплексної автоматизації технологій програмування стала можливою лише при відповідному рівні розвитку техніки. Важливою обставиною, що дозволила перейти до комплексної автоматизації, стало усвідомлення того, що не можна говорити про промислове програмування без підтримки технологічних функцій на усіх етапах життя програм.

Приблизно на початку 90-х рр. XX ст. з'явився термін - CASE-технологія (Computer Aid Software Engineering – комп'ютерна підтримка розроблення програм), яким стали позначати використання систем, що містять комплексні автоматизовані засоби підтримки розроблення і супроводу ПЗ. Найбільш вдалим виявилося використання CASE-систем у тих спеціальних областях, в яких вже

були успіхи і досвід технологічної практичної роботи, а також у тих випадках, коли ця область вже була забезпечена надійною теоретичною базою. На рис.19 наведено етапи розвитку CASE-засобів за областями застосування.

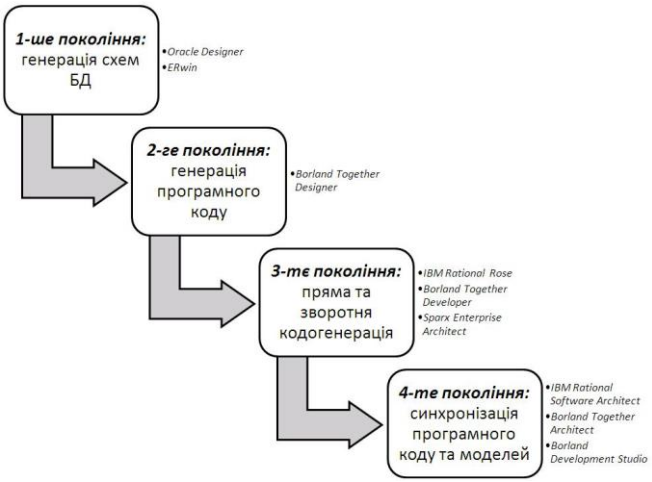


Рисунок 19 – Розвиток CASE-засобів

Першими з'явилися CASE-системи розроблення баз даних у розвинених реляційних СУБД (наприклад, Oracle Designer в системі Oracle). Наступними стали засоби генерації програмного коду та налагодження програм. Потреба підтримки не лише процесу програмування, а й етапу аналіз та проектування програмних продуктів привело до створення CASE-систем, які зв’язували моделі ПЗ із програмним кодом – спочатку лише для прямої та зворотної кодогенерації, а потім із підтримкою синхронного зв'язку коду із моделями аналізу та проектування.

Сьогодні універсальні CASE-системи будуються у рамках застосування розвинених, але все ж спеціальних методологій. Безперечний прогрес у даній сфері досягнутий для проектування, орієнтованого на моделювання на етапах аналізу і конструювання (CASE-системи 4-го покоління).

Складність процесу розроблення інформаційних систем викликала появу візуальних засобів моделювання (рис.20). У рамках об'єктно-орієнтованого підходу Object management group (OMG) розроблена спеціальна уніфікована мова моделювання UML (Unified Modeling Language) [29, 30], що розглядається як основа проектування в методології ітеративного нарощування

можливостей об'єктно-орієнтованих програмних систем. Широкого вжитку для моделювання процесів різних галузей екномічної діяльності набула родина візуальних мов IDEF (Integration Definition) [31], що є стандартом моделювання Міністерства оборони США. Паралельно із нотаціями UML та IDEF розвиваються методологія та нотація для професійного моделювання бізнес-процесів ARIS (ARchitecture of Integrated Information Systems).

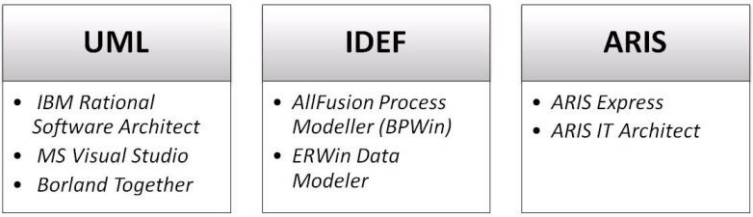
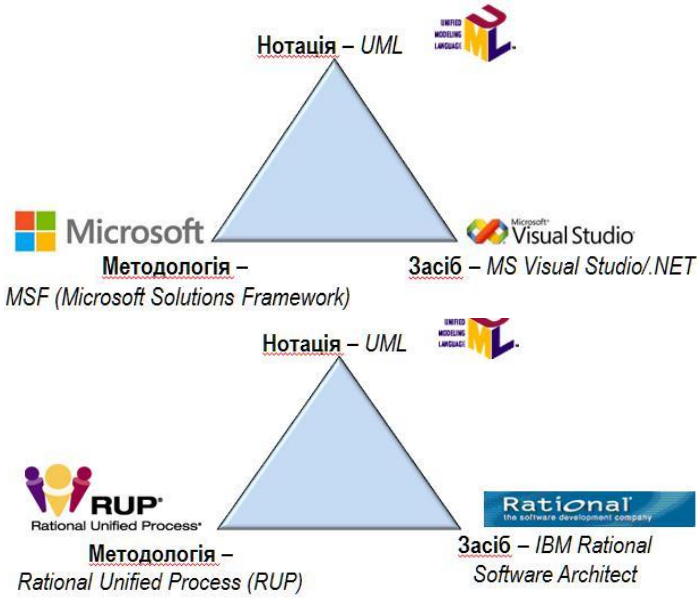


Рисунок 20 – Нотації візуального моделювання систем

Візуальне моделювання мовою UML

У даному курсі будемо розглядати нотацію UML (uml.org), що є широкозастосовною при проектуванні ПЗ, і підтримується більш ніж 50 CASE-інструментами. На базі цієї мови побудований ряд CASE-систем загального призначення з розвиненими засобами (рис.21).

Інструменти використовують споріднені або кардинально відмінні методології розроблення, але для запису моделей вони використовують одну мову моделювання.



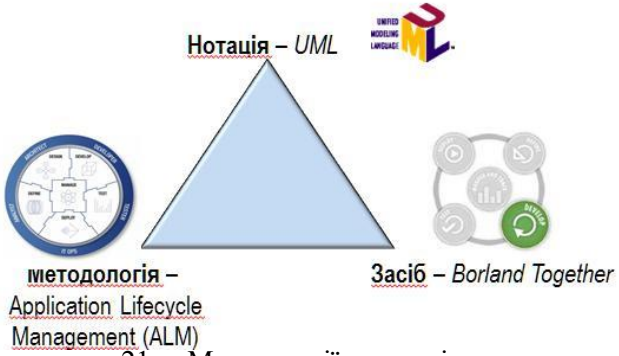


Рисунок 21 Методології та інструменти візуального моделювання

Інструменти використовують спорібнені або кардинально відмінні методології розроблення, але для запису моделей вони використовують одну мову моделювання - UML.

Необхідність використання UML залежить від складності проекту (рис.22). Обов'язково потрібно використовувати візуальне моделювання у технічно складних масштабних проектах, які розробляються для великої кількості користувачів. Використання візуальних моделей для невеликих проектів, які вирішують нескладні завдання, недоцільне, оскільки це лише збільшить вартість проекту та час розроблення.

Рисунок 22 – Використання UML залежно від складності проекту

Базові терміни та нотація

UML створена для візуального моделювання систем на основі об'єктно-орієнтованого підходу. Ця мова створювалася для оптимізації процесу розроблення програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту. Вона дозволяє будувати моделі для усіх фаз ЖЦ ПЗ.

Мова була створена як результат «війни методів» моделювання у 1995 р. та увібрала у себе риси нотацій Грейді Буча (Grady Booch), Джеймса Рамбо (James Rumbaugh), Айвара Якобсона (Ivar Jacobson) і багатьох інших (рис.23). З 1997 р. розробленням мови займається консорціум OMG (Object Management Group), створений у 1989 році для розроблення індустріальних стандартів з їх наступним використанням у процесі створення масштабованих неоднорідних розподілених об'єктних середовищ.

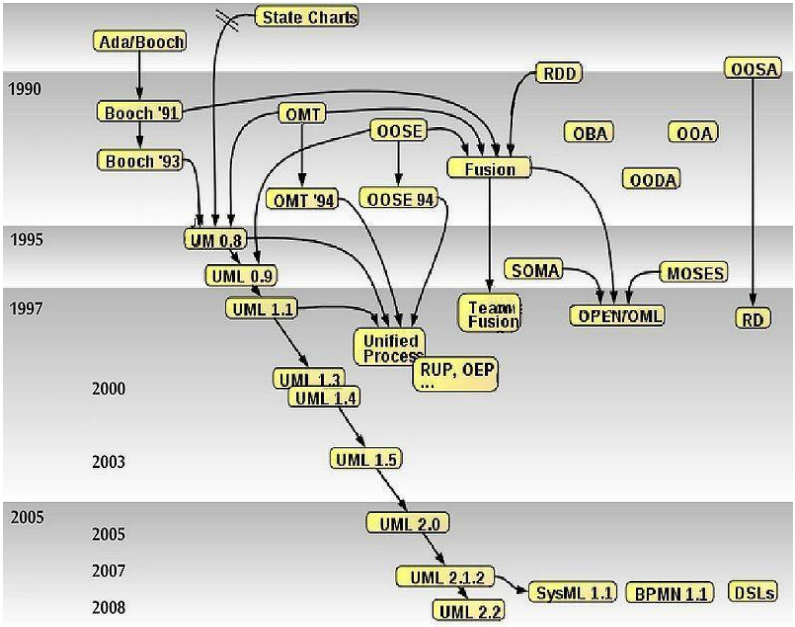


Рисунок 23 – Історія розвитку об'єктно-орієнтованих методів

Універсальність UML підтверджує її широке застосування, особливо у галузі створення програмних систем та затвердження у 2005 р. версії UML 1.4.2 як стандарту ISO/IEC 19501:2005. Найновіша стабільна версія мови UML 2.4.1 вийшла у серпні 2011 р.

Моделювання за допомогою мови UML грунтується на таких

принципах:

абстрагування - у модель необхідно включати тільки ті елементи проектованої системи, які мають безпосереднє відношення до виконання нею своїх функцій;

багатомодельність - ніщо єдина модель не може з достатнім ступенем точності описати різні аспекти системи. Можна описувати систему кількома взаємозалежним уявленнями, кожне з яких відображає певний бік її структури або поведінки;

ієрархічність - при описі системи використовуються різні рівні

абстрагування і деталізації у рамках фіксованих уявлень. При цьому перше уявлення системи описує її в найбільш загальних рисах і є представленням концептуального рівня, а наступні рівні розкривають різні сторони системи із зростаючим ступенем деталізації аж до фізичного рівня. Модель фізичного рівня в мові UML відображає компонентний склад проектованої системи з точки зору її реалізації на апаратурній і програмній платформах конкретних виробників.

Елементи (класифікатори) мови UML можна поділити на три групи (рис.24).

Рисунок 24 – Класифікатори мови UML

Сутності (entity) – абстракції, що є основними об'єкто-орієнтованими елементами мови UML, за допомогою яких будуються моделі. В UML визначено чотири типи сутностей: структурні, поведінкові, сутності групування та сутності-примітки.

Структурні сутності - це іменники в моделях UML. Як правило, вони є статичними частинами моделей, які відповідають концептуальним або фізичним елементам системи. Існує *сім*

різновидів структурних сутностей: Клас (Class), Інтерфейс (Interface), Кооперація (Collaboration), Варіант використання/Прецедент (Use case), Активний клас (Active class), Компонент (Component), Вузол (Node).

Поведінкові сутності є динамічними складовими моделі UML, які описують поведінку моделі в часі і просторі. Це дієслова мови. Існують два основних типи поведінкових сутностей: Взаємодія (Interaction) та Автомат (State).

Сутності групування є організуючими частинами моделі UML. Це блоки, на які можна розкласти модель. Первинна сутність групування – пакет (Package).

Сутності-примітки - пояснювальні частини моделі UML. Це коментарі для додаткового опису, роз'яснення або зауваження до будь-якого елемента моделі. Є тільки один базовий тип анотаційних елементів – примітка (Note).

Для зв'язування сутностей у моделях UML визначено чотири типи *відносин* (рис.25):

- ***Залежність (dependency)*** - це семантичне відношення між двома сутностями, при якому зміна однієї з них, незалежної, може вплинути на семантику іншої, залежної.

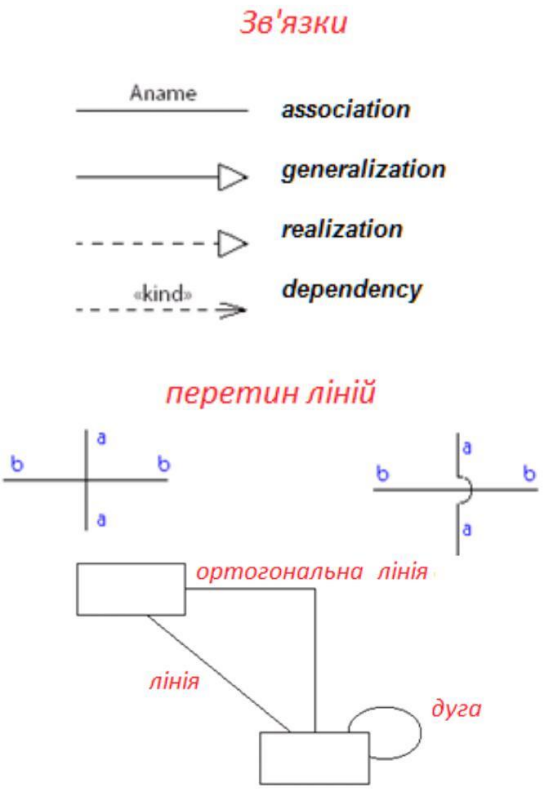


Рисунок 25 – Нотація відносин мови UML

Асоціація (association) - структурне відношення, що описує сукупність змістовних або логічних зв'язків між об'єктами.

Узагальнення (generalization) - це відношення, при якому об'єкт-нащадок (child) може бути підставлений замість об'єкта-батька (parent). При цьому відповідно до принципів об'єктно-орієнтованого програмування нащадок успадковує структуру і поведінку свого батька.

Реалізація (realization) є семантичним відношенням між класифікаторами, при якому один класифікатор визначає зобов'язання, а інший гарантує його виконання. Відношення реалізації спостерігаються у двох випадках:

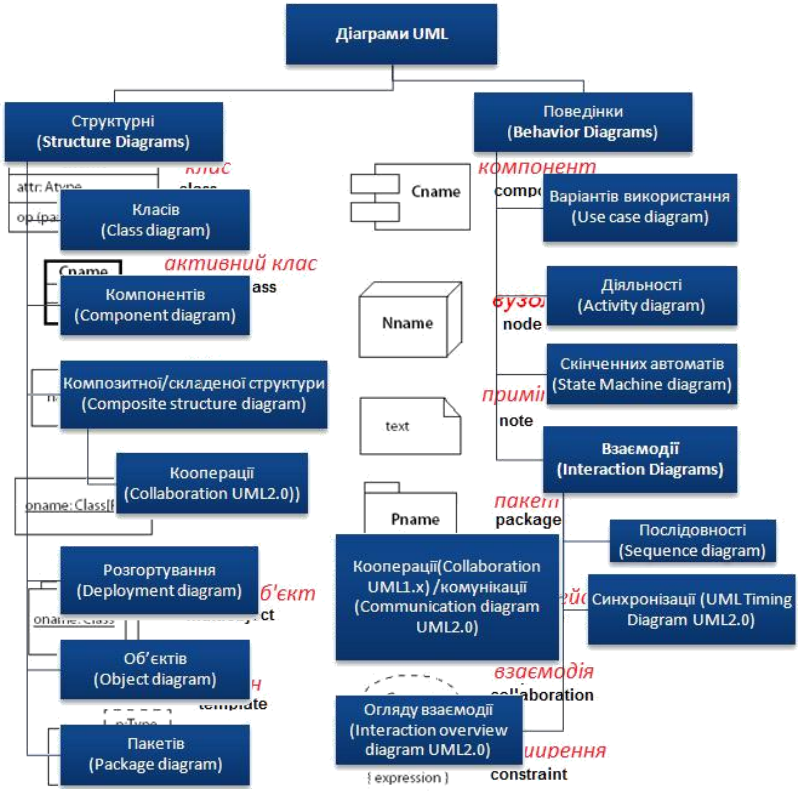
- між інтерфейсами, реалізують класами або компонентами;
- між варіантами використання, реалізують кооперації.

Усі різновиди сутностей та відносини UML в діаграмах мають свій спосіб графічного зображення (рис.25, 26).

Рисунок 26 – Нотація сутностей мови UML

В UML використовуються 13 діаграм [29], які поділені на дві основні групи (рис.27):

структурні діаграми (Structure Diagrams) – моделі, призначені для опису статичної структури сутностей або елементів системи, включаючи їх класи, інтерфейси, атрибути та відношення;



діаграми поведінки (Behavior Diagrams) – моделі, призначені для опису процесу функціонування елементів системи, включаючи їх методи та взаємодію між ними, а також процес зміни станів окремих елементів та системи в цілому.

Кожна діаграма уточнює уя влення про систему. При цьому діаграма варіантів використання є концептуальною моделлю системи, початковою для побудови усіх інших діаграм. Діаграма класів є логічною моделлю, що відбиває статичні аспекти структурної побудови системи, а діаграма діяльності, що також є різновидами логічної моделі, відображає динамічні аспекти її функціонування.

Рисунок 27 – Діаграми мови UML

Методології розроблення ПЗ

Найбільш помітними є такі методології розроблення ПЗ:

Rational Unified Process (RUP), що пропонує ітеративну модель розроблення, що включає чотири фази: початок, дослідження, побудову та впровадження [33]. Проходження через чотири основні фази називається циклом розроблення, кожен цикл завершується генерацією працездатної версії системи. У ході супроводу продукт продовжує розвиватися і знову проходить ті самі фази. Розроблення ПЗ на базі RUP базується на створенні та супроводі моделей на базі UML.

Microsoft Solution Framework (MSF) – методологія, що розроблялася для підвищення керованості процесів розроблення окремого підприємства (Microsoft), а потім стала застосовуватися розробниками, які використовують продукти Microsoft. Методологія, подібна до RUP, також включає чотири фази: аналіз, проектування, розроблення, стабілізацію, є ітераційною, припускає використання об'єктно-орієнтованого моделювання. MSF порівняно з RUP здебільшого орієнтована на розроблення бізнес-додатків.

eXtreme Programming (XP) – методологія, що приділяє основну увагу ефективній комунікації між замовником і виконавцем упроовж усього проекту з розроблення ІС для відслідковування зміни вимог [19]. В основу підходу покладена командна робота та постійне тестування прототипів.

Гнучке розроблення ПЗ (Agile) – клас методологій розроблення програмного забезпечення на основі ітеративної моделі ЖЦ, в якій вимоги та рішення еволюціонують через співпрацю між самоорганізовуваними командами [35].

Методологія Rational Unified Process (RUP)

Уніфікований процес розроблення ПЗ *Rational Unified Process* (RUP) [33] виник як відповідь на потребу розробників у процесі, що об'єднує безліч аспектів розроблення програм і відповідає таким вимогам:

- забезпечує керівництво діяльністю команди; керує завданнями окремого розробника і команди в цілому; показує, які артефакти необхідно розробити;
- надає критерії відстеження та вимірювання продуктів і функціонування проекту.

Уніфікований процес розроблення ПЗ передбачає розподіл ЖЦ програмного забезпечення на фази (рис.28).

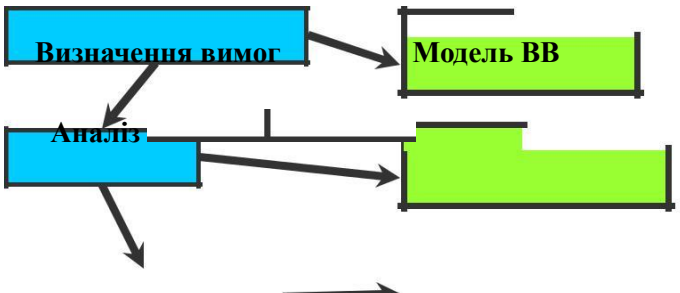
Рисунок 28 Робочі процеси ЖЦ в уніфікованому процесі розроблення ПЗ

Специфіка уніфікованого процесу полягає у трьох словосполученнях – керований варіантами використання, архітектурно-орієнтований, ітеративний та інкрементний.

Уніфікований процес керується варіантами використання

Програмна система створюється для обслуговування її користувачів. В уніфікованому процесі поняття ***користувач*** стосується когось чи чогось (наприклад, іншої системи, зовнішньої щодо даної системи), що взаємодіє із системою, яка розробляється. У відповідь на вплив користувачів система здійснює послідовність дій, які забезпечують користувачеві відчутний і значущий для нього результат. Користувача в RUP називають ***актором*** *(actor)* – категорія користувачів, що використовує певну частину функцій системи (відповідного варіанта використання).

Взаємодія актора із системою називається варіантом використання, або прецендентом. ***Варіант використання (ВВ),*** або ***прецендент,*** *(use case)* – це частина функціональності системи, необхідна для отримання користувачем значущого для нього, відчутного і вимірюваного результату. ВВ забезпечують функціональні вимоги. Сума всіх ВВ становить ***модель ВВ*** *(use-case model),* що описує повну функціональність системи. Ця модель заміняє традиційно опис функцій системи. На основі прецедентів в уніфікованому процесі будуються інші моделі для кожного етапу ЖЦ (рис.29).



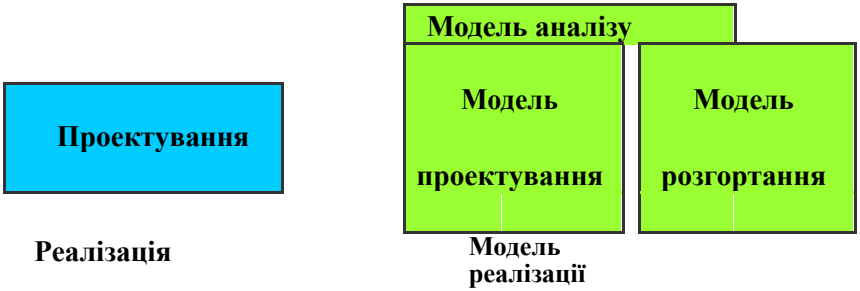

Модель аналізу
Проектування
Модель проектування
Модель розгортання
Реалізація
Модель реалізації
Тестуванн
Модель тестування

Рисунок 29 Моделі ПЗ на кожному етапі ЖЦ

За набором реалізацій ВВ будується концептуальна модель системи – ***модель аналізу*** *(analysis model),* що є детальним описом усіх вимог та потрібних для реалізації робіт. Ґрунтуючись на моделі ВВ, розробники створюють серію моделей проектування і реалізації, які здійснюють прецеденти.

Модель проектування *(design model)* є об'єктною моделлю, що містить опис фізичної реалізації ВВ та сконцентрована на тому, які функціональні та нефункціональні вимоги разом із обмеженнями середовища розроблення реалізують систему.

Модель розгортання *(deployment model)* – визначає фізичний розподіл системи по вузлах, перевіряє, чи можуть ВВ бути реалізовані у вигляді компонентів, які виконуються в цих вузлах.

Модель реалізації *(implementation model)* – дає опис, як елементи моделі проектування реалізуються у вигляді компонентів, таких, як файли із кодом програм та виконувані файли.

Модель тестування *(test model)* – дає опис, як виконувані компоненти моделі реалізації тестуються на цілісність та проходять системні тести.

Тестери тестують реалізацію для того, щоб гарантувати, що компоненти моделі реалізації правильно виконують варіанти використання.

Таким чином, прецеденти не лише ініціюють процес розроблення, але й слугують для зв'язку окремих його частин. Процес розроблення проходить серії робочих процесів, кожен з яких ініціюється ВВ: прецеденти визначаються, вони проектуються, і врешті-решт варіанти використання є вхідними даними, за якими тестери створюють тестові приклади. Оскільки прецеденти дійсно керують процесом, вони не виділяються ізольовано, а розробляються в парі з архітектурою системи. Відповідно і архітектура системи, і ВВ розвиваються впродовж життєвого циклу ПЗ.

Уніфікований процес, орієнтований на архітектуру

Архітектура програми включає в себе найбільш важливі статичні і динамічні аспекти системи та будується на основі вимог до результату. Як було відмічено вище, вимоги відбиваються у варіантах використання. Однак прецеденти також залежать від безлічі інших чинників, таких, як вибір платформи для роботи програми (тобто комп'ютерної архітектури, операційної системи, СКБД, мережевих протоколів), доступність готових блоків багаторазового використання (наприклад, каркасу GUI), спосіб розгортання, успадковані системи і нефункціональні вимоги (наприклад, продуктивність і надійність). ***Архітектура ПЗ*** – це представлення всього проекту із виділенням важливих характеристик без акценту на деталях.

Кожен продукт має функції і форму. Одне без іншого не існує. Функції відповідають ВВ, а форма - архітектурі. В реальних умовах

архітектура і прецеденти розробляються паралельно. Архітектура повинна бути спроектована так, щоб дозволити системі розвиватися не тільки в момент початкового розроблення, але і в майбутніх поколіннях. Щоб знайти таку форму, архітектор повинен працювати, повністю розуміючи ключові функції, тобто ключові варіанти використання системи. Ці ключові варіанти використання становлять 5-10% усіх ВВ, але вони дуже важливі, оскільки містять функції ядра системи.

Архітектор виконує такі кроки: визначає платформу системи – створює грубий ескіз архітектури, починаючи з тієї частини, що не пов'язана з ВВ. Хоча ця частина архітектури не залежить від прецедентів, архітектор повинен у загальних рисах розуміти варіанти використання до створення ескізу архітектури; далі архітектор працює із підмножиною виділених ВВ, кожен з яких відповідає одній із ключових функцій розроблюваної системи. Кожен з обраних прецедентів детально описується і реалізується в поняттях підсистем, класів і компонентів після того як ВВ описані та повністю розроблені, більша частина архітектури досліджена. Створена архітектура, у свою чергу, буде базою для повного розроблення інших ВВ. Цей процес продовжується до того часу, поки архітектура не буде визнана стабільною.

Уніфікований процес є ітеративним та інкрементним

Процеси розроблення комерційного ПЗ є серйозними, часто тривалими проектами. Для підвищення керованості та якості процесу доцільно розділити таку роботу на невеликі частини. Кожен міні-проект є ітерацією, результатом якої буде приріст. *Ітерації* належать до кроків робочих процесів, а *приріст (інкремент)* - до виконання проекту. Для максимальної ефективності ітерації повинні вибиратися і виконуватися за планом, що дозволяє вважати кожну ітерацію міні-проектом. Ітеративність передбачає повторення робіт з розроблення у різних фазах (рис.30).

Розробники вибирають завдання, які повинні бути вирішені у ході ітерації, враховуючи два фактори:

- у ході ітерації необхідно працювати з групою ВВ, що підвищує можливість застосування продукту під час подальшого розроблення;
- у ході ітерації необхідно займатися найбільш серйозними ризиками.

Послідовні ітерації використовують артефакти розробки в тому вигляді, в якому вони залишилися при закінченні попередньої ітерації. У ході кожного міні-проекту для обраних ВВ проводиться послідовне розроблення - аналіз, проектування, реалізація та тестування. Зрозуміло, прирощення необов'язково адитивне, особливо на ранніх фазах життєвого циклу. На кожній ітерації розробники визначають і описують відповідні ВВ, створюють проект, що використовує обрану архітектуру як напрямну, реалізують проект у вигляді компонентів та перевіряють відповідність

компонентів варіантами використання. Якщо ітерація досягла своєї мети, процес розроблення переходить на наступну ітерацію. Якщо ітерація не виконала свого завдання, розробники повинні переглянути свої рішення і спробувати інший підхід.

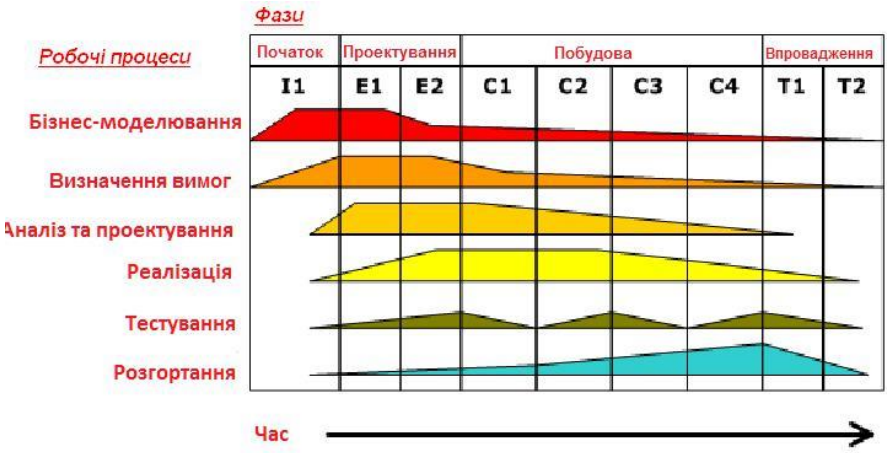


Рисунок 30 – Ітеративність уніфікованого процесу розроблення ПЗ

Для отримання максимальної економії команда, що працює над проектом, повинна вибирати тільки ті ітерації, які потрібні для досягнення мети проекту. Для цього необхідно розмістити ітерації в логічній послідовності. Непомічені раніше проблеми приведуть до збільшення ітерацій або зміни їх послідовності, а це збільшує кількість зусиль і часу для виконання процесу розроблення. Мінімізація непомічених проблем є однією з цілей зниження ризиків.

Керований ітеративний процес має такі переваги [32]:

- керована ітерація обмежує фінансові ризики витратами на одне прирощення. Якщо розробникам потрібно повторити ітерацію, організація втрачає лише зусилля, витрачені на одну ітерацію, а не
- вартість усього продукту;
- керована ітерація знижує ризик непостачання продукту на ринок у заплановані терміни. При ранньому виявленні відповідного ризику час, що витрачається на його нейтралізацію, вноситься в план на ранніх стадіях, коли співробітники менш завантажені, ніж у кінці планового періоду. При традиційному підході, коли серйозні проблеми вперше проявляються на етапі тестування системи, час, необхідний для їх усунення, як правило, перевищує час, що залишився за планом для завершення всіх робіт, що майже завжди
- приводить до затримок поставок;
- керована ітерація прискорює темпи процесу розроблення в цілому,

оскільки для більш ефективної роботи розробників і швидкого отримання ними гарних результатів короткий і чіткий план ефективніший;

- керована ітерація визнає, що бажання користувачів та пов'язані з ними вимоги не можуть бути чітко визначені на початку розроблення. Вимоги уточнюються в послідовних ітераціях, що полегшує адаптацію до змін вимог.

Усі характеристики уніфікованого процесу – керований варіантами використання, архітектурно-орієнтований, ітеративний та інкрементний процес розроблення – однаково важливі. Архітектура надає структуру, що направляє роботу в ітераціях, у кожній з яких варіанти використання визначають цілі та спрямовують роботу.

Моделі уніфікованого процесу розроблення ПЗ

Модель варіантів використання (use-case model) є концептуальною моделлю системи, що описує повну функціональність системи. Для побудови моделі варіантів використання в команді розробників виділяються співробітники, які виконують відповідні функції та створюють певні артефакти (рис.31). На рис.32 наведено схему робочого процесу розроблення моделі ВВ.

Співробітник – позначення ролі, що доручена одній або кільком особам, визначає потрібні для цієї дії якості та здібності.

Артефакт *(artifact)* – частина інформації, що створюється, змінюється або використовується співробітником під час роботи системи, визначає область відповідальності та дає змогу керувати своїми версіями

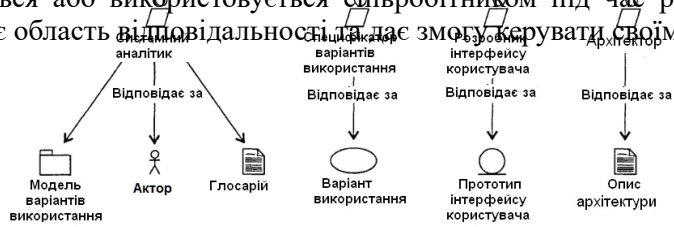


Рисунок 31 – Співробітникі, які будують модель варіантів використання

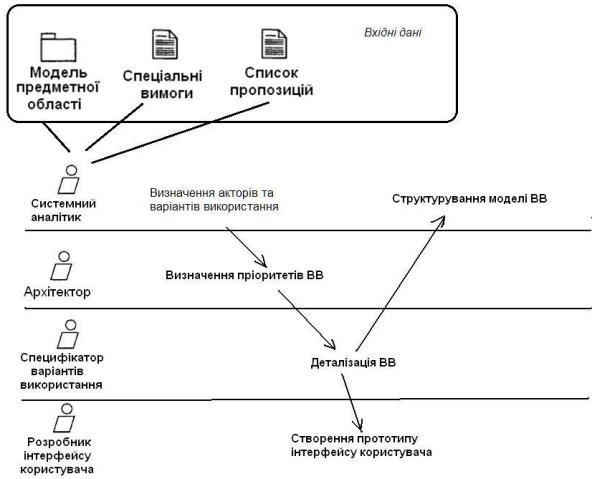


Рисунок 32 – Робочий процес побудови моделі ВВ

Модель аналізу *(analysis model)* будується на базі моделі ВВ для більш точного розуміння вимог, створення їх простого опису та для виявлення структури системи, в тому числі архітектури. На рис.33, 34 показані учасники процесу аналізу, створювані артефакти та схема робочого процесу розроблення моделі

Рисунок 33 – Співробітники, які будують модель аналізу

У табл.3 подано порівняння моделі аналізу із моделлю ВВ, з якого видно, що модель аналізу використовується розробниками, тоді як модель ВВ повинна бути зрозумілою і замовникові, і розробникам.

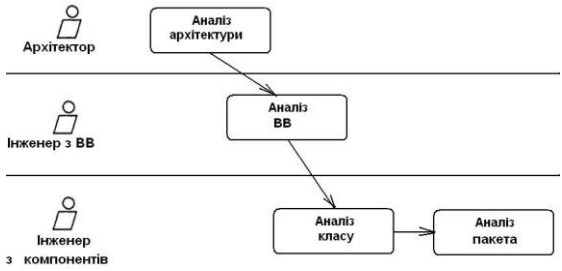


Рисунок 34 – Робочий процес побудови моделі аналізу

Таблиця 3 – Зіставлення моделі аналізу та моделі варіантів використання

Характеристика	**Модель ВВ**	**Модель аналізу**
Використовує мову	замовника	розробника
Містить вигляд системи	зовнішній	внутрішній
Модель структурована за	варіантами використання	класами та пакетами
Використовується	для домовлення між замовником та розробниками	розробниками для розуміння системи
Містить надлишковість, несумісні вимоги	так	ні
Визначає	функції системи та ВВ для подальшого аналізу	як функції реалізуються у реалізаціях ВВ

Модель проектування *(design model)* є об'єктною моделлю, сконцентрованою на вимогах, функціональних та нефункціональних, які разом із обмеженнями середовища розроблення реалізують систему.

Зіставлення моделей проектування та аналізу (табл.4) показує, що модель проектування є формалізованим описом проекту системи, сконцентрованим на її реалізації.

Модель проектування має такі *властивості*: має ієрархічну структуру; реалізації ВВ є стереотипами кооперації; модель є кресленням

реалізації.

Модель проектування розробляється разом із моделлю розгортання, оскільки проект системи неможливо створити без визначення фізичного розподілу системи за розрахунковими вузлами, що містить **модель розгортання** *(deployment model)*.

Також ця модель перевіряє, чи можуть ВВ бути реалізовані у вигляді компонентів, які виконуються у цих вузлах

Таблиця 4 – Зіставлення моделей проектування та аналізу

Характеристика	**Модель аналізу**	**Модель проектування**
Тип моделі	концептуальна	фізична
Залежність	від проекту	від реалізації
Рівень формалізації	низький	високий
Кількість стереотипів Класів	Три – сутність (entity), керування (control), межа (boundary)	будь-яка
Розмір	невеликий	значний
Детальність	ескіз проекту системи	опис проекту системи з увагою до послідовності
Підтримка у ЖЦ	може не підтримуватись	підтримується весь ЖЦ
Визначає	структуру	систему, зберігаючи структуру моделі аналізу

Розробники моделей проектування та розгортання, створювані ними артефакти наведені на рис.35.

Рисунок 35 – Співробітники та артефакти моделі проектування

Модель проектування розробляється відповідно до заданого порядку (рис.36):

1. Ідентифікуються класи.
2. Виділяються відповідальності.
3. Проектуються класи та реалізації ВВ.
4. Класи проектування збирають у підсистеми
5. Визначають інтерфейси між підсистемами.

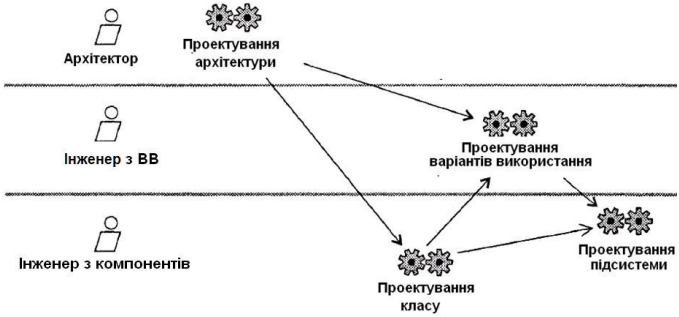


Рисунок 36 – Робочий процес побудови моделі проектування

Модель реалізації *(implementation model)* дає опис реалізації моделі проектування у вигляді компонентів програмного продукту. Елементи моделі та її розробники наведені на рис. 37.

Рисунок 37 – Співробітники та артефакти моделі реалізації

В уніфікованому процесі створення ПЗ передбачається покрокове розроблення. Результатом кожного кроку є ***білд*** *(build)* – виконувана версія системи.

Після визначення складових компонентів моделі створюється опис інтерфейсів їх взаємозв'язку та розробляється ***план складання***, що дає опис послідовності ітерацій. Для кожного білду план містить опис:

- функцій, які потрібно реалізувати у білді (це перелік ВВ та/або сценаріїв або їх частин);
- частини моделі реалізації, що стосуються білду (перелік підсистем та компонентів, потрібних для реалізації функціональності).

У результаті виконання робочого процесу будується модель реалізації системи (рис.38).

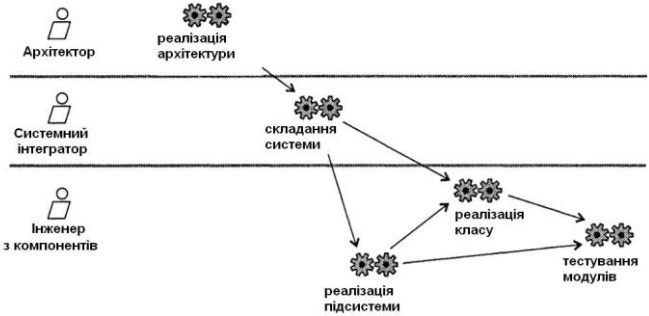


Рисунок 38 – Робочий процес побудови моделі реалізації

Модель тестування *(test model)* описує, як виконувані компоненти моделі реалізації тестуються на цілісність та проходять системні тести. У розробленні моделі тестування задіяні чотири співробітники (рис. 39,40,41) – інженер з тестування (розробник тестів), інженер з компонентів, тестувальник цілісності та системний тестувальник

Рисунок 39 – Артефакти, розроблювані інженером з тестування

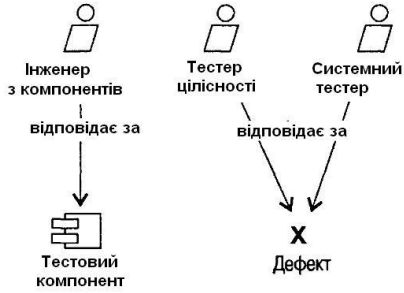
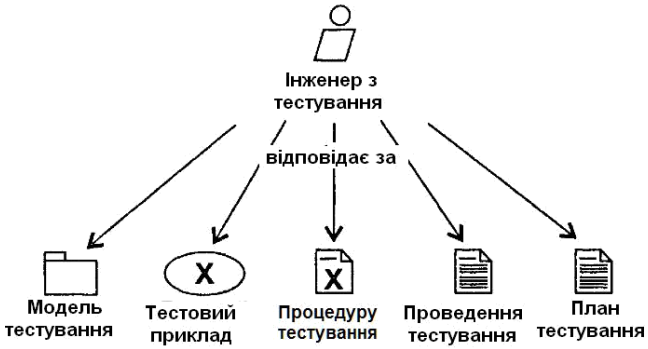


Рисунок 40 – Артефакти моделі тестування, розроблювані інженером з компонентів, тестувальником цілісності та системним тестувальником

Модель тестування розробляється (рис. 39) з урахуванням вимоги, що кожен білд є об'єктом тестування і керується системою контролю версій системи. У моделі тестування розробляються ***тестові приклади*** – шляхи тестування системи, що містить предмет тестування, вхідні дані, результат та умови тестування, – та ***тестові процедури*** – методики запуску одного/кількох тестових прикладів або їх частин. Обов'язково складається ***план тестування***, що містить опис стратегії тестування, виділених ресурсів



та графіка робіт.

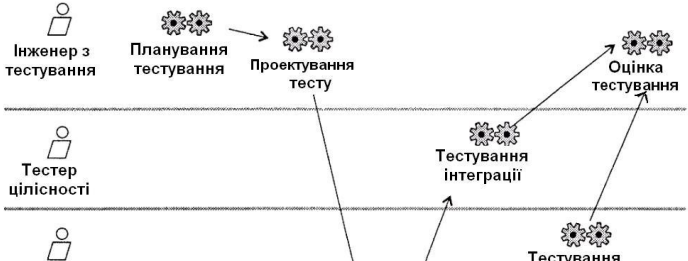


Рисунок 41 – Робочий процес побудови моделі тестування

Методологія RUP реалізована у програмному продукті Ration Software Architect компанії IBM. Як CASE-система, Ration Software Architect має велику кількість інструментів, корисних для підтримки зв'язку перших етапів проектування з етапом реалізації програм (кодування), а також з етапом оцінки. Зокрема перевіряється, що моделювання на різних етапах погоджено, що модельні угоди, визначення класів, інших елементів моделей та їх взаємозв'язку несуперечливі. Рівень автоматичного аналізу високий настільки, що дозволяє будувати за моделями так звані реалізації за замовчуванням. Це заготовки програмного коду, що включають у себе описи класів та їх методів у тому вигляді, що можна зробити з моделей. Програміст доповнює заготовки фрагментами, деталізує конкретну реалізацію.

У Ration Software Architect та інших UML CASE- системах підтримується побудова реалізацій за замовчуванням за моделями загального, а не спеціального призначення. Реалізація за замовчуванням є лише одним із прийомів підтримки зв'язків між етапами життєвого циклу розроблення програмного забезпечення. Саме ідея комплексної підтримки зв'язаності робочих продуктів різних етапів, а не окремі прийоми, які з'являлися і раніше, - головне для даної CASE-системи. Програмне втілення цієї ідеї, нехай навіть з істотними недоробками, необхідно віднести до явних переваг даного інструментарію.

Методологія Microsoft Solution Framework (MSF)

На цей час абсолютна більшість користувачів в Україні працюють на продуктах компанії Microsoft. Ця компанія підтримує університети по усьому світові, надає ряд ПЗ для підтримки основних процесів господарської діяльності. Тому при вивченні технологій програмування не можна пропустити досвід Microsoft у галузі створення програм. Аналіз результатів виконання програмних проектів, виконаний службою Microsoft Consulting Services, виявив, що лише чверть проектів можна визнати успішними, половина проектів мала великі проблеми (рис. 42).

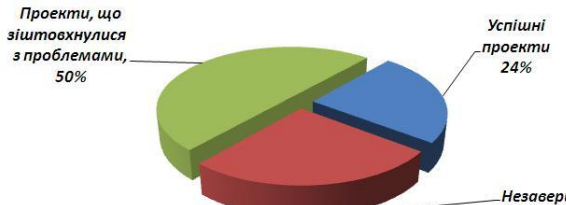


Рисунок 42 – Статистика успіхів проектів з розроблення ПЗ компанії Microsoft

Основними причинами невдач були визнані такі:

- постійна зміна вимог; нечіткі або неповні специфікації;
- низька якість коду;
- занадто широка постановка завдання;
- помилка в підборі кадрів;
- погана організація роботи;
- нечітко сформульовані цілі.

Для подолання цих труднощів був запропонований набір моделей Microsoft Solution Framework (MSF), в якому врахований досвід, накопичений групами розроблення програмних продуктів.

В основу MSF покладено *вісім базових принципів (*foundational principles*)*:

Сприяння відкритої комунікації (Foster open communications). Модель процесу MSF запроваджує вільний інформаційний потік сред членів команди та зацікавлених сторін (key stakeholders) для забезпечення однакового розуміння завдань. Документування ходу проекту та доступ до цих даних членів команди, зацікавлених сторін та замовників.

Робота у напрямку спільного бачення проекту (Work toward a shared vision). Модель процесу MSF запроваджує фазу формування концепції (Envisioning Phase) та окремий ключовий момент затвердження бачення (milestone Vision/Scope Approved) для формування спільного бачення проекту. Бачення включає детальне розуміння цілей та завдань для досягнення рішення поставленої проблеми. Спільне бачення виявляє припущення команди та замовників, потрібні для отримання рішення.

Надання прав і можливостей членам команди (Empower team members). Розширення прав і можливостей членів команди для прийняття ними відповідальності за виконану роботу. Таке збільшення відповідальності може бути затверджене у графіках, де фіксується дата закінчення робіт, що також може бути засобом виявлення можливих затримок проекту.

Визначення індивідуальної та спільної відповідальності (Establish clear accountability and shared responsibility). Модель командної групи MSF ґрунтується на принципі важливості роботи кожного для отримання якісного рішення проблеми. Усі члени групи розділяють відповідальність за проект.

Зосередження на бізнес-цілях (Focus on delivering business value). Рішення повинне приносити користь організації у вигляді додавання вартості бізнесу.

Це додавання досягається тільки після повного розгортання рішення у виробничому середовищі.

Бути гнучкими, очікувати на зміни (Stay agile, expect change). MSF припускає, що у виробничому середовищі на рішення постійно впливають зміни. Команда повинна бути обізнаною та готовою до керування змінами вимог.

Інвестування у якість (Invest in quality). У MSF кожен член команди відповідальний за якість вирішення завдання.Для підтримки якості протягом проекту формується команда тестувальників. Це гарантує, що рішення відповідає рівню якості, визначеному зацікавленими сторонами.

Навчання за досвідом (Learn from all experiences). MSF вимагає використання досвіду, отриманого у попередніх проектах. Це дозволяє знайти найкращі методики розроблення.

MSF складається із *двох моделей* та *трьох дисциплін*:

- моделі командної групи;
- моделі процесу;
- дисципліни управління проектами;
- дисципліни управління ризиками;
- дисципліни управління підготовкою.

Модель командної групи (MSF Team Model) описує, як організувати колективи і якими принципами керуватися для досягнення максимального успіху в розробленні програм. Різні колективи можуть по-своєму застосовувати на практиці різні елементи цієї моделі – все залежить від масштабу проекту, розміру колективу і кваліфікації його учасників та моделі процесу розроблення.

Формування колективу є складним завданням, що повинне виконуватися психологами та враховувати такі основні положення: не повинне бути команди з одних лідерів; не повинне бути команди з одних виконавців; у випадку невдачі команда розформовується; система штрафів (якщо проект провалюється - карають усіх).

Модель командної групи визначає тільки ролі, кожна з яких може виконуватися кількома людьми (рис. 43). Цікаво, що в моделі командної групи не передбачено єд иноначальності – усі ролі важливі, усі ролі рівноправні, тому MSF називають моделлю рівних (team of peers).

•

Рисунок 43 – Модель командної групи (MSF Team Model)

Program management – керування програмою.

Виконавець цієї ролі відповідає за організацію (але не керує): здійснює ведення графіка робіт, ранкові 15-хвилинні наради, забезпечує відповідність стандартам і специфікаціям, фіксацію порушень, написання технічної документації.

Product management – керування продуктом. Виконавці цієї ролі відповідають за спілкування із замовником, написання специфікації, роз'яснення завдань розроблювачам.

Development – найбільш традиційна роль – ***розроблення і початкове тестування продукту***.

User expirience – підвищення ефективності роботи користувачів, написання користувальницької документації.

Release management – розгортання релізу продукту, супровід і його технічна підтримка.

Test – визначення відповідності показників якості релізу встановленим значенням. Виявлення й усунення недоробок, виправлення помилок, інші функції QA.

У MSF затверджується, що таку модель можна масштабувати, розбиваючи систему за функціями.

Модель процесу (MSF Process Model) визначає, коли і які роботи повинні бути виконані.

Основні принципи і практичні прийоми, на яких ґрунтується модель:

- ітеративний підхід (послідовний випуск версій); підготовка чіткої документації; урахування невизначеності майбутнього; облік компромісів; керування ризиками;
- підтримка відповідального відношення колективу до строків випуску продукту; розбивка великих проектів на більш дрібні керовані частини; щоденне складання проекту; постійний аналіз ходу робіт.

Process model має три *основні особливості*: розбивка всього процесу на фази; використання опорних точок (milestones); ітеративність.

Увесь процес розбивається на п’ять взаємозалежних фаз (рис. 44). Перш ніж переходити до наступної фази, на попередній повинні бути отримані певні результати (досягнуті головні опорні точки). Фактично процес ітеративний і відповідає спіральній моделі ЖЦ ПЗ.

Рисунок 44 – Модель процесу (MSF Process Model)

Envisioning Phase – вироблення єдиного розуміння проекту всіма членами колективу. Ця фаза закінчується розробленням формалізованого документа, що містить:

- *problem statement* – опис завдання на розроблення ПЗ обсягом не більше однієї сторінки;
- *vision statement* – опис того, від чого відштовхується розроблення і яким результатом закінчується;
- *solution concept* - що буде впроваджене в результаті вирішення поставленої проблеми
- *user profiles* – опис потенційних користувачів системи;
- *business goals* – опис бізнес-функцій, виконання яких за допомогою розробленого ПЗ поверне інвестиції;
- *design goals* – конкретні цілі й обмеження програмного продукту, його конкретні властивості.

Planning Phase – планування чергового циклу розроблення:

- функціональні специфікації;
- план-графік робіт;
- оцінка ризиків.

Developing Phase – розроблення, причому рекомендуються різні технологічні прийоми, наприклад, повторне використання фрагментів коду, програмування за контрактом, написання захищеного від помилок ПЗ та ін.

Stabilizing Phase – створення стабільної β-версії, готової до використання.

Важливу роль відіграють *опорні точки (milestones)*, у яких аналізується стан робіт і виробляється їхня синхронізація. У цих точках додаток або його специфікації не заморожуються. Опорні точки дозволяють проаналізувати стан проекту і внести необхідні корективи, наприклад, переналагоджуватися під вимоги, що змінилися, замовника або відреагувати на ризики, можливі в ході подальшої роботи. Для кожної опорної точки визначається, які результати повинні бути отримані до цього моменту.

Кожна фаза процесу розроблення завершується *головною опорною точкою (major milestone)* – моментом, коли всі члени колективу синхронізують отримані результати. Призначення таких точок у тому, що вони дозволяють оцінити життєздатність проекту. Їхні результати видимі не тільки колективу розроблювачів, але й замовникові. Після аналізу результатів колектив розробників і замовник спільно вирішують, чи можна переходити на наступну фазу. Таким чином, головні опорні точки - це критерії переходу з однієї фази проекту на іншу.

Усередині кожної фази визначаються *проміжні опорні точки (interium milestones).* Вони, як і головні, слугують для аналізу й синхронізації досягнутого, а не для заморожування проекту. Але на відміну від головних опорних точок проміжні видні тільки членам колективу розробників.

Ітеративність процесу MSF полягає в його багаторазовому повторенні упродовж усього циклу розроблення та існування продукту. На кожній успішній ітерації у продукт включаються тільки ті нові інструменти та функції, які задовольняють постійно змінювані вимоги бізнесу.

Методологія eXtreme Programming (XP)

Екстремальне програмування *(eXtreme Programming, XP)* – спрощена методологія організації виробництва для невеликих і середніх за розміром команд розробників, які займаються розробленням програмного продукту в умовах незрозумілих або швидко змінних вимог [19].

Програмування відповідно до методик XP доводить використання загальноприйнятих принципів програмування до екстремальних рівнів:

- перегляд коду виконується постійно (з урахуванням того, що програмування ведеться парами);
- кожен учасник проекту тестує код програми постійно (тестування модулів), навіть замовники проводять функціональне тестування;
- проектування є складовою частиною повсякденної роботи кожного розробника (перероблення коду);
- розроблення виконується з урахуванням вимоги збереження в системі найбільш простого дизайну, що забезпечує поточний необхідний рівень функціональності (простіші речі надійніші в роботі);
- увага до архітектури системи на кожному етапі проекту;
- інтеграційне тестування виконується після кожної найменшої зміни у системі (триваюча інтеграція);
- ітерації невеликі – тривають години, кілька днів (постійне планування).

У розділі конспекту «Керування та організація робіт» наведені основні ризики розроблення ПЗ, виділені К.Беком. Для зменшення кількості можливих ризиків та усунення їх негативного впливу і розроблена

методологія XP. У табл.5 наведені найпоширеніші ризики розроблення та методи їх усунення за допомогою екстремального програмування.

Таблиця 5 – Методи усунення ризиків розроблення ПЗ у методології XP

Ризик	**Метод подолання у XP**
Зміна графіка	Планується короткострокове розроблення версій ПЗ
Закриття проекту	Замовник визначає найменший перелік найважливіших функцій системи, які визначають якість системи
Система втрачає корисність	Постійне розроблення та виконання тестів над системою не дозволяють накопичуватися дефектам, що впливають на роботу системи
Велика кількість дефектів і недоліків	Тести постійно створюються і виконуються не лише розробниками, а й замовниками, які перевіряють функціональність системи
Невідповідність ПЗ розв'язуваній проблемі	Представник замовника є постійним членом команди розробників. Специфікація постійно перевіряється і змінюється за потреби
Зміна характеру бізнесу	Бізнес не встигає помітно змінити напрямок діяльності у ході короткострокового розроблення версії програми
Нестача	Задачі із найбільшим пріоритетом

функціональних можливостей	реалізуються у першу чергу
Плинність кадрів	Програмісти самостійно оцінюють обсяг робіт та термін виконання, що підвищує їх зацікавленість

Зміна графіка робіт є найпоширенішою проблемою під час створення ПЗ. Неуважність до розтягування робіт може зірвати увесь проект і стати причиною усіх інших ризиків. *На кожну версію* програми при використанні ХР *виділяється один-два місяці*. У рамках кожної версії планується кілька *ітерацій* для отримання запланованої функціональності, кожна з яких *триває один-чотири тижні*. *Замовник обов'язково перевіряє функціональність*, отриману в ході чергової ітерації, тобто постійно забезпечується зв'язок із замовником, що отримує уявлення про поточний стан робіт. Кожна ітерація розділяється на кілька *задач*, які *тривають кілька днів*. *Найважливіші завдання реалізуються в першу чергу.* Тобто при зміщенні строків робіт можна бути впевненим, що не робробленими залишилися завдання низького пріоритету. Використання коротких етапів проекту для розроблення чергової версії програми дозволяє гнучко керувати строками робіт.

Для формування стилю розроблення, що дозволить досягти потрібної якості рішення, в ХР пронується керуватись чотирма цінностями [19]:

комунікацією *(communication)* – дисципліна ХР, спрямована на забезпечення безперервної комунікації усіх учасників проекту. Під час тестування, програмування в парах та попереднього оцінювання робіт замовники, розробники та менеджери змушені постійно спілкуватись;

простотою *(simplicity)* – неможливо заздалегідь точно визначити, як буде розвиватися проект, тому вирішувати необхідно лише завдання, що виникають сьогодні, у найбільш простий спосіб;

зворотним зв'язком *(feedback)* – спосіб отримати точні та конкретні дані про стан проекту. Постійне виконання тестів для перевірки усіх змін у системі забезпечує програміста звороним зв'язком про якість роботи. Кожен новий опис замовником вимог до системи одразу ж оцінюється розробниками і забезпечує замовника зворотним зв'язком із інформацію про якість опису. Менеджер, який контролює строки виконання робіт, забезпечує усіх учісників проекту інформацією про виконання планових термнів розроблення. В ХР найбільш важливі функції системи реалізуються в першу чергу у реально працюючий продукт, тому замовник досить швидко може оцінити хід виконання та якість розроблення, а також відповідність замовленню;

хоробрість *(courage)* – для того, щоб розроблення не втратила своєї актуальності, в ХР рішення повинні прийматися з максимальною швидкістю, для чого потрібна певна сміливість.

Для визначення методів вирішення проблем розроблення ПЗ на основі обраних цінностей потрібно керуватися такими *фундаментальними принципами* оцінки методів вирішення поставленого завдання:

швидкий зворотний зв'язок – інформація про стан системи повинна якомога швидше надходити до зацікавлених сторін проекту. У рамках дисципліни ХР ці дані повинні із максимальною швидкістю надходити, інтерпретуватися та на основі їх аналізу швидко виконуватися модифікації системи;

прийнятна простота – кожна проблема повинна вирішуватись у найбільш простий спосіб. У рамках ХР значні зусилля (тестування, переробка коду, комунікації) покладаються для вирішення завдань сьогодення таким чином, щоб завтра внесення змін не потребувало великих зусиль;

поступова зміна – кардинальні зміни часто приводять до провалу проекту, щоб заплановані модифікації мали успіх, їх потрібно реалізовувати як серію невеликих змін, після кожної з яких перевіряється працездатність системи;

прийнятна зміна – найвигідніша стратегія – та, що дозволяє вирішити найбільш важливу проблему і залишає максимальну свободу дій;

якісна робота – кожен учасник проекту повинен прагнути максимально якісно виконувати свої завдання і сприяти якісній роботі інших.

Кожен принцип втілює заявлені цінності. З-поміж альтернативних методів, які втілюють принципи, в ХР пропонується обирати метод рішення, який відповідає виконанню більшості принципів У рамках ХР пропонуються методи для виконання чотирьох основних видів діяльності у ході розроблення ПЗ:

Кодування – від якості коду залежать робота та надійність системи. Якісне кодування потребує постійного навчання та удосконалення. Також код містить інформацію про стан проекту у найльш стислій та чітко зрозумілій формі;

Тестування – постійне тестування зменшує вартість внесення змін у розроблення та зменшує ризики, пов’язані із змінами. Виконання усіх запланованих тестів свідчить про успішне завершення чергової версії або усього проекту;

Слухання – відкрите чесне спілкування – як між розробниками, так і між розробниками і замовниками. Програмісти повинні почути бізнес-вимоги замовника, щоб виконати завдання, а замовник повинен чути зауваження розробників, щоб краще розуміти свої потреби;

Проектування – проект системи впливає на якість ПЗ і на легкість внесення змін. Правильний проект створює структуру, що організовує логіку системи. Логічно пов’язані фрагменти ПЗ повинні поєднуватись у незалежні частини системи. При правильному дизайні програми внесення змін в

одному елементі системи не буде потребувати змін у інших.

Для розв'язання виділених проблем розроблення в методології XP використовуються такі методики [19]:

Гра в планування *(planning game)* – швидко визначає обсяг робіт, що необхідно виконати у наступній версії. Для цього розглядаються бізнес-пріоритети та технічні оцінки. Якщо з часом план перестає відповідати дійсності, відбувається оновлення плану. Замовники повинні приймати рішення стосовно обсягів робіт, пріоритетності завдань та строків випуску версій. Розробники повинні відповідати за оцінку часу, потрібного на реалізацію вимог, наслідки прийнятих рішень, організацію процесу розроблення та детальне планування робіт.

Невеликі версії *(small releases)* – найперша спрощена версія системи, що реалізує найбільш важливу функціональність, швидко вводиться в експлуатацію. Наступні версії випускаються через відносно короткі проміжки часу. ***Метафора*** *(metaphor)* – проста загальнодоступна і загальновідома історія, що коротко описує, як працює вся система. Ця історія керує усім процесом розроблення.

Простий дизайн *(simple design)* – у кожен момент часу система повинна бути спроектована так просто, як це можливо. Виявлена надмірна складність усувається одразу.

Тестування *(testing)* – програмісти постійно пишуть тести для модулів, а замовники – тести, які демонструють працездатність і завершеність тієї чи іншої можливості системи. Умова продовження розроблення – усі тести спрацьовують без помилок.

Переробка *(refactoring)* – програмісти реструктурують систему, не змінюючи її поведінки. При цьому вони усувають дублювання коду, покращують комунікацію, спрощують код і підвищують його гнучкість.

Програмування парами *(pair programming)* – увесь розроблювальний код пишеться двома програмістами на одному комп'ютері.

Колективне володіння *(collective ownership)* – у будь-що момент часу будь-що член команди може змінити будь-що код у будь-якому місці системи.

Безперервна інтеграція *(continuous integration)* – система інтегрується і збирається кожного разу, коли завершується рішення чергового завдання (можливо кілька разів за день).

40-годинний тиждень *(40-hour week)* – програмісти працюють не більше 40 годин на тиждень. Ніколи не можна працювати понаднормово два тижні поспіль.

Замовник на місці розроблення *(on-site customer)* – до складу команди входить реальний користувач системи. Він доступний упродовж усього робочого дня і здатний відповідати на запитання про систему.

Стандарти кодування *(coding standards)* – програмісти пишуть увесь код у відповідності до правил, які забезпечують комунікацію за

допомогою коду.

Гнучке розроблення ПЗ на основі Agile

Постійна зміна вимог під час розроблення ПЗ, необхідність забезпечення ефективної співпраці команди розробників потребували методів, які б забезпечили якість розроблення та супроводу програмних систем і уникнули недоліків занадто формалізованих методів, більшість з яких базувалась на каскадній моделі ЖЦ. У 90-х рр. XX ст. активно стали виникати різноманітні підходи до створення ПЗ, які забезпечували гнучку роботу програмістів. У результаті у 2001 році був написаний **Маніфест гнучкого розроблення** *(Agile manifesto)*, що зафіксував цінності ефективних підходів до розроблення програмних продуктів, таких, як XP, Feature driven development, Scrum, Adaptive software development, Pragmatic Programming (прагматичне програмування). Текст маніфесту [35] підписали 17 найавторитетніших фахівців у цій галузі діяльності – Кент Бек (Kent Beck), Алістер Коуберн (Alistair Cockburn), Мартін Фаулер (Martin Fowler) та інші:

Agile-маніфест розроблення програмного забезпечення

Ми постійно відкриваємо для себе досконаліші методи розроблення програмного забезпечення, займаючись розробленням безпосередньо та допомагаючи у цьому іншим. Завдяки цій роботі ми змогли зрозуміти, що: ***Люди та співпраця*** *важливіші за процеси та інструменти;* ***Працюючий продукт*** *важливіший за вичерпну документацію;* ***Співпраця із замовником*** *важливіша за обговорення умов контракту;* ***Готовність до змін*** *важливіша за дотримання плану.*

Хоча цінності, що справа, важливі, ми все ж цінуємо більше те, що зліва.

У результаті з'явився термін – ***Гнучке розроблення програмного забезпечення*** *(Agile software development)* – клас методологій розроблення програмного забезпечення, що базуються на ітеративному розробленні, в якому вимоги та рішення еволюціонують через співпрацю між самоорганізовуваними багатофункціональними командами.

У маніфесті озвучені основні принципи гнучкого розроблення ПЗ:

- Найвищий пріоритет – *задоволення потреб замовника* шляхом завчасного та регулярного постачання програмного забезпечення.
- *Схвальне ставлення до змін*, навіть на завершальних стадіях розроблення. Agile-процеси використовують зміни для забезпечення конкурентоспроможності замовника.
- *Працюючий продукт* необхідно *випускати* якомога *частіше*, з періодичністю від двох тижнів до двох-трьох місяців.
- Упродовж усього проекту *розробники і представники бізнесу повинні працювати разом* щодня.
- Над проектом *повинні працювати вмотивовані професіонали*, які

працюють у зручних умовах, із повною підтримкою та довірою менеджменту проекту.

- *Особиста комунікація* – найефективніший та найпрактичніший метод обміну інформацією в команді.
- *Працюючий продукт – головний показник прогресу.*
- Інвестори, розробники і користувачі повинні мати можливість підтримувати *постійний ритм роботи*. Agile допомагає налагодити такий сталий процес розроблення.
- Постійна увага до *технічної досконалості і якості проектування* підвищує гнучкість проекту.
- *Простота* – мистецтво мінімізації зайвої роботи – дуже необхідна.
- *Найкращі* вимоги, архітектурні та технічні *рішення виникають у командах, які здатні самоорганізовуватись*. Команда *постійно шукає способи підвищення ефективності* та відповідно коригує свою роботу.

Більшість гнучких методологій націлені на мінімізацію ризиків шляхом зведення розроблення до серії коротких циклів
– ітерацій, які, як правило, тривають один-два тижні. Кожна ітерація є програмним проектом у мініатюрі і включає всі завдання, необхідні для отримання мінімального приросту функціональності: планування, аналіз вимог, проектування, кодування, тестування та документування. Гнучкий програмний проект передбачає в кінці кожної ітерації отримання працездатного, готового до встановлення у реальному середовищі продукту. Після закінчення кожної ітерації команда виконує переоцінку пріоритетів розроблення.

Гнучкі методи орієнтовані на різні аспекти життєвого циклу розроблення програмного забезпечення. Деякі акцентують увагу на практичних завданнях (екстремальне програмування, прагматичне програмування, Agile-моделювання), інші зосереджені на управлінні програмними проектами (Scrum). Також серед agile-методів існують підходи, які забезпечують повне охоплення життєвого циклу розроблення – метод розроблення динамічних систем (dynamic systems development method, DSDM) та уніфікований процес розроблення (RUP), розглянутий вище. Більшість гнучких методів для використання упродовж життєвого циклу ПЗ потрібно доповнювати іншими підходами, окрім DSDM та RUP.

Особливістю гнучких методів є те, що на даний час відсутні дані про провальні agile-проекти, але є результати опитувань, які підтвердили їх успішне використання [36]. Тому і потужні компанії-розробники активно їх запроваджують у свою діяльність, наприклад, Oracle запроваджує гнучкі методи управління ЖЦ продуктів (Agile product lifecycle management), фірма IBM є власником продуктів, які підтримують використання методології RUP.

Потрібно виділити *фактори, які можуть негативно вплинути* на

використання гнучких методів: масштабні зусилля у галузі розвитку (більше 20 розробників); розподілена у просторі команда; примусове впровадження гнучкого процесу усупереч вимогам команди розробників; Небажане використання agile-методів у критично важливих системах (табл.6), де відмова ПЗ неможливий ні в якому разі (наприклад, управління повітряним рухом).

Таблиця 6 – Порівняння використання гнучких та формальних методів

Agile-методи	**Методи із чітким планом**	**Формалізовані методи**
Низька критичність	Висока критичність	Життєво важлива критичність
Досвідчені Розробники	Молодші Розробники	Досвідчені розробники
Вимоги часто Змінюються	Вимоги змінюються не часто	Обмежені вимоги
Малі команди Розробників	Великі групи Розробників	Вимоги обов'язково моделюються
Культура, що залежить від змін	Стала культура Розроблення	Екстремальна увага на якості розроблення

Agile акцентує увагу на безпосередньому спілкуванні між учасниками проекту. Більшість agile-команд розміщені в одному офісі (*bullpen*). До команди обов'язково включають представників замовника (замовники, які визначають продукт, менеджери продукту, бізнес-аналітики або користувачі). Також до команди входять тестувальники, дизайнери інтерфейсу, технічні автори та менеджери.

Із agile-методів потрібно виділити ***Scrum***, що встановлює правила керування процесом розроблення та дозволяє використовувати вже існуючі практики кодування, коригуючи вимоги або вносячи тактичні зміни. Використання цієї методології дає можливість виявляти і усувати відхилення від бажаного результату на найбільш ранніх етапах розроблення програмного продукту. Scrum-методи ітераційні та інкрементні. Кожна ітерація (спринт) триває упродовж 15-30 днів і повинна закінчитися приростом функціональних можливостей. Під час спринту обов'язково відбуваються зустрічі дійових осіб:

Планування спринту *(Planning Meeting)* – відбувається на початку ітерації.

Не більше 4-8 годин.

Мітинг (Daily Scrum) – відбувається кожного дня упродовж спринту. Триває 15 хвилин.

Демонстрація (Demo Meeting) **–** відбувається в кінці ітерації (спринту). Обмежена 4 годинами.

Ретроспектива (Retrospective Meeting). Усі члени команди розповідають про своє ставлення до ходу спринту. Обмежено 1-3 годинами.

Дійові особи розподіляються на дві групи: *повністю задіяні* у процесі розроблення («свині»):

- Власник Продукту (Product Owner);
- Керівник (ScrumMaster);
- Команда (Scrum Team);

причетні до розроблення («кури»):

- Користувачі (Users);
- Клієнти, Продавці (Stakeholders);
- Експерти-консультанти (Consulting Experts).

Віддаючи перевагу безпосередньому спілкуванню, agile-методи зменшують обсяг письмової документації порівняно з іншими методами, що викликає критику прихильників більш формалізованих підходів.

Патерни проектування

Створюючи об'єкт, у тому числі й програмний продукт, розробник часто стикається із завданнями, які вже хто-небудь вирішив. У 70-ті рр. XX ст архітектор Кристофер Александер (Christopher Alexander) [37] запропонував шаблони проектування будинків та міст. Через десятиліття ця ідея переросла у процесі розроблення ПЗ до шаблонів проектування інтерфейсу користувача, запропонованих Вардом Каннінгемом (Ward Cunningham) та Кентом Беком [38]. Далі ідея була активно підхоплена та розвинута у вигляді каталогу патернів ООП. Цей каталог [39] став дуже популярним серед розробників і часто згадується як патерни GoF («Gang of Four», або «банда чотирьох», – за кількістю авторів). Ідея повторного використання не тільки коду, а й архітектурних та проектних рішень виявилася настільки успішною, що сьогодні патерни проектування широко застосовуються в різних методиках розроблення програмного забезпечення.

У роботі [39] під ***патернами проектування*** об'єктно-орієнтованих систем розуміють опис взаємодії об'єктів і класів, адаптованих для вирішення спільної задачі проектування в конкретному контексті.

Існує багато патернів розроблення програмних систем, які відмінні сферою застосуання, масштабом, змістом, стилем опису. Наприклад, залежно від області використання існують такі патерни, як патерни аналізу, проектування, кодування, рефакторингу, тестування, реалізації корпоративних застосувань, шаблони роботи з базами даних, шаблони розподілених додатків, шаблони роботи із багатопотоковістю, шаблони

документування та ін.

В останнє десятиліття шаблони впроваджені навіть у роботу менеджера процесу розроблення ПЗ (Джеймса Коплі, Ніла Харрісона «Organizational Patterns of Agile Software Development», Тома Демарко, Тіма Лістера «Adrenaline Junkies and Template Zombies: Understanding Patterns of Project Behavior»).

У даний час найбільш популярними патернами є патерни проектування. Однією з найпоширеніших класифікацій таких патернів є класифікація за ступенем деталізації та рівнем абстракції розглянутих систем. Патерни проектування програмних систем поділяються на три категорії [40] (рис.44).

Архітектурні патерни, найбільш високорівневі, описують

структурну схему програмної системи в цілому. У даній схемі зазначаються окремі функціональні складові системи (підсистеми), а також взаємовідносини між ними. Прикладом архітектурного патерну є добре відома програмна парадигма "модель-вигляд-контролер" (model-view-controller - MVC).

Рисунок 44 – Шаблони проектування програмних систем

Ідіоми, низькорівневі патерни, мають справу з питаннями реалізації певної проблеми з урахуванням особливостей мови програмування. При цьому часто одні й ті самі ідіоми для різних мов програмування мають різний вигляд або взагалі відсутні. Наприклад, в C++ для усунення можливих втрат пам'яті можуть використовуватися інтелектуальні покажчики. Інтелектуальний покажчик містить покажчик на ділянку динамічно виділеної пам'яті, що буде автоматично звільнений при виході із зони видимості. У середовищі Java такої проблеми просто не існує, оскільки там використовується автоматичне складання сміття. Як правило, для використання ідіом потрібно глибоко знати особливості застосовуваної мови програмування.

Завдання кожного патерну – дати чіткий опис проблеми та її

вирішення у відповідній області. У загальному випадку опис патерну завжди містить такі елементи [38]:

Назва патерну – унікальне змістове ім'я, що однозначно визначає дану задачу або проблему і її рішення *Розв'язувана задача* – надається розуміння того, чому розв'язувана проблема дійсно є такою, чітко описує її межі.

Рішення – зазначається, як саме дане рішення пов'язане з проблемою, наводяться шляхи її вирішення.

Результати використання патерну – як правило, це переваги, недоліки та компроміси.

На рис.45 показана класифікація шаблонів проектування інформаційних систем [41].

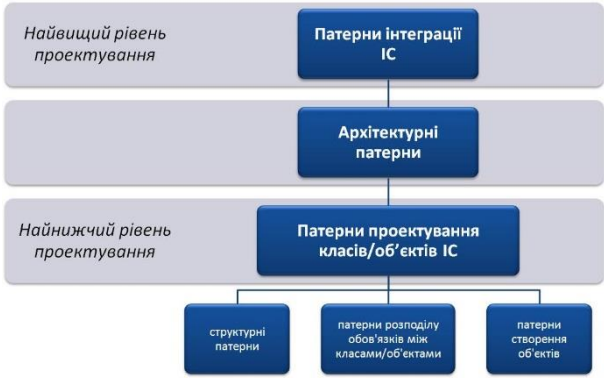


Рисунок 45 – Шаблони проектування інформаційнх систем

Архітектурні патерни також об'єднуються у групи:

- структурні, архітектурні патерни – слугують для організації класів або об'єктів системи у базовій підструктурі;
- патерни управління – для забезпечення необхідного системного функціоналу.

У свою чергу, патерни управління розділені на патерни централізованого управління (патерни, в яких одна з підсистем повністю відповідає за управління, запускає і завершує роботу решти підсистем), патерни управління, які передбачають децентралізоване реагування на події (згідно з цим патернам на зовнішні події відповідає відповідна підсистема), та патерни, які описують організацію зв'язку з базою даних

Патерни інтеграції інформаційних систем знаходяться на верхньому рівні класифікації патернів проектування. Аналогічно патернам більш

низьких рівнів класифікації серед патернів інтеграції виділена група структурних патернів. Структурні патерни описують основні компоненти інтегрованої метасистеми. Для опису взаємодії окремих корпоративних систем, включених до інтегрованої метасистеми, використовується група патернів, об'єднаних відповідно до методу інтеграції. Інтеграція корпоративних інформаційних систем передбачає організований обмін даними між системами, для якого використовується відповідний патерн. Необхідно зазначити, що на відміну від патернів проектування класів/об’єктів і архітектурних системних патернів віднесення окремого патерну інтеграції до того чи іншого виду є менш умовним.

Про шаблони проектування знають розробники, проектувальники, архітектори, але про це явище абсолютно нічого не відомо користувачам,для яких і розроблялася мова шаблонів Александера [37]. На сьогоднішній день основна роль шаблонів - це повторне використання досвіду в різних областях розроблення ПЗ, усунення комунікаційного бар'єру всередині команди розробників і між ними, підвищення якості створюваного продукту за рахунок використання перевірених роками рішень.

3.4. Стандарти проектування інформаційних систем. Методологія CDM

Реальне застосування будь-якої технології проектування ПЗ ІС не можливе без розробки **стандартів**, яких мають дотримуватися всі учасники проекту (це особливо актуально при великій кількості розробників). До них належать – стандарти проектування, оформлення проектної документації та інтерфейсу кінцевого користувача із системою.

Стандарт проектування встановлює:

а) набір необхідних моделей (діаграм) на кожній стадії проектування і ступінь їх деталізації;

б) правила фіксації проектних рішень на діаграмах, у тому числі правила іменування об’єктів, набір атрибутів для всіх об’єктів і правила їх заповнення на кожній стадії, правила оформлення діаграм тощо;

в) вимоги до конфігурації робочих місць розробників, включаючи настроювання операційної системи та CASE-засобів;

г) механізм забезпечення спільної роботи над проектом, у тому числі правила інтеграції підсистем проекту, правила підтримки проекту в однаковому для всіх розробників стані, правила аналізу проектних рішень на несуперечність.

Стандарт оформлення проектної документації установлює:

а) комплектність, склад і структуру документації на всіх стадіях проектування; б) вимоги до оформлення документації;

в) правила підготовки, розгляду, узгодження і затвердження документації із

зазначенням граничних термінів для кожної стадії;
г) вимоги до засобів підготовки документації;
д) вимоги до настроювання CASE-засобів для забезпечення підготовки документації відповідно до встановлених правил.

Стандарт інтерфейсу користувача із системою регламентує:
а) правила оформлення екранних елементів і елементів управління; б) правила використання клавіатури і миші; в) правила оформлення текстів допомоги; г) перелік стандартних повідомлень; д) правила обробки реакції користувача.

Питання для самоконтролю

1. Перелічіть візуальні нотації для моделювання предметної області та визначте області їх застосування.
2. Опишіть основні класифікатори мови UML.
3. Які діаграми використовуються в мові UML? Для чого вони використовуються?
4. Які методології розроблення вам відомі? Наведіть приклади.
5. Перелічіть переваги гнучкого розроблення ПЗ та зазначте її недоліки.
6. Перелічіть та поясніть основні характеристики уніфікованого процесу розроблення ПЗ.
7. Які моделі програмної системи розробляються в уніфікованому процесі RUP? Порівняйте їх.
8. Визначте особливості методології Microsoft Solution Framework.
9. Які цінності та принципи покладені в основу методів eXtreme Programming?
10. Що є особливостями методів Scrum?
11. Порівняйте методології RUP, MSF та XP

РОЗДІЛ 4. ФОРМУВАННЯ ТА СПЕЦИФІКАЦІЯ ВИМОГ ДО СТВОРЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

4.1. Вимоги до інформаційних систем та їх класифікація

Процес створення інформаційної системи характеризується тим, що на всіх його стадіях та етапах без винятку доводиться виробляти, обгрунтовувати та приймати численні рішення щодо принципів структурної побудови системи, засобів і принципів реалізації різних функцій і процесів управління, розробки всіх видів забезпечення.

Ці рішення приймаються як в умовах вірогідно-визначених даних, так і при даних, що мають стохастичний характер з відомими чи невідомими, а можливо, не існуючими, розподіленими вірогідностями. Тому рішення завжди слід приймати, базуючись на основних вимогах до створюваної системи, що визначаються на початку створення інформаційної системи.

Головна вимога — інформаційна система має забезпечувати підвищення ефективності виробничо-господарської діяльності економічного об’єкта, тобто приводити до корисних техніко-економічних, соціальних чи інших результатів. Наприклад: зниження чисельності управлінського персоналу, підвищення якості функціонування об’єкта, підвищення якості керування та ін.

Вимоги поділяються на дві групи:

1) вимоги, визначені державними стандартами, методичними матеріалами галузі замовника;

2) вимоги, які відбивають специфіку економічного об’єкта.

Це означає, що потрібно вивчити державні стандарти, методичні матеріали та об’єкт, який автоматизуємо, і виявити всі його особливості.

Основні вимоги до інформаційної системи згідно з ГОСТ 24.104–85 «Автоматизированные системы управления. Общие требования» поділяються на такі групи:

1) до інформаційної системи в цілому;

2) функцій інформаційної системи;

3) підготовленості персоналу;

4) видів забезпечення;

5) безпеки інформаційної системи.

Особливу роль відіграє дослідження економічного об’єкта, оскільки впровадження інформаційної системи завжди пов’язано зі зміною інформаційних потоків, із застосуванням нових методів і засобів виконання деяких функцій керування. Ці зміни виникають з двох причин:

1) згідно з методами та засобами, які впроваджуються, вдосконалюється інформаційна система;

2) зменшується різноманітність функцій управління, внесена самими працівниками з різних суб'єктивних чи об'єктивних причин.

Ця різноманітність виникає в результаті того, що використовуються оригінальні методи та засоби виконання певних робіт і загальноприйнята методика спрощується, модифікується та пристосовується до конкретних окремих потреб на даному об'єкті.

Багато функцій і процесів також не мають формалізованого опису та інструкцій для їх точного виконання. Тому до вивчення об'єкта доцільно залучити відповідних працівників підрозділів, що дасть змогу виявити особливості застосування тієї чи іншої методики в конкретних умовах, а також вимоги цих користувачів. І від того, наскільки детально та точно будуть визначені функціональні схеми об'єкта, схеми організаційної структури, інформаційні потоки та її об'ємно-часові характеристики, методики реалізації окремих процесів і їх взаємозв'язку, залежатиме подальший процес створення інформаційної системи.

Метою дослідження об'єкта є:

1) дослідження галузі його діяльності;

2) виявлення об'єктів і характеру існуючих інформаційних потоків, взаємозв'язків як усередині об'єкта, так і з зовнішнім світом;

3) визначення інформаційних потреб об'єкта;

4) встановлення організаційних, технічних і технологічних перед-умов для введення інформаційної системи;

5) побудова нових інформаційних моделей.

У процесі підготовки обстеження ознайомлюються з вхідними матеріалами та документацією для створення інформаційної системи; визначають цілі обстеження; планують обстеження; організують робочі групи; вибирають чи розробляють інструктивно-методичні матеріали для проведення обстеження; збирають і аналізують дані про зарубіжні та вітчизняні аналоги. Змістовно роботи підготовки можна подати у вигляді наступної послідовності:

1. Підготувати і затвердити керівництвом рішення про централізоване планування даних. Визначити склад групи проведення аналізу.

2. Визначити функціональні сфери організації (підприємства).

3. Скласти структурну схему організації «підрозділи - функціональні сфери».

4. Оцінити обсяг аналізу і підготувати графік аналізу.

5. Розробити приклади функцій, дій і об'єктів.

6. Зв'язатися з керівниками функціональних служб. Вибрати аналітиків серед користувачів.

7. Навчити аналітиків-користувачів.

Дослідження можуть проводитися в трьох напрямках чи при їх поєднанні з переважанням якогось із них:

1) дослідження об’єкта (його існуючого стану);

2) дослідження методичних і літературних джерел (у яких запропоновано розроблене іншими);

3) дослідження вимог користувачів.

Узагалі залежно від характеру розроблюваної інформаційної системи визначають цілі обстеження, уточнюють об’єкти в цілому та елементарні об’єкти спостереження, а також програму та організаційний план обстеження. Зміст обстеження випливає із загального циклу розробки моделі системи управління та обумовлення її вимог.

4.2. Етапи розробки вимог до інформаційних систем

Етап «Формування вимог до інформаційної системи»

Процес дослідження починається з отримання загальних знань про розвиток і функціонування об’єкта. На основі безпосереднього вивчення та аналізу зібраних матеріалів розробники будують загальну (концептуальну) модель, а потім створюють робочу модель системи управління.

На 1-му етапі «Обстеження об’єкта та обгрунтування створення інформаційної системи» виконують:

1) збирання даних про об’єкт автоматизації та види діяльності, які виконуються;

2) оцінку якості функціонування об’єкта та видів діяльності, виявлення проблем, які можна вирішити засобами автоматизації;

3) оцінку (техніко-економічну, соціальну і т.п.) доцільності створення інформаційної системи.

Вивчення системи управління передбачає такі роботи:

досліджується організаційна структура управління об'єктом, штати, фонд заробітної плати управлінського персоналу, який виконує функції планування й управління виробничо-господарскою діяльністю об`єкта;

визначаються функції та зміст робіт, які виконуються окремими підрозділами підприємства і які становлять інтерес для подальшого проектування системи;

вивчаються застосовувані методи планування, обліку, звітності, стимулювання;

вивчаються виробничі та конструкторсько-технологічні особливості підприємства (потужності устаткування і виробничих площ, наявність робочої сили, особливості технології виготовлення виробів тощо);

визначаються зв`язки даного підприємства з іншими підприємствами;

вивчається номенклатура випуску продукції та оцінка попиту на продукцію;

виконуються роботи з техніко-економічного аналізу діяльності під-

приємства для визначення обмежень на управління ним;

визначається досягнутий рівень механізації і автоматизації виробничих та управлінських процесів;

розробляються рекомендації щодо поліпшення управління на основі застосування сучасних технологічних та обчислювальних засобів.

Вивчення інформаційної системи об'єкта передбачає:

вивчення процесів формування показників і документів, а також маршрутів їхнього руху (документообігу);

одержання детальних відомостей про склад і зміст інформаційних потоків за джерелами-виникнення, періодичністю, напрямком руху, частотою періодів даних, місткості окремих повідомлень, об'ємом і щільністю, ступенем взаємозв'язку і постійністю інформації, за видами носіїв тощо;

виявлення інформаційних зв'язків між економічними розрахунками;

з'ясування методів і прийомів обробки інформації, алгоритмів розрахунків, що існують;

визначення обсягів і роботомісткості обробки економічної інформації; виявлення потреби в інформації різних підрозділів, а також ступінь задоволення цієї потреби на момент обстеження.

На стадії обстеження об'єкта виконуються роботи з технічного аналізу наявності засобів організаційної та очислювальної техніки на підприємстні, іхні експлуатаційні можливості. З'ясовується необхідність придбання додаткових сучасних обчислювальних засобів.

Під час виконання усіх перерахованих робіт застосовуються різні методи їх організації.

На 2-му етапі «Формування вимог користувача до інформаційної системи» здійснюють:

1) підготовку вихідних даних для формування вимог до інформаційної системи (характеристика об’єкта автоматизації, опис вимог до системи, межі витрат на розробку, введення в дію, очікувана ефективність системи, умови створення і функціонування системи);

2) формулювання та оформлення вимог користувача до інформаційної системи.

На 3-му етапі «Оформлення звіту про виконану роботу та заявки на розробку інформаційної системи (тактико-технічне завдання)» оформлюють звіт про виконані роботи на даній стадії і заявки на розробку інформаційної системи, тактико-технічного завдання чи іншого документа з аналогічним змістом.

Етап «Розробка концепції інформаційної системи»

На етапах 1 «Вивчення об’єкта» і 2 «Проведення необхідних науково-дослідних робіт» організація-розробник детально вивчає об’єкт автоматизації та проводить необхідні науково-дослідні роботи, пов’язані з

пошуком шляхів і оцінкою можливості реалізації вимог користувача, оформлює та затверджує звіт про науково-дослідні роботи.

На етапі 3 «Розробка варіантів концепції інформаційної системи та вибір варіанта концепції інформаційної системи, який задовольняє вимоги користувача» в загальному випадку здійснюють:

1) розробку альтернативних варіантів концепції створюваної інформаційної системи і планів щодо їх реалізації;

2) оцінку необхідних ресурсів з їх реалізації та забезпечення функціонування;

3) порівняння вимог користувача та характеристик запропонованої системи і вибір оптимального варіанта;

4) визначення порядку оцінки якості та умов приймання системи;

5) оцінку ефективності, отриманої від системи.

На етапі 4 «Оформлення звіту про виконану роботу» здійснюються підготовка та оформлення звіту, який містить опис виконаних робіт та обгрунтування запропонованого варіанта концепції інформаційної системи.

Етап «Технічне завдання»

Технічне завдання (ТЗ) визначає вимоги до функцій, усіх видів забезпечення, які реалізуються, регламентує організацію розробки, обсяги та витрати, а також перелік компонентів і функцій, передбачених у складі кожної черги.

Черговість розробки інформаційної системи та склад черг зумовлюються важливістю комплексу функцій, який приймається для даної системи, можливістю придбання та введення в експлуатацію необхідних технічних засобів відповідного технічного рівня, підготовленістю до впровадження системи, необхідністю мінімізації сумарних витрат, створеною інформаційною базою системи, можливістю використання в наступних розробках результатів проектування та впровадження першої черги інформаційної системи.

Технічне завдання розробляють згідно з ГОСТ 34.602–89 «Техническое задание на АСУ». Воно є основним вихідним документом для розробника і замовника інформаційної системи, згідно з яким здійснюється її розробка та приймання приймальною комісією.

Порядок розробки технічного завдання відповідає ГОСТ 34.601–90. Розробка проходить у чотири етапи: розробка, оформлення, погодження та затвердження.

На першому етапі організація-розробник з участю замовника розробляє проект ТЗ на інформаційну систему на базі вимог, заявки, тактико-технічного завдання і т. ін.

Також може провадитися конкурсний вибір варіанта технічного завдання з наступною розробкою кінцевого варіанта.

При розробці ТЗ визначають вимоги до інформаційної системи, до складу науково-дослідних робіт, які виконуються на наступних стадіях створення інформаційної системи, встановлюють послідовність проведення робіт, пов'язаних з її створенням, а також розроблюють окремі ТЗ на компоненти і види забезпечення.

При розробці вимог до інформаційної системи (ГОСТ 24.104–85) уточнюють цілі створюваної інформаційної системи, детально описують функціональну структуру інформаційної системи, уточнюють склад функцій, які автоматизуються, а також вимоги до якості їх виконання, формують вимоги до тимчасового регламенту розв'язання задач і їх класифікації і до частин інформаційної системи та видів забезпечення, попередньо вибирають склад обчислювальної техніки, визначають перелік задач, які забезпечують реалізацію функцій управління, які автоматизуються.

Так, на етапі «Визначення послідовності проведення робіт зі створення інформаційної системи» визначають черговість створення інформаційної системи, склад стадій та етапів створення інформаційної системи, організації-виконавці, розробляють план-графік створення інформаційної системи, а також план організаційно-технічних заходів щодо підготовки об'єкта до введення інформаційної системи в дію. На третьому етапі необхідність погодження проекту ТЗ на ІС з органами державного нагляду та з іншими зацікавленими організаціями визначають замовник і розробник та проводять спільно. Термін погодження проекту ТЗ у кожній організації не повинен перевищувати 15 днів з дня його отримання. Рекомендується розсилати копії одразу в усі організації.

Зауваження щодо проекту ТЗ мають бути представлені з технічним обгрунтуванням. Рішення щодо зауважень повинні бути прийняті розробником і замовником до затвердження ТЗ. Якщо при погодженні проекту ТЗ виникли розбіжності між розробником і замовником чи між іншими організаціями, то складається протокол розбіжностей і конкретне рішення приймається в установленому порядку. Погодження можна оформлювати окремим документом.

На четвертому етапі затвердження ТЗ на ІС здійснюються керівництвом організацій розробника та замовника системи. ТЗ до передачі на затвердження має перевірити служба нормоконтролю організацій розробника ТЗ, а в разі потреби підтвердити метрологічна експертиза.

На стадії «Формування вимог до інформаційної системи» складають звіт за ДСТУ 3008-95 "Документація. Звіти у сфері науки і техніки. Структура і правила оформлення" та заявку на розробку інформаційної системи.

Основна частина звіту містить такі розділи (РД 50-34.698-90 "Автоматизовані системи. Вимоги до змісту документів":

1. Характеристика об'єкта та результати його функціонування.

2. Опис діючої інформаційної системи.

3. Опис недоліків діючої інформаційної системи.

4. Обгрунтування необхідності вдосконалення діючої інформаційної системи.

5. Цілі, критерії та обмеження створення інформаційної системи.

6. Функції та задачі створюваної інформаційної системи.

7. Передбачувані техніко-економічні результати створення ІС.

8. Висновки і пропозиції.

У першому розділі описують тенденції розвитку, вимоги до обсягу, номенклатури та якості результатів функціонування, а також взаємодії об'єкта із зовнішнім середовищем.

При виявленні фактичних показників функціонування визначають існуючі показники та тенденції їх зміни в часі.

Другий розділ містить опис функціональної та інформаційної структур системи, якісні та кількісні характеристики, які розкривають взаємодію її компонентів у процесі функціонування.

У третьому розділі наведено результати діагностичного аналізу, при якому оцінюють якість функціонування та організаційно-технологічний рівень системи, виявляють недоліки в організації та технології функціонування інформаційних процесів і визначають ступінь їх впливу на якість функціонування системи.

У четвертому розділі, аналізуюючи відповідність показників функціонування об'єкта висунутим вимогам, треба оцінити ступінь відповідності прогнозованих показників потрібним і визначити необхідність удосконалення інформаційної системи шляхом її автоматизації.

У п'ятому розділі сформульовано виробничо-господарські, науково-технічні та економічні цілі й критерії створення інформаційної системи, а також схарактеризовано обмеження при створенні інформаційної системи.

У шостому розділі обгрунтовано вибір переліку функцій, що автоматизуються, і комплексів задач із зазначенням послідовності їх впровадження; крім того, окреслено вимоги до характеристик реалізації функцій і задач відповідно до діючих нормативно-технічних документів, які визначають загальні технічні вимоги до інформаційної системи конкретного виду, а також додаткові вимоги до інформаційної системи в цілому та її частин, що враховують специфіку створюваної інформаційної системи.

У сьомому розділі вміщено:

1) перелік основних джерел економічної ефективності, отримуваних у результаті створення інформаційної системи (в тому числі, економія виробничих ресурсів, поліпшення якості продукції, підвищення продуктивності праці та ін.), і оцінку змін основних техніко-економічних і соціальних показників виробничо-господарської діяльності об'єкта (наприклад, номенклатура та обсяги виробництва, собівартість продукції,

рентабельність, відрахування у фонди економічного стимулювання, рівень соціального розвитку);

2) оцінку очікуваних витрат, пов'язаних із створенням і експлуатацією інформаційної системи, розподіл їх по чергах створення інформаційної системи та роках;

3) очікувані узагальнені показники економічної ефективності інформаційної системи.

У восьмому розділі виділяють такі підрозділи:

1. Висновки про виробничо-господарську необхідність і техніко-економічну доцільність створення інформаційної системи.

2. Пропозиції щодо вдосконалення організації та технології процесу діяльності об'єкта.

3. Рекомендації щодо створення інформаційної системи.

У першому підрозділі вміщено:

1) порівняння очікуваних результатів створення інформаційної системи із заданими цілями й критеріями створення інформаційної системи (за цільовими показниками і нормативними вимогами);

2) принципове вирішення питання про створення інформаційної системи (позитивне чи негативне).

У другому підрозділі містяться пропозиції щодо вдосконалення виробничо-господарської діяльності, а також організаційної і функціональної структур системи, методів діяльності, видів забезпечення інформаційної системи.

У третьому підрозділі викладено рекомендації:

1) щодо видів створюваної інформаційної системи, її сумісності з іншими інформаційними системами і частиною відповідної системи, яка не автоматизується;

2) організаційної і функціональної структури створюваної інформаційної системи;

3) складу й характеристик компонентів і видів забезпечення інформаційної системи;

4) організації використання додаткових засобів обчислювальної техніки – наявних і тих, які будуть придбані;

5) раціональної організації розробки і впровадження інформаційної системи;

6) визначення основних і додаткових, зовнішніх і внутрішніх джерел і видів фінансування та матеріального забезпечення розробки інформаційної системи;

7) забезпечення виробничих умов створення інформаційної системи;

8) інші рекомендації щодо створення інформаційної системи.

Заявку на розробку інформаційної системи складають у довільній формі. Вона містить пропозиції організації-користувача до організації-

розробника на виконання робіт зі створення інформаційної системи і його вимоги до системи, умов і ресурсів для створення інформаційної системи.

На стадії «Розробка концепції інформаційної системи» складають звіт, в основній частині якого наводять:

1) опис результатів вивчення об'єкта автоматизації;

2) опис і оцінку переваг і недоліків розроблених альтернативних варіантів концепції створення інформаційної системи;

3) порівняльний аналіз вимог користувача до інформаційної системи і варіантів концепції інформаційної системи на предмет задоволення вимог користувача;

4) обгрунтування вибору оптимального варіанта концепції і опис запропонованої інформаційної системи;

5) очікувані результати та ефективність реалізації обраного варіанта концепції інформаційної системи;

6) орієнтовний план реалізації вибраного варіанта концепції інформаційної системи;

7) необхідні витрати ресурсів на розробку, введення в дію і забезпечення функціонування інформаційної системи;

8) вимоги, які гарантують якість інформаційної системи;

9) умови приймання інформаційної системи.

Вимоги, які включаються до технічного завдання, мають відповідати сучасному рівню розвитку науки й техніки, забезпечувати випереджуючий розвиток інформаційної системи щодо кращих сучасних вітчизняних і зарубіжних аналогів. Ці вимоги не повинні обмежувати ініціативи розробника в пошуку й реалізації ним технічних рішень, спрямованих на досягнення потрібних показників.

Технічне завдання (ГОСТ 34.602–89 «Техническое задание на создание автоматизированной системы») на інформаційну систему має складатися з таких розділів: загальні відомості; призначення й цілі створення (розвитку) системи; характеристика об'єктів автоматизації; вимоги до системи; склад і зміст робіт зі створення системи; порядок контролю й приймання системи; вимоги до складу і змісту робіт з підготовки об'єкта автоматизації до введення системи в дію; вимоги до документування; джерела розробки.

Після затвердження ТЗ на інформаційну систему можуть розроблятися ТЗ на частини інформаційної системи.

Залежно від вигляду, призначення, специфічних особливостей об'єкта управління та умов функціонування інформаційної системи дозволяється доповнювати зміст розділів чи вводити додаткові розділи.

У розділі «Загальні відомості» вказують:

повне найменування системи та її умовне позначення;

код теми чи код (номер) договору;

найменування підприємства (об'єднання) розробника і замовника (користувача) системи та їхні реквізити;

перелік документів, на основі яких створюється система, ким і коли затверджені ці документи;

планові терміни початку і закінчення робіт зі створення системи;

відомості про джерела і порядок фінансування робіт;

порядок оформлення і пред'явлення замовнику результатів робіт із створення системи (її частин), з виготовлення і налагодження окремих засобів (технічних, програмних, інформаційних) і програмно-технічних (програмно-методичних) комплексів системи.

Розділ «Призначення і цілі створення (розвитку) системи» складається з таких підрозділів:

1) призначення системи;

2) цілі створення системи.

У розділі «Характеристика об'єкта автоматизації» наводять:

короткі відомості про об'єкт автоматизації чи посилання на документи, які містять таку інформацію;

відомості про умови експлуатації об'єкта автоматизації і характеристики навколишнього середовища.

Розділ «Вимоги до системи» складається із таких підрозділів:

вимоги до системи в цілому;

вимоги до функцій, які виконує система;

вимоги до видів забезпечення.

Розділ «Склад і зміст робіт зі створення (розвитку) системи» має містити перелік стадій і етапів робіт зі створення системи згідно з ГОСТ 34.601–90, терміни їх виконання, перелік організацій – виконавців робіт, посилання на документи, що підтверджують згоду цих організацій на участь у створенні системи, чи запис, який визначає відповідальність за виконання цих робіт.

У розділі «Порядок контролю і приймання системи» вказують:

види, склад, обсяг і методи випробувань системи та її складових частин;

загальні вимоги до приймання робіт по стадіях, щодо порядку погодження і затвердження приймальної документації;

статус приймальної комісії.

У розділі «Вимоги до складу і змісту робіт з підготовки об'єкта автоматизації до введення системи в дію» необхідно навести перелік основних заходів, яких треба вжити в процесі підготовки об'єкта до введення інформаційної системи в дію, та їх виконавців.

У розділі «Вимоги до документування» наводять:

погоджений розробником і замовником системи перелік комплектів і видів документів, які підлягають розробці, перелік документів, що

випускаються на машинних носіях, вимоги до мікрофільмування документації;

вимоги щодо документування комплектуючих елементів міжгалузевого використання згідно з вимогами ECTS;

при відсутності державних стандартів, які визначають вимоги до документування елементів системи, додатково включають вимоги до складу і змісту таких документів.

У розділі «Джерела розробки» потрібно перелічити документи і інформаційні матеріали, на базі яких розроблювалось ТЗ і які необхідно використати в процесі створення системи.

До складу ТЗ включають додатки: розрахунок очікуваної ефективності системи, оцінку науково-технічного рівня системи.

Формальні специфікації:

Так звані формальні методи розробки програмних систем широко не використовуються. Багато компаній, які розробляють програмне забезпечення, не вважають економічно вигідним застосування цих методів в процесі розробки.

Термін "формальні методи" має на увазі ряд операцій, до складу яких входять створення формальної специфікації системи, аналіз і доказ специфікації, реалізація системи на основі перетворення формальної специфікації до програм і верифікація програм. Всі ці дії залежать від формальної специфікації програмного забезпечення. Формальна специфікація - це системна специфікація, записана на мові, словник, синтаксис і семантика якого визначені формально. Необхідність формального визначення мови передбачає, що ця мова ґрунтується на математичних концепціях. Тут використовується область математики, яка називається дискретною математикою і ґрунтується на алгебрі, теорії множин та алгебри логіки.

В інженерії ПО визначені три рівні специфікації програмного забезпечення. Це призначені для користувача і системні вимоги і специфікація структури програмної системи. Призначені для користувача вимоги найбільш узагальнені, специфікація структури найбільш детальна. Формальні математичні специфікації знаходяться десь між системними вимогами і специфікацією структури. Вони не містять деталей реалізації системи, але повинні представляти її повну математичну модель.

Розробка специфікація і проектування можуть виконуватися паралельно, коли інформація від етапів розробки специфікації передається до етапів проектування і навпаки.

Рисунок 22 – Процес розробки формальної специфіки

Створення формальної специфікації вимагає детального аналізу системи, який дозволяє виявити помилки і невідповідності в специфікації неформальних вимог. Проблеми у вимогах, які залишаються невиявленими до останніх стадій процесу розробки ПО, зазвичай вимагають великих витрат на виправлення.

Існує два основні підходи до розробки формальної специфікації, які використовуються для написання детальних специфікацій нетривіальних програмних систем.

1. Алгебраїчний підхід, при якому система описується в термінах операцій і їх відносин.

2. Підхід, орієнтований на моделювання, при якому модель системи будується з використанням математичних конструкцій, таких, як безлічі і послідовності, а системні операції визначаються тим, як вони змінюють стану системи.

Для розробки формальних специфікацій послідовних і паралельних систем в даний час створено кілька мов, представлених нижче в таблиці.

Таблиця 2 – Мови розробки формальних специфікацій

Тип мови	**Послідовні системи**	**Паралельні системи**
Алгебраїчний	Larch, OBJ	Lotos
Оснований на моделях	Z, VDM, B	CSP, мережі Петрі

Великі системи зазвичай розбиваються на підсистеми, які розробляються незалежно один від одного. Підсистеми можуть використовувати інші підсистеми, тому необхідною частиною процесу специфікування є визначення інтерфейсів підсистем. Якщо інтерфейси визначені й узгоджені, підсистеми можна розробляти незалежно один від одного.

Інтерфейс підсистеми часто визначається як набір абстрактних типів даних і об'єктів, при цьому тільки через інтерфейс доступні опис даних і операції над ними. Тому специфікацію інтерфейсу підсистеми можна розглядати як об'єднання специфікацій компонентів, що в підсумку і складе опис інтерфейсу підсистеми.

Точні специфікації інтерфейсів підсистем необхідні для розробників, які пишуть програмний код, який звертається до сервісів інших підсистем. Специфікації інтерфейсів містять інформацію про те, які сервіси доступні в інших підсистемах і як отримати до них доступ.

Ясний і однозначний інтерфейс підсистем зменшує ймовірність помилок у взаєминах між ними.

Алгебраїчний підхід спочатку був розроблений для опису інтерфейсів абстрактних типів даних, де типи даних визначаються скоріше специфікаціями операцій над даними, ніж способом уявлення самих даних. Це дуже схоже на визначення класів об'єктів.

Алгебраїчний підхід до формальних специфікацій визначає абстрактний тип даних в термінах операцій над даними.

Структура специфікації об'єкта складається з чотирьох компонентів:

- Введення, де оголошується клас (sort) об'єктів. Клас - це загальна назва для безлічі об'єктів. Він зазвичай реалізується як тип даних. Введення може також включати оголошення імпорту (imports), де вказуються імена специфікацій, що визначають інші класи. Імпорт специфікацій робить ці класи доступними для використання.

- Описова частина, в якій неформально описуються операції, асоційовані з класом. Це робить формальну специфікацію простіший для розуміння. Формальна специфікація доповнює це опис, забезпечуючи однозначний синтаксис і семантику операцій.

- Частина сигнатур, в якій визначається синтаксис інтерфейсу об'єктного класу або абстрактного типу даних. Тут описуються імена операцій, кількість і типи їх параметрів, а також класи вихідних результатів операції.

- Частина аксіом, де визначається семантика операцій за допомогою створення ряду аксіом, які характеризують поведінку абстрактного типу даних. Ці аксіоми пов'язують операції створення об'єктів класу з операціями, перевіряючими їх значення.

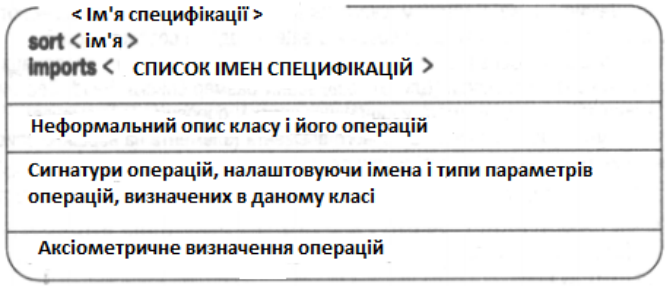


Рисунок 23 – Структура алгебраїчної специфікації

Процес розробки формальної специфікації інтерфейсу підсистеми включає наступні дії.

1. Структурування специфікації. Подання неформальній специфікації інтерфейсу у вигляді безлічі абстрактних типів даних або об'єктних класів. Також неформально визначаються операції, асоційовані з кожним класом.

2. Іменування специфікацій. Задаються імена для кожної специфікації абстрактного типу, визначаються параметри специфікацій (якщо вони необхідні) і імена визначаються класів.

3. Визначення операцій. На підставі списку виконуваних інтерфейсом функцій для кожної специфікації визначається пов'язаний з нею набір операцій. Необхідно передбачити операції по створенню примірників класів, по зміні значень примірників класів і по перевірці цих значень. Ймовірно, доведеться додати нові функції до спочатку певним списком функцій інтерфейсу.

4. Неформальна специфікація операцій. Написання неформальній специфікації для кожної операції, де має бути вказано, як операції впливають на визначений клас.

5. Визначення синтаксису операцій. Визначення синтаксису і параметрів для кожної операції. Це частина сигнатури формальної специфікації.

6. Визначення аксіом. Визначення семантики операцій шляхом опису умов, які повинні виконуватися для різних

комбінацій операцій.

Операції над абстрактним типом даних зазвичай відносяться до одного з двох класів:

1. Операції конструювання, які створюють або змінюють об'єкти класу. Зазвичай їх називають Create (Створити), Update (Змінити), Адд (Додати) або Cons (Конструювання).

2. Операції перевірки, які повертають атрибути класу. Зазвичай їм дають імена, відповідні іменам атрибута, або імена, подібні Eval (Значення), Get (Отримати) і т.п.

Хорошим емпіричним правилом для написання алгебраїчної пеціфікації є створення аксіом для кожної операції конструювання із застосуванням всіх операцій перевірки. Це означає, що якщо є m операцій конструювання і n операцій перевірки, то повинно бути визначено m x n аксіом.

Операції конструювання, пов'язані з абстрактним типом даних, часто дуже складні і можуть визначатися через інші операції конструювання та перевірки. Якщо операції конструювання визначені за допомогою інших операцій, то необхідно визначити операції перевірки, використовуючи більш примітивні конструкції.

У специфікації списку операціями конструювання є Create, Cons і Tail, які створюють списки. Операціями перевірки є Head і Length, які використовуються для отримання значень атрибутів списку.

Операція Tail не є примітивною конструкцією, тому для неї можна не визначати аксіоми з використанням операцій Head і Length, але в такому випадку Tail необхідно визначити за допомогою примітивних конструкцій.

При написанні алгебраїчних специфікацій часто використовується рекурсія. Результат операції Tail − список, сформований з вхідного списку шляхом видалення верхнього елементу. Це визначення підказує, як використовувати рекурсію для побудови даної операції. Операція визначається на порожніх списках, потім рекурсивно переходить на непусті списки і завершується, коли результатом знову буде порожній список.

Іноді простіше зрозуміти рекурсивні перетворення, використовуючи короткий приклад. Припустимо, що є список [5, 7], де елемент 5 − початок (вершина) списку, а елемент 7 − кінець списку. Операція Cons ([5, 7], 9) повинна повернути список [5, 7, 9], а операція Tail, застосована до цього списку, повинна повернути список [7, 9]. Наведемо послідовність рекурсивних перетворень, що приводить до цього результату.

Tail ([5,7,9]) =
= Tail (Cons ([5, 7], 9)) =
= Cons (Tail ([5, 7]), 9) =
= Cons (Tail (Cons ([5], 7)), 9) =
= Cons (Cons (Tail ([5]), 7), 9) =
= Cons (Cons (Tail (Cons ([], 5)), 7), 9) =
= Cons (Cons ([Create], 7), 9) =
= Cons ([7], 9) =
= [7, 9]

Тут систематично використовувалися аксіоми для Tail, що призвело до очікуваного результату. Аксіому для операції Head можна перевірити

подібним способом.

Прості алгебраїчні методи підходять для опису інтерфейсів, коли операції, асоційовані з об'єктом, що не залежать від стану об'єкта. Тоді результати будь-якої операції не залежить від результатів попередніх операцій. Якщо ця умова не виконується, алгебраїчні методи можуть стати громіздкими. Більш того, я думаю, що алгебраїчні опису поведінки систем часто штучні і важкі для розуміння.

Альтернативним підходом до створення формальних специфікацій, який широко використовується в програмних проектах, є специфікація, заснована на моделях системи. Такі специфікації використовують моделі станів системи. Системні операції визначаються за допомогою змін станів системної моделі. Таким чином визначається поведінка системи.

4.3. Модель системи. Типи системних моделей при аналізі систем

Моделі систем

Однією з широко використовуваних методик документування системних вимог є побудова ряду моделей системи. Ці моделі використовують графічні уявлення, що показують рішення як вихідної завдання, для якої створюється система, так і розроблюваної системи. Моделі є сполучною ланкою між процесом аналізу вихідної задачі і процесом проектування системи.

Моделі можуть уявити систему в різних аспектах:

- Зовнішнє уявлення, коли моделюється оточення або робоче середовище системи.
- Опис поведінки системи, коли моделюється її поведінку.
- Опис структури системи, коли моделюється системна архітектура або структури даних, які обробляються системою.

Найбільш важливим аспектом системного моделювання є те, що воно опускає деталі. Модель є абстракцією системи і легше піддається аналізу, ніж будь-яке інше уявлення цієї системи. В ідеалі уявлення системи повинно зберігати всю інформацію щодо уявного об'єкта. Абстракція є спрощенням і визначається вибором найбільш важливих характеристик системи.

Типи системних моделей, які можуть створюватися в процесі аналізу систем:

- Модель обробки даних. Діаграми потоків даних показують послідовність обробки даних в системі.
- Композиційна модель. Діаграми "сутність-зв'язок" показують, як системні сутності складаються з інших сутностей.
- Архітектурна модель. Ці моделі показують основні підсистеми, з яких будується система.
- Класифікаційна модель. Діаграми успадкування класів показують, які об'єкти мають загальні характеристики.

• Модель "стимул-відповідь". Діаграми зміни станів показують, як система реагує на внутрішні і зовнішні події.

Рисунок 24 − Приклад моделі оточення

Прості структурні моделі зазвичай доповнюються моделями інших типів, наприклад моделями процесів, які показують взаємодії в системі, або моделями потоків даних, які показують послідовність обробки і переміщення даних усередині системи і між іншими системами в навколишньому середовищі.

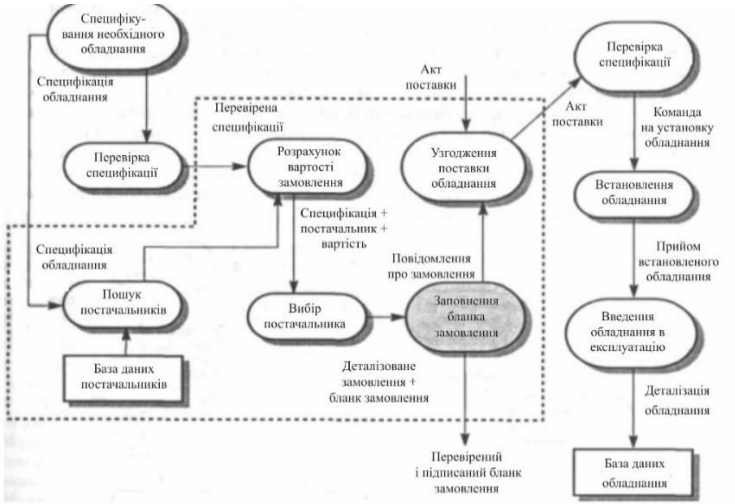


Рисунок 25 − Модель процесу придбання обладнання

Поведінкові моделі

Ці моделі використовуються для опису загальної поведінки системи.

Зазвичай розглядають два типи поведінкових моделей − модель потоків даних і модель кінцевого автомата. Ці моделі можна використовувати окремо або разом, у залежності від типу системи, що розробляється.

Моделі потоку даних − це інтуїтивно зрозумілий спосіб показу послідовності обробки даних усередині системи. Нотації, що використовуються в цих моделях, описують обробку даних за допомогою системних функцій, а також зберігання та переміщення даних між системними функціями.

У діаграмах потоків даних можуть використовуватися такі символи: закруглені прямокутники відповідають етапам обробки даних; стрілки, забезпечені примітками з назвою даних, представляють потоки даних; прямокутники відповідають сховищ або джерел даних.

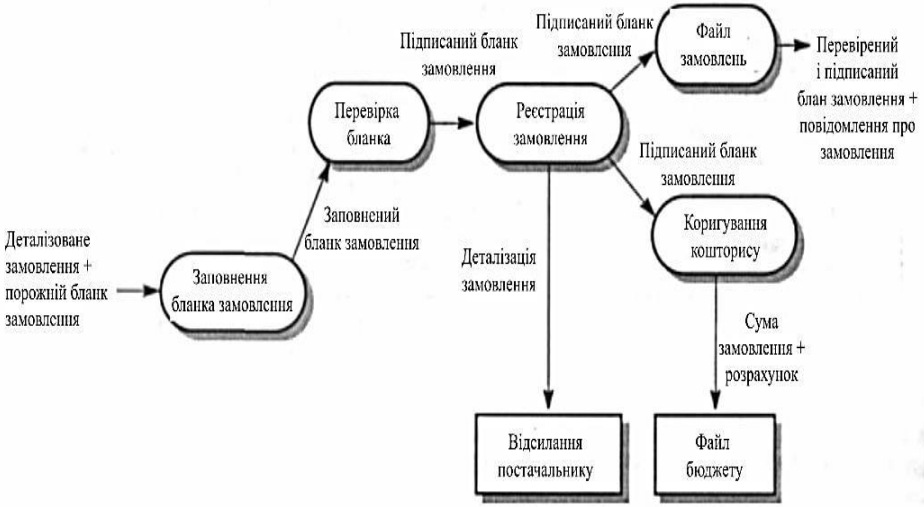


Рисунок 26 − Модель потоків даних

Моделі потоків даних показують функціональну структуру системи, де кожне перетворення даних відповідає одній системній функції. Іноді моделі потоків даних використовують для опису потоків даних в робочому оточенні системи. Така модель показує, як різні системи і підсистеми обмінюються інформацією. Підсистеми оточення не зобов'язані бути простими функціями.

Рисунок 27 − Діаграма потоків даних комплексу CASE-засобів

Моделі кінцевих автоматів використовуються для моделювання поведінки системи, що реагує на внутрішні або зовнішні події. Така модель показує стан системи і події, які служать причиною переходу системи з одного стану в інший.

Моделі кінцевих автоматів є невід'ємною частиною методів проектування систем реального часу. Такі моделі визначаються діаграмами станів, які стали основою системи нотацій в мові моделювання UML.

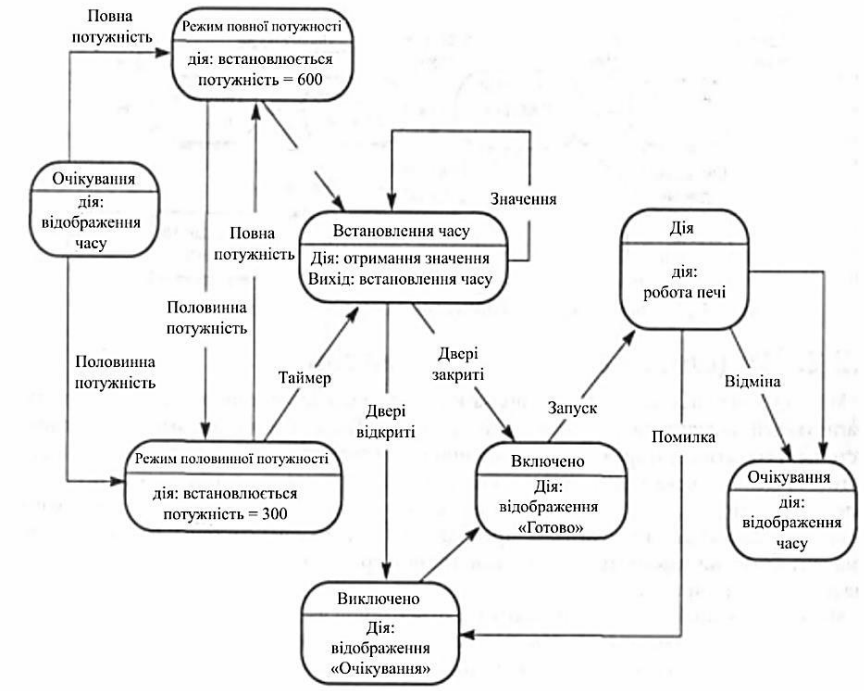


Рисунок 28 − Модель кінцевого автомату

Найбільш широко використовується методологією моделювання даних є моделювання типу "сутність-зв'язок". Для опису структури оброблюваної інформації моделі даних часто використовуються спільно з моделями потоків даних.

Проекти структури ПО представляються орієнтованими графами. Вони складаються з набору вузлів різних типів, з'єднаних дугами, що відображають зв'язки між структурними вузлами.

Подібно до всіх графічним моделям, моделі "сутність-зв'язок" недостатньо деталізовані, тому вони зазвичай доповнюються більш докладним описом об'єктів, зв'язків і атрибутів, включених в модель.

Ці описи збираються в словники даних або репозиторії.

Об'єктні моделі, можуть використовуватися як для представлення даних, так і для процесів їх обробки. В цьому відношенні вони об'єднують моделі потоків даних і семантичні моделі даних. Вони також корисні для класифікації системних сутностей і можуть представляти суті, складаються з інших сутностей.

Клас об'єктів − це абстракція безлічі об'єктів, які визначаються загальними атрибутами і сервісами (операціями).

Об'єкти − це виконувані суті з атрибутами і сервісами класу об'єктів. Об'єкти представляють собою реалізацію класу. На основі одного класу можна створити багато різних об'єктів.

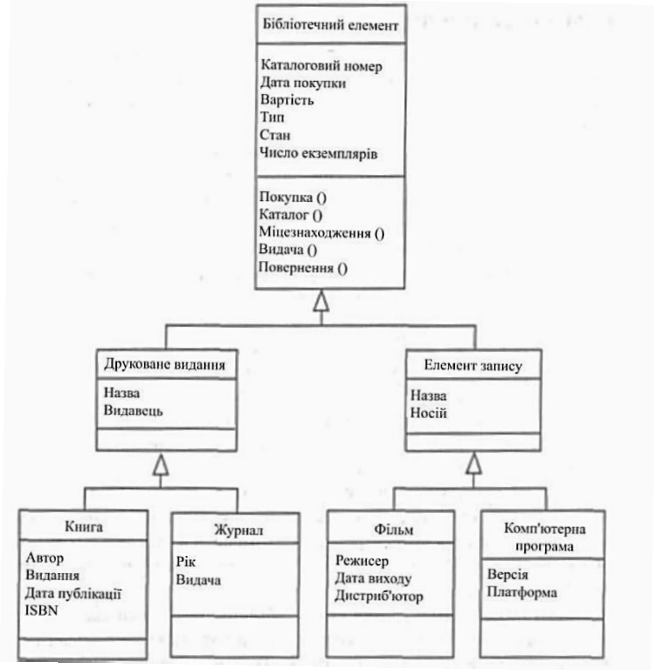


Рисунок 29 − Модель класів

Важливим етапом об'єктно-орієнтованого моделювання є визначення класів об'єктів, які потім систематизуються. Мається на увазі створення схеми класифікації, яка показує, як класи об'єктів пов'язані один з одним за допомогою загальних атрибутів і сервісів.

Схема класифікації організована у вигляді ієрархії успадкування, на вершині якої представлені найбільш загальні класи об'єктів. Більш спеціалізовані об'єкти успадковують їхні атрибути і сервіси. Ці об'єкти можуть мати власні атрибути і сервіси.

В нотації UML успадкування показуються зверху вниз. Тут стрілка (з закінченням у вигляді трикутника) виходить з класу, який успадковує атрибути з класу.

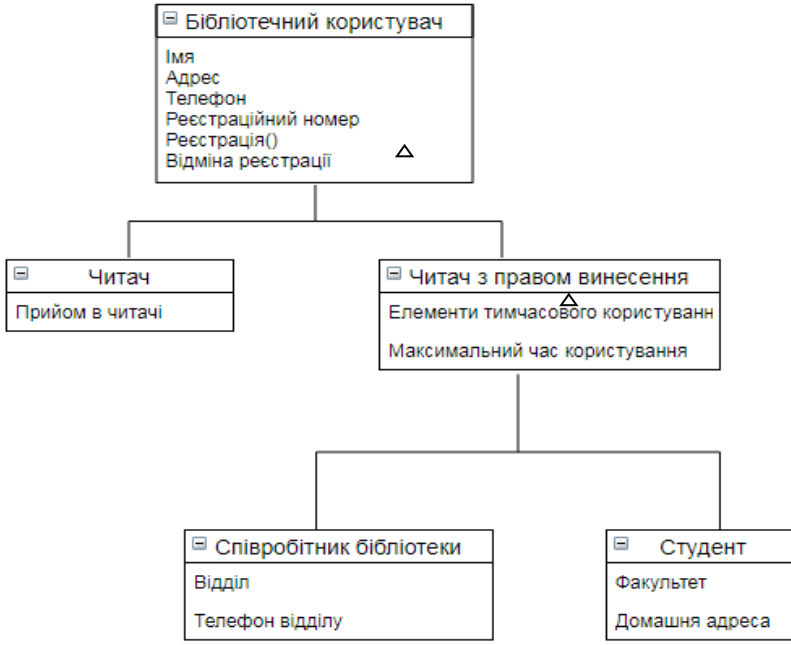


Рисунок 30 – Моделі наслідування

Об'єкти можуть створюватися з декількох об'єктів. Такий об'єкт агрегується з сукупності інших об'єктів. Класи, що представляють такі об'єкти, можна змоделювати, використовуючи модель агрегування об'єктів.

В UML для показу агрегування об'єктів використовується зв'язок з закінченням ромбовидної форми.

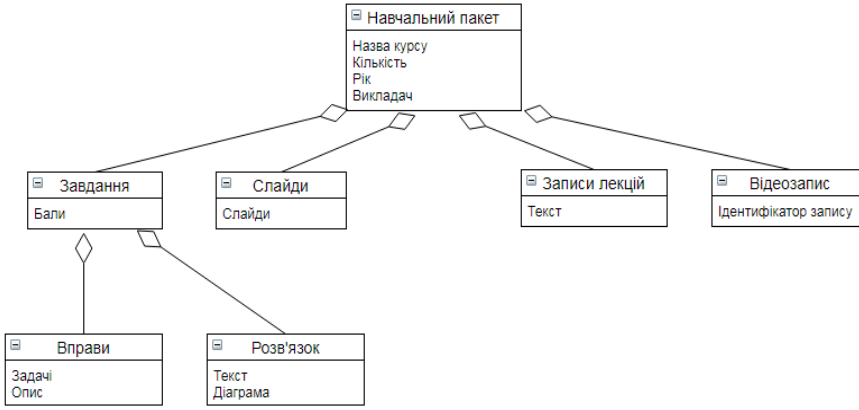


Рисунок 31 – Агрегування об’єктів

У верхній частині розташовані об'єкти. Операції позначаються поміченими стрілками, а послідовність операцій читається зверху вниз.

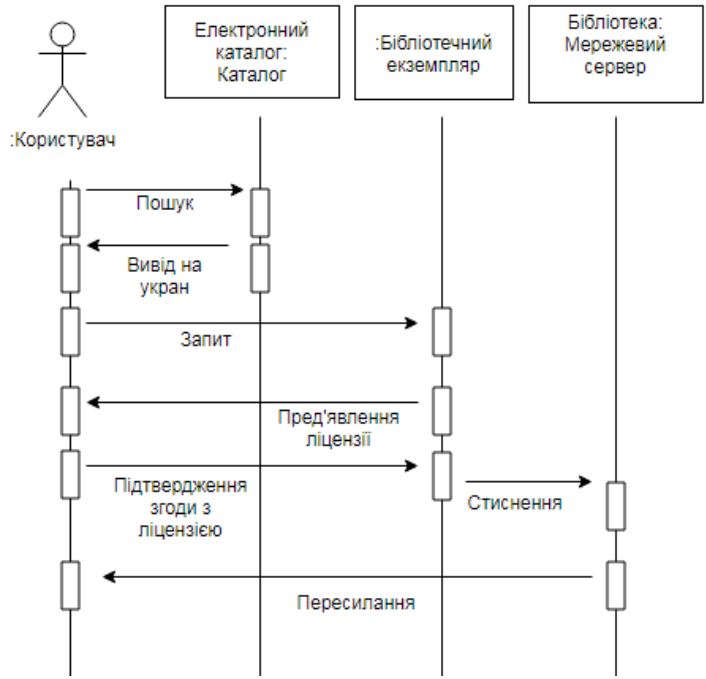


Рисунок 32 – Моделювання поведінки об’єктів

CASE-засоби проектування

Інструментальні засоби аналізу і проектування ПЗ створені для підтримки моделювання систем на етапах аналізу і проектування процесу розробки програмного забезпечення. Вони підтримують створення, редагування і аналіз графічних нотацій, що використовуються в структурних методах.

Рисунок 33 – Структура CASE – системи

Засоби, які входять в пакет інструментальних засобів:

- Редактори діаграм призначені для створення діаграм потоків даних, ієрархій об'єктів, діаграм "сутність-зв'язок" і т.д.
- Проектування ПЗ і створюють звіт про помилки і дефекти в системній архітектурі.
- Центральний репозиторій дозволяє проектувальнику знайти потрібний проект і відповідну проектну інформацію.
- Словник даних зберігає інформацію використовуються в структурі системи.
- Засоби генерування звітів на основі інформації з центрального сховища автоматично генерують системну документацію.
- Засоби створення екранних форм.
- Засоби імпортування та експортування дозволяють обмінюватися інформацією з центрального сховища різних інструментальних засобів.
- Генератори програмного коду автоматично генерують програми на основі проектів, що зберігаються в центральному репозиторії.

Питання для самоконтролю:

1. Перед вами поставлено завдання "продажу" методів формальної специфікації організації, що розробляє програмне забезпечення. Як ви будете пояснювати переваги формальної специфікації скептично налаштованим розробникам ПО?

2. Поясніть, чому необхідно визначати інтерфейси підсистем якомога точніше і чому алгебраїчна специфікація найбільш підходить для специфицирования інтерфейсів підсистем.

3. Абстрактний тип даних, який представляє стек, має наступні операції:

a. New (Створити) − створює порожній стек;

b. Push (Додати) − додає елемент в вершину стека;

c. Top (Вершина) − повертає елемент на вершині стека;

d. Retract (Видалити) − видаляє елемент з вершини стека і повертає модифікований стек;

e. Empty (Порожній) − повертає значення істини, якщо стек порожній.

Визначте цей абстрактний тип даних, використовуючи алгебраїчну специфікацію.

4. Ви системний інженер і вас просять назвати найкращий спосіб розробки програмного забезпечення для серцевого стимулятора, критичного щодо забезпечення безпеки. Ви пропонуєте розробити формальну специфікацію системи, але ваша пропозиція відкинуто менеджером. Ви вважаєте, що його доводи не обгрунтовані і базуються на упередженнях. Чи буде етичною розробка системи з використанням методів, які ви вважаєте невідповідними?

5. Розробіть модель робочого оточення для інформаційної системи лікарні. Модель повинна передбачати введення даних про нові пацієнтах і систему зберігання рентгенівських знімків.

6. Створіть модель обробки даних в системі електронної пошти. Необхідно окремо змоделювати відправку пошти та її отримання.

7. Розробіть модель класів об'єктів для системи електронної пошти. Якщо ви виконали вправу, опишіть відмінності і подібності між моделлю обробки даних і об'єктної моделлю.

8. Намалюйте модель кінцевого автомата керуючої системи для телефонного автовідповідача, який реєструє вхідні повідомлення і показує число прийнятих повідомлень на дисплеї. Система повинна з'єднувати власника телефону з абонентом після введення їм послідовності чисел (телефонного номера абонента), а також, маючи записані повідомлення, повторювати їх по телефону.

РОЗДІЛ 5. СТАНДАРТИ ТА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНИХ СИСТЕМ

5.1. Архітектурне проектування

Архітектурним проектуванням називають перший етап процесу проектування, на якому визначаються підсистеми, а також структура управління і взаємодії підсистем.

Метою архітектурного проектування є опис архітектури програмного забезпечення. Модель системної архітектури часто є відправною точкою для створення специфікації різних частин системи. В процесі архітектурного проектування розробляється базова структура системи, тобто визначаються основні компоненти системи та взаємодії між ними.

Підсистема - це система (тобто задовольняє "класичного" визначенню "система"), операції (методи) якої не залежать від сервісів, що надаються іншими підсистемами. Підсистеми складаються з модулів і мають певні інтерфейси, за допомогою яких взаємодіють з іншими підсистемами.

Модуль - це зазвичай компонент системи, який надає один або кілька сервісів для інших модулів. Модуль може використовувати сервіси, підтримувані іншими модулями. Як правило, модуль ніколи не розглядається як незалежна система. Модулі зазвичай складаються з ряду інших, більш простих компонентів.

Етапи, загальні для всіх процесів архітектурного проектування:

1.Структурування системи. Програмна система структурується у вигляді сукупності відносно незалежних підсистем. Також визначаються взаємодії між підсистемами.

2.Моделювання управління. Розробляється базова модель управління взаємовідносинами між частинами системи.

3.Модульна декомпозиція. Кожна певна на першому етапі підсистема розбивається на окремі модулі. Тут визначаються типи модулів і типи їх взаємозв'язків.

Результатом процесу архітектурного проектування є документ, що відображає архітектуру системи. Він складається з набору графічних схем уявлень моделей системи з відповідним описом. В описі повинно бути вказано, з яких підсистем складається система і з яких модулів складається кожна підсистема. Графічні схеми моделей системи дозволяють поглянути на архітектуру з різних сторін.

Як правило, розробляється чотири архітектурні моделі:

1.Статична структурна модель, в якій представлені підсистеми або компоненти, що розробляються в подальшому незалежно.

2.Динамічна модель процесів, в якій представлена організація процесів під час роботи системи.

3.Інтерфейсна модель, яка визначає послуги, що надаються кожною підсистемою через загальний інтерфейс.

4.Моделі відносин, в яких показані взаємини між частинами системи, наприклад потік даних між підсистемами.

Моделі архітектури можуть залежати від не функціональних системних вимог:

1. Продуктивність. Якщо критичним вимогою є продуктивність системи, слід розробити таку архітектуру, щоб за всі критичні операції відповідало як якнайменше підсистем з максимально малим взаємодією між ними. Щоб зменшити взаємодію між компонентами, краще використовувати крупно модульних компоненти, а не дрібні структурні елементи.

2. Захищеність. В цьому випадку архітектура повинна мати багаторівневу структуру, в якій найбільш критичні системні елементи захищені на внутрішніх рівнях, а перевірка безпеки цих рівнів здійснюється на більш високому рівні.

3. Безпека. В цьому випадку архітектуру слід спроектувати так, щоб за всі операції, що впливають на безпеку системи, відповідало якомога менше підсистем. Такий підхід дозволяє знизити вартість розробки і вирішує проблему перевірки надійності.

4. Надійність. В цьому випадку слід розробити архітектуру з включенням надлишкових компонентів, щоб можна було замінювати і оновлювати їх, не перериваючи роботу системи.

5. Зручність супроводу. В цьому випадку архітектуру системи слід проектувати на рівні дрібних структурних компонентів, які можна легко змінювати. Програми, що створюють дані, повинні бути відокремлені від програм, що використовують ці дані. Слід також уникати структури спільного використання даних.

На першому етапі процесу проектування архітектури система розбивається на кілька взаємодіючих підсистем. На самому абстрактному рівні архітектуру системи можна зобразити графічно за допомогою блок-схеми, в якій окремі підсистеми представлені окремими блоками. Якщо підсистему також можна розбити на кілька частин, на діаграмі ці частини зображуються прямокутниками всередині великих блоків. Потоки даних і / або потоки управління між підсистемами позначається стрілками. Така блок-схема дає загальне уявлення про структуру системи.

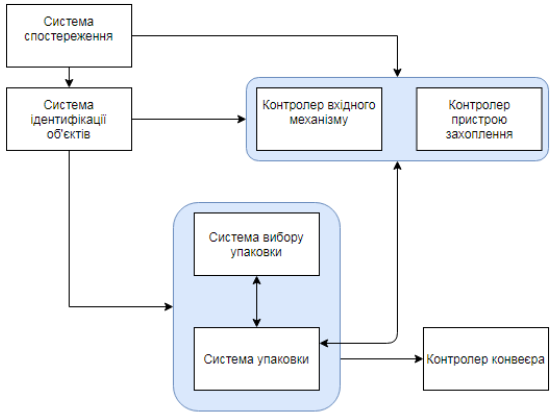


Рисунок 34 – Блок-схема системи управління автоматичним запакуванням

Для того щоб підсистеми, складові систему, працювали ефективніше, між ними повинен йти обмін інформацією. Обмін можна організувати двома способами:

1. Всі спільно використовувані дані зберігаються в центральній базі даних, доступної всіх підсистем. Модель системи, заснована на спільному використанні бази даних, часто називають моделлю сховища.

2. Кожна підсистема має власну базу даних. Взаємообмін даними між підсистемами відбувається за допомогою передачі повідомлень.

3. Модель сховища

4. Спільне використання великих обсягів даних ефективно, оскільки не потрібно передавати дані з однієї підсистеми в інші.

5. Підсистеми повинні бути узгоджені з моделлю сховища даних. Це завжди призводить до необхідності компромісу між вимогами, що висуваються до кожної підсистемі. Компромісне рішення може знизити їх продуктивність. Якщо формати даних нових підсистем не підходять під узгоджену модель представлення даних, інтегрувати такі підсистеми складно або неможливо.

6. Підсистема, в яких створюються дані, не потрібно знати, як ці дані використовуються в інших підсистемах.

7. Оскільки відповідно до узгодженої моделлю даних генеруються великі обсяги інформації, модернізація таких систем проблематична. Переклад системи на нову модель даних буде дорогим і складним, а часом навіть неможливим.

8. У системах з репозиторієм такі кошти, як резервне копіювання, забезпечення безпеки, управління доступом і відновлення даних, централізовані, оскільки входять в систему управління репозиторієм.

9. До різних підсистем пред'являються різні вимоги, що стосуються безпеки, відновлення та резервування даних. У моделі сховища до всіх підсистем застосовується однакова політика.

10.Модель спільного використання сховища прозора: якщо нові підсистеми сумісні з узгодженою моделлю даних, їх можна безпосередньо інтегрувати в систему.

11.Складно розмістити репозиторії на декількох машинах, оскільки можуть виникнути проблеми, пов'язані з надмірністю і порушенням цілісності даних.

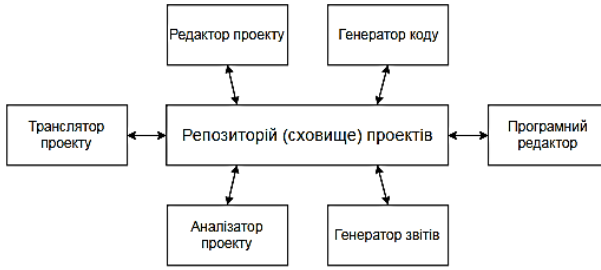


Рисунок 35 – Архітектура інтегрованого набору CASE - засобів

Модель клієнт/сервер

Модель архітектури клієнт / сервер - це модель розподіленої системи, в якій показано розподіл даних і процесів між декількома процесорами. Модель включає три основних компоненти:

1. Набір автономних серверів, що надають послуги іншим підсистемам.

2. Набір клієнтів, які викликають послуги, що надаються серверами. У контексті системи клієнти є звичайними підсистемами. Допускається паралельне виконання декількох екземплярів клієнтської програми.

3. Мережа, за допомогою якої клієнти отримують доступ до сервісів.

Найбільш важлива перевага моделі клієнт / сервер полягає в тому, що вона є розподіленою архітектурою. Її ефективно використовувати в мережевих системах з безліччю розподілених процесорів. В систему легко додати новий сервер і інтегрувати його з іншою частиною системи або ж оновити сервери, які не впливаючи на інші частини системи.

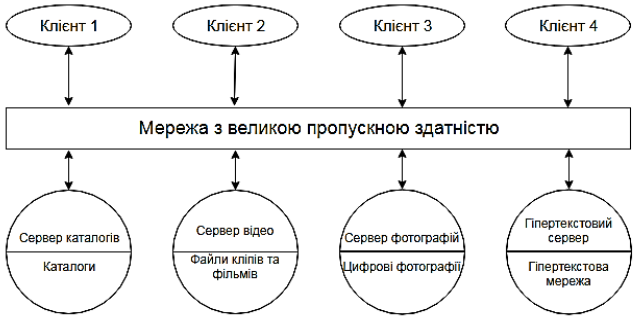


Рисунок 36 – Архітектура бібліотечної системи фільмів і фотографій

Модель абстрактної машини

Модель архітектури абстрактної машини (іноді звана багаторівневою моделлю) моделює взаємодію підсистем. Вона організовує систему у вигляді набору рівнів, кожен з яких надає свої сервіси. Кожен рівень визначає абстрактну машину, машинний мову якої (послуги, що надаються рівнем) використовується для реалізації наступного рівня абстрактної машини.

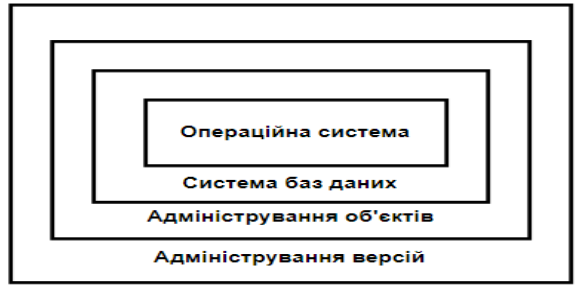


Рисунок 37 – Модель абстрактної машини для системи адміністрування версій

У структурних моделях немає (і не повинно бути) ніякої інформації по управлінню. Однак розробник архітектури повинен організувати підсистеми згідно деякої моделі управління, яка доповнювала б наявну модель структури. У моделях управління на рівні архітектури проектується потік управління між підсистемами.

Можна виділити два основних типи управління:

1. Централізоване управління. Одна з підсистем повністю відповідає за управління, запускає і завершує роботу інших підсистем.

2. Управління, засноване на подіях. Тут замість однієї підсистеми,

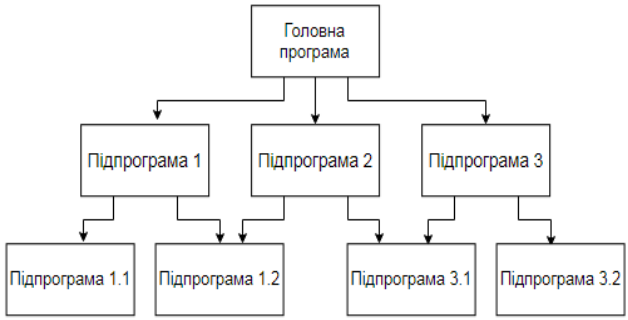
відповідальної за управління, на зовнішні подій може відповідати будь-яка підсистема. Події, на які реагує система, можуть відбуватися або в інших підсистемах, або в зовнішньому оточенні системи.

Централізоване управління

У моделі централізованого управління одна з систем призначається головною і управляє роботою інших підсистем. Такі моделі можна розбити на два класи:

Модель виклику-повернення. Це відома модель організації виклику програмних процедур "зверху вниз", в якій управління починається на вершині ієрархії процедур і через виклики передається на більш нижні рівні ієрархії. Дана модель може бути застосована тільки в послідовних системах.

Модель диспетчера. Застосовується в паралельних системах. Один системний компонент призначається диспетчером і управляє запуском, завершенням і координуванням інших процесів системи. Процес може протікати паралельно з іншими процесами. Модель такого типу може бути застосована також в послідовних системах, де керуюча програма викликає



окремі підсистеми в залежності від значень деяких змінних стану.

Рисунок 38 – модель виклику – повернення

Рисунок 39 – Модель диспетчеру для системи реального часу

Управління, засноване на подіях.

У моделях централізованого управління, як правило, управління

системою визначається значеннями деяких змінних її стану. На противагу таким моделям існують системи, управління якими засновано на зовнішні події.

Моделі передачі повідомлень. У цих моделях подія являє собою передачу повідомлення всіх підсистем. Будь-яка підсистема, яка обробляє цю подію, відповідає на нього.

Моделі, керовані перериваннями. Такі моделі зазвичай використовуються в системах реального часу, де зовнішні переривання реєструються оброблювачем переривань, а обробляються іншим системним компонентом.

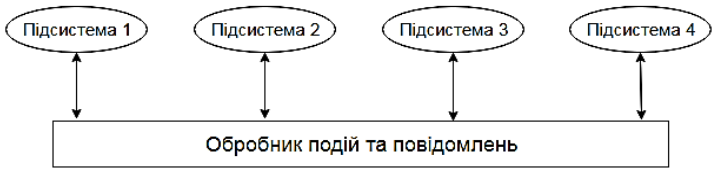


Рисунок 40 – Модель управління, заснована на передачі повідомлень

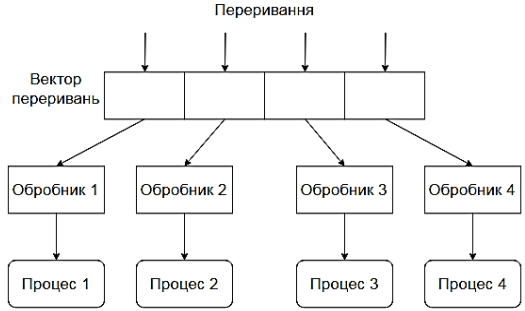


Рисунок 41 – Модель управління, заснована на перериванні

Після етапу розробки системної структури в процесі проектування слід етап декомпозиції підсистем на модулі.

Поширені дві моделі, які використовуються на етапі модульної декомпозиції підсистем:

1. Об'єктно-орієнтована модель. Система складається з набору взаємодіючих об'єктів.

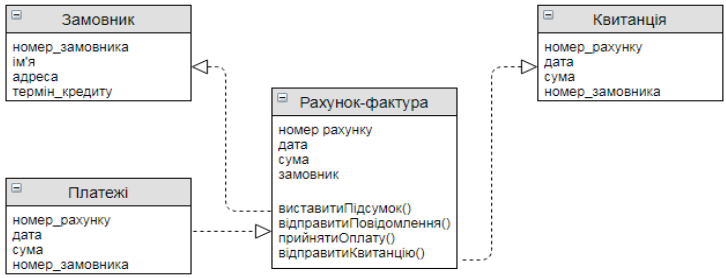
2. Модель потоків даних. Система складається з функціональних модулів, які отримують на вході дані і перетворять їх певним чином у вихідні дані. Такий підхід часто називається конвеєрним.

В об'єктно-орієнтованої моделі модулі є об'єктами з власними станами і певними операціями над цими станами. У моделі потоків даних модулі виконують функціональні перетворення.

Об’єктні моделі

Об'єктно-орієнтована архітектурна модель структурує систему у

вигляді сукупності слабо пов'язаних об'єктів з чітко визначеними інтерфейсами. Об'єкти викликають послуги, що надаються іншими



об'єктами.

Рисунок 42 – Об’єктна модель системи обробки рахунків

Моделі потоків даних

Дані проходять через послідовність перетворень. Кожен крок обробки даних реалізований у вигляді перетворення. Дані, що надходять на вхід системи, проходять через всі перетворення і досягають виходу системи. Перетворення можуть виконуватися послідовно або паралельно. Обробка даних може бути пакетної або по елементної.

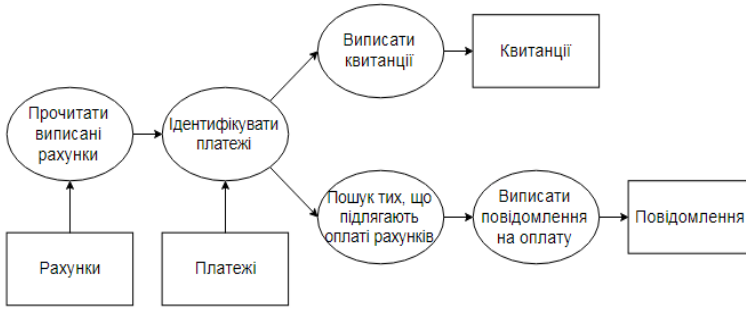


Рисунок 43 – Модель потоків даних для системи обробки рахунків

Поряд з основними моделями, використовуються архітектурні моделі, характерні для конкретної предметної області додатки. Ці моделі називаються проблемно-залежними архітектурами.

Можна виділити два типи проблемно-залежних архітектурних моделей:

1. Моделі класів систем. Відображають класи реальних систем, увібравши в себе основні характеристики цих класів. Як правило,

архітектурні моделі класів зустрічаються в системах реального часу, наприклад в системах збору даних, моніторингу і т.д.

2. Базові моделі. Більш абстрактні і надають розробникам інформацію по загальній структурі будь-якого типу систем.

Моделі класів систем

Модель компілятора найбільш відомий приклад архітектурної моделі класу систем. Компоненти, з яких складається компілятор, можна організовувати відповідно до різних архітектурними моделями. Можна застосувати архітектуру потоків даних, в якій таблиця ідентифікаторів служить сховищем спільно використовуваних даних.

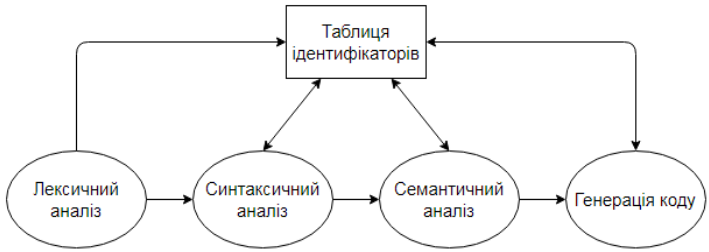


Рисунок 44 – Модель компілятора

Однак такі моделі виявляються менш ефективними, якщо компілятор інтегрований з іншими мовними засобами, наприклад системою редагування структур, інтерактивним відладчик, програмою підготовки друкованих документів і т.п. В цьому випадку компоненти системи можна організувати відповідно до моделі сховища.

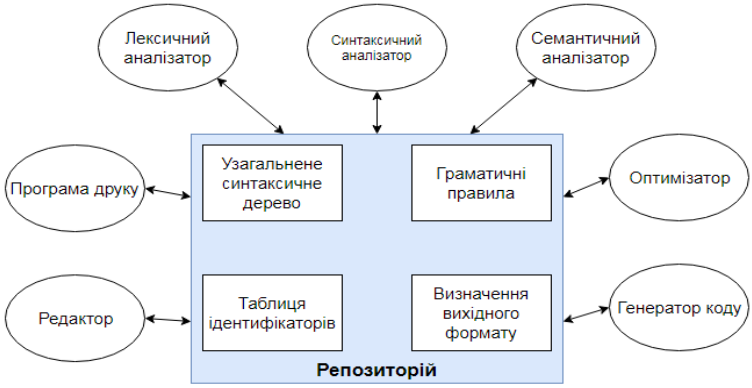


Рисунок 45 – Модель репозиторія

Базові архітектури

Базові моделі являють собою ідеалізовану архітектуру, в якій відображені особливості, властиві системам, що працюють в даній предметній області.

Прикладом базової архітектури може служити модель OSI.

Базові моделі зазвичай не розглядаються в якості методів реалізації. Їх основне призначення - служити еталоном для порівняння різних систем в будь-якої предметної області, тобто базова модель є стандартом при оцінці різних систем.

Проектування з повторним використанням компонентів.

У більшості інженерних розробок процес проектування заснований на повторному використанні вже наявних компонентів.

Повторне використання компонентів дозволить істотно скоротити витрати на розробку ПЗ. Тільки за допомогою систематичного повторного використання ПЗ можна зменшити витрати на його створення і обслуговування, скоротити терміни розробки систем і підвищити якість програмних продуктів.

Щоб повторне використання ПЗ було ефективним, його необхідно враховувати на всіх етапах процесу проектування ПЗ або процесу розробки вимог. Під час програмування можливе повторне використання на етапі підбору компонентів, які відповідають вимогам. Однак для систематичного повторного використання необхідний такий процес проектування, в ході якого постійно розглядалася б можливість повторного використання вже існуючої архітектури, де система була б явно організована з наявних компонентів ПЗ. 2. Конфігураційна спеціалізація, при якій різні версії додатка створюються для управління різними периферійними пристроями.

3. Функціональна спеціалізація, при якій створюються різні версії додатка для замовників з різними вимогами.

Процес адаптації сімейства додатків для створення нової програми складається з кількох етапів:

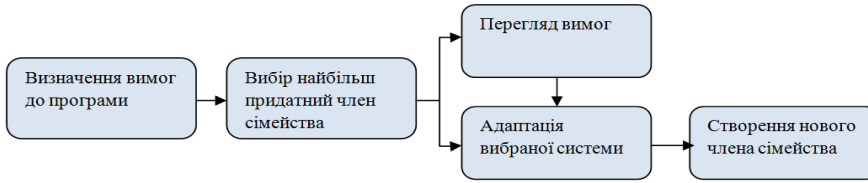


Рисунок 49 - Процес адаптації сімейства додатків

Проектні патерни

Спроби повторно використовувати діючі компоненти постійно обмежуються конкретними рішеннями, прийнятими системними

розробниками. Один із способів вирішення цієї проблеми - повторне використання більш абстрактних структур, які не містять деталей реалізації. Такі структури розробляються спеціально для того, щоб відповідати певному додатком.

Патерн - це опис проблеми та методу її рішення, що дозволяє в подальшому використовувати це рішення в різних умовах. Патерн не є детальної специфікацією. Швидше, він являє собою опис, в якому акумульовані знання і досвід. Патерн - рішення загальної проблеми.

При створенні ПЗ проектні патерни завжди пов'язані з об'єктно орієнтованим проектуванням. Щоб забезпечити загальність, патерни часто використовують такі об'єктно-орієнтовані поняття, як успадкування і поліморфізм. Однак загальний принцип використання патернів однаково застосуємо при будь-якому підході до проектування ПО.

Основні елементи проектного патерну:

1. Змістовне ім'я, яке є посиланням на патерн.

2. Опис проблемної області з перерахуванням всіх ситуацій, в яких можна використовувати патерн.

3. Опис рішень з окремим описом різних частин рішення та їх взаємовідносин. Це не опис конкретного проекту, а шаблон проектних рішень, який можна використовувати різними способами.

4. Опис "вихідних" результатів - це опис результатів і компромісів, необхідних для застосування патерну. Зазвичай використовується для того, щоб допомогти розробникам оцінити конкретну ситуацію і вибрати для неї найбільш підходящий патерн.

Часто в опис паттерна вводяться також розділи мотивації (обгрунтування корисності патерну) і застосовності (опис ситуацій, в яких можна використовувати патерн).

Як приклад розглянемо патерн Оглядач. Даний патерн використовується тоді, коли необхідні різні уявлення стану об'єкта. Він виділяє потрібний об'єкт і представляє його в різних формах:

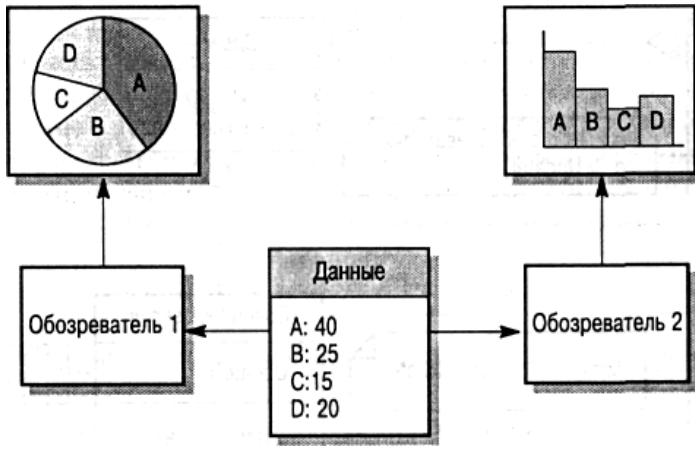


Рисунок 50 - патерн Оглядач - використовується тоді, коли необхідні різні уявлення стану об'єкта. Він виділяє потрібний об'єкт і представляє його в різних формах.

Проектування інтерфейсу користувача

Проектування обчислювальних систем охоплює широкий спектр проектних дій - від проектування апаратних засобів до проектування інтерфейсу користувача.

Організації-розробники часто наймають фахівців для проектування апаратних засобів і дуже рідко для проектування інтерфейсів.

Види інтерфейсів: текстовий і графічний

Графічні інтерфейси мають ряд переваг:

- Їх відносно просто вивчити і використовувати. Користувачі, які не мають досвіду роботи з комп'ютером, можуть легко і швидко навчитися працювати з графічним інтерфейсом.
- Кожна програма виконується в своєму вікні (екрані). Можна перемикатися з однієї програми в іншу, не втрачаючи при цьому дані, отримані в ході виконання програм.
- Режим повноекранного відображення вікон дає можливість прямого доступу до будь-якого місця екрану.

На схемі зображено ітераційний процес проектування призначеного для користувача інтерфейсу. Найбільш ефективним підходом до проектування інтерфейсу користувача є розробка із застосуванням моделювання призначених для користувача функцій.

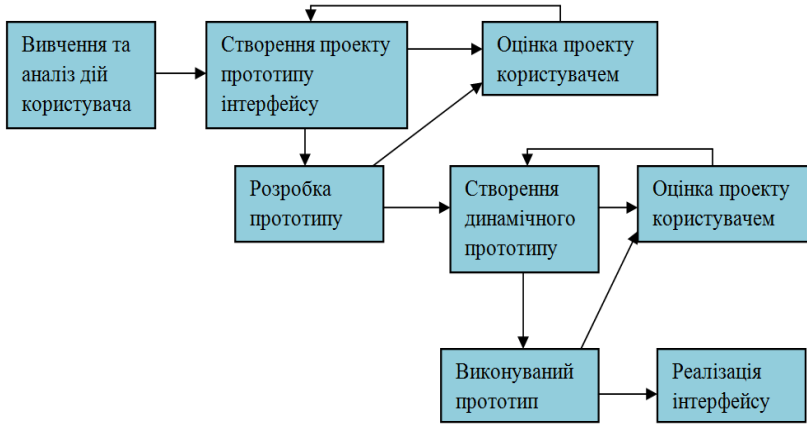


Рисунок 51 - Процес проектування інтерфейсу користувача

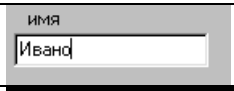
Таблиця 5 - Принципи проектування інтерфейсів

Принцип	**Опис**
Облік знань користувача	Необхідно використовувати терміни і поняття, взяті з досвіду майбутніх користувачів системи, а об'єкти, керовані системою, повинні бути безпосередньо пов'язані з робочим середовищем користувача.
Узгодженість	Команди меню системи повинні бути одного формату, параметри повинні передаватися в усі команди однаково і пунктуація команд повинна бути схожою. Зручно, коли для всіх типів об'єктів системи підтримуються однакові методи.
Мінімум несподіванок	Одне і теж дія в різних ситуаціях повинно приводити до аналогічної реакції.
Здатність до відновлення	У інтерфейсах повинні бути засоби, що запобігають помилки користувача, а також дозволяють коректно відновити інформацію після помилок. Це підтвердження деструктивних дій і можливість скасування дій.
Керівництво користувача	Інтерфейс повинен надавати необхідну інформацію в разі помилок користувача і підтримати засоби контекстно-залежною довідки.
Облік різнорідності користувачів	В інтерфейсі повинні бути засоби для зручної взаємодії з користувачами, що мають різний рівень кваліфікації і різні можливості.

Розробниками інтерфейсів передбачені 5 основних стилів взаємодії

користувача з системою.

Таблиця 6 - Стилі взаємодії з користувачем

1.	Безпосереднє маніпулювання	
2.	Вибір з меню	
3.	Заповнення форм	
4.	Командний мову	
5.	Природна мова	

Таблиця 7 - Переваги і недоліки стилів взаємодії

Стиль взаємодії	**Основні переваги**	**Основні недоліки**	**Приклади додатків**
Пряме маніпулювання	Швидка і інтуїтивно зрозуміла взаємодія Легке в вивченні	Складна реалізація. Підходить тільки там, де є зоровий образ завдань і об'єкта	Відеоігри; системи автоматичного проектування
Вибір з меню	Скорочення кількості помилок користувача.	Повільний варіант для досвідчених користувачів. Може бути складним, якщо меню складається з великої кількості вкладених пунктів	Головним чином системи загального призначення
Заповнення форм	Введення з клавіатури	Займає простір	Системи управління запасами; обробка мінімальний на екрані фінансової інформації
Командна мова	Просте введення даних.	Важка у вивченні	Операційні системи; бібліотечні системи
Природна мова	Легкий в вивченні, потужний і гнучкий.	Підходить недосвідченим користувачам. легкий в	Складно запобігти помилкам введення Вимагає великого

		налаштуваннях	ручного набору Системи розкладу; системи зберігання даних WWW

Модель з розділеними інтерфейсом командного мови і графічним інтерфейсом лежить в основі деяких операційних систем, зокрема Linux.

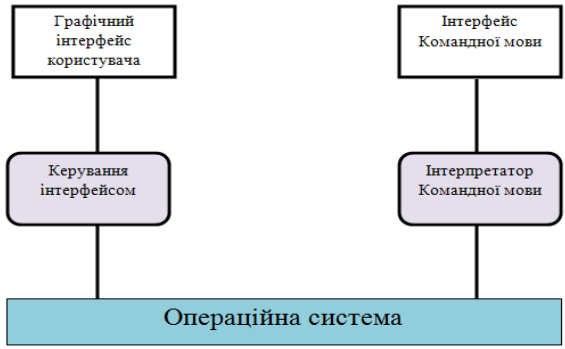


Рисунок 52 - Модель з розділеними інтерфейсом

За допомогою візуальних засобів інформацію можна представляти графічно, наприклад у вигляді графіків і діаграм

Приймаючи рішення за поданням даних, розробник повинен враховувати ряд факторів:

- Що потрібно користувачеві: точні значення даних або співвідношення між значеннями?
- Наскільки швидко будуть відбуватися зміни значень даних? Чи потрібно негайно показувати користувачеві зміна значень?
- Чи повинен користувач робити які-небудь дії у відповідь на зміну даних?
- Чи потрібно користувачеві взаємодіяти з відображається інформацією за допомогою інтерфейсу з прямим маніпулювання
- Інформація повинна відображатися в текстовому (описово) або числовому форматі? Чи важливі відносні значення елементів даних?

Альтернативи

Часто візуальне представлення інформації наочніше, ніж табличний аналог.

Використання в інтерфейсах кольору:

- Використовуйте обмежена кількість квітів
- Використовуйте різні кольори для показу змін в стані системи

- Для допомоги користувачу використовуйте колірне кодування
- Використовуйте колірне кодування продумано і послідовно
- Обережно використовуйте доповнюють кольору

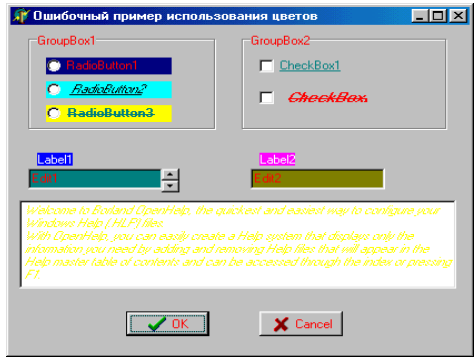


Рисунок 54 - Приклад неправильного використання кольорів

Засоби підтримки користувача

Одним з основних аспектів проектування користувача є довідкова система. Довідкову систему додатки складають:

- повідомлення, що генеруються системою у відповідь на дії користувача;
- діалогова довідкова система;
- документація, що поставляється з системою.

Таблиця 8 - Фактори проектування текстових повідомлень

Фактор	Опис
Зміст	Довідкова система повинна знати, що робить користувач, і реагувати на його дії повідомленнями відповідного змісту.
Професійний рівень користувача	Повідомлення повинні містити відомості, що відповідають професійному рівню користувачів.
Досвід користувача	У той же час початківцям користувачам такі повідомлення здадуться складними, мало зрозумілими і дуже короткими.
Стиль повідомлень	Повідомлення повинні мати позитивний, а не негативний відтінок. використовувати активний, а не пасивний тон звернення. Не повинно бути образ та спроб пожартувати.

Культура	Розробник повідомлень повинен бути знайомий з культурою тієї країни, де продається система. Повідомлення, цілком доречне в культурі однієї країни, може виявитися неприйнятним в іншій.

Перше враження користувача про систему засновано на повідомленнях про помилки, вони повинні:

- Бути послідовними і конструктивними
- Бути ввічливими, короткими, не містити образ.
- Чи не містять звукових сигналів, які можуть збити з пантелику відвідувачів.

Бажано:

- Зв'язати повідомлення з контекстно-залежною довідкою.
- Включити в повідомлення варіанти виправлення помилок.

У зв'язку з тим, що довідкова система має ієрархічну структуру, де на верхніх рівнях ієрархії міститься більш повна інформація, а на нижніх - більш докладна, може виникнути така ситуація: користувач заходить в систему отримавши повідомлення про помилку і потім переміщається в системі по посиланнях. Через деякий час він заплутується і йому необхідно починати все спочатку. Щоб таких ситуацій не виникало інформацію зручно відображати в декількох вікнах.

Тексти довідкової системи повинні бути:

- Написані спільно з творцями програми.
- Продумано так, щоб його можна було прочитати у вікні малого розміру (тільки необхідна інформація).
- Адаптовані до недосвідченого користувача.

Документація користувача повинна містити 5 документів:

- Функціональний опис, в якому коротко представлені функціональні можливості системи. Прочитавши функціональний опис, користувач повинен визначити, чи це система, яка йому потрібна.
- Документ по інсталяції системи, в якому міститься інформація по установці системи.
- Довідник з основних функцій, що представляє неформальне введення в систему, що описує її "повсякденне" використання.
- Довідник, в якому описані можливості системи і їх використання, представлений список повідомлень про помилки і можливі причини їх появи, розглянуті способи відновлення системи після виявлення помилок.
- Керівництво адміністратора, необхідне для деяких типів

програмних систем. У ньому дано опис повідомлень, що генеруються системою при взаємодії з іншими системами, і описані способи реагування на ці повідомлення.

Рисунок 56 - Документація користувача

Разом з перерахованими посібниками необхідно надавати іншу зручну в роботі документацію. Для досвідчених користувачів системи зручні різного виду предметні покажчики, які допомагають швидко переглянути список можливостей системи і способи їх використання.

Оцінювання інтерфейсу

Це частина загального процесу тестування і атестації систем ПО, в якому оцінюється зручність використання і ступінь відповідності інтерфейсу вимогам користувача.

Таблиця 10 - Показники зручності використання.

Показник	**Опис**
Вивчення	Кількість часу навчання, необхідне для початку продуктивної роботи.
Швидкість роботи	Швидкість реакції системи на дії користувача.
Стійкість	Стійкість системи до помилок користувача.
Відновлення	Здатність системи відновлюватися після помилок користувача.
Адаптованість	Здатність системи "підлаштовуватися" до різних стилів роботи користувача.

Існують прості і не дорогі методики оцінювання, що дозволяють виявити окремі дефекти в інтерфейсах.

• Анкети, в яких користувачі оцінюють інтерфейс. Ці відомості дають можливість розробникам зафіксувати, користувачі з яким рівнем знань мають проблеми з інтерфейсом.

• Спостереження за роботою користувачів. Дозволяють відслідковувати, які використовуються сервіси, що здійснюють помилки, як користувачі взаємодіють з системою.

• Відеоспостереження типового використання системи. Може виявитися корисним для виявлення проблем, але для уточнення

використовуються інші методи оцінювання.

• Додавання в систему програмного коду, який збирав би інформацію про найбільш часто використовуваних системних сервісах і найбільш розповсюджені помилки. Сприяє зміні інтерфейсу так, щоб доступ до найбільш часто використовується операціями був мінімальний.

5.2. Стандартизація розробки інформаційних систем

Розвиток будь-якої галузі економіки обов'язково супроводжується формалізацією використовуваних підходів та появою стандартів різного рівня. На ранніх етапах окремі підприємства формалізують внутрішні процеси, щоб запезпечити повторюваність результатів процесу або створення певного продукту. Для полегшення взаємодії підприємств та зручності споживачів розробляються *галузеві стандарти*. Розвиток кожного виду господарської діяльності приводить до потреби у державних засобах забезпечення якості продукції або процесу, тому розробляються та затверджуються *державні стандарти*. Для поліпшення умов співробітництва, розроблення загальнозрозумілих правил конкуренції на міжнародному ринку створюються об'єднання галузевих органів стандартизації, результатом діяльності яких є регіональні стандарти (діють у обмеженому переліку держав, які приєдналися) або *міжнародні стандарти*.

Одним із перших стандартів, що мав істотний вплив на розвиток теорії проектування та розроблення ІС, був стандарт BSP (Business System Planning). Даний стандарт був розроблений компанією IBM у середині 70-х рр. XX ст. Процес BSP передбачав виділення в ході розроблення ІС таких кроків: отримання підтримки керівництва, визначення процесів підприємства, визначення класів даних, проведення інтерв'ю, обробка та організація результатів інтерв'ю. Найважливіші кроки процесу BSP спостерігаються у більшості формальних методик.

На сьогодні діють такі стандарти, які регламентують процес розроблення ПЗ:

- ГОСТ 34.601-90 [7] – державний стандарт, що поширюється на автоматизовані системи і встановлює стадії та етапи їх створення. У стандарті міститься опис змісту робіт на кожному етапі.
- ISO/IEC 12207. Systems and software engineering – Software Life Cycle Processes [3] – міжнародний стандарт на процеси розроблення та організацію життєвого циклу ПЗ. Поширюється на всі види замовленого ПЗ. Стандарт не містить опису фаз, стадій та етапів.
- Guide to the Software Engineering Body of Knowledge (SWEBOK) [25] – Керівництво до зведення знань з програмної інженерії – галузевий стандарт Інституту інженерів з радіоелектроніки та електротехніки (IEEE), що систематизує основні види діяльності з програмної

інженерії.

Міжнародні стандарти ISO

Базовим стандартом розроблення ПЗ є **ISO 12207. Systems and software engineering – Software Life Cycle Processes**, в якому

усі процеси ЖЦ ПЗ розподілені на три групи (рис.14).

Рисунок 14 – Процеси ЖЦ ІС відповідно до стандарту ISO 12207

Допоміжні процеси призначені для підтримки виконання основних процесів, забезпечення якості проекту, організації верифікації та тестування ПЗ. **Організаційні процеси** визначають дії та завдання замовників та розробників для керування процесами у ході проекту.

Для підтримки практичного використання стандарту ISO 12207 розроблені такі технологічні документи: Керівництво для ISO/IEC 12207 (ISO/IEC TR 24748-3:2011 Systems and software engineering - Life cycle management - Part 3: Guide to the application of ISO/IEC 12207 (Software life cycle processes)) та Керівництво з використання ISO/IEC 12207 в керуванні проектами (ISO/IEC TR 16326:2009 Systems and software engineering - Life cycle processes - Project management).

У 2002 р. був опублікований стандарт на процеси життєвого циклу систем **ISO/IEC 15288 Systems and software engineering - System life cycle processes** [26], у розробленні якого брали участь фахівці різних галузей: системної інженерії, програмування, управління якістю, людськими ресурсами, безпекою та ін. Даний документ враховує практичний досвід створення систем в урядових, комерційних, військових та академічних організаціях і може бути застосований для широкого класу систем, але його основне призначення – підтримка створення комп'ютеризованих систем. На цей час діє версія стандарту 2008 р. У стандарті ISO/IEC 15288:2008 у структурі ЖЦ виділені групи процесів за видами діяльності.

Стадії створення системи, передбачені у стандарті ISO/IEC 15288:2008, та основні результати, які повинні бути досягнуті до моменту їх завершення, наведені в таблиці 2.

Таблиця 2 – Стадії створення систем (ISO/IEC 15288)

Ном. поряд.	Стадія	Опис
1	Формування концепції	Аналіз потреб, вибір концепції та проектних рішень
2	Розроблення	Проектування системи
3	Реалізація	Створення системи
4	Експлуатація	Введення в експлуатацію та використання системи
5	Підтримка	Забезпечення функціонування Системи
6	Зняття з експлуатації	Зупинення використання, демонтаж, архівування системи

Стандарти ISO/IEC 12207 та ISO/IEC 15288 мають єдину термінологію і розроблені таким чином, щоб могли використовуватись одночасно у проекті Також потрібно відмітити, що у процесі промислового розроблення ПЗ обов'язково використовуються стандарти якості серії **ISO 9000.** Серія ISO 9000 [19] (управління якістю) містить у собі такі стандарти:

- ISO 9000-1. Керування якістю і гарантії якості. Частина 1. Посібник з вибору й використання.
- ISO 9000-2. Керування якістю й гарантії якості. Частина 2. Загальний посібник із застосування стандартів ISO 9001,
- ISO 9002 і ISO 9003.
- ISO 9000-3. Керування якістю й гарантії якості. Частина 3. Посібник із застосування стандарту ISO 9001 при розробленні, установці й супроводі ПЗ.
- ISO 9000-4. Керування якістю й гарантії якості. Частина 4. Посібник з керування надійністю програм.

Основний стандарт **ISO 9001:2009** [26] задає модель системи якості для процесів проектування, розроблення, виробництва, установки й обслуговування (продукту, системи, послуги). У моделі ISO 9000 лише

згадуються вимоги, які повинні бути реалізовані, але не говориться, як це можна зробити. Тому для побудови повноцінної системи якості за ISO, крім основної моделі ISO 9001, необхідно використовувати допоміжні галузеві та рекомендаційні стандарти.

Стандарти організації IEEE

У 1963 р. у результаті злиття Інституту радіотехніків (Institute of Radio Engineers, IRE) і Американського інституту інженерів-електриків (American Institute of Electrical Engineers, AIEE) була створена міжнародна некомерційна асоціація технічних фахівців, світовий лідер у галузі розроблення стандартів з радіоелектроніки та електротехніки Інститут інженерів з радіоелектроніки та електротехніки IEEE (Institute of Electrical and Electronics Engineers). Дана міжнародна організація об'єднує понад 400 тис. фахівців із 170 країн. IEEE здійснює інформаційну і матеріальну підтримку фахівців для організації та розвитку наукової діяльності в електротехніці, електроніці, комп'ютерній техніці та інформатиці.

Керівництво до зведення знань із програмної інженерії (SWEBOK, 2004) містить опис 10 галузей знань [25]:

Software requirements – програмні вимоги;
Software design – дизайн (архітектура);
Software construction – конструювання програмного забезпечення;
Software testing – тестування;
Software maintenance – експлуатація (підтримка) програмного забезпечення;
Software configuration management – управління конфігураціями;
Software engineering management – управління у програмній інженерії;
Software engineering process – процеси програмної інженерії; Software engineering tools and methods – інструменти та методи;
Software quality – якість програмного забезпечення.

Для кожної галузі SWEBOK містить опис ключових елементів у вигляді підобластей (subareas). Для кожної підобласті наведена декомпозиція у вигляді списку тем (topics) із їх описом.

Стандарт зрілості компанії-розробника ПЗ CMM

Говорячи про стандартизацію процесів підприємства потрібно розглянути модель зрілості технологічних процесів організації Capability Maturity Model (CMM) [28], розроблену Інститутом інженерів програмного забезпечення (Software Engineering Institute, SEI) та корпорацією Mitre під керівництвом Уоттса Хамфри (Watts Humphrey).

Методологія CMM розроблялася й розвивалася в США як засіб, що дозволяє вибирати кращих виробників ПЗ для виконання держзамовлень у першу чергу міністерства оборони. Для цього були розроблені критерії оцінки зрілості ключових процесів компанії та визначений набір дій, необхідних для їхнього подальшого вдосконалювання. У підсумку методологія виявилася надзвичайно корисною для більшості компаній, що

прагнуть якісно поліпшити існуючі процеси проектування, розроблення, тестування програмних засобів і звести керування ними до зрозумілих і легко реалізованих алгоритмів і технологій, описаних у єдиному стандарті. У подальшому ця модель переросла у методологію підвищення якості процесів підприємства Capability Maturity Model Integration (CMMI).

Застосування CMMI дозволяє поставити розроблення ПЗ на промислову основу, підвищити керованість ключових процесів і виробничу культуру в цілому, гарантувати якісну роботу й виконання проектів у строк.

Основою для створення CMM стало базове положення про те, що фундаментальна проблема "кризи" процесу розроблення якісного ПЗ полягає не у відсутності нових методів і засобів розроблення, а в нездатності компанії організувати технологічні процеси й керувати ними.

Для оцінки ступеня готовності підприємства розробляти якісний програмний продукт CMM використовує ключове поняття ***зрілість організації (Maturity)***. *Незрілою* вважається організація, у якій:

- відсутнє довгострокове й проектне планування;
- процес розроблення програмного забезпечення і його ключові моменти не ідентифіковані, реалізація процесу залежить від поточних умов, конкретних менеджерів і виконавців;
- методи і процедури не стандартизовані й не документовані;
- результат не визначений реальними критеріями, які встановлюються за запланованими показниками із застосуванням стандартних технологій і розроблених метрик;
- процес вироблення рішення відбувається стихійно, на грані мистецтва.

У цьому випадку велика ймовірність появи несподіваних проблем, перевищення бюджету або невиконання строків здачі проекту. У такій компанії, як правило, менеджери й розробники не керують процесами - вони змушені займатися поточними та спонтанними проблемами Основні ознаки *зрілої організації*:

- у компанії є чітко визначені й документовані процедури керування вимогами, планування проектної діяльності, керування конфігурацією, створення й тестування програмних продуктів, відпрацьовані механізми керування проектами;
- ці процедури постійно уточнюються й удосконалюються;
- оцінки часу, складності й вартості робіт ґрунтуються на накопиченому досвіді, розроблених метриках і кількісних показниках, що робить їх достатньо точними;
- актуалізовано зовнішні й створені внутрішні стандарти на ключові процеси й процедури;
- існують обов'язкові для всіх правила оформлення методологічної

програмної й користувальницької документації;

- технології незначно змінюються від проекту до проекту на основі стабільних і перевірених підходів і методик;
- максимально використовуються напрацьовані у попередніх проектах організаційний і виробничий досвід, програмні модулі, бібліотеки програмних засобів;
- активно апробуються і впроваджуються нові технології, виробляється оцінка їхньої ефективності.

СММ визначає п'ять рівнів технологічної зрілості компанії (рис.16), за якими замовники можуть оцінювати потенційних претендентів на підпис контракту, а розробники – удосконалювати процеси створення ПЗ.

Кожний із рівнів, крім першого, складається з декількох ключових областей процесу (Key Process Area), що містять цілі (Goal), зобов'язання щодо виконання (Commitment to Perform), можливість виконання (Ability to Perform), виконувані дії (Activit Performed), їхній вимір і аналіз (Measurement and Analysis) та перевірку впровадження (Verifying Implementation).

Таким чином, СММ фактично є комплексом вимог до ключових параметрів ефективного стандартного процесу розроблення ПЗ та засобом його постійного поліпшення. Виконання цих вимог збільшує ймовірність досягнення підприємством поставлених цілей у сфері якості.

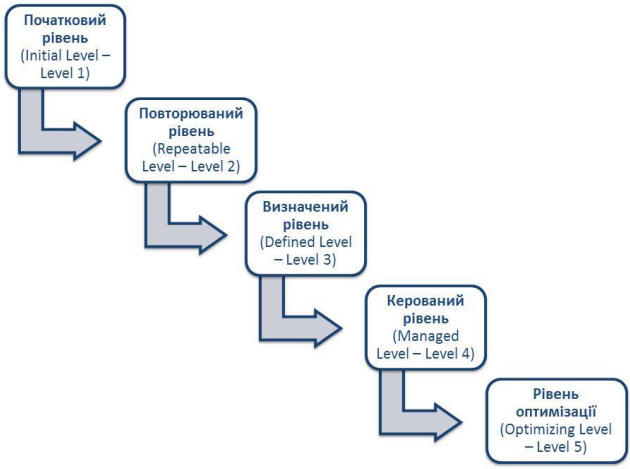


Рисунок 16 – Рівні зрілості компанії за СММ

Початковий рівень *(Initial Level – Level 1).*

На даному рівні компанія може одержати замовлення, розробити і передати замовникові програмний продукт. Стабільність розробок відсутня. Лише деякі процеси визначені, результат цілком залежить від зусиль

окремих співробітників. Успіх одного проекту не гарантує успішності наступного. До цієї категорії можна віднести будь-яку компанію, що хоч якось виконує взяті на себе зобов'язання.
Ключові області процесу цього рівня не зафіксовані.

Повторюваний рівень *(Repeatable Level – Level 2).*

Цьому рівню відповідають підприємства, які володіють певними технологіями керування й розроблення. Керування вимогами та планування у більшості випадків ґрунтуються на розробленій документованій політиці й накопиченому досвіді. Установлені й уведені в повсякденну практику базові показники для оцінки параметрів проекту. Менеджери відслідковують виконання робіт і контролюють тимчасові й виробничі витрати.

У компанії розроблені деякі внутрішні стандарти й організовані спеціальні групи перевірки якості QA. Зміни версій кінцевого програмного продукту й створених проміжних програмних засобів відслідковуються в системі керування конфігурацією. Є необхідна дисципліна дотримання встановлених правил. Ефективні методики й процеси установлюються, що забезпечує можливість повторення успіху попередніх проектів у тій самій прикладній області.
Ключові області процесу розроблення ПЗ цього рівня наведені на рис.17.

Визначений рівень *(Defined Level – Level 3).*

Рівень характеризується деталізованим методологічним підходом до керування (описані й закріплені у документованій політиці типові дії, необхідні для багаторазового повторення: ролі й відповідальність учасників, стандартні процедури й операції, порядок дій, кількісні показники й метрики процесів, формати документів та ін.).

Для створення й підтримки методологій в актуальному стані в організації підготовлена й постійно функціонує спеціальна група. Компанія регулярно проводить тренінги для підвищення професійного рівня своїх співробітників.
Починаючи з цього рівня, організація практично перестає залежати від особистісних якостей конкретних розроблювачів і не має тенденції опускатися на нижчі рівні. Ця незалежність обумовлена продуманим механізмом постановки завдань, планування заходів, виконання операцій і контролю виконання.

Управлінські й інженерні процеси документовані, стандартизовані й інтегровані в уніфіковану для всієї організації технологію створення ПЗ. Кожний проект використовує затверджену версію цієї технології, адаптовану до особливостей поточного проекту. *Ключові області* процесу розроблення ПЗ цього рівня наведені на рис.17.

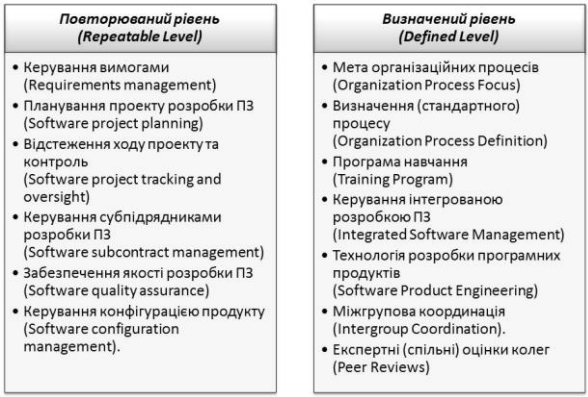


Рисунок 17 – Ключові області повторюваного та визначеного рівнів зрілості компанії

Керований рівень *(Managed Level – Level 4).*

Рівень, на якому розроблені та закріплені у відповідних нормативних документах кількісні показники якості. Більш високий рівень керування проектами досягається за рахунок зменшення відхилень різних показників проекту від запланованих. При цьому тенденції зміни продуктивності процесу можна відділити від випадкових варіацій на підставі статистичної обробки результатів вимірювань у процесах.

Ключові області процесу розроблення ПЗ цього рівня наведені на рис.18.

Рівень оптимізації *(Optimizing Level – Level 5).*

Для цього рівня заходи щодо вдосконалювання розраховані не лише на існуючі процеси, але й на запровадження, використання нових технологій і оцінку їхньої ефективності. Основним завданням усієї організації на цьому рівні є постійне вдосконалювання існуючих процесів, що в ідеалі спрямовано на запобігання відомим помилкам або дефектам і попередження можливих. Застосовується механізм повторного використання компонентів від проекту до проекту (шаблони звітів, формати вимог, процедури й стандартні операції, бібліотеки модулів програмних засобів).

Ключові області процесу розроблення ПЗ цього рівня наведені на

рис.18.

Рисунок 18 – Ключові області керованого рівня зрілості та рівня оптимізації компанії

СММ визначає такий мінімальний набір вимог: реалізувати 18 ключових областей процесу розроблення ПЗ, що містять 52 цілі, 28 зобов'язань компанії, 70 можливостей виконання (гарантій компанії) і 150 ключових практик.

У результаті аудиту та атестації компанії присвоюється певний рівень, що після наступних аудитів може підвищуватися або знижуватися. Кожний наступний рівень в обов'язковому порядку містить у собі всі ключові характеристики попередніх. У зв'язку з цим сертифікація компанії щодо одного з рівнів припускає безумовне виконання всіх вимог більш низьких рівнів.

До *переваг* моделі СММ належить те, що вона орієнтована на організації, які займаються розробленням програмного забезпечення. У даній моделі вдалося більш детально визначити вимоги, специфічні для процесів, пов'язаних з розробленням ПЗ. Із цієї причини в СММ наведені не тільки вимоги до процесів організації, але й приклади реалізації таких вимог.

Основний *недолік* СММ полягає в тому, що модель не авторизована як стандарт ні міжнародними, ні національними органами зі стандартизації. Втім, СММ давно стала промисловим стандартом. До *недоліків* моделі також необхідно віднести більші зовнішні накладні витрати на приведення процесів компанії у відповідність до моделі СММ, ніж при використанні моделей Міжнародного стандарту ISO 9000.

Питання для самоконтролю

1. Які види стандартів вам відомі? Наведіть приклади.
2. Що відповідно до стандарту СММ повинна запровадити компанія-розробник програмних систем, щоб якісно виконувати замовлення?
3. Які міжнародні стандарти ISO регулюють розроблення інформаційних систем?
4. Які міжнародні стандарти якості використовуються під час розроблення інформаційних систем?
5. Які документи відповідно до державних стандартів України розробляються у ході створення програмного забезпечення?
6. Які групи процесів у ході розроблення програмного забезпечення виділяють стандарти ISO?
7. Які стандарти організації IEEE регламентують розроблення програмних систем?
8. Які області охоплює керівництво SWEBOK?

9. Які державні стандарти України регламентують розроблення інформаційних систем та програмного забезпечення?
10. Які дії, передбачені стандартом ISO 12207, виконуються в ході розроблення програмних продуктів?
11. Які рівні зрілості компанії виділяє стандарт CMM? Чим вони характеризуються?
12. На якому рівні зрілості компанія може контролювати якість створюваного програмного забезпечення?

РОЗДІЛ 6. МУЛЬТИМЕДІЙНІ ТЕХНОЛОГІЇ ДЛЯ ОСВІТНІХ ІНФОРМАЦІЙНИХ СИСТЕМ

6.1. Використання мультимедійних технологій в начальному процесі закладів вищої освіти

Інформаційні технології — вимога сьогодення, що дозволяє створити суспільство, засно- ване на знаннях. Новітні інформаційні технології стрімко ввійшли в усі сфери нашого життя, стали такою ж реальністю, як телефонний зв'язок чи подорожування літаком. Вони спрощують спілкування та співробітництво. Суспільство, яке дбає про своє майбутнє, має усвідомити колосальні можливості, привнесені новими інформаційними технологіями, та навчитися грамотно застосовувати їх у навчально-виховному процесі.

Ведуться дискусії з приводу ефективності та доцільності використання інформаційних технологій на уроках чи заняттях у закладі вищої освіти, але не використовувати їх було б безглуздям. Можливості сучасного уроку й системи освіти значно розширюються завдяки використанню мультимедійних, інтерактивних технологій, Інтернету. Сьогодні перед педагогами стоїть важливе завдання — виховати та підготувати молодь, спроможну активно включатися в якісно новий етап розвитку сучасного суспільства, пов'язаний з інформацією. Ні для кого не є новиною, що дити- на опановує комп'ютер раніше, ніж навчається грамотно писати та читати. Згідно з сучасною концепцією навчання дедалі більше уваги приділяється оптимізації та індивідуалізації шкільної освіти.

Проблемами підвищення якості вищих навчальних дисциплін засобами мультимедійних технологій займались Н. В. Морзе, Н. П. Дементієвська, І. В. Засядько, Л. І. Скалій, Л. Й. Ястребов та інші.

В Україні використання інформаційних технологій у навчальному процесі, окрім уроків інформатики, почалося декілька років тому. До цього мало хто використовував інформацію з Інтернету з дидактичною метою. Але за останні роки наше суспільство зробило крок уперед.

Комп'ютери стрімко увійшли в різноманітні сфери нашої повсякденної діяльності, тому широке запровадження комп'ютерної техніки в процесі навчання є важливим завданням. Найо- птимальнішими є такі етапи оволодіння цими технологіями.

Необхідним є осмислення критеріїв доцільності використання мультимедійних засобів в процесі навчання:

- підвищується ефективність навчання, ніж при використанні традиційних засобів на- вчання;

- неможливість реалізації певних засобів навчання у вигляді матеріальних об’єктів
- (оригінали у природних або штучних умовах);
- недостатня наочність та зрозумілість або надлишкова складність відповідних верба- льно-знакових, графічних (статичних або динамічних), знакових, логічно- математичних моделей.

Використання мультимедійних технологій не може забезпечити суттєвого педагогічного ефекту без суб’єкта, оскільки ці технології є тільки засобами навчання, ефективність яких за- лежить від умінь їх використання викладачем для досягнення педагогічних цілей на основі глибокого вивчення можливостей їх використання. Чим вища професійна підготовка викладача, тим більша ефективність використання мультимедійних технологій.

Сьогодні у закладах вищої освіти з метою активізації пізнавальної діяльності студента потрібно використовувати усі можливі мультимедійні технології для проведення лекцій, семінарських, лабораторних і самостійних занять. Найбільш ефективними є такі мультимедійні продукти, як: інтерактивні довідники та матеріали для самоосвіти (словники, енциклопедії, ат- ласи, самовчителі різних мов тощо); освітні програми разом з іграми або освітні програми (ін- терактивні, ігрові, розважальні), мета яких — викликати інтерес і бажання пізнавати більше, підвищити якість засвоєння нового матеріалу.

Вдосконалення персональних комп’ютерів дозволяє достатньо широко використовувати мультимедійні технології, які є сукупністю різних засобів навчання: текстів, графічних зобра- жень, музики, відео і мультиплікації в інтерактивному режимі, тим самим розширюючи можливості вдосконалення навчально-виховного процесу.

Щоб оволодіти мультимедійними технологіями, доцільно ознайомити педагогів з арсена- лом дидактичних можливостей мультимедійних засобів навчання, в тому числі:

- урізноманітнення форм подання інформації; різні типи навчальних завдань;
- створення навчальних середовищ, які забезпечують «занурення» учня, студента в уявний світ, у певні соціальні й виробничі ситуації;
- широке застосування ігрових технологій;
- реалізація можливості відтворення фрагменту навчальної діяльності (предметно-
- змістового, предметно-операційного і рефлексивного);
- активізація навчальної роботи учнів, студентів, посилення їх ролі як суб’єкта навча- льної діяльності; посилення мотивації навчання.

Серед головних дидактичних функцій, що реалізовані за допомогою комп’ютерних тех- нологій у процесі вивчення навчальних дисциплін,

науковці визначають наступні: пізнавальна; розвивальна; дослідницька; комунікативна.

На це актуалізував увагу К. Д. Ушинський: «Педагог... має подбати про те, щоб якомога більше органів чуття — око, вухо, голос, чуття мускульних рухів і навіть, якщо можливо, нюх і смак — узяли участь в акті запам'ятовування. За такого дружнього сприяння всіх органів в акті засвоєння ви переможете найлінивішу пам'ять». А далі завдання за викладачем, тому що вдало підібраний дидактичний матеріал або інтерактивні засоби завершують процес ви- вчення нового матеріалу, даючи можливість виконати завдання самостійно, щоб усе зрозуміти.

Досвід свідчить, чим ширше застосовуються комп'ютери у процесі навчання (не тільки інформатики), то тим раніше учні починають працювати з комп'ютером і ефективніші резуль- тати навчання. Пізніше той самий учень стає студентом і у нього є хороша база ще зі школи во- лодіння комп'ютером. Такі студенти набагато краще пристосовуються до сучасного процесу навчання у ЗВО за допомогою новітніх технологій.

За останні роки в торговій мережі поряд з підручниками, посібниками та зошитами з'явилося чимало автоматизованих навчальних курсів з іноземних мов, які є мультимедійними продуктами. Ці програми (електронні підручники, комп'ютерні задачники, навчальні посібни- ки, гіпертекстові інформаційно-довідкові системи — архіви, каталоги, довідники, енциклопедії, тестуючі та моделюючі програми-тренажери тощо) розробляються на основі мультимедійних технологій, які виникли на стику багатьох галузей знання.

Аналіз практики використання мультимедійних продуктів (створення програм) показує, що склалося своєрідне зачароване коло: спеціалісти, зайняті змістом свого навчального курсу, не мають можливості, а часом і бажання брати участь у процесі створення мультимедійних програм. Програмісти виконують їх непрофесійно, без урахування досвіду педагогічної роботи, специфіки змістового параметру конкретного навчального матеріалу. Разом з тим, особливо для педагогів-початківців, необхідні добре структуровані та змістовно наповнені програми, готові до використання в реальному навчальному процесі. Досвідчений педагог не завжди готовий сприймати та використовувати навіть добре продумані готові програми, оскільки він, як педа- гог-професіонал, має свою концептуальну лінію, своє бачення розв'язання предметної пробле- ми.

Досвід роботи у вищій школі показує, що більшість викладачів пропонує або демонстра- цію навчального матеріалу у вигляді фрагментів, або вимушує студента працювати з величез- ним обсягом програм, створених у вигляді енциклопедій. При цьому вони написані як бази да- них або як тренажери і призначені для його індивідуальної роботи з персональним комп'ютером. Викладачу-предметнику в даному випадку

відводиться роль простого консуль- танта або пасивного спостерігача, навіть за наявності мережевих комунікацій. І такий ви- клад матеріалу не кожного викладача задовольняє.

Кожен педагог на сучасному етапі розвитку мультимедійних технологій в змозі створити сам свою презентацію (об'єднувати в одній комп'ютерній програмно-технічній системі текст, звук, відеозображення, графічне зображення та анімацію (мультиплікацію) для лекції, практич- ного заняття чи самостійної роботи, використовуючи готовий матеріал (словники, енциклопе- дії; атласи, самовчителі різних мов; електронні підручники; відео-фрагменти фізичних дослідів, вправ з фізичного виховання, побудови фігури з геометрії тощо), а також своє особисте бачення погляду, виходячи з досвіду роботи професіоналів, які чудово знають свою предметну галузь із змістової та методичної точки зору, можуть дати реальний і бажаний ефект у процесі створення навчальних програм. Такі заготовки лекцій можна зберігати, тільки завжди удосконалюючи. Перший раз створюючи таку презентацію, підготовка до лекції потребує великої затрати зусиль та часу у викладача, але в майбутньому значно полегшить йому роботу.

Найбільш доцільно використовувати мультимедійні технології на заняттях, які вимагають від викладача максимального використання наочності, а від студента постійної уваги, а також лекцій, які несуть велике теоретичне навантаження.

Зміст мультимедійного матеріалу має відповідати вимогам навчальної програми, містити не лише теоретичну, а й практичну інформацію та не перетворюватися на видовище замість навчальної роботи. Доцільно підібрані презентації можуть повністю замінити інші засоби навчан- ня, що є однією із переваг класів інформаційно-комунікаційних технологій.

Для ефективного використання мультимедійних засобів навчання викладачі повинні ви- рішили такі завдання:

- доцільність використання засобів навчання під час вивчення навчальної дисципліни; дидактичні функції кожного із засобів;
- місце мультимедійних та інших засобів у процесі вивчення всієї теми;
- послідовність викладу навчального матеріалу з допомогою мультимедійної програми.

У процесі роботи потрібно створити алгоритм підготовки до проведення навчальних за- нять з використанням мультимедійних засобів та розробити методичні рекомендації щодо створення мультимедійного супроводу занять. Необхідний матеріал викладач підбирає у біб- ліотечно-інформаційному центрі, використовує інші джерела інформації, в тому числі можли- вості мережі Internet, записує на електронні носії інформацію хронометрований відео- та аудіоматеріал. Створення різноманітних

презентацій — справа доступна. Це підтверджує той факт, що створення простих презентацій розпочинається з шкільних років.

Презентація як комп'ютерний документ є послідовністю слайдів, тобто електронних сто- рінок. Слайди створюються з використанням попередньо дібраного матеріалу для заняття.

Демонстрація такого документу може відбуватися на екрані монітору комп'ютера чи на великому екрані за допомогою спеціальних пристроїв — мультимедійного проектора, плазмен- ного екрану, мультимедійного плато, телевізора тощо. На екран із текстовою інформацією мо- жна подати різноманітну графіку (статичне зображення, малюнки, фотографії, схеми, таблиці, мультиплікація, динамічне зображення, відеофрагменти), звук (мова, музика, функціональні шуми й звуки), гіперпосилання на окремі сайти. Текст і зображення можуть бути кольоровими, супроводжуватися фонограмою та звуковими ефектами чи голосовим коментарем диктора. Ча- стіше демонстрацію презентації супроводжує доповідь окремої людини.

При демонстрації об'єкти можуть відразу відображатися на слайдах, а можуть з'являтися на них поступово, в певний час, визначений доповідачем для підсилення наочності доповіді та акцентування на особливо важливі моменти її змісту. За потреб доповідач може порушити ви- значену заздалегідь послідовність демонстрації слайдів і перейти до будь-якої сторінки в дові- льному порядку. Програма PowerPoint, що входить до пакету Microsoft Office, дозволяє ство- рювати презентації з ефектами анімації, включає можливості малювання простих об'єктів і внесення зміни до малюнків і фотографій, відображення графіків і діаграм.

За *способом подання слайдів* можна розрізняти презентації:

1. Для супроводу лекції, виступу — із записом голосу лектора чи усним супроводом.

2. Слайд-шоу — без супроводу лектора або із записаним голосом доповідача.

3. Комбінована — з усним супроводом, із записаним голосом, частиною якої може бути слайд-шоу.

Коли потрібно показати недоступні для безпосереднього спостереження явища та проце- си в розвитку й динаміці, мультимедійні засоби навчання мають безперечну перевагу над ін- шими засобами. Тому доцільно використовувати їх для фіксації уваги учня на окремих части- нах статичного матеріалу.

Кожен із застосованих видів предмета має власні виражальні засоби та дидактичні мож- ливості, що спрямовані на забезпечення оптимізації процесу навчання. Тому мультимедійні програми як своєрідний засіб навчання можуть забезпечити принципово нову якість: обмін ін- формацією між студентом і технічною системою відбувається у діалоговій формі, за

нерегла- ментованим сценарієм, який кожного разу сприймається студентом по-новому, за його розсу- дом, а сама комп'ютерна технологія навчання органічно вписується в класичну систему, розви- ває і раціоналізує її, при цьому забезпечуючи нові можливості щодо організації паралельного навчання і контролю знань, надає реальну допомогу практичному впровадженню індивідуалі- зованого навчання.

Для викладачів, які не ознайомлені з основами роботи з мультимедіа, з дидактичним осо- бливостями використання презентацій у навчальному процесі та не мають змоги ознайомитися з такими матеріалами, потрібно організовувати спеціальне навчання (семінари, тренінги).

Воно цілеспрямовано не лише на використання програми PowerPoint, а на дидактичні особливості учительських та учнівських мультимедійних презентацій, методику їх проектуван- ня та створення. Розробку критеріїв оцінювання учнівських презентацій започатковано в про- грамі «Intel®Навчання для майбутнього», яка реалізується в Україні з кінця 2004 року. Це потужний та унікальний засіб для формування в студентів вміння виступати перед аудиторією, коротко формулювати думку, структурувати доповідь, використовувати мультимедійні засоби і можливості для ілюстрування ідеї, гіпотези, висновків. В учнів чи студентів формуються навички стисло, чітко, зручно представити результати досліджень за допомогою вдало підібраних діаграм і графіків, а також відтворити найяскравіші переконливі факти для імітації думок, ідей.

На практичному занятті з використанням мультимедійних технологій викладач може продемонструвати в комп'ютерному класі з одного комп'ютера на всі, за допомогою певних програм (наприклад, NetOpSchool), деякі взірці виконання завдань, можна також через мультимедійний проектор на мультимедійну дошку вивести зміст завдання для обговорення на парі. На лабораторне заняття результат своєї роботи кожен студент може висвітлити через проектор на екран.

Мультимедійні програми використовуються як засіб впровадження самостійної роботи учнів, студентів, мультимедійна інформація — як інструктивний та ілюстративний матеріал. Також мультимедію можна використовувати у поєднанні з іншими засобами навчання. Академік С. Г. Шаповаленко так пояснював сутність та необхідність такого поєднання: «засоби на- вчання обслуговують усі моменти навчання: сприймання, осмислення, закріплення та застосу- вання інформації. Далеко не завжди один і той самий засіб може обслуговувати всі ці моменти, тому що функціональні можливості його обмежені». Тому залежно від того, яке завдан- ня необхідно розв'язати, учитель-предметник залучає ті або інші засоби. Так виникають ком- плекси засобів навчання**.**

Нові технології не тільки забезпечують викладачів та студентів новими засобами та ре- сурсами, але й змінюють самі способи комунікації

між викладачами та студентами.

Отже, впровадження мультимедійних технологій, підвищує якість освіти, активізує на- вчальну та виховну діяльність студента, виявляє творчі здібності студентів, вдосконалюється самостійна робота, забезпечується двонаправленість.

Новий підхід характеризується використанням інтерактивних методів, які забезпечують двонаправлений потік інформації викладач студент і студент студент незалежно від форми заняття.

6.2. Види та основні характеристики мультимедійних технологій навчання

Характеризуючи мультимедіа-технології, слід зазначити, що вони є розділом нових інформаційних технологій. При визначенні співвідношення понять мультимедіа і гіпермедіа-технологій встановлено, що особливістю системи гіпермедіа є використання технології гіпертексту і гіперзображення, які допускають лінійний, довільний і асоціативний перегляд відповідно до логіки побудови зв'язків. У той же час гіперсистеми можуть містити графічну, звукову, відеоінформацію. При цьому мультимедіасистема в чистому вигляді не містять гіпертехнологію. Обидві технології з'єднуються у структурі інтерактивних мультимедійних навчальних середовищ. Аналіз основних форм 20 представлення інформації, методів і засобів їх обробки дозволив визначити, що мультимедіа-технологія багатопланова, складно структурована, що вимагає для реалізації рішення безлічі завдань з різних областей людських знань.

Застосування мультимедійних технологій поділяється на:

загальне або індивідуальне користування;

для професіоналів або для рядового споживача;

для застосування інтерактивного і неінтерактивного;

для використання інформації за місцем або на відстані.

Варто більш детально зупинитися на кожному з перерахованих пунктів.

1. Технології загального або індивідуального користування. Відносно технологій загального користування можна виділити наступні види: інтерактивні термінали, деякі технології презентацій за допомогою комп'ютера, ті, що ширяться по мережах. У свою чергу, до технологій індивідуального користування можна віднести мультимедійні робочі місця, навчальні класи, мультимедійні комп'ютери для ведення різних документів. До основних місць їх застосування можна віднести громадські зони, а також будинки і робочі місця споживачів.

2. Технології для професіоналів і рядових споживачів. У цю категорію можна віднести робочі зони мультимедіа (комп'ютерна графіка, проекти і т.п.). Також сюди можуть входити системи, що застосовуються

поатківцями. Вони, як правило, використовуються в громадських місцях, це системи з вбудованими мікропроцесорами, які призначені для функціонування в побуті. Це ігрові приставки, CD-I, Play Station.

3. Використання інформації за місцем і на відстанях. Стрімкий розвиток на початковому етапі мультимедіа можна пояснити швидким процесом розвитку стаціонарних комп'ютерів, які сьогодні є будинку у кожного. Тоді стало можливим запис і зберігання інформації на спеціально призначених компакт-дисках. Сучасність диктує свої правила. Сьогоднішній стрімкий розвиток цифрових мереж середньої та високої пропускної здатності дозволяє говорити про стрімкий розвиток дистанційних мультимедійних технологій.

4. Застосування інтерактивних і неінтерактивних технологій. Підходячи до даної категорії, слід акцентувати увагу на тому, що велика кількість фахівців не згодні з тим, що неінтерактивні системи можна назвати мультимедійними. Але важливо розуміти, що їх кількість може істотно збільшитися. Так, неінтерактивні мультимедіа застосовуються для залучення уваги і розваги аудиторії за допомогою демонстрації презентацій і виставок. І.Д. Бех зазначає, що використання мультимедійних технологій в процесі професійної підготовки сучасного вчителя повинне бути комплексним і інтегрованим, охоплювати весь курс навчання і здійснюватися при викладанні різних предметів [10, с. 44].

До визначальних дидактичних особливостей (характеристик) ІКТ Т.О. Бутенко відносить такі [21, с. 10]:

комп'ютерна візуалізація і комп'ютерне моделювання навчальних відомостей про об'єкти, процеси і явища, як реальних, так і віртуальних;

зберігання великих обсягів даних і забезпечення мобільного доступу до них;

забезпечення оперативного (миттєвого) зворотного зв'язку між учасниками навчального процесу;

автоматизація обчислювальних процесів і інформаційно-пошукової діяльності;

автоматизація процесів управління навчальною діяльністю і контроль за засвоєнням навчального матеріалу. Експериментально встановлено, що при усному викладі матеріалу за хвилину слухач сприймає і здатний обробити до однієї тисячі умовних одиниць інформації, а при «підключенні» органів зору до 100 тисяч таких одиниць. Тому абсолютно очевидна висока ефективність використання в навчанні мультимедіа, основа яких - зорове і слухове сприйняття матеріалу.

Сьогодні мультимедійні технології вищої школи мають бути для студента надійним дидактичним інструментом, за допомогою якого він може самостійно набути необхідних знань з предмету, які повинні бути достатніми, досяжними, діагностованими (як викладачем, так і самим

студентом).

Актуальною проблемою, яка потребує якнайшвидшого вирішення, залишається проблема впровадження інформаційно-комунікативних технологій і мультимедіа в тому числі, при підготовці студентів. Така підготовка, підкреслює Т.Г. Величко, «повинна бути наскрізною протягом усього періоду навчання. Вона не повинна зводитися до отримання студентами операторських навичок, а повинна передбачати засвоєння інформаційних технологій навчання без впровадження сучасних технологій навчання досягнення високої якості освіти і забезпечити успішну реалізацію нового змісту середньої освіти неможливо» [28, с. 17].

Мультимедійні технології розглядаються як частина набору інструментів і вибір інструменту повинні відповідати змісту навчальних програм. З педагогічної точки зору, загальноприйнято, що мультимедійні технології є потенціалом, щоб змінити і додати новий вимір до навчання. Термін «мультимедійні технології» має багато значень, але в даному випадку використовується для позначення цифрової інформації з використанням будь-яких інтегрованих комбінацій аудіо, відео, зображення (двовимірні, тривимірні), і текст. У найбільш примітивних формах, терміном «мультимедіа» іноді називають зміст презентації з використанням комбінацій звуку, зображень (статичних, анімаційних і т.д.), відео і тексту. З цієї точки зору, будь-яка презентація, яка включає в себе використання, наприклад, відеомагнітофон і слайд-шоу можна вважати мультимедіа. Відмінною рисою цифрових мультимедійних технологій, які потрібно використовувати при підготовці студентів (на відміну від примітивних форм), є здатність підтримувати тісний контакт з користувачем.

По суті, ми можемо розрізняти дві області застосування ММТ при підготовці майбутніх вчителів [32]:

1. Усередині закладу вищої освіти: це відноситься до всіх засобів, які сприяють відвідування безпосередньо лекційних, практичних, лабораторних занять і т.д. Тут мається на увазі використання відео- і аудіо-інструментів, персональних комп'ютерів та інших гаджетів з (як зазначалося раніше) тісною взаємодією студента і відповідної техніки, і створення для цього оптимально можливого інформаційного простору, для формування в кожної особистості інформаційної культури.

2. За межами освітньої установи: це відноситься до комунікативних технологій, таких як Web, чати, форуми групи новин, для далеких обміну матеріалами, і так далі. Потужність цих інструментів полягає в можливості взаємодіяти і співпрацювати для того, щоб ефективно передавати знання, є дистанційне навчання. У той час як в минулому, для навчання зазвичай потрібно одночасна присутність викладача і студентів для взаємодії, тепер можна вчитися на великих відстанях завдяки даним технологіям.

Питання про мультимедійні додатки в освітньому середовищі також

передбачає розрізнення двох основних груп додатків, що відносяться до поведінки студентів: пасивним або інтерактивним. Перша група, це інструменти які вчителі використовують просто для підвищення потужності пояснення навчального матеріалу: відео, звук, зображення, графіки і так далі. У цьому випадку студенти не взаємодіють з інструментами мультимедіа, які означають, що поточний зміст інформації представлений на занятті не змінюється згідно поведінки студента. Інтерактивні мультимедійні засоби можуть змінювати поточний зміст відповідно до поведінки студента: студенти можуть змінити зміст відповідно до їх власних інтересів, рівнів, або поставленого завдання перед ним. Інтерактивні інструменти мультимедійних технологій можна використовувати за тією схемою, що і пасивні, як звуки, відео і тексти, але вони також можуть отримати спеціальні завдання: записи, перезапису зміни форм і т.д. Мультимедійні продукти надають широкі можливості для різних аспектів навчання.

Одними з основних можливостей і переваг засобів мультимедіа в разі їх застосування в навчальному процесі є:

- одночасне використання декількох каналів сприйняття студента в процесі навчання, за рахунок чого досягається інтеграція інформації, що доставляється різними органами почуттів;
- можливість симулювати складні реальні експерименти;
- візуалізація абстрактної інформації за рахунок динамічного представлення процесів;
- можливість розвинути когнітивні структури та інтерпретації студентів.

Наявність мультимедійних засобів і проведення занять дає викладачам можливість планувати такі види діяльності, які вносять елемент зацікавленості в навчальний процес. Вони дозволяють створити активне кероване комунікативне середовище, в якому здійснюється навчання. Взаємодія студента з комп'ютером, таким чином, з простого обміну інформацією або виконання команд перетворюється на багатогранну діяльність в цьому середовищі, завдяки чому перед студентом відкриваються дійсно необмежені можливості.

Мультимедійні технології стають все більш популярними в сфері освіти як засіб мотивації студентів в навчанні і надання їм багато способів висловити свої ідеї і показати їх результати роботи на практиці. Це також дозволяє вчителю використовувати свої навчальні програми в інноваційній сучасній манері, бути гнучким у донесенні інформації до кожного студента. При використанні мультимедійного засобу викладач стає посередником, консультантом або керівником, допомагає студентам отримати доступ, організовувати та створювати проблемні ситуації для ще більшого залучення і заохочення кожного студента до навчання (що є необхідним і важливим в наш час). До засобів мультимедіа відносяться пристрої мовного введення і

виведення інформації; широко поширені вже зараз сканери (оскільки вони дозволяють автоматично вводити в комп'ютер друковані тексти та малюнки); високоякісні відео- (video-) і звукові- (sound-) плати, плати відеозахвату (video grabber), що знімають зображення з відеомагнітофона або з відеокамери і вводять його в ПК; високоякісні акустичні та відеозаписів системи з підсилювачами, звуковими колонками, великими відеоекранами.

Мультимедіа діляться на програмні і апаратні. Апаратна сторона мультимедіа може бути представлена як стандартними засобами - відеоадаптерами, моніторами, дисководами, накопичувачами на жорстких дисках, так і спеціальними засобами - звуковими картами і звуковими колонками. Програмна сторона без апаратних мультимедіа позбавлена сенсу. Програмні засоби поділяються на прикладні та спеціалізовані. Прикладні - це самі додатки Windows, що представляють користувачеві інформацію в тому чи іншому вигляді. Спеціалізовані - це засоби створення мультимедійних додатків - мультимедіа проектів (наприклад, програма для створення мультимедіа презентацій MicroSoft Power Point,). Сюди входять графічні редактори, редактори відеозображень (наприклад, Adobe Premier),засоби для створення і редагування звукової інформації і т.д..

Мультимедійні презентації - комплексні освітні ресурси, здатні об'єднувати всі цифрові освітні ресурси, необхідні для конкретного заняття. Інструментом для створення мультимедійних презентацій є програма PowerPoint, що входить в офісний пакет будь-якого персонального комп'ютера з ОС Windows. Для створення ігрового мультимедійного середовища використовують цю ж програму але з підтримкою макросів (Pptm). В даному розширенні студент не тільки знайомиться, пізнає світ, а й може «доторкнутися» за допомогою комп'ютерної миші. При цьому ускладнюється завдання, збільшується інтерес студента і безпосередньо його участь в навчанні конкретної теми. І тут можливості презентації будуть використані не тільки для наочного уявлення основних понять теми, але і як частково ігрова діяльність кожного студента. Найчастіше інтерактивні дошки використовуються в навчально - виховних закладах як дидактичні засоби, для активного засвоєння великої кількості навчального матеріалу, тобто як мультимедійна технологія. Наприклад, при використанні даного програмного забезпечення можна створювати дидактично - ігрові тести для самоконтролю. Також для створення мультимедійних презентацій можна використовувати і інші програми, які можуть забезпечувати інтерактивність створюваних навчальних матеріалів. Неперевершеним інтерактивним функціоналом володіють продукти корпорації Digital Workshop (OpusCreator, OpusPro), але Matchware Mediator перевершує їх простотою і зручністю у використанні. Основними елементами мультимедіа-технології є числа, текст, анімація, звук, відео. Для кожного елемента існують технічні засоби отримання і відображення, програмне забезпечення, алгоритми обробки,

засоби зберігання та передачі, методичні рекомендації щодо використання.

Під технічним забезпеченням мультимедіа-технологій в роботі прийнята сукупність взаємопов'язаних і взаємодіючих технічних засобів для введення, зберігання, переробки, передачі різнорідних даних і програм, організації спілкування людини-ЕОМ з метою використання мультимедійних технологій. Принциповою особливістю сучасних ЕОМ стала можливість обробки різнорідної інформації в реальному часі. Цьому сприяли досягнення в області цифрової обробки сигналів. Головна перевага цифрової технології - це здатність передавати дані з мінімальними втратами.

Суть цифрової обробки полягає в перетворенні аналогових звукових і відео сигналів в цифрову форму за допомогою аналогово-цифрових перетворювачів і виконання над ними різних операцій, таких як корекція, фільтрація, стиск, аналіз спектрів, виявлення, розпізнавання, обробка зображень. При цьому головним недоліком є великі обсяги інформації, що вимагають реалізації ефективних методів стиснення.

Для вирішення цих проблем були розроблені стандарти мультимедійного комп'ютера, що визначають вимоги до комп'ютерів, призначених для обробки мультимедійних даних. Вимоги стандарту визначають основні характеристики комп'ютера, такі як: тип мікропроцесора, швидкодія, обсяг ОЗУ, обсяг жорсткого диска, частоту і час вибірки в аналого-цифровому і цифро-аналоговому перетворювачах. Основні периферійні компоненти, такі як CD-ROM, сканери, звукові та відеотехніка, відеомагнітофони є невід'ємною частиною мультимедійного технічного забезпечення.

Аналіз процесу конфігурації мультимедійних комп'ютерів показав труднощі, з якими можуть зіткнутися вчителі, котрі впроваджують мультимедіа технології в навчальний процес. До них відноситься, перш за все, забезпечення максимальної продуктивності комп'ютера, так як обробка графічної, звукової та відео інформації вимагає наявності досить великих ресурсів пам'яті, швидкодії. Існує два рішення цієї проблеми. Перша полягає в використанні більш потужних комп'ютерів. Друга - у використанні технології самоорганізації Plug and Play для вибору оптимальної конфігурації системи. Оскільки компонентами технології Plug and Play є операційна система, BIOS (базова система введення / виведення) і безпосередньо апаратура, то технологія Plug and Play це не тільки ідеологія архітектури комп'ютера, специфікація для виробників компонент, але і критерій відбору апаратних засобів для реалізації оптимальної конфігурації комп'ютера з метою отримання максимальної продуктивності.

Під програмним забезпеченням мультимедіа технологій слід розуміти сукупність програм, призначених для обробки елементів мультимедіатехнології: тексту, звуку, зображення, відео, а також програм, інтегруючих ці дані в єдине середовище. Так як програмне забезпечення

ЕОМ ділиться на системне і прикладне, то його класифікацію запропоновано представляти наступним чином. Перший рівень програмного забезпечення складають драйвери різних нестандартних пристроїв, програмне забезпечення, що служить для підтримки технічних засобів. Другий рівень програмного забезпечення, що дозволяє розширити можливості використання мультимедійних можливостей комп'ютера - це прикладне програмне забезпечення. У Microsoft Office входять, принаймні, три програми, що дозволяють створювати мультимедійні додатки - це PowerPoint, Microsoft Publisher і FrontPage. Всі ці програми використовують основні досягнення в області проектування мультимедійних додатків: наявність гіпертексту, інтелектуального інтерфейсу, обробки різнорідних даних, можливість публікації в Інтернеті. Явним недоліком навчального мультимедійного проекту, створеного на основі прикладних програм є те, що вони можуть виконуватися тільки в середовищі, використовуваному для їх проектування, вони прив'язані до операційної системи і використовуваного браузеру. Але з огляду на, що більшість комп'ютерів українських шкіл базується на платформі Intel, використовують операційну систему, це прикладне програмне забезпечення відповідає критерію універсальності, цілком придатні для використання в Internet мережі будь-якого навчального закладу.

Третім рівнем мультимедійного програмного забезпечення є сімейство інструментальних авторських систем, які дозволяють створювати мультимедійні навчальні проекти, що працюють незалежно від засобів розробки. Вони мають вбудовані засоби реалізації інтерактивності, вони як і раніше залежать від архітектури комп'ютера і операційних систем, мають задану структуру проекту. Як правило, ця структура являє собою набір карток або сторінок, які можуть бути пов'язані між собою в стек, в книгу. Кожна сторінка може містити інформацію будь-якого типу: текст, графіку, звук, анімацію.

Вбудована мова програмування використовується для створення більш гнучких зв'язків і вирішення завдань навігації і інтерактивності. Всі системи мають інтелектуальний інтерфейс, підтримують механізм OLE. До таких програм можна віднести Multimedia Tool Book компанії Asymetrix, Authorware Professional for Windows компанії Macromedia, HyperCard фірми Apple. Четвертим рівнем мультимедійного програмного забезпечення є програми обробки елементів мультимедіа технології. Цей рівень позначимо як допоміжне програмне забезпечення. Він складається з текстових редакторів, програм комп'ютерної графіки та анімації, програм обробки звукової, відеоінформації.

На ринку існує ціле сімейство текстових і графічних редакторів. Що ж стосується програм звукової і відео обробки, то вони, як правило, прив'язані до апаратури, кожна звукова карта, відеокарта, карта захоплення

відео супроводжується своїми драйверами і програмами. Так як, в даний час існує два способи представлення графічної інформації в комп'ютері растровий і векторний, що володіють різними принципами побудови зображення на екрані монітора,то і в програмне забезпечення мультимедійних технологій повинні бути включені обидва види графічних редакторів [74, с. 38].

Методичне забезпечення мультимедійних технологій - це сукупність методичних посібників і методичних вказівок по обробці різнорідної інформації, методичні рекомендації щодо використання різних видів інформації в навчальному процесі. В даний час цього можна досягти, якщо розташовувати електронні образи всіх навчально-методичних матеріалів на WEB-серверах локальних мультимедійних інформаційних систем. Для формування електронного образу навчально-методичного посібника в даний час використовується термін електронний підручник. Так як основою електронного підручника є сторінка, то для його створення використовується об'єктна модель документа, яка включає в себе об'єкт документ і його дочірні об'єкти, колекції для формування складових частин.

Основним об'єктом підручника є документ, який може складатися з різних колекцій: фреймів, зображень, форм, посилань. Як стандарт для електронних підручників, що зберігаються на WEB серверах університету можна запропонувати структуру, засновану на використанні колекції фреймів, посилань для подання змісту і реалізації навігації в підручнику.

Організаційне забезпечення являє собою сукупність документів, що встановлюють правила і форму оформлення навчально-методичних матеріалів і розміщення їх на сервері, перелік стандартів файлів різнорідних даних, що використовуються для розміщення в мультимедійних базах даних, правила і технологію доступу до програм обробки мультимедійних даних і до WEB- сервера навчально-методичної літератури [78, с. 341]. Таким чином, інформаційно - комунікативні засоби дозволяють значно підвищити (у порівнянні з традиційними формами, методами і засобами навчально - методичного забезпечення) технологічність викладання і освоєння професійних дисциплін, вдосконалення за рахунок оптимізації і програмування інформаційного середовища, автоматизації процесу викладу навчального матеріалу і контролю знань студентів, при підготовці майбутніх вчителів технологій.

Перспективою дослідження є розробка методичних рекомендацій щодо адаптації майбутніх вчителів технологій в інформаційно - комунікативному середовищі, розробка завдань, спрямованих на використання ІКТ та педагогічних програмних засобів педагогічної діяльності кожного студента. Однією з першочергових завдань підготовки майбутніх вчителів технологій до реалізації власного професійного потенціалу в умовах інформатизації освіти є створення відповідних умов для

розвитку умінь самостійно здобувати професійні знання, використовувати їх для розробки та впровадження методично доцільного педагогічного програмного забезпечення.

6.3. Новітні мультимедійні технології в освітніх інформаційних системах

Використання мультимедійних засобів в освіті зводиться до декількох базових методів педагогічної діяльності, які, відповідно до принципів взаємодії студента з комп'ютерно орієнтованими засобами навчання, можна розділити на два основні класи.

До першого віднесемо ті, де студентам відводиться роль пасивного спостерігача й отримувача інформації. До таких програмних засобів навчального призначення переважно належать ті, у яких здійснюється керування процесом подання інформації з боку викладача або розробника.
До другого класу відносяться інтерактивні освітні засоби мультимедія у тому сенсі, що в них закладено можливість активної участі студента, який може самостійно обирати шлях вивчення певної теми, визначати послідовність вивчення тощо.

Актуальність цього дослідження полягає в тому, що мультимедійні технології завдяки своїм можливостям дають змогу активно використовувати в процесі навчання комп'ютерні засоби та можуть застосовуватися при проведенні практично будь-яких видів навчальних занять. Використання мультимедійних технологій дає можливість отримувати велику кількість навчальної інформації в доступній формі із мінімальними витратами ресурсів. Саме завдяки цьому вони набувають все більшого розповсюдження та популярності в системі сучасної освіти у вищій школі.

У сучасній освіті значно зріс обсяг і рівень навчального матеріалу, а отже, й вимоги до викладачів. Істотну допомогу в роботі викладачів надають технічні засоби навчання (ТЗН).

ТЗН – це система засобів, що складається з двох взаємопов'язаних частин: специфічних носіїв навчальної інформації (відеострічки, диски, наочні приладдя, навчальні посібники та ін.) і апаратури, за допомогою якої може бути подано або створено навчальну інформацію. З визначення ТЗН випливає, що проблема їх використання має два діалектично поєднані аспекти: педагогічний і технічний.

Педагогічний аспект охоплює питання, які пов'язані із створенням специфічних носіїв навчальної інформації, їх змісту відповідно до дидактичних вимог навчального процесу, а також із розробкою методики їх застосування. Технічний аспект охоплює питання створення або пристосування апаратури, яка задовольняла б педагогічні і технічні вимоги

щодо подання (вироблення) навчальної інформації. Це – створення електронних посібників й апаратури для їх подання [4].

Значну частину технічних засобів навчання складають аудіовізуальні засоби подання навчальних матеріалів (АВ ТЗН). Особливо важливу роль вони відіграють при масовому аудиторному навчанні. У роботі АВ ТЗН присутні такі інформаційні процеси, як введення, вивід, відображення та видалення інформації.

На рис. 1 наведена схема процесу створення (введення), передачі й сприйняття інформації під час масового аудиторного навчання. Тут джерелом є АВ ТЗН, в який викладач увів навчальну інформацію. Технічний засіб відтворює на екрані або будь-яким способом виводить закладену в нього аудіо або візуальну інформацію, а викладач може додати до неї щось нове (наприклад, коментарі, пояснення, обговорення). Вся ця інформація передається й сприймається аудиторією.

На практиці майже завжди застосовується більш гнучкий та ефективний – інтерактивний спосіб викладання [5]. Інтерактивне навчання – це співнавчання, взаємонавчання (колективне, групове, навчання співпраці), де студент і викладач є рівноправними, рівнозначними суб'єктами навчання. Воно ефективно сприяє формуванню цінностей, навичок і вмінь, створенню атмосфери співпраці, взаємодії, дає змогу педагогу стати справжнім лідером колективу. Сутність інтерактивного навчання полягає в тому, що навчальний процесс відбувається за умов постійної, активної взаємодії усіх студентів з широким залученням як традиційних технічних засобів навчання, так і створених на базі інформаційних комп'ютерних технологій [4].

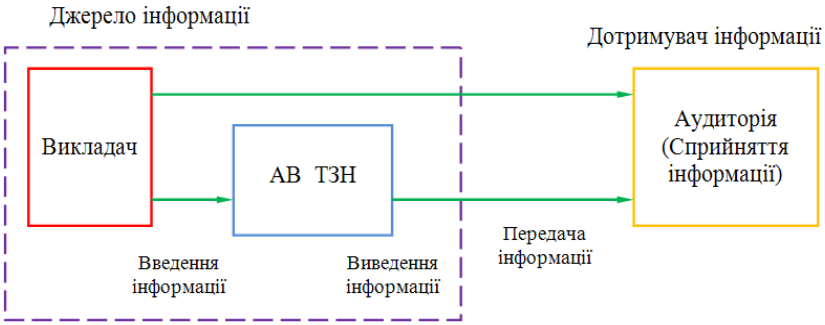


Рис. 1. Інформаційні процеси при навчанні з використанням аудіовізуальних технічних засобів навчання

При інтерактивному навчанні завжди має місце зворотний зв'язок. Інтерактивні аудіовізуальні технічні засоби навчання (ІАВ ТЗН) мають два контури зворотного зв'язку: внутрішній і зовнішній (рис. 2). Тут викладач не тільки сприймає інформацію, але й оперативно управляє технічним засобом

навчання і вводить нову інформацію. Інший зворотний зв'язок у вигляді стану аудиторії враховується викладачем або оперативно при проведенні занять, або при підготовці до них.

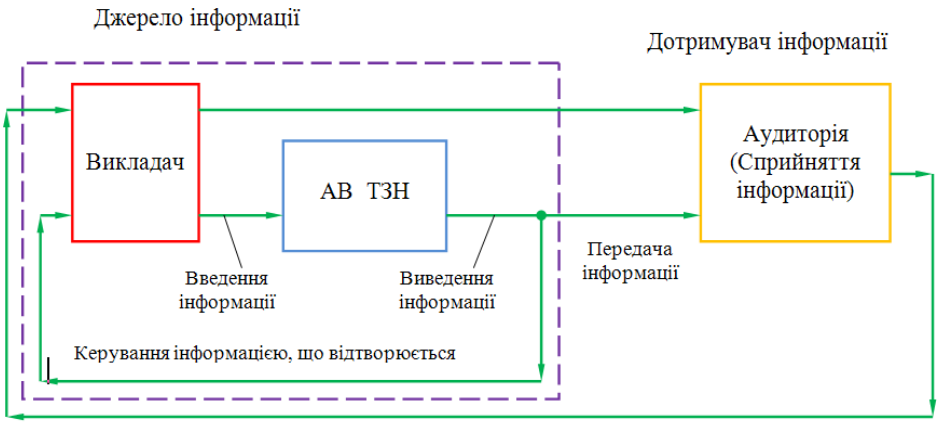


Рис. 2. Інтерактивні зв'язки при навчанні з використанням аудіовізуальних технічних засобів навчання

У наш час існує достатній досвід роботи з інтерактивними технічними засобами індивідуального використання. По суті, будь-який персональний комп'ютер є таким засобом. Що стосується інтерактивних колективних засобів для аудиторного навчання, то їх споживчі якості ще не повною мірою задовольняють потреби користувачів, але вони швидко розвиваються. *Термін "мультимедіа"* - латинського походження, який поширився за рахунок англомовних джерел і в перекладі означає "багато середовищ".

У всесвітній доповіді з освіти ЮНЕСКО (1998р.) "мультимедіа" називають здатність подавати текст, зображення та звук користувачеві. У словниках мультимедія визначається як сукупність комп'ютерних технологій, які одночасно використовують декілька середовищ: графіку, текст, відео, фотографію, анімацію, звукові ефекти, високоякісний звуковий супровід.

За даними ЮНЕСКО, при слуховому сприйнятті закріплюється 15% мовленнєвої інформації, при зоровому- 25% візуальної інформації; слухаючи та дивлячись одночасно, людина запам'ятовує 65% інформації, що їй повідомляється. В цьому і допомагають мультимедійні технології.

Сучасні підходи до використання комп'ютерів при вивченні спеціальних дисциплін засновуються на двох важливих інноваційних технологіях останніх років - мультимедійних технологіях (МТ) та глобальній мережі інтернет.

Ефективність застосування мультимедійних технологій в першу чергу залежить від професійної підготовки викладача до їх використання і від

умов, у яких вони застосовуються.

Функція МТ в структурі навчання полягає в тому, що, по-перше, вони виступають джерелом змістовної інформації, по-друге, щоб взяти на себе окремі функції викладача та студента, по-третє, представляти собою специфічні особливості різних видів наочностей, придатних для вирішення певних задач.

Значна частина вимог пов'язана із забезпеченням інтерактивності аудіовізуальних засобів.Тому саме інтерактивні засоби викликають найбільший інтерес. На рис 3. наведено загальну структуру інтерактивного аудіовізуального (ІАВ) засобу масового навчання.

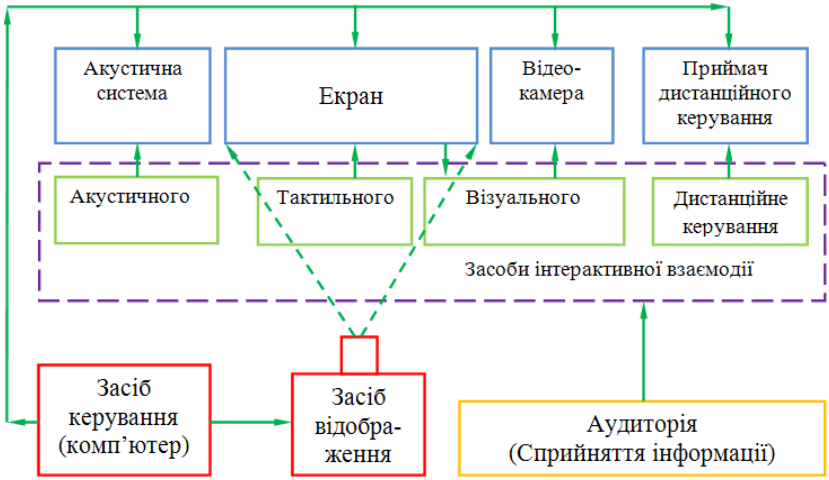


Рис. 3. Загальна структура інтерактивного аудіовізуального комплексу ІАВ ТЗН

Основним елементом ІАВ ТЗН є відображення й сприйняття візуальної інформації зможливістю управляти нею безпосереднім дотиком до екрана або засобами дистанційного керування. Значну роль відіграє вивід звукової інформації, а також можливість уведення й розпізнавання мови. Як видно з рис.3, зворотний зв'язок забезпечується засобами інтерактивної взаємодії за чотирма каналами: акустичним (мікрофони), тактильним (дотик до екрана), візуальним (сканери, відеокамери), дистанційного керування (стандартні інфрачервоні й радіоканали). У наш час випускаються різні інтерактивні технічні засоби, а також існує ціла низкадуже цікавих ідей і розробок.

Наприклад, управляти курсором мишки за допомогою дистанційного керування або бездротової мишки, вносити доповнення в матеріал безпосередньо дотиком до екрану інтерактивних дошок та ін.

Сьогодні найбільшу популярність набули комплекси, що

складаються з комп'ютера, мультимедійного проектора й інтерактивної дошки. Усі компоненти цих комплексів випускаються серійно, мають добре налагоджене програмне забезпечення й знайшли широке застосування в навчальному процесі різних навчальних закладів. Ці комплекси задовольняють майже всі основні вимоги, і це стало головним, що визначило їхню високу популярність. Єдине, що в них є негативним, це вартість і компактність. При відповідній доробці й удосконаленні вони можуть стати ідеальним технічним засобом для аудиторних занять вищих та інших навчальних закладах.

Сучасні досягнення у галузі інформаційної технології надають викладачу змогу використовувати автоматизовані інформаційні системи для найрізноманітніших потреб, не вдаючись при цьому до написання власних програм. У студентів різні характери, різні здібності. Побудувати навчальний процес з урахуванням індивідуальності кожного студента без допомоги техніки викладач не може. Комп'ютер саме й надає можливість роботи з урахуванням особливостей кожного студента. Треба чітко усвідомити, що комп'ютер - це не лише наставник, це інструмент творчості, що розвиває, заохочує до пошуків.

Комп'ютер має великий вплив на зміст, метод, організаційні форми навчання, роль викладача у навчальному процесі. Визнаний потужним засобом підтримки процесу навчання, комп'ютер докорінно змінює організаційно-методичну структуру заняття, використання найбільш прогресивних і раціональних форм і методів навчання, вимагає чіткої наукової організації навчально-виховного процесу.

При комп'ютеризації освіти традиційна дидактична система навчання "викладач- студент", "студент-студент" замінюється новою багатоаспектною системою навчання "викладач-комп'ютер-студент", "комп'ютер-студент", "студент-комп'ютер-студент".

Використання мультимедійних засобів в освіті зводиться до декількох базових методів педагогічної діяльності, які, відповідно до принципів взаємодії студента з комп'ютерно орієнтованими засобами навчання, можна розділити на два основні класи. До першого віднесемо ті, де студентам відводиться роль пасивного спостерігача й отримувача інформації. До таких програмних засобів навчального призначення переважно належать ті, у яких здійснюється керування процесом подання інформації з боку викладача або розробника. До другого класу відносяться інтерактивні освітні засоби мультимедія у тому сенсі, що в них закладено можливість активної участі студента, який може самостійно обирати шлях вивчення певної теми, визначати послідовність вивчення тощо.

Окремі мультимедійні ресурси або засоби навчання мають лінійну структуру подання інформації, за якої студент послідовно знайомиться з навчальним матеріалом, причому ця послідовність строго визначена змістом

певної предметної галузі або окремого навчального курсу.

Лінійна послідовність подання інформації різного роду за допомогою мультимедійних засобів навчального призначення рекомендується у тих випадках, коли студенти або зовсім не володіють, або володіють досить обмеженими попередніми знаннями з певної предметної галузі, що викликає потребу оглядового викладу навчального матеріалу. Такий стиль викладання можна також рекомендувати і тоді, коли студентам потрібно сформувати початковий рівень фундаментальних знань у предметній галузі.

Нелінійні мультимедійні засоби відіграють значну роль у самоосвіті, особливо у системах відкритого або дистанційного навчання. За такого підходу студент займає в процесі навчання активну роль, самостійно визначаючи необхідний матеріал та шляхи його засвоєння.

Мультимедійні засоби навчального призначення, що засновані на нелінійних способах подання та організації інформації, більш доцільно використовувати у ситуаціях, коли студенти вже володіють певними початковими знаннями з предметної галузі, яких досить для того, щоб вони могли самостійно формувати питання і визначати для себе навчальні завдання.

Під час добору та формування змісту мультимедійних ресурсів потрібно враховувати, що мультимедія включає одночасно декілька каналів сприйняття. На відміну від традиційних друкованих джерел інформації для сучасних мультимедійних інформаційних засобів і телекомунікаційних технологій характерне одночасне використання зорових, слухових і тактильних інформаційних каналів. Добір мультимедійної інформації повинен відповідати трьом загальним принципам: висока ефективність, коректність, достовірність.

Одним із суттєвих чинників добору змісту і побудови мультимедійних ресурсів є необхідність ознайомлення студентів з основними формами інтерактивного спілкування у сучасних інформаційних мережах, до яких відносятся телеконференції, чати, електронна пошта.

Таким чином використання процедур інформаційного пошуку, як і довільна робота студентів з інформаційним наповненням мультимедійних ресурсів і телекомунікаційних систем, дозволяє сформувати у них певні навички в галузі структуризації і класифікації інформації, яку згодом можна використовувати у навчальній діяльності.

Розвиток інформаційно-комунікаційних технологій дозволяє створювати інтерактивні засоби масового навчання, які задовольняють найвищі вимоги. У наш час створено і створюється велика кількість апаратних технічних засобів різного типу, що дає можливість подавати навчальну інформацію у поєднанні з їхньою наочною демонстрацією.

Також важливим чинником застосування мультимедійних технологій є те, що студент перетворюється з об'єкта навчання в суб'єкт навчання, тобто

процес пізнання переходить із категорії «вчити» до категорії «вивчати» свідомо і самостійно через «занурення» студента в особливе інформаційне середовище, яке найкраще мотивує і стимулює вивчення практично будь-якої навчальної дисципліни.

Розглянемо найпопулярніші сервіси для створення дидактичних матеріалів.

Дидактичний матеріал — особливий тип наочного навчального посібника, переважно карти, таблиці, набори карток з текстом, цифрами або малюнками, реактиви, рослини тощо, які роздаються учням для самостійної роботи в класі і вдома або демонструються вчителем перед усім класом.

До сервісів для створення дидактичних матеріалів можна віднести ті, що дозволяють створити щось на зразок карток чи таблиць.

ClassTools

Онлайн сервіс для створення інтерактивних Flash-ресурсів і, перш за все, дидактичних ігор для уроків ClassTools.NET. За допомогою цього сервісу ви можете в лічені хвилини створити свою дидактичну гру або створити навчальну діаграму, скориставшись одним із шаблонів. Алгоритм роботи досить простий. Набираєте за шаблоном питання і відповіді. За допомогою Генератора ігор підбираєте найбільш підходящий для вас варіант. Запускаєте. Є можливість зберегти ігри на комп'ютері у вигляді HTM-файла, розмістити на сторінках сайтів і блогів, поділитися посиланням. Є можливість «запаролити» режим редагування готової роботи. Більшість дидактичних ігор можна успішно використовувати з інтерактивною дошкою. Сервіс також дозволяє викладачам і школярам створювати інтерактивні Flash-діаграми для ефективного проведення презентацій, захисту проектів, подання діаграм, аналітичних доповідей, планування заходів і т.д. Для початку роботи реєструватися не потрібно. Сервіс на англійській мові, але підтримує кирилицю.

BrainFlips

Онлайн сервіс для створення карток BrainFlips. За допомогою сервісу можна виготовити картки по предмету викладання і тут же почати працювати з ними. Карти-завдання об'єднуються в колоди. У картку можна додати відео, аудіо або фото для того, щоб включити всі канали сприйняття інформації. Також можна користуватися картками інших учасників сервісу. Формат використання карток вибирається викладачем. Сервіс створений спеціально для вчителів. Є можливість створювати групи, підключати до групи учасників. Сервіс на англійській мові, але підтримує кирилицю. Назви груп, карток, колод карток і описів лише англійською мовою. Для початку роботи необхідно зареєструватися.

Flashcard Machine

Онлайн сервіс Flashcard Machine створений для підготовки дидактичних матеріалів в ігровій формі у вигляді наборів карток. Матеріали

на картках можуть бути у вигляді тексту, зображень, звуку, посилань. Питання готового набору карток при запуску тасуються випадковим чином. Для початку роботи необхідно зареєструватися. Сервіс підтримує кирилицю. Є можливість виступати в ролі вчителя, студента і організувати групову роботу з картками. Є велика колекція готових карток, розкладена по темам, віком.

JeopardyLabs

Онлайн сервіс JeopardyLabs призначений для генерації тематичних вікторин. Для початку роботи на сервісі не потрібно реєструватися. Тільки ввести пароль для редагування. Сервіс підтримує кирилицю. Після заповнення даними сервіс запропонує посилання для роботи з вікториною.

JigsawPlanet

Онлайн сервіс для генерації пазлів з вихідних графічних зображень (фотографій) JigsawPlanet. Для початку роботи необхідно зареєструватися. Потім користувач створює альбом (и) і завантажує тематичні зображення, з яких сервіс пропонує створити різні за складністю та формою пазлів ігри. Створені роботи можна зберігати на сторінках сайтів у вигляді альбомів і як окремі роботи. Можна поділитися роботами в соціальних сервісах і за допомогою електронної пошти. Роботи можна створювати з загальним доступом (публічні) - для тих, хто має посилання, і приватні.

LearningApps

Сервіс LearningApps призначений для створення інтерактивних навчально-методичних посібників з різних предметів. Сервіс заснований на роботі з шаблонами (заготовками) для створення роботи. Тематика різноманітна: від роботи з картами до розгадування кросвордів і створення карт знань. Сервіс підтримує кілька мов (російська мова підтримується на окремих шаблонах при заповненні контенту). Для початку роботи необхідно зареєструватися. Є велика колекція робіт, російською мовою зустрічаються тільки поодинокі матеріали, тому можна розраховувати тільки на свої роботи.

Wixie

Wixie дозволяє малювати, додавати текст, додати картинку, і багато іншого. Це дає вам можливість спробувати Wixie (повна версія орієнтована на навчальні заклади) і апробувати в практичній діяльності. Безкоштовна версія не вимагає реєстрації і підтримує кирилицю. Школярі можуть використовувати в Wixie інструменти малювання, змінювати параметри тексту, картинок і вбудовувати голосові записи при розробці електронних публікацій і флеш-анімації.

WordLearner

Онлайн сервіс для створення дидактичних матеріалів (робочих листів, головоломок, вправ, карток та ігор) WordLearner. Для початку роботи необхідно зареєструватися (як студент, педагог, представник освітнього

закладу). Сервіс англійською мовою, підтримує кирилицю. Є можливість створювати групи, класи, реєструвати школярів і вести статистику роботи в групі.

Zondle

Освітній сайт Zondle не тільки надає безкоштовні онлайн дидактичні ігри для початкової та середньої школи, а й пропонує вчителю проявити творчість, підготувати захоплюючі ігри за допомогою одного з навчальних предметів. Досить зареєструватися в Zondle, вибрати тему і створити список обраних ігор.

Можливі три рівні створення освітнього ресурсу:

1. Створення гри за шаблоном: Це найпростіший варіант. Учитель набирає ряд завдань по конкретній темі. Потім ви можете подивитися, як ваші завдання будуть реалізовуватися в наявних на сайті різних іграх. Кожна з ікон являє собою готову гру. Зупиняєте свій вибір на одній з них і вносите свої корективи. Ваші гри зберігаються на віддаленому сервері. Ви можете також вбудувати їх на свій сайт або блог.
2. Створення авторського пакета: Пакет являє собою послідовність сторінок, які можуть містити текст, зображення, відео, аудіо і, звичайно, Zondle ігри та обрані предметні теми.
3. Створення гри з нуля: Ви самі підбираєте персонажі, тло, стаціонарні об'єкти, ландшафт. Підбираєте звукові ефекти, а також ефекти анімації і пересування. І вже під цю гру придумуєте завдання. Конструктор дозволяє створювати ігри не тільки для індивідуальної роботи учня на комп'ютері, але і використовувати великий екран для фронтальної та групової роботи або інтерактивну дошку.

На сайті створено спільноту вчителів, яке обмінюється створеними ресурсами. Для початку роботи необхідно зареєструватися, створити матеріали, позначити клас, і почати працювати з використанням нових можливостей. Втім, ви можете скористатися ресурсами навіть без реєстрації.

PurpozeGames

Сервіс для створення тематичних ігор онлайн PurpozeGames. Для початку роботи необхідно зареєструватися. Сервіс підтримує кирилицю. Можливі два варіанти створення ігор:

1. Прив'язуючи до точки на зображенні питання з однозначною відповіддю.
2. Можливість дати альтернативний відповідь. За результатами гри ведеться рейтингування.

Study Stack

Онлайн сервіс Study Stack для створення дидактичних матеріалів для освіти. Порядок роботи з вашими матеріалами: це робота з текстом (питання і відповіді) і робота з графічними зображеннями та коментарями до них. Набравши один раз комплект запитань і відповідей, ви отримуєте декілька

варіантів для генерації дидактичних матеріалів в ігровій формі. Готові роботи легко можна вмонтувати на сторінки сайтів, блогів, поділитися інформацією в соціальних мережах. Для початку роботи необхідно зареєструватися або скористатися аккаунтом від Facebook. Сервіс підтримує кирилицю. Крім ваших робіт ви можете скористатися колекцією робіт, створених педагогами світу.

6.4. Інтеграція медіа в освітній процес

Масштаби впливу медіа на різні сфери життя людини зростають з кожним днем. Це вимагає від людини не лише знання сучасних технічних пристроїв та уміння з ними працювати, а й певного рівня критичного мислення (здатності інтерпретувати інформацію, отриману із засобів масової інформації, розуміння різноманітних медіатекстів), навичок самостійної роботи, пов'язаної з пошуком, обробкою, презентацією інформаційного матеріалу тощо.

Сучасні учні, студенти є активними споживачами інформації різноманітних медіа. Уже в 60ті роки XX сторіччя у провідних країнах світу у педагогічній науці сформувався спеціальний напрям – медіаосвіта, покликаний допомогти їм краще адаптуватися в умовах медіа культури. Цілі і задачі медіаосвіти вченими різних країн формулюються по- різному, але в педагогічному аспекті даний напрям вивчає взаємодію медіа з процесами навчання і виховання. Тому метою нашої роботи є дослідження інтеграції медіа в освітній процес.

Розвиток засобів масової інформації/ комунікації і залучення їх до навчально- виховного процесу значно активізували творчі шукання педагогів у багатьох країнах. Існуючий процес комп'ютеризації та інформатизації сучасного суспільства визначив напрями вітчизняних і зарубіжних досліджень, які аналізують різні аспекти проблемиінтеграції сучасних медіа в освітній процес:

- Соціально-філософська і гуманітарна взаємодія людини і медіа в процесі спілкування;
- вплив окремих медіа на розвиток, виховання і навчання людини;
- проблема насильства через сучасні засоби інформації і комунікації.

В. Робак зазначає, що у Німеччині працюють науково-дослідні інститути, що виконують дослідження цієї галузі. Німецький вчений Б. Щерб став першим професором медіапедагогіки. Він вважає, що медіапедагогіка, насамперед, застосовується у позаурочній роботі і саме там має найкращі перспективи. На його думку, настав час широкого впровадження у навчальних закладах медіапедагогічних і медіапсихологічних курсів, поряд із курсами з комп'ютерних

технологій(Робак В., 2006). Інший німецький дослідник професор С. Ауфенангер зазначає, що медіапедагогіка як наукова галузь зазнала істотних змін, ставши комплексною наукою, що акумулює відомості з багатьох дисциплін (загальної психології, філософії, освітніх технологій, психології розвитку тощо) (Шариков А. В., 1990). З цією думкою погоджується директор Мюнхенського науково-дослідного інституту медіапедагогіки Х. Тойнер, яка стверджує, що медіапедагогіка – це міждисциплінарна галузь, дотична до всіх наук (Мантуленко В. В., 2004). Професор Нюрнберзького університету Д. Шпангел вважає нагальним завданням для медіапедагогіки вироблення свого ядра, яке на даний момент досить невизначене (Робак В., 2006). На наш погляд, справедливою є думка В. Робака, який зазначає, що в центрі цього ядра мають бути медіадидактика й теорія медіавиховання.

У медіапедагогіці німецькі дослідники виділяють такі взаємопов'язані напрями як суспільно-критична та політично вмотивована медійна педагогіка. Перша ставить за мету зміну суспільства за допомогою таких її засобів як здатність ідеологічної критики, впливу на медіасистему, використання альтернативних медій; друга має за мету боротьбу проти маніпуляцій за допомогою медій.

Німецький дослідник професор С.Ауфеннагер описує три можливих варіанти реагування освітньої системи майбутнього на вимоги медіасуспільства: перший – консервативний варіант; другий виходить з ідеї інтеграції, але розглядає її в аспекті асиміляції в традиційних методикодидактичних підходах; третій покликаний показати, як може здійснюватись інтеграція нових медіатехнологій в освіту в педагогічному супроводі, і що в процесі конкуренції з медіаринком освітня система повинна більш критично підходити до його розгляду.

Подальший розвиток цивілізації залежить від того, як будуть будуватись взаємостосунки медіа з особистістю людини, медіа і культури, медіа і освіти.

Інтеграція медіаосвітиу традиційне навчання передбачає, насамперед, отримання позааудиторної інформації у контексті базової освіти. При інтеграції медіаосвітиу традиційні навчальні курси велику роль відіграє позааудиторна інформація. Джерелами її можуть бути:

- засоби масової інформації на друкованій основі (книги, газети, журнали);
- електронні засоби передачі інформації (радіо, ефірне, кабельне, супутникове телебачення, комп'ютерні мережі, комп'ютерні ігри, мережа «Інтернет» тощо).

Позааудиторна інформація може бути застосована у начальних цілях, оскільки вона розширює уявлення учнів, студентів про оточуючий світ і процеси, які в ньому відбуваються. Інформація, яка отримується з різноманітних медіаджерел, на відміну від навчального матеріалу, має низку

привабливих для молоді властивостей: вона емоційно забарвлена, актуальна, доступна для розуміння, не вимагає заучування і не підлягає оцінці. Вплив медіа на молоду людину полягає, насамперед, в тому, що цей вплив «неорганізований». Він має характер сприйняття, який відрізняється від сприйняття інформації на занятті.

Існує декілька підходів до використання засобів масової комунікації та інформації:

- інтеграційний (через уже наявні навчальні предмети);
- факультативний (через роботу факультативів, гуртків, клубів);
- спеціальний (введення нового предмету, спецкурсу).

Медіа, інтегровані в традиційні предмети навчального циклу, стають одним із засобів підвищення ефективності навчання, якщо задовольняють наступні вимоги:

- сприяють підвищенню пізнавального інтересу;
- відповідають практичним потребам викладача і студента;
- універсальні у використанні;
- застосовуються з опорою на базову навчальну програму.

При застосуванні сучасних медіа у навчальному процесі значно змінюється роль педагога і самого навчального процесу, його методи і зміст. Сучасні медіатехнології дозволяють індивідуалізувати і активізувати освітній процес у межах колективного навчання. З'явилась можливість у масовому масштабі вирішувати особливий тип задач, спрямованих на рефлексію молоддю своєї діяльності, її саморегуляції, що реалізується навіть в умовах індивідуального навчання.

Поряд з великим дидактичним потенціалом сучасних медіа існує і ряд психологічних проблем, які достатньо багатоаспектні. Це проблеми місця самих медіа у навчальному процесі та у реалізації навчаючих систем, взаємодія молодої людини з медіа, особливості їх діалогу, актуальним стає наукове обґрунтування проблем захисту людини від маніпулювання за допомогою медіазасобів.

Деякі особливості застосування цих технічних засобів в освітньому середовищі можна обґрунтувати на основі розуміння сутності електронних медійних технологій, яку складають наступні якості:

- інтерактивність, яка сприяє організації взаємодії і самоатестації викладача;
- мультимедійність, яка подає об'єкти та процеси не традиційним текстовим способом, а за допомогою фото, відео, графіки, анімації, звука, що створює психологічні умови, які сприяють сприйняттю і запам'ятовуванню матеріалу. Застосування інформаційних технологій навчання забезпечує психо-фізіологічну орієнтованість навчання, яка передбачає підвищення ефективності навчання і створення оптимальних функціональних станів, які підвищують здатність мозку до засвоєння

матеріалу. Сучасні медіа забезпечують одночасну роботу декількох каналів подачі інформації, коли різні середовища доповнюють одне одного. Перед студентами відкриваються можливості творчого використання багатьох інформаційних джерел, кожне з яких має свою мову:

- здатність до моделювання, яке являє собою, насамперед, моделювання реальних об'єктів і результатів дослідження; моделювання за допомогою медіа допомагає вивчати об'єкт чи явище в природних умовах, з різних точок зору, задіяти наочно-образні компоненти мислення, що відіграє виключно важливу роль у навчанні, в тому числі при роз'ясненні і засвоєнні багатьох понять;комунікативність, тобто можливість безпосереднього спілкування, оперативність передачі інформації, контроль за станом процесу; комунікативність вирішує питання доведення інформації в найкоротші терміни, дозволяє дистанційно керувати навчальним процесом, забезпечує його кваліфікованими педагогами незалежно від місцезнаходження студентів; продуктивність, тобто автоматизація нетворчих, рутинних операцій, на які витрачається багато сил і часу.

Сучасні засоби надання інформації дозволяють суттєво підвищити ступінь ергономічних вимог до навчальних матеріалів: обрати розмір і тип шрифту, розмістити в тексті не лише малюнки, а й звукові фрагменти чи кінокліпи. Виникає нова, з точки зору ергономіки, можливість: учень сам підбирає найбільш ергономічні особисто для нього характеристики матеріалу, що вивчається. Сучасний учень, студент може на свій розсуд ілюструвати текст, що вивчається, зробивши сприйняття особистісним. Він може самостійно перетворювати інформацію, яка отримується з мережі, підбирати аргументи, будуючи їх за певною логікою доказовості, яка відображає його власні уявлення, напрям думки. У результаті студенти більш глибоко проникають в сутність питання, у них з'являється інтерес більш активно працювати з навчальною літературою.

Використання медіа орієнтоване на індивідуалізацію навчання у межах єдиного навчально-виховного процесу. При індивідуалізації навчання кожен студент опановує активну, зорієнтовану конкретно на нього діяльність. При цьому пробуджується мисленнєвий процес, більш повно реалізуються пізнавальні потреби, стимулюється творча активність, студенти отримують можливість обирати оптимальний темп навчання, контролювати і коригувати вивчення матеріалу, причому результати роботи не віддалені, їх видно безпосередньо на занятті. Медіа мають також великі виховні можливості: привчають до охайності та організованості. Засоби графіки, музичні фрагменти знімають напруження. Робота з медіа розвиває уміння планувати свою діяльність, приймати відповідальні рішення.

Якщо традиційна система навчання стимулює мотивацію досягнення (отримання хороших оцінок, успішна здача екзамену тощо), то використання інформаційних і комунікаційних технологій забезпечує формування

пізнавальних мотивів школяра, студента, які сприяють стійкій активності пізнання, підвищують ефективність засвоєння знань.

Всі технології, які будуть впроваджуватись у найближчому майбутньому, стануть називатись технологіями третього тисячоліття, яке стане переломним в розвитку людства. Тому проблема інтеграції медіа в освітній простір потребує пильної уваги з боку філософів, економістів, соціологів, політологів, педагогів, психологів тощо, оскільки ці проблеми можуть бути вирішені лише комплексно.

Таким чином, сучасні медіа – комплексний засіб опанування людиною оточуючого світу в його соціальних, психологічних, художніх, інтелектуальних аспектах, що викликає необхідність вивчення і врахування дидактичного потенціалу нових медіа.

Дослідження інтеграції медіа в освітній процес показують, що відбувається теоретичне осмислення світового досвіду у цій галузі освіти. У засвоєнні такого досвіду, безумовно, багато позитивних моментів, а саме: запозичення нових методик, методів викладання, вивчення теоретичних підходів тощо. Поряд з цим, при запозиченні цілей, задач необхідне обов'язкове врахування української дійсності і менталітету. Тому активізація досліджень у цій галузі є актуальним напрямом розвитку сучасної педагогіки. Перспективними напрямами досліджень є інтеграція медіа у викладання дисциплін різних циклів.

Питання для самоконтролю:

1. Що покладено в основу різноманітних класифікацій медіа?
2. Яким чином автор виділив 3 групи медіатехнологій?
3. В залежності від каналу по якому поступає медіаінформація, які технології?
4. Яким чином можна вдосконалити процес професійної підготовки майбутніх вчителів?
5. Що таке "педагогічні умови формування професійної компетентності майбутнього вчителя"?
6. Які педагогічні умови формування професійної компетентності майбутнього вчителя виділив автор?
7. Що відноситься до тематичних (психолого-педагогічних, організаційно-педагогічних, організаційно-дидактичних, мотиваційних) умов формування професійної компетентності майбутнього вчителя?
8. Якими методичними принципами належить керуватися при проектуванні застосування технології SMART Board у навчальному процесі ?
9. Опишіть основні етапи методичної підготовки вчителя до використання технології SMART Board на уроці.
10. Якими педагогічними вміннями повинен володіти вчитель для того, щоб використання технології SMART Board у навчальному процесі сприяло

розвиткові й формуванню стійких пізнавальних інтересів учнів?

11. Опишіть особливості відтворення статичних навчальних елементів засобами SMART Board.

12. Опишіть особливості відтворення динамічних навчальних елементів засобами SMART Board.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Обеспечение систем обработки информации программное. Термины и определения. ГОСТ 19781-90. – [Чинний від 1992-02-01 до 2007-12-10] – 16 с.– (Міждержавний стандарт).
2. Systems and software engineering – Software Life Cycle Processes. ISO 12207:2008. – [Чинний від 2008-02-01] – II, 122 с.– (Міжнародний стандарт).
3. Бьярне Страуструп. Программирование. Принципы и практика использования C++; пер. с англ. Д.Клюшин. Москва: Вильям,. 2011. – 1248 с.
4. Alistair Cockburn. Methodology per project. Humans and Technology Technical Report, TR 99.04, Oct.1999 7691 Dell Rd, Salt Lake City, UT 84121 USA. [Електронний ресурс]. Режим доступу: http://alistair.cockburn.us/Methodology+per+project.
5. IEEE Standard Glossary of Software Engineering Terminology, Глосарій. IEEE Std 610.12-1990. – (Галузевий стандарт).
6. Автоматизированные системы. Стадии создания. ГОСТ 34.601-90. – [Чинний від 1991-01-01] – 10 с.– (Міждержавний стандарт).
7. Техническое задание на создание автоматизированной системы. ГОСТ 34.602-89 – [Чинний від 1990-01-01] – 12 с.– (Міждержавний стандарт).
8. Spiral model. [Електронний ресурс]. Режим доступу: http://en.wikipedia.org/wiki/Spiral_model.
9. Standard Glossary of terms used in Software Testing. Version 1.2, ISTQB, 2006. [Електронний ресурс]. – Режим доступу: www.istqb.org/downloads/glossary.
10. 1061-1998 IEEE Standard for Software Quality Metrics Methodology. – (Галузевий стандарт).
11. Проектный треугольник. [Електронний ресурс]. Режим доступу: http://office.microsoft.com/ru-ru/project-help/HA010351692.aspx.
12. Спиральная модель. [Електронний ресурс]. Режим доступу: http://ru.wikipedia.org/wiki Спиральная_модель.
13. Кент Бек. Экстремальное программирование. – СПб.: Изд-во «Питер», 2002. – 224 с.
14. Брукс Ф. Как проектируются и создаются программные комплексы. Мифический человеко-месяц. Очерки по системному программированию. – СПб.: Изд-во «Символ-Плюс», 2000. – 304 с.
15. ДСТУ ISO 9000:2007. Системи управління якістю. Основні положення та словник термінів. – К.: Держспоживстандарт, 2008. – [Чинний від 2008-01-01] – 35 с.– (Державний стандарт).

16. Alistair Cockburn. Methodology per project. Humans and Technology Technical Report, TR 00.04, Jan.00. [Електронний ресурс]. Режим доступу: http://alistair.cockburn.us.

17. Alistair Cockburn, Laurie Williams. The Costs and Benefits of Pair Programming. Proceedings of the First International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2000). [Електронний ресурс]. Режим доступу: http://collaboration.csc.ncsu.edu/laurie/Papers/ XPSardinia.PDF.

18. Сэм Канер. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бізнес-приложений/ Сэм Канер, Джек Фолк, Енг Кек Нгуен; пер. с англ. – К.: Изд-во «ДиаСофт», 2001. – 544 с.

19. IEEE Guide to the Software Engineering Body of Knowledge (SWEBOK), 2004. – (Галузевий стандарт). [Електронний ресурс]. – Режим доступу: http://www.computer.org/portal/web/swebok/htmlformat.

20. ISO/IEC 15288 Systems and software engineering - System life cycle processes. – [Чинний від 2008-03-18] – 70 с.– (Міжнародний стандарт).

21. ДСТУ ISO 9001:2009. Системи управління якістю. Вимоги. – К.: Держспоживстандарт, 2009. – [Чинний від 2009-06-22] – 80 с.– (Державний стандарт).

22. M.C. Paulk, C.V. Weber, B. Curtis, M.B. Chrissis et al The Capability Maturity Model: Guidelines for Improving the Software Process. Addison-Wesley, Boston. 1995. – 456 с.

23. OMG Unified Modeling Language Specification Version 1.5, March 2003 formal/03-03-01. [Електронний ресурс]. Режим доступу: www.omg.org.

24. Буч Г., Рамбо Дж., Джекобсон А. UML. Руководство пользователя. – СПб.: Изд-во «ДМК-Пресс», «Питер», 2001. – 432 с.

25. Patricia Griffith Friel and Thomas M. Blinn (1989). "Automated IDEF3 and IDEF4 Systems Design Specification Document". Technical report. NASA Johnson Space Center. [Електронний ресурс]. Режим доступу: http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/ 19910023489_1991023489.pdf.

26. Шеер А.-В. ARIS - моделирование бизнес-процессов. – Москва: Вильямс, 2009. – 224 с.

27. Якобсон А., Буч Г., Рамбо Дж. Унифицированный процесс разработки программного обеспечения. – СПб.: Изд-во «Питер», 2002. – 492 с.

28. Introduction to the Microsoft Solutions Framework. [Електронний ресурс]. Режим доступу: http://technet.microsoft.com/en-us/library/bb497060.aspx.

29. Agile manifesto. [Електронний ресурс]. Режим доступу:

http://agilemanifesto.org.

30. Agile software development. [Електронний ресурс]. Режим доступу: http://en.wikipedia.org/wiki/ Agile_software_development.

31. C. Alexander et al. A pattern language: towns, buildings, construction. New York: Oxford University Press, 1977. – 120 с.

32. Введение в паттерны проектирования. [Електронний ресурс]. Режим доступу: http://cpp-reference.ru/patterns/ introduction.

33. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб.: Изд-во «Питер», 2007. – 366 с.

34. Buschmann F., Meunier R., Rohnert H., Sommerlad P., Stal M. Pattern-Oriented Software Architecture, Volume 1, A System of Patterns. Wiley, 1996. – 476 с.

Навчальний посібник

Проектування та розробка мультимедійних освітніх інформаційних систем

Відповідальний за випуск Г.О. Шліхта
Комп'ютерне верстання Г.О. Шліхта

Підп. до друку 24.03.2019, поз.

Формат 60х84/16. Ум. друк. арк. ___.
Обл.-вид. арк. __.
Тираж 20 пр.
Зам. №
Собівартість вид. грн к.

Видавець і виготовлювач Рівненський державний гуманітарний університет