МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІСЬКОГО ГОСПОДАРСТВА імені О. М. БЕКЕТОВА**

І. А. Чуб, М. В. Новожилова, М. П. Пан

ІННОВАЦІЙНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

КОНСПЕКТ ЛЕКЦІЙ

(*для студентів денної і заочної форм навчання освітнього рівня «магістр»
за спеціальністю 122 – Комп'ютерні науки,
освітньо-професійна програма «Комп'ютерні науки. Управління проектами»*)

**Харків
ХНУМГ ім. О. М. Бекетова
2018**

Чуб І. А. Інноваційні інформаційні технології: конспект лекцій (для студентів денної і заочної форм навчання освітнього рівня «магістр» за спеціальністю 122 – Комп'ютерні науки, освітньо-професійна програма «Комп'ютерні науки. Управління проектами» / І. А. Чуб, М. В. Новожилова, М. П. Пан; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків: ХНУМГ ім. О. М. Бекетова, 2018. – 119 с.

Автори:
д-р техн. наук, проф. І. А. Чуб,
д-р фіз.-мат. наук, проф. М. В. Новожилова,
канд. техн. наук, доц. М. П. Пан

Рецензенти:

І. В. Гребеннік, доктор технічних наук, професор, завідувач кафедри системотехніки (Харківський національний університет радіоелектроніки),

С. В. Яковлєв, доктор фізико-математичних наук, професор (Національний аерокосмічний університет імені М. Є. Жуковського «Харківський авіаційний інститут»),

Рекомендовано кафедрою прикладної математики і інформаційних технологій, протокол № 8 від 29.11.2018.

Конспект лекцій складено з метою допомогти студентам спеціальності 122 – Комп'ютерні науки освітньо-професійної програми «Комп'ютерні науки. Управління проектами» під час підготовки до занять, заліків і екзаменів.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ОЗНАК, СИМВОЛІВ,ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

ІІТ – інноваційні інформаційні технології

ІКТ – інформаційно-комунікаційні технології

ІС – інформаційна система

ІТ – інформаційна технологія

ОПР – особа, що приймає рішення

ПЗ – програмне забезпечення

СКБД – система керування базою даних

СППР – система підтримки прийняття рішень

СУП – система управління проектами

CMM – Capability Maturity Model – модель зрілості можливостей

EDMS – Electronic Document Management – система управління документами підприємства

CSS – Cascading Style Sheets – каскадні таблиці стилів

ERP – Enterprise Resource Planning – планування ресурсів підприємства

HTML – Hyper Text Markup Language – мова розмітки гіпертексту

HTTP – Hyper Text Transfer Protocol – протокол передачі гіпертексту

FTP – File Transfer Protocol – протокол передачі файлів по мережі

OLTP – OnLine Transaction Processing – обробка транзакцій в реальному часі

OLAP – Online Analytical Processing – інтерактивна аналітична обробка

RDF – Resource Description Framework – структура опису ресурсу

SQL – Structured Query Language – мова структурованих запитів

WBS – Work Breakdown Structure – структура декомпозиції робіт

XML – eXtensible Markup Language – розширювана мова розмітки

ВСТУП

Стрімкий розвиток і розповсюдження нових інформаційних технологій набуває сьогодні характеру глобальної інформаційної революції, що здійснює зростаючий вплив на політику, економіку, управління, фінанси, науку, культуру та інші сфери людської життєдіяльності у рамках національних кордонів і світі у цілому. Інформаційно-комунікаційні технології сьогодні – один з найважливіших факторів, що впливають на формування суспільства XXI століття, надають унікальні можливості у сфері пересування капіталу, товарів і послуг, стають основою формування нового типу економіки – «економіки знань», «інформаційної економіки», «кіберекономіки». Інформація і знання стає стратегічним ресурсом, масштаби використання якого – один з факторів соціально-економічного розвитку держави та її повноцінного входження у світову економіку.

Враховуючи вищевикладене, мету навчальної дисципліни «Інноваційні інформаційні технології» можна сформулювати в такий спосіб: надання теоретичних та практичних знань з використання сучасного програмного забезпечення та міжнародної мережі Інтернет для розв'язання задач управління проектами.

Завдання дисципліни – системо-логічне вивчення інноваційних інформаційних технологій обробки критичної інформації, що в кінцевому результаті сприятиме професійній адаптації в сучасному інформаційному просторі.

Предметом вивчення дисципліни «Інноваційні інформаційні технології» є теоретичні принципи та концепції, які формують системно-інформаційний підхід до аналізу навколишнього світу, вивчення інформаційних процесів, методів і способів отримання, обробки, передачі і збереження інформації з використанням сучасних комп'ютерних технологій.

У результаті вивчення навчальної дисципліни студент повинен:

знати: і системно застосовувати методи аналізу та моделювання прикладної області, виявлення інформаційних потреб і збору вихідних даних для проектування програмного забезпечення.

зміти: обґрунтовувати вибір методів формування вимог до програмної системи, розробляти, аналізувати та систематизувати вимоги розробляти і оцінювати стратегії проектування програмних засобів; набувати нові наукові і професійні знання, вдосконалювати навички, прогнозувати розвиток програмних систем та інформаційних технологій.

мати такі компетентності:

– здатність аналізувати предметні області, формувати, аналізувати та моделювати вимоги до програмного забезпечення (спеціальна компетентність 1 (СК 1) освітньо-професійної програми Комп'ютерні науки. Управління проектами);

– здатність розвивати та реалізовувати нові конкурентоспроможні ідеї в сфері інформаційних технологій (СК 3);

– здатність розробляти і координувати процеси, фази та ітерації життєвого циклу програмних систем на основі застосування відповідних моделей, методів та технологій розробки програмного забезпечення (СК 6);

– здатність застосовувати та критично оцінювати нові методології управлінні проектами, ґрунтуючись на фахових у цих областях наукових літературних джерелах (СК 7);

– здатність використовувати сучасні інформаційні технології для управління проектами (СК 8);

– здатність аналізувати існуючі бізнес-процеси, проектувати сучасні ефективні бізнес-процеси з використанням принципів інформаційних технологій (СК 9).

Вивчення цієї дисципліни базується на відомостях з дисципліни «Методологія управління проектами та програмами». На результатах

вивчення цієї дисципліни базується дисципліна «Інтелектуальні системи підтримки прийняття рішень в управлінні проектами».

Методи організації та здійснення навчально-пізнавальної діяльності (за джерелом передачі навчальної інформації) є такими: словесні – лекції; наочні – ілюстрації, презентації; практичні – вправи, навчальна праця, практичні роботи.

Під час вивчення дисципліни використовуються індуктивні, дедуктивні та аналітичні методи передачі та сприймання навчальної інформації.

Методи самостійного оволодіння знаннями студентів, формуванням умінь і навичок: продуктивні – проблемні; репродуктивні – пояснювально-ілюстративні.

Методи, що сприяють успішному засвоєнню знань, умінь: розв’язання типових задач, виконання вправ, конспектування лекцій, складання математичних моделей, розробка алгоритмів, програмування.

У якості методів контролю виступають: тестування, тобто опитування студентів за кожним змістовим модулем, контроль виконання практичних завдань, виконання залікових модульних завдань, самоконтроль, а також практична перевірка умінь і навичок за темами.

Підсумковий контроль проводиться за екзаменаційними білетами.

1 ОСНОВНІ НАПРЯМИ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ В УПРАВЛІННІ ПРОЕКТАМИ

1.1 Генеза інформаційних технологій

Основою інформаційного суспільства є інформаційно-комунікаційні технології (ІКТ). Розвиток та впровадження ІКТ в освіті постійно досліджуються науковцями міжнародних організацій: ЮНЕСКО, ООН, Європейського Союзу, Ради Європи та інших.

Поняття «технологія» походить з грецької technê – мистецтво, майстерність, ремесло + logos – знання, вчення, тобто «знання про майстерність», але на сьогодні дане поняття тлумачать по- різному.

Визначення 1.1. Технологією є сукупність способів обробки чи переробки матеріалів, інформації, виготовлення виробів, проведення різних виробничих операцій, надання послуг тощо.

Надамо ще одне визначення.

Визначення 1.2. Технологія – це сукупність методів, засобів і реалізації людьми конкретного складного процесу шляхом поділу його на систему послідовних взаємопов’язаних процедур і операцій, що виконуються більш або менш однозначно і мають на меті досягнення високої ефективності певного виду діяльності

Поняття інформаційної технології з’явилося з виникненням інформаційного суспільства, основою соціальної динаміки в якому є не традиційні, матеріальні, а інформаційні ресурси – знання, наука, організаційні чинники, інтелектуальні здібності людей, їх ініціатива і творчість.

Визначення 1.3. Інформаційна технологія – це сукупність методів та технічних засобів, що використовується для збирання, створення, організації, зберігання, опрацювання, передавання, подання і використання інформації,

розширюючи знання людей і розвиваючи їх можливості в управлінні технічними і соціальними процесами.

Таким чином, інформаційна технологія – це сукупність засобів і методів, за допомогою яких здійснюється процес обробки інформації.

Розглянемо поняття «Інноваційні інформаційні технології».

Уведемо узагальнене означення.

Визначення 1.4. Інноваційні технології – радикально нові чи вдосконалені технології, які істотно поліпшують умови виробництва або самі виступають товаром та носять елементи наукової новизни (нового знання).

Як результат – інноваційні (нові) інформаційні технології (ІІТ) – це сукупність засобів і методів по отриманню, обробці, зберіганню та передачі інформації з використанням електронної техніки некваліфікованим користувачем.

Визначення 1.5. Інформаційні технології, що базуються на використанні персональних комп'ютерів, комп'ютерних мереж і засобів зв'язку, утворюють інформаційно-комунікаційні технології.

Терміни «інформаційні технології» та «інформаційно-комунікаційні технології» з'являються в англомовних джерелах з початку 70-х років. Вираз information technologies з'являється у зарубіжній літературі в 1964 р. в статті Джона Дайболда (John Diebold).

Уперше докладний аналіз концептуального апарату і перспектив розвитку інформаційних технологій провів академік В. М. Глушков, який визначив інформаційну технологію як людино-машинну технологію збирання, оброблення та передавання інформації.

Інформаційно-комунікаційні технології (ІКТ, від англ. information and communications technology – ICT) – іноді вживають як синонім до інформаційних технологій (ІТ), хоча ІКТ є більш загальним терміном, що підкреслює роль уніфікованих технологій та інтеграцію телекомунікацій (телефонних ліній, бездротових з'єднань), комп'ютерів, програмного забезпечення (операційні системи, мережеві протоколи, пошукові системи

тощо), накопичувальних та аудіовізуальних систем, що надають можливість користувачам створювати та зберігати данні, змінювати їх, передавати ці данні іншим користувачам.

1.2 Інформаційні технології і інформаційні системи

Одне зі значень терміну «система» загалом – це сукупність елементів, які працюють разом під час виконання певного завдання.

Різновидом системи є інформаційна система, тобто організований набір елементів, що збирає, обробляє, передає, зберігає та надає дані.

До інформаційної системи входять люди, обладнання, процеси, процедури, дані та операції. Наприклад, у США під інформаційними системами розуміють усі письмові й електронні форми поширення інформації, обробки даних та обміну ідеями. Отже, до них можна віднести всі форми письмового спілкування всередині підприємства (доповіді, звіти, бюлетені та службові записки), а також всі електронні інформаційні засоби (електронну пошту та селекторні і телевізійні наради).

Таким чином, термін «Інформаційна система» в цьому разі потрібно вживати у двох значеннях:

а) як певний метод, суть якого – у раціональному поєднанні і впорядкованості всіх елементів у часі і просторі в такий спосіб, щоб кожний з них сприяв успіху діяльності всього об'єкта. З цим трактуванням пов'язано розуміння координації і синхронізації дій персоналу управління, поєднаних з метою досягнення поставлених завдань;

б) як об’єкт, який має достатньо складну, певним чином упорядковану внутрішню структуру (наприклад, виробничий процес).

Кожна інформаційна система має такі компоненти:

– структура системи – множина елементів системи і взаємозв'язків між ними. Приклад: організаційна і виробнича структура підприємства;

– функції кожного елемента системи. Приклад: управлінські функції –

прийняття рішень у певних структурних підрозділах підприємства;

– вхід і вихід кожного елемента і системи загалом. Приклад: матеріальні або інформаційні потоки, які надходять у систему або вводяться нею;

– мета й обмеження системи та її окремих елементів. Приклад: досягнення максимального прибутку; фінансові обмеження.

Отже, інформаційна система містить не тільки апаратну і програмну частини, але і інформацію, яка міститься в системі, специфічні алгоритми її обробки, та спеціалістів, котрі взаємодіють із системою.

Будь-яка ІС характеризується наявністю інформаційної технології перетворення вихідних даних у результатну інформацію. ІТ не може існувати окремо від технічного і програмного середовища.

Інформація – це реальний виробничий ресурс поряд з іншими матеріальними ресурсами. Саме через усвідомлення цієї фундаментальної ідеї виникла інформатика як наука про виробництво інформації. Причому виробництво інформації та її верхнього рівня – знань – сьогодні має вирішальний вплив на модифікацію і створення нових промислових технологій.

В інформаційній технології можна виокремити дві складники:

а) здатність генерувати за запитом інформаційний продукт;

б) засоби доставки цього інформаційного продукту в зручний час і в зручній для користувача формі.

Кожна інформаційна технологія орієнтована на обробку інформації певних видів: даних (системи програмування й алгоритмічні мови, системи управління базами даних – СУБД, електронні таблиці); текстової інформації (текстові процесори і гіпертекстові системи); статичної графіки (графічні редактори); знань (експертні системи); динамічної графіки, анімації, відеозображення, звуку (інструментарій створення мультимедійних додатків, що охоплює засоби анімації й управління відеозображенням та звуком). Інформаційні технології відрізняються за типом інформації, яка обробляється, але можуть і об'єднуватися, утворювати інтегровані системи,

що мають різні технології.

1.3 Бази даних. Пошукові системи

Визначення 1.6. База даних (БД) – впорядкований набір логічно взаємопов'язаних даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів.

Головне завдання БД – гарантоване збереження значних обсягів інформації та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином, БД складається з двох частин: збереженої інформації та системи керування нею.

З метою забезпечення ефективності доступу записи даних організовують як множину фактів як елементів даних.

Дані (date) – це компонент БД, над яким виконуються дії в ІС.

Визначення 1.7. Пошукова систе́ма (або скорочено пошуковик) – певна база даних, онлайн-служба (апаратно-програмний комплекс з веб-інтерфейсом), що надає можливість пошуку інформації в Інтернеті. У просторіччі під *пошуковою системою* розуміють веб-сайт, на котрому розміщено інтерфейс (фронт-енд) системи. Програмною частиною пошукової системи є пошукова машина (*пошуко́вий рушій*) – комплекс програм, що забезпечує функціональність пошукової системи і, зазвичай, є комерційною таємницею компанії-розробника пошукової системи.

Існує близько 50 пошукових систем. Найбільш популярними є пошуковик Google – 86,30 % всіх запитів; Yahoo – 5,30 % всіх пошукових запитів.

1.4 Програмні засоби систем управління проектами

Універсальну архітектуру програмних засобів систем управління проектами (СУП) представлено на рисунку 1.1.

Основою автоматизації управління проектами традиційно розглядається система календарно-ресурсного планування. Також в контур управління проектом можуть залучатися ще ряд інших інформаційних систем, які функціонують на підприємстві і можуть використовуватися при вирішенні деяких завдань управління проектом (від статистичних пакетів до систем фінансового планування).

Завдання створення об'єднаного інформаційного простору з існуючих автономних інформаційних ресурсів вирішується на основі інтеграційного підходу.

Рисунок 1.1 – Архітектура програмних засобів СУП

Одним з підходів до створення сучасної інформаційної технології інтеграції даних є ERP (Enterprise Resource Planning) системи.

Визначення 1.8. Enterprise Resource Planning – планування ресурсів підприємства – це клас систем для управління виробництвом, трудовими ресурсами, фінансами та активами, орієнтованих на оптимізацію ресурсів.

В організаційній сфері інтеграційний підхід виражається в формуванні управлінських структур рівню ієрархії вище штатного розкладу (керівний комітет, група управління, робоча група), і організаційно-розпорядчих документів, що описують наскрізні процеси, що стосуються не тільки персоналу проекту, але і постійні структурні підрозділи підприємства (ресурсні підрозділи, фінансова служба, служба логістики, служба тощо).

В IT-області інтеграційний підхід виражається в необхідності створення контуру взаємопов'язаних продуктів, в якому СУП зв'язується з іншими системами підприємства інформаційними та користувальницькими інтерфейсами. В обох областях рішення найчастіше не є універсальними і розробляються під вимоги конкретних замовників

Виокремимо два основні напрями автоматизації СУП:

– автоматизація стандарту управління проектами підприємства;

– автоматизація функцій управління проектами.

Автоматизація стандарту управління проектами забезпечується засобами інформаційних технологій.

Стандарт управління проектами підприємства являє собою сукупність документів, що пояснюють чи передбачають, як і в якій послідовності, в які терміни, з використанням яких шаблонів треба виконувати ті чи інші дії в процесі управління проектами.

Одним із перспективних підходів в автоматизації проектами є організація стандарту у вигляді бази знань. Цей стандарт забезпечує:

– усі необхідні сервіси по оновленню і пошуку документів, з організації взаємозв'язків між документами, перехресних посилань тощо.

– колективну роботи, в яку включається не тільки проектна група, але і постійні підрозділи підприємства (ресурсні, функціональні, спеціалізовані тощо).

У стандарті можуть бути явно чи неявно закладені вимоги до автоматизації функцій управління проектами.

До основних сфер діяльності з управління проектами, що мають бути автоматизованими, відносяться:

– власне управління проектами, яке у вузькому сенсі зазвичай розуміється як календарно-ресурсне планування;

– формування та ведення бюджету проекту;

– управління документами – як управлінськими, так і є результатами виконання проекту;

– управління діловими процесами в проектах, включаючи процеси узгодження документів. Функціональні компоненти СУП та відповідні пакети прикладних програм подано на рисунку 1.2.

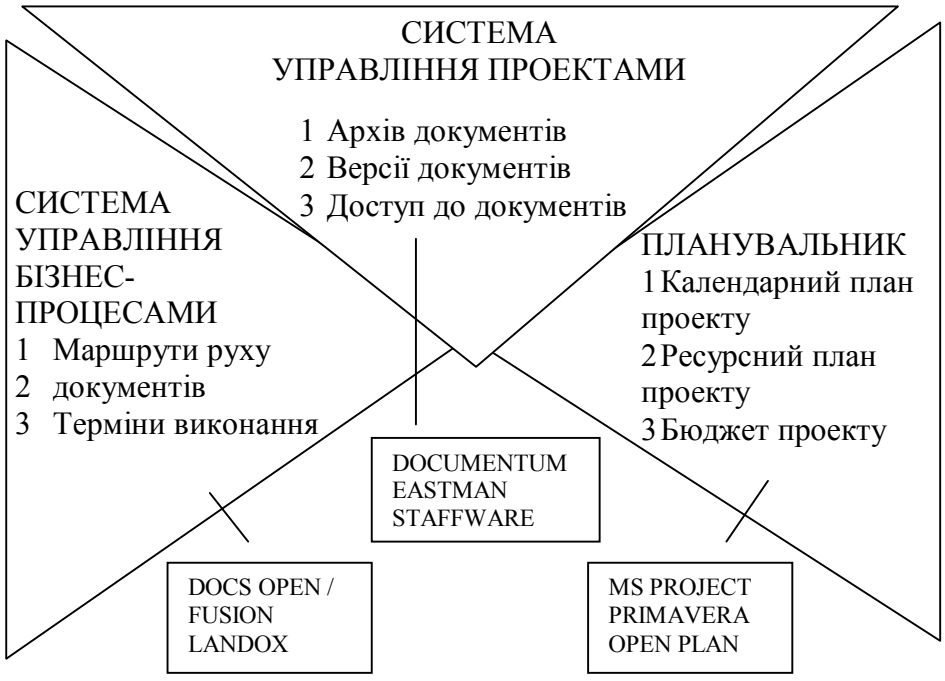


Рисунок 1.2 – Функціональні компоненти СУП

У частині календарно-ресурсного планування СУП повинна забезпечити:

– формування структури декомпозиції робіт (WBS-структури) необхідного ступеня деталізації;

– формування календарного плану, що містить тривалість робіт, їх обсяг та вартість, обмеження на дати початку та закінчення, а також технологічні залежності між роботами;

– формування обмежень за проектом, що визначають перелік трудових ресурсів, які передбачається використовувати в проекті із зазначенням доступної кількості в певний час;

– формування детального плану робіт, в якому роботам призначені ресурси – трудовитрати і матеріально-технічні ресурси;

– побудова звітів про стан проекту, в тому числі з використанням різних аналітик.

У частині фінансового планування СУП має забезпечити:

– планування та облік фінансових потоків, включаючи розрахунки із замовником та субпідрядниками

– формування завдань виконавцям і облік реально витраченого часу;

– облік непроектного і неробочого часу, відпусток та лікарняних листів;

– облік відрядних та адміністративних витрат.

В управлінні документами і діловими процесами проектів велике значення мають не тільки традиційні функції управління документами, такі як підтримка версій документів і історії роботи з ним, ведення архіву, авторизація доступу, підтримка зв'язків між документами (EDMS-функції).

Може бути, навіть більше значення набувають функції управління рухом документів і контролю термінів їх виконання (workflow-функції).

Управління документами реалізується з використанням базової функціональності промислових пакетів (Docs Open, Documentum).

Функції управління рухом документів і контролю термінів їх виконання реалізуються з використанням базової функціональності спеціалізованих програмних систем (Eastman) або промислових пакетів управління документами (Documentum).

Серед найбільш важливих функціональних можливостей відзначимо:

– наявність єдиної точки доступу до всієї інформації, що відноситься до проектів;

– створення нових документів з автоматичним зв'язуванням їх з конкретним фрагментом проекту (роботою, вузлом WBS) або з проектом в цілому;

– перехід до роботи з документами безпосередньо з середовища календарно-ресурсного планування, і навпаки;

– ініціювання процесу проходження (виконання) документа, отримання відміток про виконання, узгодження, ознайомленні або відхилення документа;

– контроль стану документа в процесі проходження ним певного бізнес-процесу, пов'язаного з управлінням проектом (наприклад, узгодження і приймання робіт).

2 СУТНІСТЬ ТА ЗАГАЛЬНІ ХАРАКТЕРИСТИКИ СУЧАСНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

2.1 Архітектура «клієнт-сервер»

Клієнт-сервер (англ. client-server) – це обчислювальна або мережева архітектура, в якій завдання або мережеве навантаження розподілені між постачальниками послуг – серверами, і замовниками послуг – клієнтами.

Фактично клієнт і сервер – це програмне забезпечення. Зазвичай ці програми розміщені на різних обчислювальних машинах і взаємодіють між собою через обчислювальну мережу за допомогою мережевих протоколів, але вони можуть бути розміщені також і на одній машині.

Програми-сервери очікують від клієнтських програм запити і надають їм свої ресурси у вигляді даних (наприклад, завантаження файлів за допомогою протоколів HTTP, FTP, BitTorrent, потокове мультимедіа або робота з базами даних) або у вигляді сервісних функцій (наприклад, робота з електронною поштою, спілкування за допомогою систем миттєвого обміну повідомленнями або перегляд веб-сторінок у всесвітній павутині).

Таким чином, сервером, як правило, виступає програма-сервер протоколу обміну гіпертекстовою інформацією HTTP, що відповідає на запити клієнтів.

З технологією HTTP тісно пов’язане таке поняття, як URL. Ця абревіатура розшифровується як Uniform Resource Locator, що можна вільно перекласти, як єдиний покажчик на ресурс. Фактично, це адреса документа.

Типовий для URL вигляд:

протокол://повне.ім’я.машини.або.адреса:порт/шлях

Тут «протокол» набуває такого значення:

HTTP – передача гіпертексту; FTP – протокол передачі файлів; TELNET – термінальний доступ; NEWS – новини Usenet; FILE – для доступу до локальних файлів.

Параметр «порт» можна не вказувати й тоді це порт, стандартний для даного протоколу. Для FTP використаються порти 20 і 21, для HTTP – 80, для TELNET – 23, для GOPHER – 70, NEWS – 119 тощо.

Параметр «шлях» специфічний для кожного протоколу, наприклад, для FTP – це шлях у файловій системі. Схожий зміст має цей параметр і для інших протоколів.

Узгодження типів переданих у рамках www документів виробляється за допомогою заголовків, якими обмінюються навігатор і WWW-сервер. Весь комплекс цих заголовків відомий як MIME, Multipurpose Internet Mail Extensions. Навігатор (браузер) повинен знати, якого типу документ він одержує, адже він повинен його інтерпретувати, показувати й взагалі щось із ним робити.

Оскільки одна програма-сервер може виконувати запити від множини програм-клієнтів, її розміщують на спеціально виділеній обчислювальній машині, налаштованій особливим чином, як правило, спільно з іншими програмами-серверами, тому продуктивність цієї машини має бути високою. Через особливу роль такої машини в мережі, специфіки її обладнання та програмного забезпечення, її також називають сервером, а машини, які виконують клієнтські програми, відповідно, клієнтами.

Розглянемо архітектуру клієнт/сервер у контексті створення баз даних. Загальна мета систем баз даних – це підтримка розробки і виконання додатків баз даних. Тому на високому рівні систему баз даних можна розглядати як систему з дуже простою структурою, що складається з двох частин (рис. 2.1) – сервера (чи машини бази даних) і набору клієнтів (чи зовнішнього інтерфейсу).

Сервер – це власне система керування базою даних (СКБД). Він підтримує всі основні функції СКБД, а саме: визначення даних, обробку

даних, захист і цілісність даних тощо. Зокрема, він надає повну підтримку на зовнішньому, концептуальному і внутрішньому рівнях, тому «сервер» у цьому контексті – це просто інше ім’я СКБД.

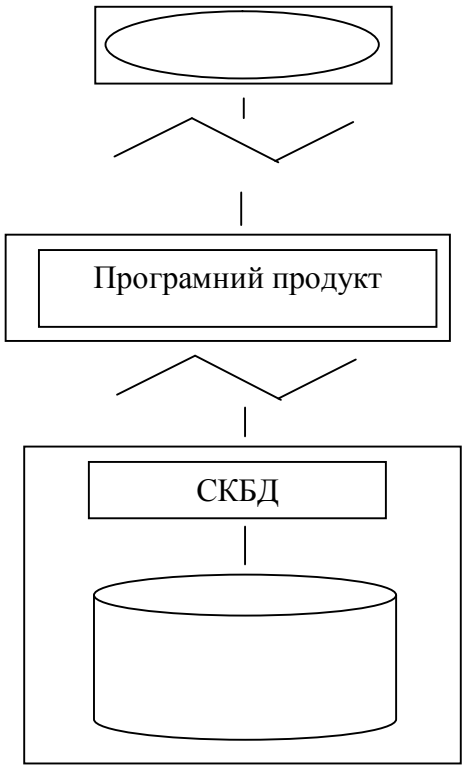


Рисунок 2.1 – Архітектура клієнт/сервер

Клієнти – це різні застосування, що виконуються «над» СКБД: застосування, котрі написані користувачами, і убудовані застосування, котрі надані постачальниками СКБД чи деякими сторонніми постачальниками програмного забезпечення (часто називаються інструментальними засобами).

Звичайно, з погляду користувачів, немає різниці між убудованими додатками і додатками, написаними користувачем – усі вони використовують інтерфейс сервера, а саме інтерфейс зовнішнього рівня.

Зауваження 2.1. Винятками є спеціальні «службові» додатки (застосування). Такі застосування іноді можуть працювати тільки безпосередньо на внутрішньому рівні системи. Такі утиліти швидше належать до безпосередніх компонентів СКБД, ніж до додатків у звичайному розумінні.

2.2 Розподілена обробка

Завдяки тому, що система в цілому може бути чітко розділена на дві частини (сервер і клієнти), з'являється можливість роботи цих двох частин – застосувань, котрі написані користувачами, і убудованих застосувань – *на різних машинах.*

Інакше кажучи, існує можливість розподіленої обробки. Розподілена обробка припускає, що окремі машини можна з'єднати комунікаційною мережею так, що обробка даних може бути розподіленою між декількома машинами мережі.

Розподілена обробка може бути найрізноманітнішою і здійснюватися на різних рівнях. Як відзначалося вище, в одному з простих випадків запускається сервер СКБД на одній машині і клієнтський додаток на іншій.

Термін «клієнт/сервер» став синонімом структури, відповідно до якої клієнт і сервер запускаються на різних машинах.

При цьому машина сервера може бути виготовлена на спеціальне замовлення, пристосована для роботи з СКБД («машина бази даних») і може забезпечити вищу продуктивність СКБД.

Машина клієнта може бути персональною станцією, яку пристосовано до потреб кінцевого користувача, і тому вона забезпечує кращий інтерфейс, повну відповідність вимогам, швидку реакцію та додаткові зручності у використанні.

Кілька різних машин клієнтів можуть мати спільну машину сервера. Тому однією базою даних можуть спільно користатися кілька окремих клієнтських систем (рис. 2.3).

Ще одна перевага виконання сервера і клієнта на окремих машинах – відповідність практичній роботі багатьох підприємств.

Машини клієнтів можуть мати свої власні дані, що зберігаються, а машина сервера може мати свої власні застосування. Тому кожна машина буде виступати в ролі сервера для одних користувачів і в ролі клієнта для

інших (рис. 2.4), тобто кожна машина підтримуватиме повну систему баз даних.

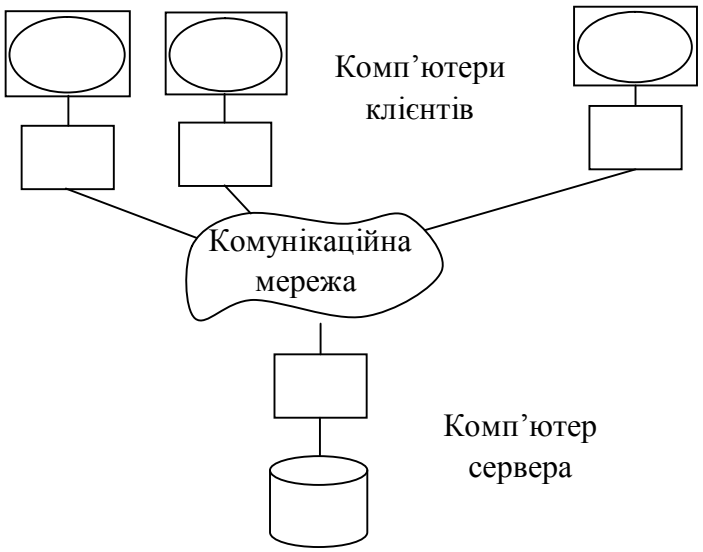


Рисунок 2.3 – Один сервер, багато клієнтів

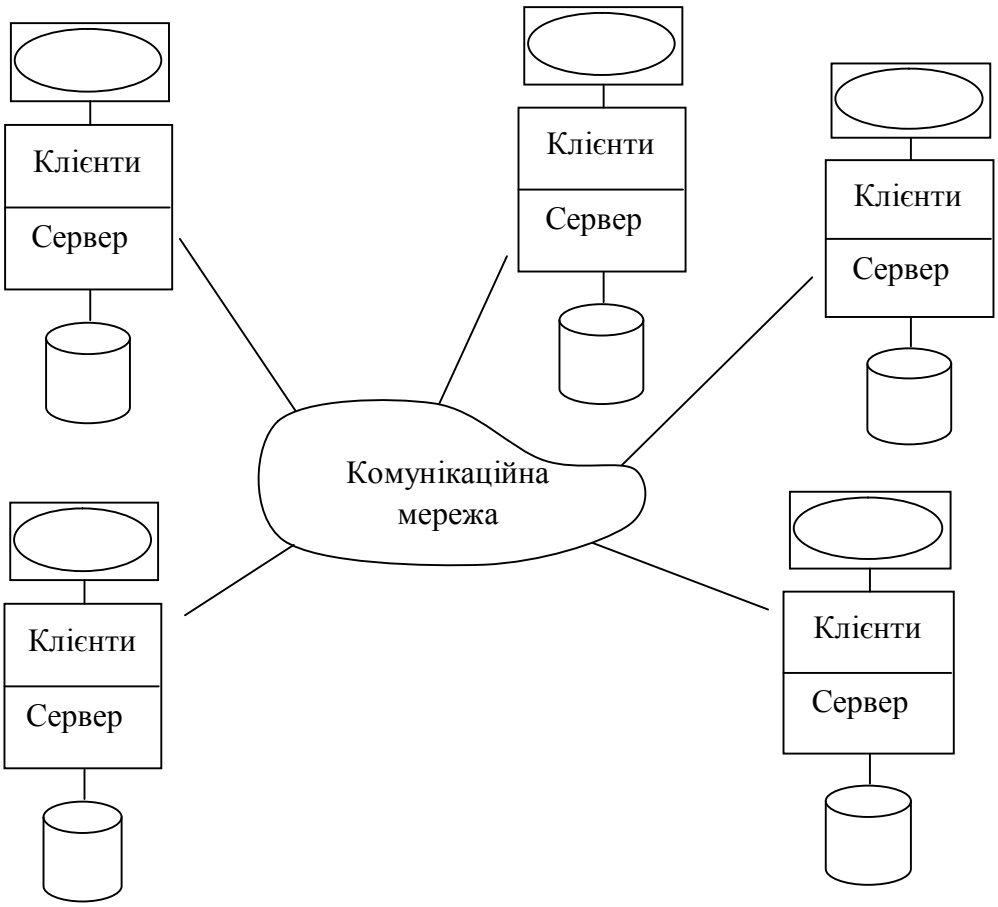


Рисунок 2.4 – Багато серверів, багато клієнтів

Остання перевага полягає в тому, що окрема машина клієнта може мати доступ до декількох різних машин серверів. Такий доступ в основному надається двома способами.

1. Клієнт може діставати доступ до будь-якої кількості серверів, кожен запит до бази даних має бути спрямований лише до одного сервера. У такій системі неможливо за один запит дістати комбіновані дані з двох чи більше серверів.

Крім того, користувач у такій системі має знати, на якій саме машині міститься певна частина даних.

2. Клієнт може дістати доступ до будь-якої кількості серверів одночасно (тобто за один запит можна одержати комбіновані дані з двох чи більше серверів). У цьому випадку сервери сприймаються клієнтом як один, і користувач може не знати, на якій саме машині міститься певна частина даних.

Другий випадок – це приклад системи, що звичайно називають розподіленою системою баз даних. Повна підтримка для розподілених баз даних означає, що окремий додаток може „прозоро” обробляти дані, розподілені на множині різних баз даних, керування якими здійснюють різні СКБД. Клієнти працюють на численних машинах з різними операційними системами, що з’єднані комунікаційними мережами. Тут поняття "прозоро" означає, що додаток виконує обробку даних так, ніби керування даними цілком здійснюється однією СКБД, що працює на окремій машині.

Відзначимо найважливіші переваги розподілених ІС.

– Надійність. Наявність множини розподілених компонентів, розташованих у різних місцях, робить АС несприйнятливою до локальних збоїв, коли один із компонентів відмовляє або стає недоступним для зв'язку з іншими.

– Ефективність. Багатофункціональні елементи розміщуються в місцях найчастішого використання і забезпечують більш швидкий доступ до даних, скорочуючи час відгуку і вартість зв'язку.

– Гнучкість. Можна поступово збільшити обчислювальну потужність завдяки об'єднанню кількох обчислювальних систем малої і середньої потужності.

2.3 Класифікація інформаційних систем

Різноманітність сфер і форм застосування сучасних інформаційних технологій породжує різноманітність способів їх класифікації.

За масштабністю інформаційні системи поділяються на такі групи:

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

Одиничні ІС (рис. 2.5) реалізуються, як правило, на автономному персональному комп'ютері без використання комп'ютерної мережі. Така система може містити декілька простих додатків із спільним інформаційним фондом. Подібні комплекси можуть бути створені за допомогою таких локальних систем управління базами даних як Clipper, FoxPro, Paradox, MS Access тощо.

Рисунок 2.5 – Класифікація ІС за масштабністю

Групові ІС орієнтовані на колективне використання інформації і найчастіше будуються на базі локальної обчислювальної мережі.

При розробці таких додатків найчастіше використовуються сервери баз даних (SQL-сервери) для робочих груп. Серед найбільш відомих таких серверів є Oracle, InterBase, Sybase, тощо.

Корпоративні ІС призначені для великих компаній і можуть підтримувати територіально віддалені вузли і мережі. Як правило, вони мають ієрархічну клієнт-серверну структуру зі спеціалізацією серверів. При розробці таких систем можуть використовуватись ті ж сервери баз даних, що й при розробці групових ІС. Для корпоративних систем найбільш поширеними є сервери Oracle, DB2, Microsoft SQL Server.

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

За сферою застосування інформаційні системи можна умовно поділити на чотири групи:

– системи обробки трансакцій (операцій з базою даних) – призначені для ефективного відображення предметної області в будь-який момент часу (OLTP – OnLine Transaction Processing);

– системи підтримки прийняття рішень – за допомогою комплексу запитів здійснюється аналіз даних в різних аспектах: часовому, просторовому тощо;

– інформаційно-довідкові системи базуються на гіпертекстових документах і мультимедійних засобах. Найбільший розвиток такі системи отримали в мережі Інтернет;

– офісні інформаційні системи – призначені для перетворення паперових документів в електронні, автоматизації діловодства і управління документообігом.

За способом організації автоматизовані ІС можуть бути класифіковані наступним чином: на основі архітектури файл-сервер; на основі архітектури

клієнт-сервер; на основі багаторівневої архітектури; на основі Інтранет-технологій.

За рівнем або сферою діяльності: державні, територіальні (регіональні), галузеві, підприємств або установ, технологічних процесів.

Державні ІС призначені для вирішення господарських проблем країни на базі використання обчислювальних комплексів та економіко-математичних методів. Наприклад:

– автоматизована система державної статистики – основне джерело статистичної інформації;

– автоматизована система планових розрахунків (Міністерство економіки України);

Для розв’язання задач статистичного аналізу даних на сучасному світовому ринку існує більше 1 000 пакетів прикладних програм. Це такі як STATISTIKA, STATGRAPHICS, WinSTAT, КВАЗАР, а також статистичні експертні системи, зокрема, СТАТЗКС, Statistical Navigator Pro.

За типом підтримки, яку ці системи забезпечують в організації управління, системи можуть бути поділені на такі групи:

– системи обробки операцій, які реєструють та обробляють дані, одержані внаслідок ділових операцій. Воно може проводитись або способом пакетного оброблення даних, або в масштабі реального часу;

– автоматизовані системи управління технологічними процесами (АСУТП), що приймають рішення з типових питань, таких, як управління виробничим процесом;

– системи співробітництва на підприємстві, які використовують комп'ютерні мережі для забезпечення зв'язку, координації та співробітництва відділів і робочих груп, що беруть участь у процесі;

– інформаційні менеджерські системи – системи забезпечення менеджменту, що продукують заздалегідь визначені звіти, подають відображення даних і результати вжитих заходів на періодичній чи

інцидентній основі або за запитом;

– системи підтримки прийняття рішень – ІС, які використовують моделі прийняття рішень.

Корпоративні системи, системи підтримки прийняття рішень та експертні системи характеризують новий етап автоматизації управління підприємством.

Системи підтримки прийняття рішень (СППР або англ, Decision Support System – DSS) призначені для підтримки прийняття рішень керівників різного рівня при вирішенні неструктурованих і слабо структурованих проблем і використовують нові засоби інформаційних технологій - програмні агенти, сховища і вітрини даних, OLAP-системи тощо.

СППР використовують не тільки загальне інформаційне забезпечення, а й загальне математичне забезпечення - бази моделей.

2.4 Структуровані та неструктуровані системи

Класифікація ІС у менеджменті організації сприяє виявленню найбільш характерних рис, притаманних ІС та дозволяє здійснювати ефективне управління. Існують різні класифікації, що переслідують певні цілі.

Класифікацію ІС за ознакою структурування задач наведено на рис.2.6.

Розрізняють три типи задач, для яких створюються інформаційні системи: структуровані (що формалізуються), неструктуровані (що не формалізуються) і частково структуровані.

Структурована задача – це задача, де відомі всі її елементи та взаємозв'язки між ними.

Неструктурована задача – це задача, в якому неможливо виділити елементи і встановити між ними зв'язки.

У структурованій задачі зміст відображається у формі математичної моделі, що має точний алгоритм розв’язання. Подібні задачі зазвичай доводиться вирішувати багаторазово, і вони носять рутинний характер.

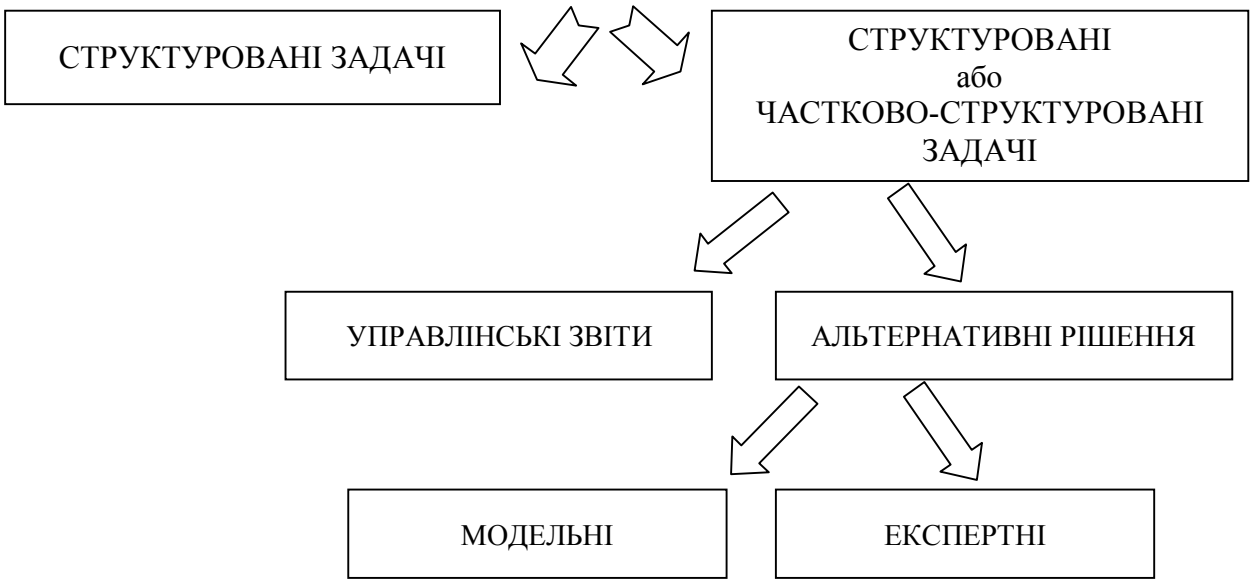


Рисунок 2.6 – Класифікація інформаційних систем за ознакою структурування задач

Метою використання інформаційної системи для розв’язання структурованих завдань є повна їх автоматизація, тобто зведення ролі людини до нуля.

Розв’язання неструктурованих задач через неможливість створення математичного опису та розробки алгоритму пов'язане з великими труднощами. Можливості використання тут інформаційної системи невеликі.

Рішення в таких випадках приймається людиною з евристичних міркувань на основі свого досвіду і, можливо, непрямої інформації з різних джерел.

Зауважимо, що в практиці роботи будь-якої організації існує порівняно небагато повністю структурованих або зовсім неструктурованих задач.

Про більшість задач можна сказати, що відома лише частина їх елементів і зв'язків між ними. Такі задачі називаються частково структурованими. У цих умовах можна створити інформаційну систему.

Отримана в ній інформація аналізується людиною, яка буде відігравати визначальну роль. Такі інформаційні системи є автоматизованими, так як в їх функціонуванні бере участь людина.

Інформаційні системи, що використовуються для розв'язання частково структурованих задач, підрозділяються на два види:

1. ІС, що орієнтовані головним чином на обробку даних (пошук, сортування, агрегування, фільтрацію) та створення управлінських звітів. Використовуючи відомості, що містяться в цих звітах, ОПР (особа, що приймає рішення);

2. ІС, що застосовуються для розробки можливих альтернатив рішення. Прийняття рішення при цьому зводиться до вибору однієї із запропонованих альтернатив.

ІС, що створюють управлінські звіти, забезпечують інформаційну підтримку користувача, тобто надають доступ до інформації в базі даних і її часткову обробку. Процедури маніпулювання даними в ІС мають забезпечувати наступні можливості:

– суміщення даних, що одержуються з різних джерел;

– швидке додавання або виключення того чи іншого джерела даних і автоматичне перемикання джерел при пошуку даних;

– управління даними з використанням можливостей систем управління базами даних.

Логічну незалежність даних цього типу від інших баз даних, що входять у підсистему інформаційного забезпечення. Запити до ІС і, отже, процедури формування відповіді на них можна поділити на рутинні і нерутинні. Рутинні процедури характеризуються заданістю вхідної і вихідної інформації, а також визначеністю алгоритму отримання останньої з першої. Виділення рутинних завдань і процедур обробки інформації дозволяє їх формалізувати, а надалі й автоматизувати, що залежить тільки від того, чи спроможні інформаційні технології, що використовуються в організації,

забезпечити інфраструктуру для цього. Якщо рутинні повсякденні дії автоматизовані, то набагато простіше обробляти нерутинні випадкові запити.

При створенні або класифікації ІС завжди виникають проблеми, що пов'язані з формалізованим описом (математичним та алгоритмічним) поточних задач. Від ступеня формалізації значним чином залежать ефективність роботи всієї системи, а також рівень автоматизації, обумовлений ступенем участі людини при ухваленні рішення на основі отриманої інформації.

Чим точніше математичний опис задачі, тим вище можливості комп'ютерної обробки даних і тим менше ступінь участі людини в процесі її рішення. Це і визначає ступінь автоматизації задачі.

Інформаційні системи, що розробляють альтернативи рішень, можуть бути модельними або експертними.

Модельні інформаційні системи надають користувачеві математичні, статистичні, фінансові та інші моделі, використання яких полегшує вироблення і оцінку альтернатив рішення. Користувач може одержати відсутню йому для прийняття рішення інформацію шляхом встановлення діалогу з моделлю в процесі її дослідження.

Основними функціями модельної інформаційної системи є:

– можливість роботи в середовищі типових математичних моделей, включаючи рішення основних задач моделювання типу «як зробити, щоб?», «Що буде, якщо?», аналіз чутливості та ін;

– досить швидка та адекватна інтерпретація результатів моделювання;

– оперативна підготовка і коректування вхідних параметрів і обмежень моделі;

– можливість графічного відображення динаміки моделі;

– можливість пояснення користувачеві необхідних кроків формування та роботи моделі.

Експертні інформаційні системи забезпечують вироблення та оцінку можливих альтернатив користувачем за рахунок створення експертних систем, пов'язаних з обробкою знань. Експертна підтримка прийнятих користувачем рішень реалізується на двох рівнях.

Робота першого рівня експертної підтримки виходить з концепції «типових управлінських рішень», відповідно до якої часто виникають у процесі управління проблемні ситуації можна звести до деяких однорідних класів управлінських рішень, тобто до деякого типового набору альтернатив. Для реалізації експертної підтримки на цьому рівні створюється інформаційний фонд зберігання та аналізу типових альтернатив.

Якщо наявна проблемна ситуація не асоціюється з класами типових альтернатив, в роботу повинен вступати другий рівень експертної підтримки управлінських рішень. Цей рівень генерує альтернативи на базі наявних в інформаційному фонді даних, правил перетворення і процедур оцінки синтезованих альтернатив. Прийняття рішення при цьому зводиться до вибору однієї із запропонованих альтернатив.

Тип інформаційної системи залежить від того, чиї інтереси вона обслуговує і на якому рівні управління. На рис 2.7. наведено один з можливих варіантів класифікації інформаційних систем за функціональною ознакою з урахуванням рівнів управління та рівнів кваліфікації персоналу.

Розглядаючи систему управління, необхідно виділити три рівні управління: стратегічний, тактичний і оперативний. На кожному з цих рівнів управління є свої завдання, при вирішенні яких виникає потреба у відповідних даних, отримати ці дані можна шляхом запитів в інформаційну систему.

Що вище за значущістю рівень управління, то менший обсяг робіт, виконуваних фахівцем і менеджером за допомогою ІС.

Однак при цьому зростають складність і інтелектуальні можливості ІС та її роль в прийнятті менеджером рішень. Будь-який рівень управління

потребує інформації із всіх функціональних підсистем, але в різних обсягах і з різним ступенем узагальнення.

Інформаційна система оперативного рівня підтримує фахівців-виконавців, обробляючи дані про угоди і події (рахунки, накладні, зарплата, кредити, потік сировини і матеріалів).

Призначення ІС на цьому рівні – відповідати на запити про поточний стан і відслідковувати потік угод у організації, що відповідає оперативному управлінню.

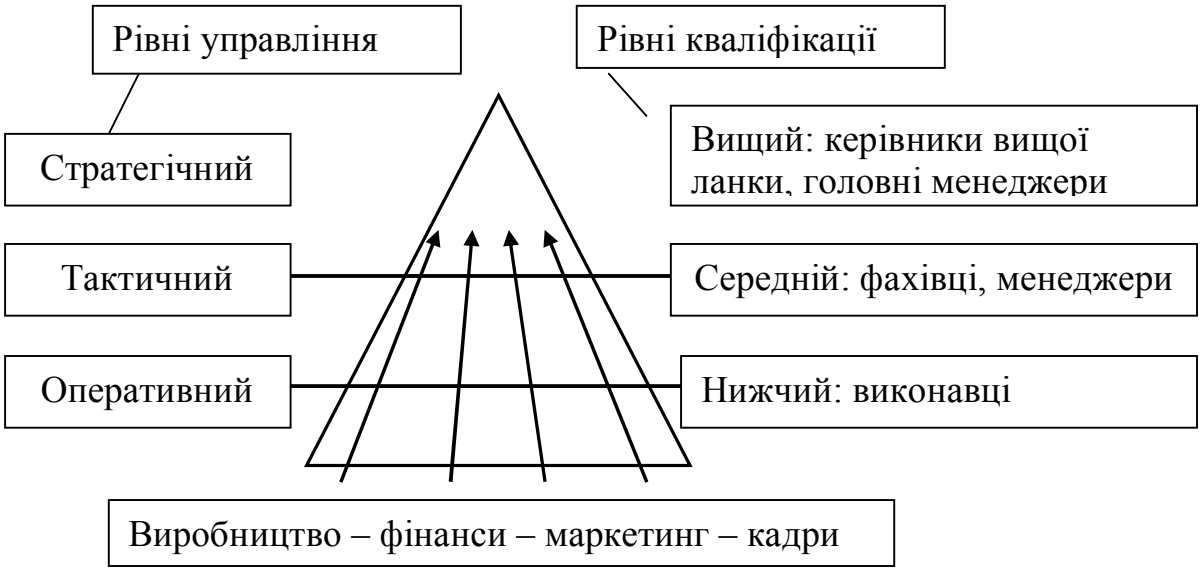


Рисунок 2.7 – Класифікація ІС за функціональною ознакою та рівнем управління

На цьому рівні інформаційна система повинна бути легкодоступною, безперервно діючою і надавати точну інформацію.

Завдання, цілі та джерела інформації на оперативному рівні заздалегідь визначені та структуровані. Рішення приймаються у відповідності із заданим алгоритмом.

Інформаційна система оперативного рівня є сполучною ланкою між організацією та зовнішнім середовищем.

ІС тактичного рівня управління використовуються працівниками (менеджерами) середньої управлінської ланки для моніторингу (постійного

стеження), контролю, прийняття рішень і адміністрування. Основні функції цих інформаційних систем: порівняння поточних показників з минулими, складання періодичних звітів за певний час, а не видача звітів по поточним подіям, як на оперативному рівні, забезпечення доступу до архівної інформації тощо.

Деякі ІС забезпечують прийняття нетривіальних рішень. На цьому рівні можна виділити два типи інформаційних систем: управлінські (для менеджменту) і системи підтримки прийняття рішень.

Управлінські ІС мають обмежені аналітичні можливості. Вони обслуговують управлінців, які потребують щоденної, щотижневої інформації про стан справ. Основне їх призначення полягає у відстеженні щоденних операцій у фірмі і періодичному формуванні строго структурованих зведених типових звітів.

Отже, характеристики управлінських ІС є такими:

– використовуються для підтримки прийняття рішень структурованих і частково структурованих задач на рівні контролю за операціями;

– орієнтовані на контроль, звітність і прийняття рішень по оперативній обстановці;

– спираються на існуючі дані та їх потоки всередині організації;

– мають малі аналітичні можливості та негнучку структуру.

СППР обслуговують частково структуровані задачі, результати яких важко спрогнозувати заздалегідь. Вони мають більш потужний аналітичний апарат з кількома моделями. Інформацію одержують з управлінських і операційних інформаційних систем. Використовують ці системи всі, кому необхідно приймати рішення: менеджери, фахівці, аналітики тощо.

Таким чином, характеристики СППР є такими:

– забезпечують вирішення проблем, розвиток яких важко прогнозувати;

– оснащені складними інструментальними засобами моделювання та аналізу;

– дозволяють легко змінювати постановки задач і вхідні дані;
– відрізняються гнучкістю і легко адаптуються до зміни умов по кілька разів на день;
– мають технологію, максимально орієнтовану на користувача.

У міру того як індустріальне суспільство трансформується в інформаційне, продуктивність економіки все більше буде залежати від рівня розвитку цих систем. Такі системи, особливо у вигляді робочих станцій і офісних систем, найбільш швидко розвиваються сьогодні в бізнесі. У цьому класі інформаційних систем можна виділити дві групи:

– інформаційні системи офісної автоматизації;
– інформаційні системи обробки знань.

Інформаційні системи офісної автоматизації внаслідок своєї простоти і багатопрофільності активно використовуються працівниками будь-якого організаційного рівня. Найбільш часто їх застосовують працівники середньої кваліфікації: бухгалтери, секретарі, клерки. Основна мета – обробка даних, підвищення ефективності їх роботи та спрощення канцелярської праці.

ІС офісної автоматизації пов'язують воєдино працівників інформаційної сфери в різних регіонах і допомагають підтримувати зв'язок з покупцями, замовниками та іншими організаціями. Їх діяльність в основному охоплює управління документацією, комунікації, складання розкладів тощо.

ІС стратегічного рівня управління. Розвиток і успіх будь-якої організації багато в чому визначаються прийнятої в ній стратегії.

Визначення 2.1. Стратегія – це набір методів і засобів вирішення перспективних довгострокових завдань.

В даний час, у зв'язку з переходом до ринкових відносин питанню стратегії розвитку та поведінки організації приділяють велику увагу, що сприяло корінній зміні у поглядах на ІС. Вони стали розцінюватися як стратегічно важливі системи, які впливають на зміну вибору цілей організації, її завдань, методів, продуктів, послуг, дозволяючи випередити

конкурентів, а також налагодити більш тісну взаємодію з споживачами і постачальниками.

Стратегічна інформаційна система – це інформаційна система, що забезпечує підтримку прийняття рішень у реалізації стратегічних перспективних цілей розвитку організації.

Інформаційні системи стратегічного рівня допомагають вищій ланці управлінців вирішувати неструктуровані задачі та здійснювати довгострокове планування. Їх основне завдання – порівняння змін, що відбуваються у зовнішньому середовищі з існуючим потенціалом організації. Вони покликані створити загальне середовище комп'ютерної та телекомунікаційної підтримки рішень в несподівано виникаючих ситуаціях. Використовуючи найдосконаліші програми, ці системи здатні в будь-який момент надати інформацію з багатьох джерел.

У наш час ще не розроблено загальну концепцію побудови стратегічних інформаційних систем унаслідок їх багатопланового використання не тільки за цілями, але й за функціями. Існують дві точки зору: одна базується на концепції, що спочатку необхідно сформулювати свої цілі та стратегії їх досягнення, а тільки потім пристосовувати інформаційну систему до наявної стратегії; друга – на те, що організація використовує стратегічну ІС при формулюванні цілей і стратегічному плануванні. Сучасним, раціональним підходом до розробки стратегічних інформаційних систем буде методологія синтезу цих двох точок зору.

У будь-якій організації бажано мати декілька взаємопов'язаних локальних ІС різного призначення, які взаємодіють між собою. На основі інтеграції ІС різного призначення за допомогою комп'ютерних мереж в організації створюються корпоративні ІС. Подібні ІС надають користувачеві можливість працювати в організації як з загальними так і з локальними базами даних.

3 ТРЕНДИ РОЗРОБКИ ВЕБ-ОРІЄНТОВАНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Основні особливості та проблеми сучасних веб-орієнтованих програмних проектів

Сьогодні існує широкий спектр програмних та технічних засобів для побудови веб-систем.

Сучасна практика господарювання висуває доволі жорсткі вимоги до побудови інформаційних систем. Щоб їх задовольнити, ІС повинна:

– мати розподілену модульну архітектуру;

– забезпечувати централізований доступ до інформації та централізоване управління;

– мінімізувати витрати на своє утримання (обслуговування серверів, налаштування конфігурації робочих станцій, оновлення версій тощо);

– підтримувати інтерфейси взаємодії з іншими системами; володіти мультиплатформеністю;

– мати мінімальні претензії до конфігурації робочих місць користувача; відповідати високим показникам якості, надійності, стабільності, безпеки, швидкодії.

Інтернет впевнено увійшов в наше життя, зокрема суспільне. Проте для кінцевого користувача типовий інформаційний ресурс все ще залишається статичною сторінкою з матеріалом сумнівної якості та походження. Власне саме поняття Веб (дослівно – павутина) не має чіткого технічного визначення, скоріше це концепція, ідея поєднання ІТ для представлення інформації різного типу в зручному для користувача вигляді та з можливістю миттєвого переходу за посиланнями. На момент масового впровадження графічного інтерфейсу користувача на платформі персональних комп’ютерів

ця концепція призвела до вибухоподібного поширення мережі Інтернет у світі.

На сьогодні значного поширення і імплементації набула методика Web 2.0 – методика проектування систем, які шляхом урахування мережних взаємодій стають тим краще, чим більше людей ними користуються. Особливістю Web 2.0. є принцип залучення користувачів до наповнення і багаторазової вивірки контента

Web 2.0 слід сприймати як суму технологій, а не як єдину технологію. При цьому на передній план виходить ідея співпраці, колективної творчості, миттєвого поширення та отримання відгуку від споживачів інформації. Інтернет перестає бути лише інформаційним джерелом, а стає платформою, місцем для спільної праці.

Характерною особливістю веб-орієнтованих інформаційних систем, зокрема систем управління контентом веб-сайтів, є наявність великої кількості документальної інформації, причому самі ці документи часто генеруються динамічно, на основі тих чи інших процедур.

Сьогодні стає все більш зрозуміло, що сучасні ІС, зокрема веб-орієнтовані системи, повинні все більше спиратися на семантику предметної області, на знання про неї.

Основу знань системи становить онтологія, тому система, яка ґрунтується на знаннях, повинна бути онтологічно орієнтованою.

Визначення 3.1 Онтологія – це загальноприйнята і загальнодоступна концептуалізація певної області знань, яка містить базис для моделювання цієї області знань і визначає протоколи для взаємодії між агентами, які використовують знання з цієї області, і, нарешті, включає домовленості про представлення теоретичних основ даної області знань.

Онтологія в інформатиці повинна мати формалізовану мову представлення, яку комп'ютер здатний обробляти без безпосередньої участі людини.

Тому веб-орієнтована інформаційна система, яка ґрунтується на знаннях, має бути орієнтованою на роботу як з онтологіями, так і з документами і множинами документів.

В основу моделі інформаційної бази веб-орієнтованої системи необхідно покласти два вузлові компоненти:

- онтологія предметної області та множина документів, а також
- зв'язки між цими компонентами.

Іншими словами, необхідно побудувати графову модель, яка складатиметься не з логічно розрізнених документів, що характерно для більшості сучасних веб-орієнтованих систем, а з описів реальних класів предметної області, їх екземплярів та зв'язків між ними, а також пов'язаних з ними документів. Тобто, можна говорити про проблему "занурення" множини документів, які можуть бути статичними або генеруватися динамічно, в загальну семантику предметної області.

Слід також зазначити, що базу знань, яку покладено в основу інтелектуальної інформаційної системи, можна розглядати як наповнення власне онтології як концептуальної схеми.

Детальніше формалізувати зв'язки між семантикою предметної області та множиною документів можна за формальною моделлю онтології.

Формально модель інформаційного наповнення веб-орієнтованої системи розглядається як трійка:

$$M = < W^*, D, L >,$$

де M – онтологія предметної області;

W* – розширена онтологія, наповнення онтології W конкретними екземплярами класів (фактично – база знань);

D – множина документів;

L – множина зв'язків між W* та D.

На найзагальнішому рівні, в термінології UML власне онтологія пов’язана з діаграмою класів, а наповнення бази знань – з діаграмою екземплярів.

Але об’єктно-орієнтована реалізація онтологічного підходу, незважаючи на спорідненість цих напрямків, має свою специфіку.

Об’єктно-орієнтоване програмування зосереджується переважно на методах класів: програміст приймає проектні рішення за операторними методами класу, тоді як розробник онтології приймає рішення, ґрунтуючись на структурних властивостях класу.

У результаті структура класу і зв’язки між класами в онтології значно відрізняються від структури подібної предметної області в об’єктно-орієнтованій програмі.

Значеннями функцій інтерпретації онтології можуть бути елементи множини документів D. При цьому мова повинна йти не тільки власне про онтологію, але і про всю базу знань, побудовану на її основі.

Якщо онтологія розглядається як трійка

$$< Q, I, F >,$$

де Q – множина класів, які відповідають поняттям предметної області;

I – множина зв’язків між ними;

F – множина функцій інтерпретації;

то розширена онтологія описується як така трійка:

$$< Q^*, I^*, F^* >,$$

де Q* – множина класів разом із їхніми екземплярами;

I* – множина зв’язків між цими елементами;

F* – множина функцій інтерпретації, визначених у найпростішому випадку на елементах з Q*, I*.

Тоді елементи D можуть бути значеннями функцій з F*. Іншими словами, будемо вважати документ d релевантним відносно W*, якщо існують хоча б один вузол i/та функція інтерпретації f такі, що d = f(w).

Важливими елементами веб-орієнтованої інформаційної системи мають стати класи, які реалізують функції інтерпретації та можуть використовуватися для динамічного формування документів, а також класи, які забезпечують відображення отриманих документів.

Наведене співвідношення може стати основою для динамічного формування переліку споріднених документів. Для документа d = f(w) спорідненими документами можуть вважатися зокрема такі:

– всі документи s такі, що s = h(w), де h пов’язане з F* (тобто документи, пов’язані з тим самим вузлом іншими функціями інтерпретації);

– всі документи s такі, що s = g(u), де функції інтерпретації g пов’язані з f, вузли и пов’язані з w.

Принципово важливим є те, що зв’язки і можливі переходи між документами повинні насамперед визначатися зв’язками між класами онтології та їх екземплярами.

Тому ключовим компонентом стає система навігації, яка на основі онтології забезпечує динамічне формування навігаційних графів, які визначають можливі переходи веб-сайтом. Якщо ввести міри, які характеризують семантичну близькість понять онтології, а також ступені важливості функцій інтерпретації, це дасть змогу проранжувати документи за мірою їх спорідненості до даного. Цікаво також дослідити, як введені так міри спорідненості співвідносяться з традиційними підходами до визначення мір схожості документів на основі аналізу власне текстів.

У цьому контексті стає очевидним, що проблема динамічного формування документів, споріднених до даного, якнайтісніше пов’язана з проблемою підвищення повноти та релевантності інформаційного пошуку. Дійсно, незалежно від того, чи користувач вийшов на певний вузол онтології в процесі навігації по сайту, чи він ввів відповідне ключове слово в полі

введення – йдеться про формування переліку документів, пов’язаних з цим вузлом, і ранжування цих документів за мірою релевантності.

Здійснівши формальний опис ІС, визначимо дві основні групи типових операцій над нею.

1. До першої групи належать операції, що модифікують наявну систему шляхом зміни її внутрішньої структури – у таких операціях бере участь одна інформаційна система.

2. Друга група операцій формується на основі операцій, традиційних для теорій формальних систем та абстрактних автоматів. У таких операціях беруть участь дві або більше інформаційні системи; результатом операції є нова система, що їх об’єднує.

Розглядаючи операції 2-ї групи, вважатимемо, що вихідний потік може розщеплюватися або дублюватися та одна з його копій – передаватися в іншу систему. У результаті застосування таких операцій до двох або більше інформаційних систем утворюється нова інформаційна система, що містить як компоненти вихідні системи. Запити до цих систем уже надходитимуть від вищого сервісу, а не від користувачів системи. Відповіді компонент спрямовуються як вхідний потік у вищу систему.

Одним з наслідків побудови моделі веб-системи є можливість формального опису її поведінки. Такий опис має імовірнісний характер та відображає асимптотичну поведінку системи.

Досліджуючи поведінку системи, розглядають абстрактний часовий параметр. Розглянемо проблему визначення часового параметра. Формальний підхід ґрунтується на виділенні певної впорядкованої множини, що розглядається як множина моментів часу. Елементи множин вхідних сигналів та реакцій системи є результатами застосування певних функцій до моментів часу.

При моделюванні систем архітектури «клієнт–сервер», враховуючи те, що будь-яка дія системи визначається запитом клієнта, за формальний параметр часу доцільно приймати номер за порядком запиту клієнта.

Множина номерів запитів – множина натуральних чисел. Існує істотна відмінність між реальним фізичним часом та формальним часом інформаційної системи: так система, що функціонує коротший час, може бути формально старішою – опрацювати більшу кількість запитів.

У межах аналізу поведінки веб-систем (зокрема конкретних апаратних та мережевих рішень) доцільно проводити зіставлення реального та формального часу. Основний акцент під час моделювання перспективної поведінки системи роблять на терміни формального часу.

Одним з головних напрямків аналізу поведінки системи є визначення її поведінки в перспективі – тобто при зростаючому до безмежності числі запитів системи.

Інформаційна система є динамічною системою в розумінні теорії формальних систем. Для визначення реакції системи на отриманий запит достатньо лише інформації про стан системи (вмісту її бази даних).

Іншою важливою вимогою до веб-орієнтованих інформаційних систем є необхідність використання сучасних стандартів і технологій з огляду на логіку та перспективи їх розвитку. Ідеться насамперед про проект Semantic Web, орієнтований на обмін даними в середовищі WWW та на їх повторне використання і особливо – на опис семантики, змісту даних, які зберігаються на веб-ресурсах. Semantic Web ґрунтується передусім на таких рекомендаціях, затверджених консорціумом W3C:

– розширена мова розмітки XML – eXtensible Markup Language;

– структура опису ресурсу RDF – (Resource Description Framework). Ця структура забезпечує можливість для взаємодії застосувань на основі аналізу описів веб-документів. RDF Schema, своєю чергою, допомагає поєднувати ці описи в єдиний словник. Специфікація RDF надає потужну інфраструктуру для підтримки обміну знаннями в Інтернеті;

– мова веб-онтологій OWL (Ontology Web Language), що надає інструментарій для побудови структурованих веб-орієнтованих онтологій,

які легко інтегруються та забезпечують можливість взаємодії даних між різними групами.

З огляду на описані вище принципи проектування веб-орієнтованих інформаційних систем, які повинні брати до уваги зв'язки між онтологією предметної області та множиною документів, а також з використанням наведених вище рекомендацій, розроблено прототип системи управління вмістом веб-ресурсів, який забезпечує такі можливості:

1. Система генерує навігаційні графи на основі онтологічного опису предметної області, може переходити від однієї навігаційної структури до іншої та від одного опису відповідного поняття предметної області до іншого залежно від завдання потрібної системи відношень та функцій інтерпретації. При цьому залишається можливість жорсткого завдання системи навігаційних посилань на основі відношення "рубрика/підрубрика".

2. Система працює з онтологією, яка містить три логічно виокремлені рівні:

– рівень даних, або онтологія предметної області: описує базові поняття, класи та зв'язки між ними, а також конкретні сутності предметної області;

– рівень структури, що забезпечує навігацію користувача по сайту;

– рівень представлення даних, або онтологія структурних шаблонів, що керує відображенням інформації у вигляді, зручному для користувача.

3. Система спирається на XML, RDF та OWL. Для того, щоб описати модель системи управління веб-сайтом, що ґрунтується на базі знань, необхідно спочатку описати вимоги, що накладаються на онтологію.

Онтологічна база знань разом з редактором онтологій має відповідати такому переліку вимог:

1. Структури: онтологія має відображати всю структуру інформаційного ресурсу.

2. Вмісту: онтологія має надавати можливість акумулювати різні типи інформації, а також дозволяти посилання на інші онтології.

3. Імпорту та експорту: вміст онтології має легко імпортуватись або експортуватись в систему і з системи відповідно.

4. Життєвого циклу інформації: онтологія має надавати можливості для точного представлення та ілюстрації всіх етапів життєвого циклу.

5. Відображення: онтологія має забезпечувати легкість внесення змін до представлення інформації на сайті.

Структура сайту визначається екземплярами одного класу – класу сторінок (Pages). Отже, до неї можна звернутись як до дерева, кореневому елементу якого відповідає початкова сторінка.

Завданням онтології є змоделювати ієрархічну деревоподібну структуру і вказати для кожного елемента його рівень та позицію.

Для цього в класі сторінок вводяться додаткові параметри (властивості), а саме:

– батьківський елемент: тобто кожен елемент структури, крім кореневого, має елемент, який визначає рівень вкладеності його у навігаційній структурі;

– позиція: властивість, що відповідає за порядок відображення певного елемента на визначеному рівні;

– ресурс: URL ресурсу (класу або індивіду онтології, що відображається в цьому розділі структури).

Система з веб-інтерфейсом, незалежно від архітектури, по суті є розподіленою і надає можливість спільної паралельної роботи з інформацією. Вся робота з системою відбувається через веб-браузер, який входить в набір стандартних програм будь-якої операційної системи. Ні до програмної, ні до апаратної частини робочого місця користувача не висувають жодних вимог, окрім як організувати мережне з’єднання з сервером і забезпечити роботу веб-браузера (технологія тонкого клієнта). Сучасний веб-інтерфейс забезпечує прийнятний для ІС рівень швидкодії та зручності використання.

Інформаційні веб-системи отримують особливу перевагу там, де є розподіл мережі користувачів системи. Це може бути філіальна структура організації, наявність віддалених співробітників або партнерів.

Так само веб-система виявляється вигіднішою в разі необхідності мобільного доступу, наприклад, коли користувач працює з різних ПК чи йому необхідно мати доступ до системи зі свого портативного комп'ютера, але не бути територіально прив'язаним. У цьому випадку перевага полягає у відсутності необхідності установки клієнтського програмного забезпечення та організації доступу до системи (за винятком доступу до Інтернету).

Обрання систем, орієнтованих на веб-доступ, пояснюється доволі просто – кросплатформеність, спрощена система використання «Logon and Go» – потрібен лише браузер, логін та пароль, доступ з корпоративної локальної мережі (Intranet) – будь-яке під'єднання; якщо потрібен доступ з мережі Internet – доступ до Internet, і системою вже можна користуватись. Тому завдяки описаним вище властивостям структура веб-сайту стає повністю керованою і підпорядкованою адміністратору. Більше того, додавання нових елементів до структури полягає лише у введенні нового елемента та визначенні для нього вищезгаданих параметрів.

3.2 Огляд та класифікація мов веб-програмування. Основні засоби веб-технологій. HTML – мова розмітки тексту

Всесвітня павутина складається з веб-сторінок, які створено у форматі HTML (HyperText Markup Language, «мова розмітки гіпертексту»). HTML – це фундаментальна, базова технологія Інтернету.

HTML не є мовою програмування, це мова розмітки тексту, що використовує спеціальні оператори – теги (tag) чи інша назва – дескриптори (descriptor) для розмітки текстового документа. Ці позначки вказують, в якому вигляді буде виведено текстовий чи інший елемент у вікні браузера.

HTML дозволяє формувати на сторінці сайту текстові блоки, додавати до них зображення, організовувати таблиці, керувати відтворенням кольору, додавати до дизайну сайту звуковий супровід, організовувати гіперпосилання з переходом до інших розділів сервера або звертатися до інших ресурсів Інтернету і компонувати всі ці елементи між собою. Документи, що створено лише засобами HTML, мають розширення .htm або .html.

Однією з основних функціональних особливостей мови HTML, завдяки якій вона і отримала свою назву, є гіперпосилання.

Визначення 3.2 Гіперпосилання (Hyperlink) – це базовий функціональний елемент HTML-документу, який реалізує зв'язок певного об’єкту веб-сторінки з іншим об’єктом. Для гіперпосилання може використовуватися як фрагмент тексту, так і графічний об'єкт, а сам гіперзв'язок можна встановлювати як між об’єктами одного сайту, так і між об'єктами, що розміщені на різних сайтах Інтернету.

HTML є мовою, що лише інтерпретується, тому для виконання коду його не потрібно компілювати. Інтерпретатор мови втілено в браузер, і він інтерпретує код безпосередньо під час відкривання документа. Якщо в коді сторінки виявлено помилку, інтерпретатор, зазвичай, не видає відповідного попередження, а просто ігнорує весь «помилковий» рядок, що може зіпсувати зовнішній вигляд завантаженої сторінки. Це є важливим для розробників, тому слід ретельно тестувати результати своєї роботи.

Визначення 3.3 Каскадна таблиця стилів (CSS – Cascading Style Sheets) – це технологія опису зовнішнього вигляду документа, що створено засобами HTML, XML і XHTML.

CSS використовується для завдання кольорів, шрифтів, розташування елементів сторінки тощо. До появи CSS оформлення веб-сторінок вказувалося безпосередньо в HTML-коді сторінки. Проте з появою CSS стало можливим принципове розділення змісту і представлення документа. За рахунок такого нововведення стало можливим легке застосування єдиного

стилю оформлення для кількох сторінок сайту, а також швидка зміна цього оформлення.

Переваги CSS:

- застосування кількох варіантів дизайну сторінки для різних пристроїв перегляду. Наприклад, для відображення на екрані монітора - дизайн буде розраховано на велику ширину. У разі друкування документу не буде роздруковане меню сайту, а у разі перегляду у мобільному комп'ютері чи телефоні, меню буде виведено після вмісту сторінки;
- зменшення часу завантаження сторінок сайту за рахунок перенесення правил представлення даних до окремого CSS-файлу. В цьому випадку браузер завантажує лише структуру документа і дані, що містяться на сторінці. CSS-файл з правилами представлення цих даних завантажується браузером лише один раз і зберігається в кеші браузера;
- простота подальшої зміни дизайну. Не потрібно виправляти кожну сторінку, достатньо лише змінити кілька правил у CSS-файлі;
- додаткові можливості оформлення. Наприклад, за допомогою CSS-правил можна застосувати обтікання певного блоку текстом або зробити так, щоб меню фіксовано знаходилося в певному місці при перегортанні сторінки.

Недоліки CSS: різні браузери можуть в різний спосіб інтерпретувати правила CSS, і відповідно, по різному відображати одні і ті ж фрагменти сторінки.

DHTML – динамічна мова розмітки тексту. DHTML (Dynamic HTML) – це набір засобів, які реалізовують інтерактивність веб-сторінки без звертання до серверу. Тобто певні дії відвідувача можуть спричинити зміну зовнішнього вигляду і вмісту сторінки.

DHTML побудовано на об'єктній моделі документа DOM (*Document Object Model*), яка розширює традиційний статичний HTML-документ. DOM забезпечує динамічний доступ до вмісту документа, його структури і стилів.

В DOM кожен елемент веб-сторінки є об'єктом, який надається до змін. DOM не визначає нових тегів чи атрибутів, а лише забезпечує можливість

програмного управління всіма тегами, атрибутами і каскадними аркушами стилів CSS.

JavaScript – мова сценаріїв. JavaScript – це мова, призначена для написання сценаріїв для інтерактивних HTML-сторінок. Мова JavaScript не має жодного відношення до мови Java. Java розроблено фірмою SUN, а JavaScript – фірмою Netscape Communication Corporation. Первинною назвою була LiveScript, але з огляду на велику популярність мови Java, назву LiveScript з комерційних міркувань було змінено на JavaScript.

JavaScript не призначено для створення автономних застосувань. Програмний код на JavaScript вписується безпосередньо в текст HTML-документа і інтерпретується браузером в міру завантаження цього документа. За допомогою JavaScript можна динамічно змінювати текст завантаженого HTML-документу і реагувати на події, які пов’язані з діями відвідувача або змінами стану документа чи вікна.

Важливою особливістю JavaScript є об'єктна орієнтованість. Програмісту є доступними численні об'єкти, такі, як документи, гіперпосилання, форми, фрейми тощо. Об’єкти характеризуються описовою інформацією (властивостями) і можливими діями (методами).

PHP – мова створення сценаріїв. PHP (*Personal Home Page*) – мова створення сценаріїв, яка давно переросла свою назву. Перша версія PHP була створена Расмусом Лердорфом в 1994 р. і була набором інструментів для відстеження поведінки відвідувачів сайту. З часом PHP з набору інструментів перетворилася на повноцінну мову програмування, а її назву було змінено як PHP HyperText Preprocessor (препроцесор гіпертексту PHP).

PHP – це серверна мова. Конструкції PHP, що вставлено в HTML-текст, виконуються сервером при кожному відвідуванні сторінки. Результат обробки конструкцій разом із звичайним HTML-текстом передається браузеру.

Основними конкурентами PHP є ASP (Active Server Pages) від компанії Microsoft і ColdFusion від компанії Allaire.

Порівняно з ними PHP має ряд переваг, зокрема:

– висока продуктивність. PHP-програми працюють швидше за ASP;

– функціональність. Розробку PHP-програми можна відокремити від власне розробки веб-сторінки, що спрощує працю і для програміста і для дизайнера;

– ціна. Мова PHP є абсолютно безкоштовною;

– простота у використанні. Програмісти, що мають досвід програмування на поширених мовах швидко зрозуміють синтаксис PHP;

– переносимість. Один і той же PHP-код можна використовувати як в середовищі Windows, так і на платформах UNIX.

ASP – активні сторінки сервера. ASP (Active Server Pages) — технологія, що є аналогічною до JavaScript і PHP. Для створення інтерактивної веб-сторінки із застосуванням макромови ASP необхідно вбудувати в її код відповідний скрипт, що віддалено нагадує Java і C. Скрипт інтерпретується і виконується безпосередньо на сервері, після чого до браузера відправляється вже готовий HTML-документ з результатами роботи сценарію ASP. Для сторінок, що містять конструкції ASP, не має значення, яке програмне забезпечення встановлено на комп'ютері користувача, але принципове значення має тип сервера, який має підтримувати дану технологію.

XML – розширена мова розмітки. Основну увагу в мові XML зосереджено на даних. При цьому структурна розмітка даних і представлення даних є строго розділеними.

Основні причини створення XML:

– спроба надати могутні засоби форматування і структуризації даних для широкого кола розробників;

– необхідність в стабільній реалізації мови структуризації документів, де можна легко створювати допоміжні інструменти, що є доступними для звичайних користувачів;

XML є метамовою, тобто спеціальною мовою, якою можна скласти повний опис класу інших мов, якими створено документи. XML містить набір правил, що дозволяють створювати унікальні застосування і підмножини даних.

У багатьох розробників виникають певні труднощі у зв'язку з абстрактністю XML і довільним використанням його методів. Насправді, XML є досить логічною і добре організованою структурою, вона має чіткий синтаксис, що змушує строго дотримуватися певних правил. Починати вивчення XML слід із застосування вже отриманих знань про HTML. XML, як і HTML, використовує теги та атрибути.

Як мова розмітки веб-документів XML має свої особливості:

– гнучкість. XML дозволяє обробляти унікальні дані, і незалежно від їх характеру надає адекватні методи для їх зберігання і обробки;

– можливість налаштування. Гнучкість XML безпосередньо пов'язана з можливістю визначати власні дескриптори, необхідність в яких виникає в процесі рішення задачі;

– узгодженість. XML відрізняється синтаксичною цілісністю і строгою структурою.

Практично всі сучасні браузери підтримують XML. Вона здатна цілком замінити HTML, як засіб розмітки веб-сторінок.

XSLT – розширена мова перетворення листів стилів. Мова XSLT (eXtensible Stylesheet Language Transformations) є транслятором, за допомогою якого можна вільно модифікувати початковий текст. XLST грає вирішальну роль в затвердженні XML як універсальної мови збереження і передачі даних. Область застосування XSLT є широкою - від електронної комерції до безпровідного Веб.

Фактична збірка результуючого документа відбувається, коли початковий документ і аркуш стилів XSLT передаються до синтаксичного аналізатора XSLT (XSLT-процесора).

Перетворення XSLT засновано на шаблонах. XSLT-процесор аналізує початковий документ і намагається знайти відповідний XSL-шаблон.

Після чого виконуються інструкції, що містяться всередині нього.

Ajax – технологія для взаємодії з сервером без перевантаження сторінок. Ajax (Asynchronous Javascript And XML) – це підхід до створення веб-застосувань за допомогою наступних технологій:

– стандартизоване представлення засобами XHTML і CSS;

– динамічне відображення і взаємодія з користувачем за допомогою DOM;

– обмін і обробка даних у вигляді XML и XSLT;

– широке застосування мови сценаріїв JavaScript;

– асинхронні запити за допомогою об'єкту XMLHttpRequest.

У стандартному веб-застосуванні обробкою всієї інформації займається сервер, браузер відповідає лише за взаємодію з користувачем, передачу запитів і виведення отриманих даних у форматі HTML.

В Ajax-застосуванні між користувачем і сервером з'являється ще один посередник – програмний механізм (рушій) Ajax. Він визначає, які запити можна обробити з боку клієнта, а які необхідно виконувати на сервері.

Поведінка сервера теж змінилася. Якщо раніше на кожен запит сервер видавав нову сторінку, то тепер він надсилає лише ті дані, які потрібні клієнту, а рушій Ajax в браузері формує з них HTML-код.

Асинхронність виявляється в тому, що далеко не кожен запит користувача скеровується до сервера, причому зворотне теж справедливо - далеко не кожна реакція сервера обумовлена запитом користувача. Велику частину запитів формує рушій Ajax, причому є можливим, щоб він передбачав запити користувача.

Зрозуміло, що за такою схемою роботи міняється якісне навантаження на сервер – якщо раніше запитів було мало, але кожен з них вимагав значних ресурсів (серверу потрібно витягнути інформацію з бази даних, сформувати з

неї веб-сторінку і відправити до браузера), то тепер завдання сервера спрощується (формувати веб-сторінки не потрібно, та й об’єм даних, що передаються є значно меншим), але доводиться обробляти більше запитів.

Створювати застосування на Ajax є трудомістким і складним завданням. Потрібно написати і відлагодити на JavaScript рушій з десятьох чи двадцятьох тисяч рядків коду плюс реалізувати серверну частину.

На сьогодні спостерігається тенденція на стрімке поширення Ajax-застосувань. Масштабні Ajax-проекти представлено у Google – Google Suggest (сервіс, що підказує найбільш популярні запити), Gmail і Google Maps. Найбільш ґрунтовну переробку програмісти Google зробили з поштовим інтерфейсом, тоді як Google Suggest і Google Maps дивують не стільки новизною підходу, скільки якістю реалізації.

Adobe Flash. Adobe Flash (раніше відома як Macromedia Flash), або просто Flash — мультимедійна платформа, що призначена для створення векторної анімації і інтерактивних застосувань (зокрема, ігор), а також для інтеграції відеороликів у веб-сторінки.

Основним призначенням даної технології є створення високоякісної інтерактивної анімації, яка має відносно невеликий розмір вихідного файлу. За допомогою Flash розробник має можливість виготовляти барвисті анімаційні заставки, певні елементи яких можуть «реагувати» на рухи миші, міні-ігри, озвучені мультиплікаційні кліпи і багато іншого.

Adobe Flash має інструменти для роботи з векторною, растровою і частково з тривимірною графікою, а також підтримує потокову трансляцію аудіо і відео.

Основними засобами розробки є пакети Adobe Flash Professional і Adobe Flash Builder, що дозволяють створювати інтерактивні застосування, зокрема, презентації, інтро-заставки, ігри і мультфільми).

Flash використовує вбудовану мову програмування ActionScript, що базується на ECMAScript. Програмні модулі імпортуються в документ як аплети і вставляються в потрібний кадр анімації, де повинна відбутися

динамічна зміна зображення. За допомогою спеціального редактора можна написати невелику програмку, що керує програванням кліпу, створити елементи, що надаються до індивідуальних налаштувань відвідувачами сайту, генерувати заставку з кількома варіантами продовження. Способів реалізації можливостей ActionScript є багато, але для використання всієї її потужності, необхідно мати певний досвід в програмуванні.

Стандартним розширенням для скомпільованих Flash-файлів (анімації, ігор і інтерактивних застосувань) є .SWF (Shockwave Flash або Small Web Format). Відеоролики у форматі Flash є файлами з розширенням FLV (Flash Video), а Flash в даному випадку використовується лише як контейнер для відеозапису. Розширення FLA відповідає формату робочих файлів в середовищі розробки.

Flash-вміст відтворюється за допомогою цілого ряду програмних засобів, але домінуюче місце на ринку займає офіційний програвач Adobe Flash Player, який поширюється як безкоштовний плагін для більшості сучасних браузерів.

4 ВЕБ-ОРІЄНТОВАНІ СИСТЕМИ

4.1 Завдання Веб-орієнтованих систем

Веб-програмування охоплює широке коло професійних задач, пов'язаних з розробкою програмних систем, що функціонують у межах інфраструктури всесвітньої мережі WWW та її сервісів. Особливої уваги заслуговує задача проектування та побудови ефективних систем керування веб-контентом (CMS), які є фундаментом будь-яких веб-проектів, що в тій чи іншій мірі застосовують автоматизоване редагування та публікацію інформації для відвідувачів веб-ресурсів.

Одним з напрямків розвитку Всесвітньої мережі WWW є керування даними та контентом. Ця галузь містить такі напрямки як керування великими об'ємами даних, керування даними на основі хмарних обчислень, керування мультимедійними даними, а також є дотичною до Веб-mining, задач кластеризації, класифікації та аналізу даних в Веб, моделювання Веб-контенту, Semantic Web тощо. Задача ефективного керування Веб-контентом набула великої значущості для багатьох галузей, що виникли на базі інфраструктури WWW, серед яких інтернет-комерція, дистанційне навчання та освітні веб-ресурси, керування великими інформаційними порталами, розробка та підтримка корпоративних порталів, створення Веб-ресурсів електронного урядування, підтримка персональних сайтів та блоґів тощо.

Серед популярних програмних рішень, що використовуються для керування контентом слід зазначити такі CMS-системи: Drupal, Joomla, Wordpress, Plone та ін. Для дистанційного навчання часто використовуються системи Moodle та aTutor, а також MediaWiki як засіб створення освітніх вікі-проектів. Перевагою цих систем, що зумовило їх популярність, є відкритий програмний код, наявність великої кількості додактів та безкоштовна ліцензія на використання. Серед пропрієтарних систем слід зазначити такі:

Magento, 1-С Бітрікс (електронна комерція); Microsoft SharePoint, Adobe Business Catalyst (комплексні SaaS-системи); Blackboard (система дистанційного навчання).

4.2 Статичні і динамічні Веб-сайти. Особливості впровадження Веб-орієнтованих систем

Величезна кількість наявних сайтів може бути розбита на 2 основні групи: статичні сайти й динамічні сайти. Розглянемо переваги і недоліки кожної групи.

Визначення 4.1. Статичним прийнято називати сайт, що складається з незмінних, тобто статичних, HTML-сторінок.

HTML-сторінка є сукупністю тексту, графічних зображень і власне мови гіпертекстової розмітки HTML, відповідальної за представлення сторінки в браузері.

Статичні HTML-сторінки створюються вручну, після чого при кожному звертанні до сайту представляються користувачу в незмінному вигляді. Щоб оновити інформацію на подібних сторінках, необхідно вручну внести зміни безпосередньо в програмний код сторінки.

Перевагами статичних сайтів є такі:

– статичні сайти створюють мінімальне навантаження на сервер, а тому невимогливі до ресурсів хостинга;

– статичні сайти завантажуються швидко;

– розробка статичних сайтів обходиться дешевше;

– перенести статичні сайти на новий хостинг дуже просто.

Серед недоліків статичних сайтів необхідно виділити складність оновлення сайту, внесення яких-небудь змін. Керування сайтом неможливе без знань і вмінь в області веб-програмування – це може викликати додаткові витрати при необхідності додавання нових матеріалів на сайт, нових розділів або категорій. А при розвитку сайту й збільшенні кількості сторінок взагалі

стає важко підтримувати цілісність проекту, стежити за правильністю програмних кодів і т.д.

На відміну від статичних, динамічні сайти набагато більш гнучкі в керуванні**.**

Визначення 4.2. Динамічні сайти являють собою сукупність тексту і графіки, мови розмітки,а також використовують також різні технології, що дозволяють конструювати веб-сторінки в динаміці.

Динамічні сайти можна розробляти «з нуля», вручну створюючи всі необхідні програмні коди, скрипти тощо. Однак набагато частіше для створення динамічних сайтів використовуються спеціальні системи керування контентом – CMS. CMS дозволяють використовувати вже готові програмні модулі та компоненти, без необхідності щоразу створювати їх «з нуля». На основі однієї CMS можна створити будь-яку кількість динамічних сайтів.

Динамічні сайти в браузері формуються з декількох частин або ж браузер заповнює інформацією вже готові шаблони сторінок.

У динамічних сайтах реалізований поділ змісту й оформлення веб-сторінок, що дозволяє оперативно змінювати інформацію на сайтах без необхідності змінювати програмні коди сторінок будь-якому користувачеві, навіть без знання веб-програмування.

В CMS для додавання і редагування матеріалів використовуються візуальні WYSIWYG-редактори (принцип «що бачу - те й одержую»).

Динамічні сайти можуть «підлаштовуватися» під своїх відвідувачів, реагуючи на їхні дії. Для цього використовуються технології серверних, клієнтських скриптів, за допомогою яких і створюються сценарії поведінки сайту при певних діях користувачів.

Крім перерахованих переваг, динамічні сайти мають і ряд недоліків**.** У порівнянні зі статичними сайтами динамічні дають більше навантаження на сервер. Таким чином, динамічні сайти вони більше вимогливі до хостингу, ресурсів сервера.

Для функціонування динамічного сайту необхідно додаткове програмне забезпечення, тоді як для відображення статичних сайтів досить тільки браузера.

При виборі типу сайту необхідно зважити на складність завдання. Нераціонально створювати складні динамічні сайти для розв'язання простих завдань**,** наприклад для реалізації сайтів-визиток з 3-5 сторінок. У цьому випадку на сайті практично не потрібне оновлення контенту, не потрібна наявність інтерактивних функцій – сайт може бути статичним.

У свою чергу, функціоналу статичного сайту недостатньо для складних інтерактивних задач. Наприклад, неможливо створити інтернет-магазин, використовуючи лише статичні HTML-сторінки.

4.3 Системи керування контентом

Визначення 4.3. Під контентом веб-сайту розуміють інформаційний вміст, що надається відвідувачам.

Контент – термін, що стійко закріпився в професійній термінології, пов'язаній з розробкою інформаційних веб-систем. Контент являє собою текст, гіпертекст, зображення, відео, аудіо, анімацію тощо.

Визначення 4.3. Система керування контентом (Content Management System, CMS) – програмне забезпечення, що дозволяє редагувати контент інформаційної системи (веб-сайту) за допомогою зручного інтерфейсу користувача та забезпечує публікацію контенту для відвідувачів разом із засобами навігації.

CMS-система забезпечує дві ключові функції роботи з контентом:

1) з точки зору відвідувача веб-сайту CMS забезпечує навігацію та подання контенту сайту шляхом організації меню та інших навігаційних елементів, а також обробки запитів на отримання веб-сторінок сайту;

2) з точки зору редактора сайту CMS забезпечує зручні інтерфейси та механізми для додавання, збереження та редагування контенту сайту,

звільняючи редактора від необхідності досконало володіти засобами розмітки веб-документів (HTML) та керування серверною структурою файлів та бази даних.

Серед інших функцій CMS – забезпечення багатокористувацького доступу до редагування контенту, підтримка реєстрації відвідувачів сайту, засоби коментування контенту тощо.

Ключовим завданням CMS очевидно є керування контентом. Натомість, розробники нових CMS часто залишають ключове завдання системи без достатньої уваги та реалізують його виключно на механічному рівні, не створюючи достатні засоби автоматизації цього процесу. У загальному сенсі керування контентом – це сукупність процесів та технологій для збору, організації та публікації інформації у різних формах та на різних носіях. Керування веб-контентом покликано забезпечити життєвий цикл інформації, яка подається відвідувачам сайту.

Керування контентом веб-сайтів охоплює не тільки фізичні процеси розміщення веб-сторінок в інтернеті. Сучасне керування контентом вимагає розглядати контент як об’єкт моделювання, структурування і формалізації для організації його ефективного зберігання, публікації та подання кінцевим споживачам у зручному вигляді із зручними інтелектуалізованими засобами навігації. У зв’язку із значущою роллю керування контентом, першочергова увага розробників CMS сконцентрована на розробці оптимальних моделей структурування та формалізації веб-контенту, виявленні та використанні закономірностей при переміщенні по контенту сайту, формалізації зв’язків між ділянками контенту та побудові зручних схем навігації.

Подібні задачі вимагають більш глибокого погляду на контент веб-сайтів, ніж організація розміщення фізичного html-тексту із зображеннями на сервері, та прокладають шлях від керування контентом до керування знаннями в веб-середовищі.

Структура CMS визначається завданнями, які вирішує система для керування контентом (рис. 4.1).

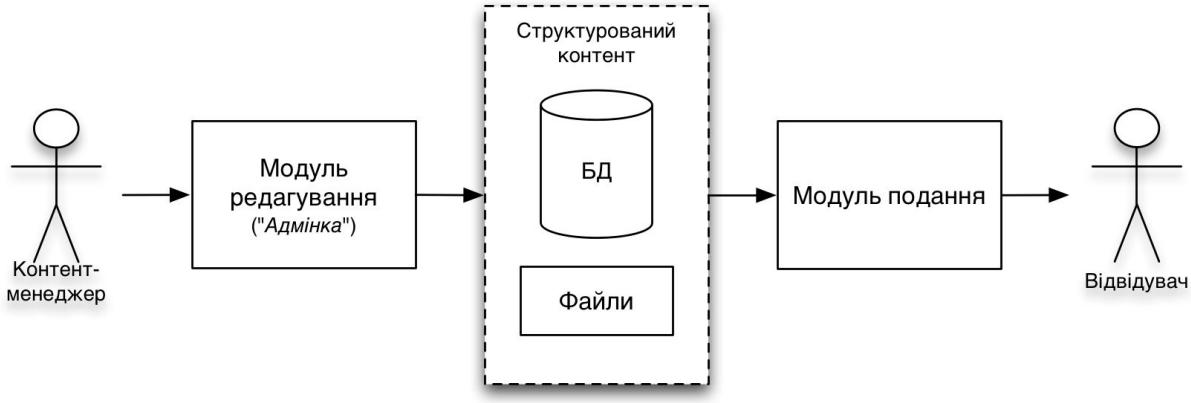


Рисунок 4.1 – Структура CMS

CMS складається з таких компонентів (рис. 4.1):

– модуль редагування (адміністративна частина), що забезпечує виконання адміністративних функцій по редагуванню контенту сайту;

– контент у структурованому вигляді, що, як правило, включає базу даних для зберігання HTML-тексту веб-сторінок та сукупність упорядкованих медіа-файлів;

– модуль подання, ядро системи, що керує логікою подання контенту відвідувачам та забезпечує навігацію по контенту.

Для реалізації CMS створюються два базові класи – клас системи (ядро) та клас сторінки. Уся робота із сторінками сайту відбувається за допомогою класу сторінки. Клас ядра відповідає за загальну логіку роботи системи та обробку ключових запитів.

Процес публікації сторінки містить наступні етапи:

– користувач ініціалізує запит до веб-сайту;

– ядро системи обробляє запит та видобуває код запитуваної сторінки;

– по коду сторінки створюється об’єкт поточної сторінки (ОПС);

– під час ініціалізації ОПС звертається до бази даних для отримання відповідного запису із таблиці сторінок;

– ядро системи розпочинає процес публікації сторінки;

– відповідно до шаблону готується заголовок документа та навігаційні елементи;

– ядро системи звертається до метода публікації в ОПС;

– система завершує формування документа та надсилає користувачеві згенеровану веб-сторінку.

Особливості безпосередньої публікації документа повинні спиратися на роботу спеціальної підсистеми шаблонізації, що дозволяє розмежувати бізнес-логіку серверного коду від подання контенту засобами HTML.

5 ПРИНЦИПИ ОРГАНІЗАЦІЇ І СХЕМА ПРОЕКТУВАННЯ БАЗ ДАНИХ

5.1 Архітектура системи баз даних

Система керування базами даних – СКБД (database management system – DBMS) – програмне забезпечення, що забезпечує інтерфейс між власне фізичною базою даних (тобто даними, що справді зберігаються) і користувачами системи.

Головна функція СКБД – дати можливість користувачеві бази даних працювати з даними, не заглиблюючись у деталі апаратного забезпечення**.**

Архітектура ANSI/SPARC* має три рівні: внутрішній, концептуальний і зовнішній (рис. 5.1).

Внутрішній рівень – це рівень, найближчий до фізичного збереження, тобто пов'язаний зі способами збереження інформації на фізичних пристроях збереження.

Зовнішній рівень найближчий до користувачів, тобто він пов'язаний із засобами подання даних для окремих користувачів.

Концептуальний рівень – це «проміжний» рівень між двома першими.

Якщо зовнішній рівень пов’язаний з індивідуальними поданнями користувачів, то концептуальний рівень пов’язаний з узагальненим поданням користувачів. Інакше кажучи, може бути кілька зовнішніх представлень, кожне з яких складається з більш-менш абстрактного подання визначеної частини бази даних, і може бути тільки одне концептуальне подання, що складається з абстрактного подання бази даних загалом. Також є єдине внутрішнє подання, яке відбиває всю базу даних як фізично збережену.

---

* ANSI/SPARC – American National Standart Institute / Study Group on Data Management Systems

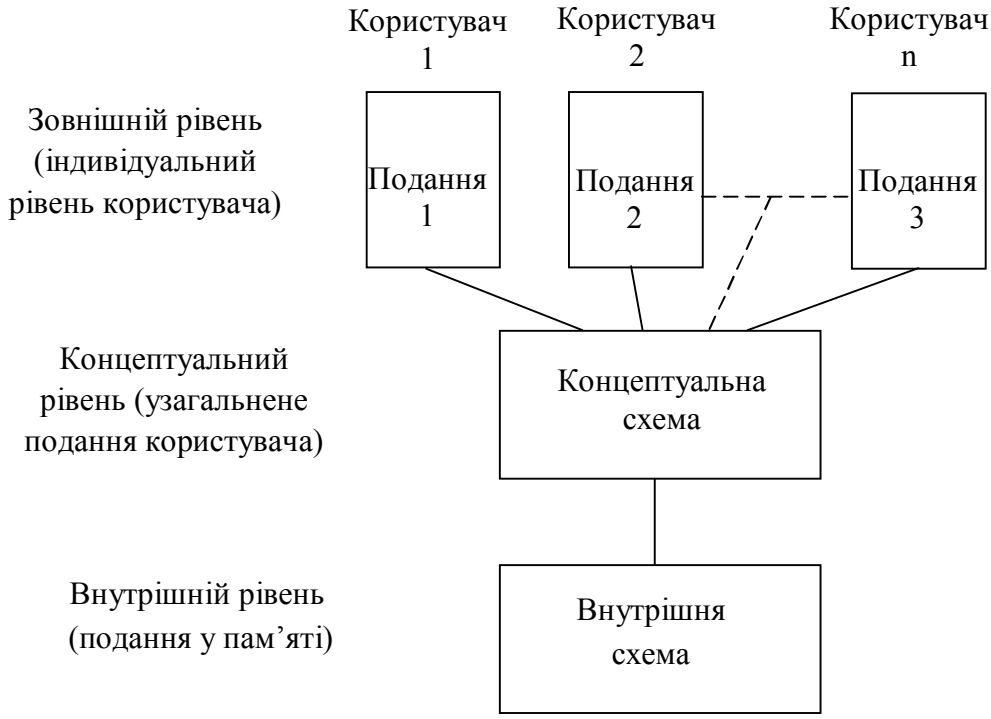


Рисунок 5.1 – Три рівні архітектури

Подання називається абстрактним, якщо воно містить логічні конструкції, які орієнтовані на користувача, такі як записи і поля, і не містить машинно-орієнтованих конструкцій, таких як біти та байти.

5.2 Зовнішній рівень

Зовнішній рівень – це індивідуальний рівень користувача. Користувач може бути прикладним програмістом або кінцевим користувачем з будь-яким рівнем професійної підготовки. Особливе місце серед користувачів займає адміністратор бази даних. (На відміну від інших користувачів його цікавить також концептуальний і внутрішній рівні). Кожен користувач має свою мову спілкування:

– для прикладного програміста це або одна з поширених мов програмування, така як C, COBOL чи PL/1, або спеціальна мова системи, що розглядається;

– для кінцевого користувача це спеціальна мова запитів або мова спеціального призначення, яка, можливо, заснована на формах і меню, створена спеціально з урахуванням вимог користувача і підтримується певним оперативним додатком.

Усі ці мови містять підмову даних, тобто підмножину операторів усієї мови, яка пов’язана лише з об'єктами та операціями баз даних. Цю підмову даних вбудовано в базову мову, яка забезпечує різні можливості, що безпосередньо не пов’язані з базами даних. Це підтримка локальних (тимчасових) змінних, обчислювальні операції, логічні операції типу if-then-else тощо. Існує одна мова, котра підтримується практично всіма сучасними СКБД – це SQL.

Окремого користувача цікавить лише деяка частина усієї бази даних; крім того, користувацьке подання такої частини буде взагалі чимось абстрактним у порівнянні зі способом фізичного збереження даних. Відповідно до термінології ANSI/SPARC подання окремого користувача називається зовнішнім поданням.

Визначення 5.1. Зовнішнє подання бази даних – це вміст бази даних, яким бачить його певний користувач (тобто для цього користувача зовнішнє подання і є база даних).

Узагалі, зовнішнє подання складається з множини примірників кожного типу зовнішнього запису, які не обов'язково збігаються зі збереженими записами. Підмова даних, якою послуговується користувач, визначена у термінах зовнішніх записів; наприклад, операція вибірки мови обробки даних проводитиме вибірку з примірників зовнішніх, а не збережених записів.

Кожне зовнішнє подання визначається засобами зовнішньої схеми, що загалом складається з визначень кожного типу записів у зовнішньому представленні. Зовнішня схема написана за допомогою мови визначення даних з користувацької підмови даних. (Тому мову визначення даних іноді називають *зовнішньою* мовою визначення даних.)

5.3 Концептуальний рівень

Концептуальне подання – це подання усієї інформації бази даних у більш абстрактній формі (як і у випадку зовнішнього подавання) проти фізичного способу збереження даних. Однак концептуальне подання суттєво відрізняється від способу подання даних окремому користувачеві. Концептуальне подання – це подання даних такими, які «вони є насправді», а не такими, якими змушений їх бачити користувач у рамках, наприклад, визначеної мови чи апаратного забезпечення, яке використовується.

Концептуальне подання складається з множини примірників кожного типу концептуального запису. Концептуальний запис зовсім не обов'язково повинний збігатися з зовнішнім записом, з одного боку, і зі збереженим записом – з іншого.

Концептуальне подання визначається за допомогою концептуальної схеми**,** що містить визначення кожного типу концептуальних записів. Концептуальна схема використовує іншу мову визначення даних – концептуальну. Для незалежності даних, не можна долучати до визначення концептуальної мови будь-який розгляд структури збереження чи методу доступу. Визначення концептуальної мови повинно стосуватися тільки до змісту інформації. Це означає, що в концептуальній схемі не може бути жодного згадування про подання збереженого файлу, послідовності збережених записів, індексування, хеш-адресацію тощо. Коли концептуальна схема справді забезпечує незалежність даних у цьому змісті, то зовнішні схеми, які визначені на базі концептуальної, свідомо забезпечуватимуть незалежність даних.

5.4 Внутрішній рівень

Третім рівнем архітектури є внутрішній рівень. Внутрішнє подання – це подання нижнього рівня усієї бази даних; воно складається з багатьох

примірників кожного типу внутрішнього запису. Термін «внутрішній запис» належить термінології ANSI/SPARC і означає конструкцію, яка називається збереженим записом. Внутрішнє подання так само, як зовнішнє і концептуальне, не пов'язане з фізичним рівнем, тому що в ньому не розглядаються фізичні записи (блоки, сторінки), і не розглядаються фізичні ділянки пристрою збереження, такі як циліндри й доріжки. Іншими словами, внутрішнє подання припускає нескінченний лінійний адресний простір; подробиці того, як адресний простір відображений на фізичному пристрої збереження, залежать від системи і навмисне не долучені до загальної архітектури.

Внутрішнє подання описується за допомогою внутрішньої схеми, що визначає не лише різні типи збережених записів, але ще наявні індекси, засоби подання збережених полів, фізичну послідовність збережених записів тощо.

5.5 Роль проектування даних в життєвому циклі інформаційних систем

Життєвий цикл інформаційної системи в загальному випадку починається в момент прийняття рішення про її створення і закінчується в момент виведення її з експлуатації. Основними його етапами (якщо опустити деталі) зазвичай є:

- проведення передпроектного обстеження;
- проектування даних;
- розробка додатків, тестування, написання документації;
- впровадження створеної ІС та навчання користувачів;
- експлуатація та супровід ІС;
- виведення з експлуатації та утилізація.

На етапі передпроектного обстеження здійснюються аналіз і моделювання бізнес-процесів, що підлягають автоматизації (іноді цей процес називається структурним моделюванням), а також формулюються вимоги до майбутнього продукту. Нерідко на цьому ж етапі проводиться вибір СКБД та інструментальних засобів. Зазвичай подібне обстеження проводиться за участю потенційних користувачів.

Етап проектування даних відбувається за участю потенційних користувачів. Іноді в процесі проектування даних створюються прототипи додатків, що дозволяють уточнити і доповнити вимоги до кінцевого продукту. Якщо передбачається, що впровадження нової ІС буде супроводжуватися виведенням з експлуатації її попередника, то приймаються рішення про те, яким чином використовувати успадковані дані, і в модель даних, яка є підсумком цього етапу, вносяться необхідні зміни.

Реальне створення програмного продукту, в тому числі клієнтських додатків і додатків, відповідальних за генерацію звітів і аналіз даних, відбувається на етапі розробки. Важливою частиною роботи в цьому випадку є також тестування та документування створюваного продукту. Даний етап зазвичай завершується створенням дистрибутива програми або його частин і документуванням процедури його інсталяції.

Вартість виправлення помилки, допущеної на попередньому етапі життєвого циклу, приблизно в десять разів перевищує витрати на її виправлення на поточному етапі. Зокрема, багато розробників стикаються з тим, що помилки проектування даних призводять іноді до написання коду великого об’єму, що так чи інакше їх компенсує, і нерідко викликають проблеми на етапі супроводу готового продукту.

Оскільки проектування даних – це етап, розташований безпосередньо за передпроектними обстеженнями, дуже важливо, щоб ця частина роботи над проектом була виконана максимально якісно.

Саме важливість цього етапу зумовила стрімке зростання популярності такої категорії програмного забезпечення, як засоби проектування даних, протягом останніх десяти років.

Багато сучасних СКБД містять візуальні засоби (нерідко входять до складу утиліт адміністрування), що дозволяють створити нову схему бази даних або переглянути вже наявну.

Проте останнім часом все більш популярними стають CASE-засоби (Computer-Aided System Engineering). Існує кілька типів CASE-засобів, але для створення баз даних найчастіше використовуються ті з них, що містять у своєму складі інструменти для створення діаграм «сутність-зв'язок» і проектування даних.

Розглянемо процес проектування даних за допомогою подібних інструментів.

Процес проектування даних можна умовно розділити на два етапи:

- логічне моделювання;
- фізичне проектування.

Результатом першого з них є так звана логічна (або концептуальна) модель даних, що виражається зазвичай діаграмою «сутність-зв'язок» або ER (Entity-Relationship) діаграмою, яка представлена в одній зі стандартних нотацій, прийнятих для відображення подібних діаграм.

Результатом другого етапу є готова база даних або DDL-скрипт для її створення.

Визначення 5.2. DDL (Data Definition Language) – мова визначення даних, підмножина SQL.

Логічне моделювання. Логічна модель даних описує факти і об'єкти, що підлягають реєстрації в майбутній базі даних. Основними компонентами такої моделі є сутності, їх атрибути та зв'язки між ними. Як правило, фізичним аналогом сутності в майбутній базі даних є таблиця, а фізичним аналогом атрибута – поле цієї таблиці.

З логічної точки зору сутність являє собою сукупність однотипних об’єктів або фактів, які називаються примірниками цієї сутності. Фізичним аналогом примірника зазвичай є запис в таблиці бази даних. Як і записи в таблиці реляційної СКБД, примірники сутності повинні бути унікальними, тобто повний набір значень їх атрибутів не повинен дублюватися. Як і поля в таблиці, атрибути можуть бути ключовими й неключовими.

На етапі логічного проектування для кожного атрибута зазвичай визначається приблизний тип даних (строковий, числовий, BLOB та ін.) Конкретизація відбувається на етапі фізичного проектування, так як різні СКБД підтримують різні типи даних і обмеження на їх довжину або точність.

Зв’язок з логічної точки зору являє собою співвідношення між сутностями, яке нерідко може бути виражене звичайними фразами, наприклад: «Клієнт розміщує замовлення» («A customer places an order» – цією фразою цілком можна описати зв’язок, зображений на рис. 4.2), де іменниками є назви пов’язаних між собою сутностей.

Зазначимо, що на етапі логічного проектування можна описати поведінку СКБД при порушенні правил цілісності, які визначаються даним зв’язком (наприклад, при видаленні відомостей про замовника, який розмістив хоча б одне замовлення, для зв’язку, зображеного на рис. 4.2).

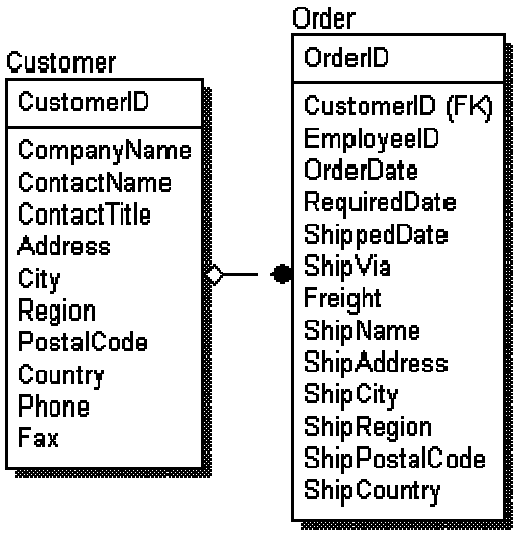


Рисунок 5.2 – Приклад зв’язку між сутностями логічної моделі

В процесі створення фізичної моделі ці шаблони перетворюються в код на процедурному розширенні мови SQL, що застосовується в конкретній СКБД.

Зазначимо, що логічна модель даних, як правило, не пов'язана з конкретною СКБД і не враховує технічні особливості конкретних платформ, що застосовуються при її подальшої фізичної реалізації.

Фізичне проектування даних. Фізичне проектування даних здійснюється на основі логічної моделі. Результатом цього процесу є фізична модель, яка містить повну інформацію, необхідну для генерації всіх необхідних об'єктів в базі даних. Для СКБД, що підтримують системний каталог, фізична модель відповідає його вмісту.

Зазвичай на підставі фізичної моделі створюється DDL -скрипт для створення об'єктів бази даних, або ці об'єкти створюються безпосередньо з CASE-засобу – переважна більшість сучасних засобів такого класу підтримує різні механізми доступу до даних і може виступати в ролі клієнтського додатка, який ініціює виконання DDL-скриптів.

У процесі фізичного проектування слід визначити найменування таблиць і типи даних для всіх полів. Якщо необхідно, на цьому ж етапі описуються подання (якщо такі будуть створюватися) і може бути створений код збережених процедур. Далі визначається, які саме об'єкти і як будуть створюватися: наприклад, за допомогою яких SQL-запитів створюються індекси, за допомогою яких об'єктів – тригерів або серверних обмежень – реалізується посилальна цілісність, чи створюються індекси для альтернативних ключів тощо.

Приклад діаграми, що відображає фізичну модель для SQL Server 7.0, що відповідає наведеній вище логічної моделі, представлений на рис. 4.3.

Як правило, сучасні засоби проектування даних підтримують кілька типів СКБД (наприклад, ERwin фірми Computer Associates підтримує більше 20 різних СКБД).

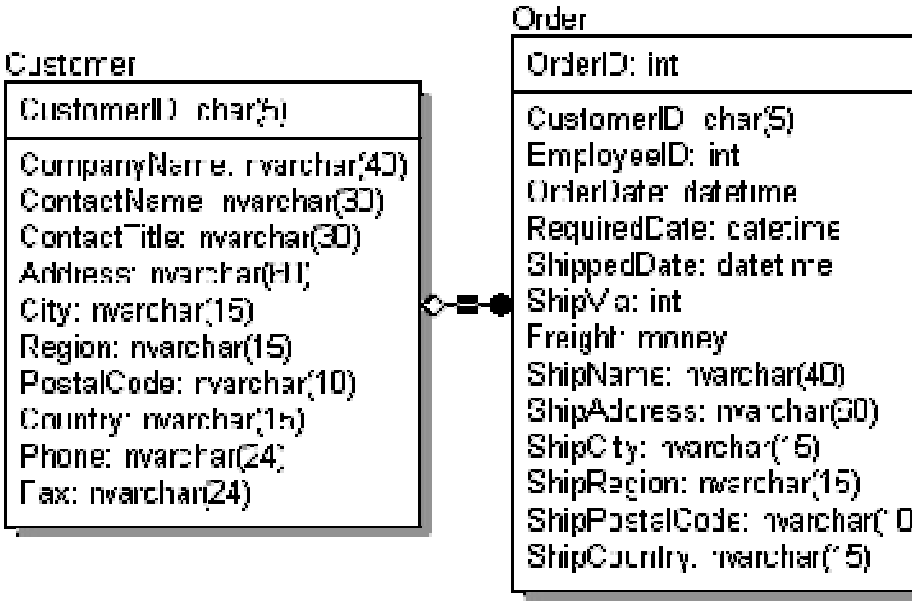


Рисунок 5.3 – Приклад найпростішої фізичної моделі даних

Рівень підтримки тієї чи іншої платформи в різних засобах проектування даних може бути різний. Наприклад, конкретний засіб може підтримувати або не підтримувати для даної СКБД такі особливості, як створення збережених процедур, генерація об’єктів фізичної пам’яті (табличних просторів, сегментів відкоту тощо), завдання розміщення об’єктів бази даних у фізичних об’єктах тощо. Тому, вибираючи засіб проектування даних для вирішення конкретного завдання, необхідно зважати на його можливості з точки зору підтримки особливостей тієї або іншої платформи – при вдалому розкладі можна заощадити чимало часу на «ручне» доведення створюваної бази даних (або DDL-скрипта для її генерації) до необхідного стану.

При цьому, чим більше можливостей і платформ підтримує конкретний засіб проектування даних, тим дорожче воно коштує (зазначимо, що CASE-засоби – це не найдешевші програмні продукти – ціни на ці програми коливаються від однієї до кількох десятків тисяч доларів).

Відзначимо, що фізичне проектування може включати і додаткову «ручну» роботу з доведення бази даних або скрипта для її генерації до

«товарного» вигляду. Наприклад, нерідко в скрипт також включаються SQL-пропозиції для заповнення деяких таблиць, дані яких більш-менш постійні, таких, наприклад, як довідник телефонних кодів різних країн, а також нестандартні тригери і процедури.

Переважна більшість сучасних засобів проектування даних надають можливість не тільки документувати модель, але і створювати по ній звіти, що містять ті чи інші відомості про моделі. Деякі засоби проектування даних дозволяють зберігати їх моделі в спеціальних репозитаріях, а також здійснювати колективне проектування.

Крім того, переважна більшість засобів проектування даних дозволяє відновлювати фізичну модель даних їх системного каталогу та представляти її у вигляді ER-діаграми (цей процес називається зворотним проектуванням – reverse engineering), а також проводити синхронізацію системного каталогу і фізичної моделі. При створенні інформаційної системи, яка повинна використовувати успадковані від попередніх їй систем дані, така особливість вельми корисна – в цьому випадку логічне проектування можна почати з модифікації вже наявної моделі (цей процес отримав назву round-trip engineering).

Найбільш популярні CASE-засоби засоби проектування даних

CASE – засіб	**Виробник**	**URL**
Designer 2000	Oracle	www.oracle.com/
ERwin	Computer Associates	www.cai.com/
PowerDesigner	Sybase	www.sybase.com/
ER/Studio	Embarcadero	www.embarcadero.com/
System Architect	Popkin Software	www.popkin.com/
Visible Analyst	Visible Systems	www.visible.com
Visio Enterprise	Microsoft	www.microsoft.com/

6 УПРАВЛІННЯ ПРОГРАМНИМИ ПРОЕКТАМИ

6.1 Управління конфігурацією програмного продукту

Основними програмними додатками, представленими на ринку систем управління проектами, є:

1) системи календарного планування:

– MS Project (розробник – фірма Microsoft);

– Time Line (розробник – компанія Time Line Solutions);

– SureTrak Project Manager (розробник – компанія Primavera).

2) професійні системи управління проектами:

– Primavera Project Planner (розробник - компанія Primavera);

– Open Plan (розробник – компанія Welcom Software Technology);

– Spider Project (розробник – компанія «Технології управління «Спайдер»).

У таблиці 6.1 проведено ранжування множини розглянутих програмних систем за основними критеріями ефективності.

Основними завданнями, алгоритми розв’язання яких представлені у відомих системах управління проектами, є такі:

– розробка розкладу виконання множини операцій проекту без урахування обмеженості ресурсів;

– розробка розкладу виконання множини операцій проекту з урахуванням обмеженості ресурсів;

– визначення критичного шляху і резервів часу виконання операцій проекту;

– визначення потреби проекту у фінансуванні, витратних матеріалах і обладнанні;

– визначення розподілу в часі завантаження поновлюваних ресурсів;

– аналіз множини можливих ризиків і планування розкладу з урахуванням ризиків;

– аналіз відповідності процесу виконання проекту плановим показникам;

– аналіз відхилень ходу робіт від запланованого і прогнозування основних параметрів проекту.

Таблиця 6.1 – Ранжування множини програмних систем за основними критеріями ефективності

Система / Критерій	MS Project	Time Line	SureTrak	P3	Open Plan	Spider Project
Навченість персоналу	1	1	1	0	0	1
Зручність інтерфейсу	1	0	1	1	1	0
Інтерактивний самовчитель	1	0	1	1	1	0
Функціональні можливості	0	0	0	1	1	1
Управління ризиками	0	1	0	1	1	1
Робота з великим числом проектів	0	1	1	1	1	1
Відкритість архітектури	1	1	1	1	1	0
Вартість ПЗ	1	1	1	0	0	0
Простота інсталяції	0	1	1	0	0	1
Моделювання доходів	0	0	0	1	0	1
Загальний бал	5	6	7	7	6	6

Систематизація найбільш популярних програмних систем управління проектами, які займають значну частку ринку відповідного програмного забезпечення, представлена в таблиці 6.2.

Таблиця 6.2 – Характеристика програмних систем управління проектами

Назва	Переваги	Недоліки
1	2	3
MS Project 2007	– ефективний аналіз графіків проектів і управління ними; – створення професійно оформлених діаграм і схем; – посилення контролю за ресурсами і фінансами	– установка в повному робочому варіанті і інтеграція вимагає досвідченого інженера; – складні цикли в роботі
Time Line 6.5	– реалізація концепції багато-проектного планування в рамках організації; – гнучкі засоби підтримки формування звітів; – зняті обмеження на розмірність проектів; – достатньо потужні алгоритми роботи з ресурсами; – зберігання всіх даних в єдиній SQL-базі даних	– відсутність можливостей опису і відображення ієрархії ресурсів організації; – через велику кількість інформації, пропонованої програмою, інтерфейс занадто перевантажений; – програма не розвивається, функціональні змоги не вдосконалюються
Primavera SureTrak	– планування і складання розкладів для невеликих проектів; – інтуїтивний інструментарій для легкого опису проектів; – швидкий доступ до інтерактивних фільтрам, призначеним для побудови звітності в реальному часі і проведення аналізу методом критичного шляху; – створення описів ресурсів, які настроюються для створення додаткових більш ефективних звітів	– відсутність засобів багатопроектного управління; – відсутність засобів

Продовження таблиці 6.2

1	2	3
		фрагментації проектів; – менша розмірність проектів, ніж в аналогах; – прості засоби створення звітів
Primavera Project Planner	– проектна інформація та результати реалізації як одного проекту, так і групи доступні для керівників, менеджерів компанії і всіх учасників системи; – координація, виявлення відхилень від стратегічного курсу компанії, - можливість внести коригуючий вплив в систему управління проектом; – ефективна взаємодія учасників проекту в рамках виконання загальної задачі	– слабкі можливості візуалізації графіки; – незручне подання звітів; – обмеження на кількість календарів; – обмеження на кількість контрольованих ресурсів
Open Plan	– відкрита архітектура з можливістю підключення зовнішніх додатків; – підтримка ієрархічних структур ресурсів; – розрахунок, прогнозування, аналіз і витрат відбувається автоматично; – аналіз ризиків; – робота з великим числом проектів; – розмежування доступу до інформації	– складність в освоєнні програми; – обмежена кількість операцій, які виконуються з ресурсами; – відсутність можливості моделювання доходів, ризиків, необхідних резервів; – досить дорогий продукт
Spider Project	– розвинені алгоритми планування використання обмежених ресурсів;	– недоробки у програмній реалізації; – немає інструментів для

Закінчення таблиці 6.2

1	2	3
	– можливість використання при складанні розкладу робіт взаємозамінних ресурсів; – можливість використання нормативно-довідкової інформації; – нескладне конфігурування	інтеграції з іншими додатками

Термін software – програмне забезпечення (ПЗ) – увів в 1958 році всесвітньо відомий статистик Джон Тьюкей (John Tukey).

З 1990 по 1995 роки велася робота над міжнародним стандартом щодо єдиного подання про процеси розробки програмного забезпечення. Як результат, було видано стандарт ISO/IEC 12207.

У 2004 році була створена фундаментальна праця «Керівництво до зведення знань по програмній інженерії» (SWEBOK), в якій було зібрано основні теоретичні й практичні відомості, накопичені в цій галузі.

Варто ввести робочі визначення ряду термінів, які будуть в подальшому використовуватись.

Визначення 6.1. Програмування – це процес відображення певної множини цілей на множину машинних команд і даних, інтерпретація яких на комп'ютері або обчислювальному комплексі забезпечує досягнення рішення поставленої задачі.

Це відображення може бути дуже простим, наприклад, запис у текстовому редакторі. А може бути багатоступінчатим і дуже складним, коли спочатку цілі відображаються на вимоги до системи, вимоги – на високорівневу архітектуру і специфікації компонентів, специфікації – на дизайн компонентів, дизайн – на початковий код.

Далі початковий код за допомогою компіляторів відображається на код розгортання, код розгортання – на виклики функцій ПЗ оточення (ОС,

проміжне ПЗ, бази даних), яке може розташовуватися на множині комп'ютерів, об'єднаних в мережу, і лише після цього – в машинні команди і дані.

Визначення 6.2. Професійне програмування (синонім виробництво програм) – діяльність, направлена на отримання доходів за допомогою програмування.

Професійне розроблення програм – це завжди колективна діяльність, в якій беруть участь мінімум дві особи: програміст і споживач.

Програмний продукт – сукупність програм і супровідної документації по їх установці, настройці, використанню і доопрацюванню. Якщо програмний продукт достатньо складний, то його розгортання у клієнтів, як правило, реалізується окремими самостійними проектами впровадження. Супровід включає усунення критичних несправностей в системі і реалізується часто не як проект, а як процесна діяльність.

Підтримка полягає в розробці нової функціональності, переробці вже існуючій функціональності, у зв'язку із зміною вимог, і поліпшенням продукту, а також включає усунення некритичних зауважень до ПЗ, виявлених при його експлуатації .

Життєвий цикл програмного продукту завершується виведенням продукту з експлуатації і зняттям його з підтримка і супроводи.

Визначення 6.3. Процес розробки ПЗ – сукупність процесів, що забезпечують створення і розвиток програмного забезпечення.

Найпоширеніший процес розробки ПЗ на практиці – це процес «як вийде». Це не означає, що процесу як такого немає. Він є і, як правило, забезпечує розробку ПЗ при прийнятних витратах і якості, але цей процес не документований, є «знанням зграї», тримається на людях і передається з покоління в покоління. Цілеспрямована робота за оцінкою ефективності і поліпшенню процесу не ведеться.

Визначення 6.4. Модель процесу розробки ПЗ – формалізоване представлення процесу розробки ПЗ.

Згідно зSWEBOK 2004, програмна інженерія включає 10 основних і 7 додаткових областей знань, на яких базуються процеси розробки ПЗ.

До основних областей знань належать такі наступні області:

1. Software requirements – програмні вимоги.

2. Software design – дизайн (архітектура).

3. Software construction – конструювання програмного забезпечення.

4. Software testing – тестування.

5. Software maintenance – експлуатація (підтримка) програмного забезпечення.

6. Software configuration management – конфігураційне управління.

7. Software engineering management – управління в програмній інженерії.

8. Software engineering process – процеси програмної інженерії.

9. Software engineering tools and methods – інструменти і методи.

10. Software quality – якість програмного забезпечення.

Додаткові області знань включають:

1. Computer engineering – розробка комп'ютерів.

2. Computer science – інформатика.

3. Management – загальний менеджмент.

4. Mathematics – математика.

5. Project management – управління проектами.

6. Quality management – управління якістю.

7. Systems engineering – системне проектування.

Все це необхідно знати і уміти застосовувати, для того, щоб розробляти ПЗ. Очевидно, управління проектами, про яке ми говоритимемо далі, лише одна з 17 областей знань програмної інженерії, і та допоміжна. Проте основною причиною більшості провалів програмних проектів є саме застосування неадекватних методів управління розробкою.

Standish Group, проаналізувавши роботу сотень американських корпорацій і підсумки виконання декількох десятків тисяч проектів,

пов'язаних з розробкою ПЗ, в своїй доповіді з красномовною назвою «Хаос» прийшла до наступних невтішних висновків:

Тільки 35 % проектів завершилися в строк, не перевищили запланований бюджет і реалізували всі необхідні функції і можливості.

46 % проектів завершилися із запізненням, витрати перевищили запланований бюджет, необхідні функції не були реалізовані в повному об'ємі.

Середнє перевищення термінів склало 120 %, середнє перевищення витрат 100 %, зазвичай виключалося значне число функцій.

19 % проектів повністю провалилися і були анулювані до завершення.

Відповіді на вічні питання «Хто винен?» і «Що робити?». Хто винен? Ніхто. Як ніхто не винен в тому, що на небі хмари, що йде дощ, що дме вітер. Оскільки самого-то «хаосу» не було, і ні, а є лише Богом дана (для атеїстів – об'єктивна) реальність, яка поміщена в особливій специфіці виробництва програм, в порівнянні з будь-якою іншою виробничою діяльністю.

Продукт праці програмістів не є матеріальним, це колективні ментальні моделі, записані мовою програмування. І цю специфіку необхідно враховувати.

Що робити? Керувати людьми. Успіх, як і провал проектів щодо розроблення ПЗ належить до сфери психології.

Гуру програмування Ф. Брукс в 1975 році писав: «Програміст, подібно поетові, працює майже безпосередньо з чистою думкою. Він будує свої замки в повітрі і з повітря, творить силою уяви».

Саме внаслідок унікальності галузі досвід професіоналів, накопичений в матеріальному виробництві і викладений в стандарті PMI PMBOK, мало сприяє успіху в управлінні програмним проектом.

Управляти розробкою ПЗ треба інакше. Творчість – це інтелектуальна діяльність людини, закони якої нам невідомі. Якби ми знали закони творчості, то і картини, і вірші, і музику, і програми вже давно б створювали комп'ютери.

Творчий початок це те, що ріднить програмування з наукою і мистецтвом. Творчість в програмуванні починається з визначення цілей програми і закінчується тільки тоді, коли в її коді, написаному на якій-небудь мові програмування, поставлена остання крапка.

Спроби розділяти програмістів на творчу еліту, архітекторів і проектувальників, і нетворчих програмістів-кодерів не мають під собою об’єктивних підстав.

Навіть якщо алгоритм програми, строго визначений математично, два різних програміста його закодують по-різному, і отримана програма матиме різні споживчі якості.

Творчість нерозривно пов'язана з натхненням, а це субстанція капризна і непередбачувана. Але без натхнення в програмуванні не обійтися. І чим складніше завдання, тим важче витягнути це натхнення з підсвідомості. Іноді для цього потрібні години, а іноді тижні.

Програмування – не мистецтво, в тому сенсі, що воно не є творчим віддзеркаленням і відтворенням дійсності в художніх образах. Про мистецтво в програмуванні можна говорити тільки в сенсі уміння, майстерності, знання справи, як і в будь-якій іншій професії.

І як в будь-якій іншій професії майстерність програміста може доставляти дійсну естетичну насолоду, але тільки для людей, причетних до цієї професії.

Ще одна особливість, яка властива творчості програміста – це постійне оновлення інформаційних технологій, які програмістові необхідно знати і успішно застосовувати в своїй роботі. Програміст має зважати і активно застосовувати на практиці гігабайти професійної інформації. Це обладнання комп'ютерів, комп'ютерних мереж і мережеві протоколи. Це операційні системи і мови програмування. Це програмні інтерфейси проміжного ПЗ і прикладних бібліотек з особливостями і перевагами їх реалізації в конкретних продуктах. Це технологічні стандарти, технології розробки тощо.

«Творчий політ» «творчий застій» – це про діячів мистецтва. Але багато менеджерів не визнають об’єктивний факт нерівномірності продуктивності програмістів та намагаються «штовхати» програмістів, неначе це примусить їх думати швидше. Не примусить. Але може примусити звільнитися або зайнятися імітацією роботи.

Дуже велика спокуса провести аналогію з цими галузями і говорити про індустріальному підході до розробки ПЗ. Не виходить.

Існує думка, що для колективної творчості програмістів скоріше доречна аналогія з створенням художнього кінофільму або театрального спектаклю. Кількість провальних проектів в цих областях не менше, ніж в програмуванні. Про це пише авторитет в управлінні програмними проектами У.Ройс: «Менеджери програмних проектів зможуть добитися більшого, якщо застосовуватимуть методи управління характерні для кіноіндустрії». І ще одна аналогія програмних проектів з кінематографом. Наявність навіть самих зоряних акторів не забезпечує успіх фільму. Тільки талановитий режисер здатний організувати і надихнути акторів на створення шедевра відкрити нові зірки. А талановитих режисерів, як, втім, і талановитих менеджерів програмних проектів, на жаль, не так багато.

6.2 Еволюція підходів до управління програмними проектами

За 50 років розвитку програмної інженерії накопичилася велика кількість моделей розробки ПЗ. Розглянемо еволюцію підходів до управління програмними проектами.

1. «Як вийде». Розімкнена система управління. Повна довіра технічним лідерам. Представники бізнесу практично не беруть участь в проекті. Планування, якщо воно і є, то неформальне і словесне. Час і бюджет, як правило, не контролюються.

2. «Водопад» або каскадна модель. Жорстке управління із зворотним зв'язком. Розрахунок опорної траєкторії (план проекту), вимірювання

відхилень, корекція і повернення на опорну траєкторію. Краще, але не ефективно.

3. «Гнучке управління». Розрахунок опорної траєкторії, вимірювання відхилень розрахунок нової уточненої траєкторії і корекція для виходу на неї. «Плани – ніщо, планування – все» (Ейзенхауер, Дуайт Девід).

4. «Метод частих постачань». Класичні методи управління перестають працювати у випадках, коли структура і властивості керованого об'єкту нам не відомі і/або змінюються з часом. Ці підходи так само не допоможуть, якщо поточні властивості об'єкту не дозволяють йому рухатися з необхідними характеристиками. Аналогічно, якщо робоча група проекту не може забезпечити необхідну ефективність і тому постійно працює в режимі авралу, то це приводить не до зростання продуктивності, а до відходу професіоналів з проекту.

5. Концепція і методологія IT проекту. У 1987 року Інститутом управління проектами (США) було запропоновано таке визначення: «Проект є завданням з певними вихідними даними і бажаними результатами цілями, які обумовлюють спосіб його вирішення».

Вадами цього визначення було визнано те, що спосіб вирішення обумовлюється не тільки і не завжди результатами його вирішення, так і те, що у визначенні проекту не згадано засоби його реалізації. В літературі зустрічаються інші визначення поняття «проект», наприклад, «проект містить в собі задум (проблему), засоби її реалізації (вирішення проблеми) і одержані внаслідок її реалізації результати». Або ще: «проект – це сукупність певних елементів (об'єктів матеріальної нематеріальної природи) і зв'язків між ними, що забезпечує досягнення поставлених цілей». Ці визначення можна вважати універсальними, методично важеними й достатньо повними. До їх недоліків можна віднести те, що в них практично відсутній змістовий аспект.

Не торкаючись теоретичних суперечок про вади і переваги різних визначень проекту, пропонуємо керуватися поняттям, яке найближче до визначення, даного в методичних матеріалах Всесвітнього банку: «Проект —

це комплекс взаємопов'язаних заходів, розроблених для досягнення певних цілей протягом заданого часу при встановлених ресурсних обмеженнях».

3. Управління ІТ проектом. Для досягнення стабільних позитивних результатів проекті повинні бути легкокерованими. Розробка легкокерованих проектів вимагає діяльності в двох паралельних напрямах.

По-перше, необхідно залучити всіх стейкхолдерів, хто робить ставку на проект (тобто всіх, хто може виграти при здійсненні проекту), до визначенні конкретних цілей проекту і засобів їх досягнення.

По-друге, необхідний пошук такого варіанту (серед наявних варіантів), який би забезпечував економне витрачання ресурсів при реалізації проекту.

4. Особливості управління ІТ проектами Коли структура і властивості керованого об'єкту нам не відомі, необхідно використовувати адаптивне управління, яке, додатково до прямих управляючих дій, направлено на вивчення і зміну властивостей керованого об'єкту. Для того, щоб зрозуміти структуру і властивості об'єкту і впливати на нього з метою їх приведення до бажаного стану, в проекті повинен бути додатковий контур зворотного зв'язку – контур адаптації. Відомо, що продуктивність різних програмістів може відрізнятися в десятки разів. Більш того, продуктивність одного і того ж програміста може відрізнятися в десятки разів в залежності від виду праці. Тому, крім чисто управлінських завдань, керівник, якщо він прагне отримати найвищу продуктивність робочої групи, повинен направляти постійні зусилля на вивчення і зміну об'єкту управління: людей і їх взаємодії.

6.3 Моделі процесу розробки програмного забезпечення

Моделі процесів розробки ІТ проектів прийнято класифікувати по «вазі» – кількості формалізованих процесів (більшість процесів або тільки основні) і детальної їх регламентації. Чим більше процесів документовано, чим детальніше вони описані, тим більше «вага» моделі.

Найбільш поширені сучасні моделі процесу розробки ПО подано на рис 5.1.

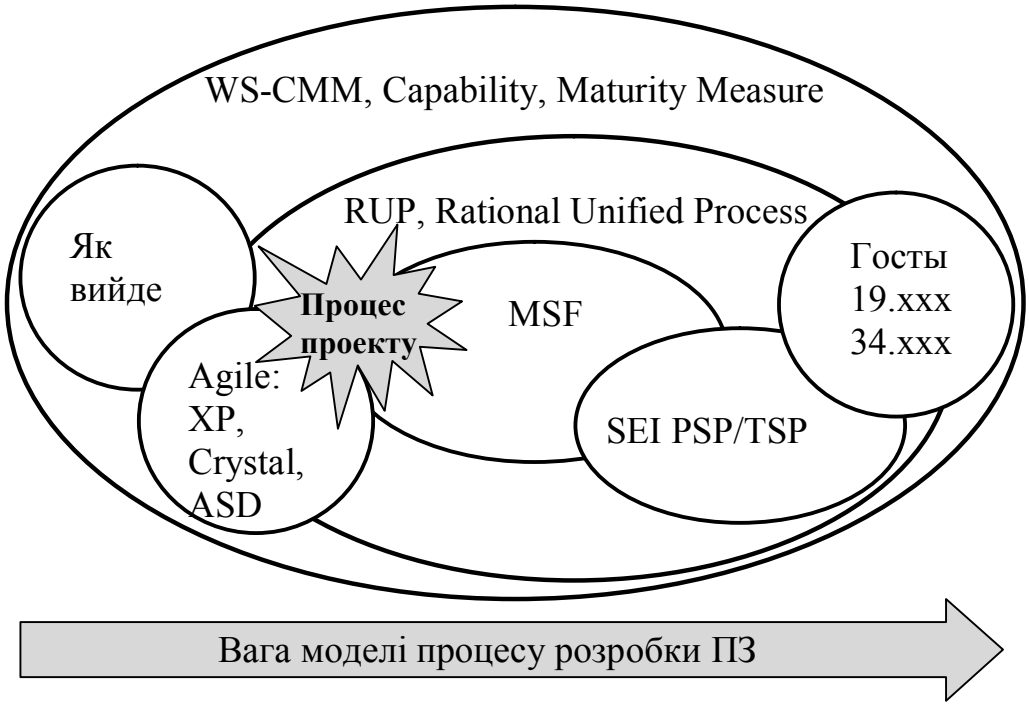


Рисунок 6.1 – Моделі процесу розробки ПЗ і їх розподіл за «вагою»

Стандарти. ГОСТ 19 «Єдина система програмної документації» і ГОСТ 34 «Стандарти на розробку і супровід автоматизованих систем» орієнтовані на послідовний підхід до розробки ПЗ. Розробка відповідно до цих стандартів проводиться по етапах, кожний з яких припускає виконання певних робіт, і завершується випуском достатньо великого числа вельми формалізованих документів.

Таким чином, строге проходження цими стандартами не тільки приводить до результату, але і вимагає дуже високого ступеня формалізації розробки.

SW-CMM – Capability Maturity Model for Software. В середині 80-х років минулого сторіччя Міністерство оборони США задумалося про те, як вибирати розробників ПО при реалізації великомасштабних програмних проектів. За замовленням військових Інститут програмній інженерії, що

входить до складу Університету Карнеги-Меллона розробив SW-CMM, в якості еталонної моделі організації розробки програмного забезпечення.

Дана модель визначає п'ять рівнів процесу розробки ПЗ.

1. Початковий – процес розробки носить хаотичний характер. Він лише зосереджується на формуванні команди конкретних виконавців.

2. Повторюваний – встановлені основні процеси управління проектами: відстежування витрат, термінів і функціональності. Впорядкування цих процесів, необхідно для того, щоб повторити попередні досягнення на аналогічних проектах.

3. Визначений – процеси розробки ПЗ і управління проектами описані і упроваджені в єдину систему процесів компанії. У всіх проектах використовується стандартний для організації процес розробки і підтримки програмного забезпечення, адаптований під конкретний проект.

4. Керований – збираються детальні кількісні дані по функціонуванню процесів розробки і якості кінцевого продукту. Аналізуються значення і динаміка цих даних.

5. Що оптимізується – постійне поліпшення процесів ґрунтується на кількісних даних по процесах і на пробному впровадженні нових ідей і технологій.

Документація з повним описом SW-CMM займає близько 500 сторінок і визначає набір з 312 вимог, яким повинна відповідати організація, якщо вона планує атестуватися за цим стандартом на 5-й рівень зрілості.

5. RUP. Уніфікований процес (Rational Unified Process, RUP) був розроблений Пилипом Крачтеном (Philippe Kruchten), Іваром Якобсоном (Ivar Jacobson) і іншими співробітниками компанії "Rational Software" як доповнення до мові моделювання UML.

Модель RUP описує абстрактний загальний процес, на основі якого організація або проектна команда повинна створити конкретний спеціалізований процес, орієнтований на її потреби. Саме ця межа RUP викликає основну критику – оскільки він може бути чим завгодно, його не

можна вважати нічим визначеним. В результаті такої загальної побудови RUP можна використовувати і як основу для традиційного стилю водопаду розробки, так і в якості гнучкого процесу.

6. MSF – Microsoft Solutions Framework (MSF) [11] – це гнучка і достатньо легковага модель, побудована на основі ітеративної розробки. Привабливою особливістю MSF є велика увага до створення ефективної проектної команди. Для досягнення цієї мети MSF пропонує достатньо нестандартні підходи до організаційної структури розподілу відповідальності і принципам взаємодії усередині команди.

7. PSP/TSP. Одна з останніх розробок Інституту програмної інженерії Personal Software Process / Team Software Process]. Personal Software Process визначає вимоги до компетенцій розробника. Згідно цієї моделі кожен програміст повинен уміти: враховувати час, витрачений на роботу над проектом; враховувати знайдені дефекти; класифікувати типи дефектів; оцінювати розмір завдання; здійснювати систематичний підхід до опису результатів тестування; планувати програмні завдання; розподіляти їх за часом і складати графік роботи. виконувати індивідуальну перевірку проекту і архітектури; здійснювати індивідуальну перевірку коду; виконувати регресійне тестування.

Team Software Process орієнтується на самокеровані команди чисельністю 3–20 розробників. Команди повинні: встановити власні цілі; скласти свій процес і плани; відстежувати роботу; підтримувати мотивацію і максимальну продуктивність. Послідовне застосування моделі PSP/TSP дозволяє зробити нормою в організації п'ятий рівень CMM – модель зрілості можливостей (Capability Maturity Model).

Agile. Основна ідея всіх гнучких моделей полягає в тому, що вживаний в розробці ПЗ процес повинен бути адаптивним. Вони декларують своєю вищою цінністю орієнтованість на людей і їх взаємодію, а не на процеси і засоби.

По суті, так звані гнучкі методології це не методології, а набір практик, які можуть дозволити (а можуть і ні) добиватися ефективної розробки ПЗ, ґрунтуючись на ітеративності, інкрементальності, самокерованості команди і адаптивності процесу.

6.4 Вибір моделі процесу

Важкі і легкі моделі виробничого процесу мають свої переваги і свої недоліки. Ефективність сильно залежить від індивідуальних здібностей виконавців.

Алістер Коуберн, один з авторів «Маніфесту гнучкої розробки ПЗ» проаналізував дуже різні програмні проекти, які виконувалися по різним моделям від абсолютно полегшених і «гнучких» до важких (CMM-5) за останні 20 років [15, 16].

Він не виявив кореляції між успіхом або провалом проектів і моделями процесу розробки, які застосовувалися в проектах. Звідси він зробив висновок про те, що ефективність розробки ПЗ не залежить від моделі процесу, а також про те, що : У кожного проекту повинна бути своя модель процесу розробки.

У кожної моделі – свій час. Це означає, що не існує єдиного правильного процесу розробки ПЗ, в кожному новому проекті процес повинен визначатися кожен раз наново, залежно від проекту, продукту і персоналу, у відповідності з «Законом 4-х П». Абсолютно різні процеси мають застосовуватися в проектах, в яких беруть участь 5 осіб, і в проектах, в яких беруть участь 500 осіб.

Якщо продуктом проекту є критичне ПЗ наприклад, система управління атомною електростанцією, то процес розробки повинен сильно відрізнятися від розробки, наприклад, сайту «відпочинь.ua». І, нарешті, по-різному слід організовувати процес розробки в команді вчорашніх студентів і в команді визнаних професіоналів.

Що треба робити для успіху програмного проекту. Стів Макконнелл приводить тест програмного проекту на виживання. Це дуже показовий чек-лист з 33-х пунктів.

Керівник програмного проекту повинен його періодично використовувати для внутрішнього аудиту своїх процесів.

Щоб програмний проект став успішним, необхідно:

1. Чітко ставити цілі.

2. Визначати спосіб досягнення мети.

3. Контролювати й керувати реалізацією.

4. Аналізувати загрози й протистояти їм.

5. Створювати команду.

1. Ставимо цілі

1.1 Концепція визначає ясні недвозначні цілі.

1.2 Всі члени команди рахують концепцію реалістичною.

1.3 У проекті є обґрунтування економічної ефективності.

1.4 Розроблений прототип призначеного для користувача інтерфейсу.

1.5 Розроблена специфікація цільових функцій програмного продукту.

1.6 З кінцевими користувачами продукту налагоджений двосторонній зв'язок

2. Визначаємо спосіб досягнення мети

2.1 Є детальний письмовий план розробки продукту.

2.2 У список завдань проекту включені «другорядні» завдання (управління конфігураціями, конвертація даних, інтеграція з іншими системами).

2.3 Після кожної фази проекту оновлюється розклад і бюджет.

2.4 Архітектура і проектні рішення документовані.

2.5 Є план забезпечення якості, що визначає тестування і рецензування.

2.6 Визначений план багатоетапного постачання продукту.

2.7. У плані враховано навчання, вихідні, відпустки, лікарняні.

2.8 План проекту і розклад схвалений всіма учасниками команди.

3. Контролюємо і управляємо реалізацією

3.1 У проекту є куратор. Це такий топ-менеджер виконуючої компанії, який особисто зацікавлений в успіху даного проекту.

3.2. У проекту є менеджер, причому тільки один!

3.3 У плані проекту визначені «бінарні» контрольні точки.

3.4 Всі зацікавлені сторони можуть отримати необхідну інформацію про хід проекту.

3.5 Між керівництвом і розробниками встановлені довірителі відносини.

3.6 Встановлена процедура управління змінами в проекті.

3.7 Визначені особи, відповідальні за рішення про ухвалення змін в проекті.

3.8 План, розклад і статусна інформація за проектом доступна кожному учасникові.

3.9 Код системи проходить автоматичне рецензування.

3.10 Застосовується система управління дефектами.

4. Аналізуємо загрози.

4.1 Складаємо список ризиків проекту. Здійснюється його регулярний аналіз і оновлення.

4.2 Керівник проекту відстежує виникнення нових ризиків.

4.3 Для кожного підрядчика визначена особа, відповідальна за роботу з ним.

5. Працюємо над створенням команди

5.1 Досвід команди достатній для виконання проекту.

5.2 У команди достатня компетенція в прикладній сфері.

5.3 У проекті є технічний лідер.

5.4 Чисельність персоналу достатня.

5.5 У команди є достатня згуртованість.

5.6. Всі учасники прихильні проекту.

Оцінка і інтерпретація тесту.

Оцінка: сума балів, кожен пункт оцінюється від 0 до 3:

– 0 – навіть не чули про це;

– 1 – чули, але поки не вживаний;

– 2 – застосовується частково;

– 3 – застосовується повною мірою.

Поправочні коефіцієнти:

– для малих проектів (до 5 осіб) – 1.5;

– для середніх (від 5 до 20 осіб) – 1.25.

Результат: 90 – чудовий результат. 100 % шансів на успіх.

Цей чек-лист перераховує, що потрібно робити, щоб програмний проект був успішним. але не дає відповіді на запитання, як це потрібно робити.

Таким чином, головний принцип є таким: не люди мають підстроюватся під вибрану модель процесу, а модель процесу повинна підстроюватися під конкретну команду, щоб забезпечити її найвищу продуктивність.

Щоб програмний проект став успішним, необхідно:

1. Чітко ставити цілі.

2. Визначати спосіб досягнення мети.

3. Контролювати і управляти реалізацією.

4. Аналізувати погрози і протидіяти їм.

5. Створювати команду.

6.5 Концепції удосконалення управління IT проектами

Як самостійна області знань управління проектами почало формуватися на початку XX століття. У цій дисципліні поки немає єдиних міжнародних стандартів. Найбільш відомі центри компетенції: PMI, Project

Management Institute, PMBOK - американський національний стандарт ANSI/PMI 99-001-2004. IPMA, International Project Management Association.

Приблизно 50 років тому людство почало жити в новій суспільно-економічній формації, яка називається інформаційне або постіндустріальне суспільство. Це епоха змін, глобалізації і інтелектуального капіталу.

Інновації – невід'ємний атрибут нашого часу. Відома фраза – «Якщо у вас повільний доступ в Інтернет, ви можете назавжди відстати від розвитку інформаційних технологій».

Практика повинна постійно перебудовуватися стосовно нових і нових умов.

Приклад. Hewlett-Packard отримує велику частку прибутку на товарах, які рік назад навіть не існували.

Глобалізація. Загальна взаємозалежність і взаємозв'язок. Транснаціональні компанії. Бізнес йде туди, де дешевше робоча сила. Інтернет. Конкуренція без меж.

Приклад. Google.

За допомогою Інтернету можна вийти на ринок, на якому більше 100 млн. споживачів.

Проста мобілізація засобів і зусиль вже не може забезпечити прогрес. Пригадаємо Ф. Брукса: «Якщо проект не укладається в терміни, то додавання робочої сили затримає його ще більше».

Людству відомі два види діяльності. Репродуктивна діяльність (праця) є зліпком, копією з діяльності іншої людини або копією своїй власній діяльності, освоєній в попередньому досвіді.

Така діяльність, як, наприклад, праця токаря в будь-якому механічному цеху, або рутинна повсякденна діяльність менеджера-управлінця на рівні разів і назавжди засвоєних технологій.

Продуктивна діяльність (творчість) – діяльність, направлена на отримання об'єктивно нового або суб'єктивно нового (для даного працівника)

результату. Репродуктивна діяльність йде в минуле. У постіндустріальному суспільстві інтелект – основна виробнича сила.

Ідею багатства тепер пов'язують не з грошима, а з людьми, не з фінансовим капіталом, а з «людським».

Ринок праці перетворюється на ринок незалежних фахівців і його учасникам все більше відомо про можливі варіанти вибору. Працівники інтелектуальної праці починають самостійно визначати собі ціну.

У будь-якому товарі, зробленому в США, частка зарплати становить 70 %. Але це в середньому за всіма товарами.

Що стосується розробки ПЗ, то майже все, що в цій галузі проводиться, створюється за допомогою інтелекту. Все менший об'єм людської діяльності може бути організований в виді операцій, що повторюються.

Проект – це основа інновацій. Зробити те, до чого інші компанії ще не додумалися, зробити це щонайшвидше, інакше це зроблять інші. Запропонувати споживачеві якісніший продукт або такий продукт потреба в якому споживач навіть не може поки усвідомити.

Критерії успішності проекту. Завдання проекту – досягнення конкретної бізнес-цілі, при дотриманні обмежень «залізного трикутника» (рис.6.3)

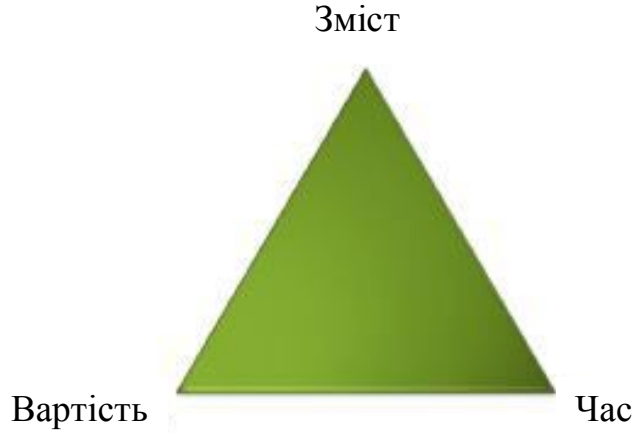


Рисунок 6.3 – Залізний трикутник обмежень проекту

Це означає, що жоден з кутів трикутника не може бути змінений без роботи впливу на інші. Наприклад, щоб зменшити час, потрібно буде збільшити вартість і/або скоротити зміст. Згідно поточної редакції стандарту PMBOK, проект вважається успішним якщо задоволені всі вимоги замовника і учасників проекту.

Отже, проект розробки ПЗ на сьогодні має не три, а чотири чинники успіху:

1. Виконаний у відповідності із специфікаціями.

2. Виконаний в строк.

3. Виконаний в межах бюджету.

4. Кожен учасник команди йде з роботи в 18:00 з відчуттям успіху.

Цей четвертий чинник успіху повинен стати відтворним, якщо підприємство хоче бути ефективним. Для успішного проекту характерно постійне відчуття його учасниками задоволення і гордості за результати своєї роботи, відчуття оптимізму. Немає нічого згубнішого для проекту, чим байдужість або смуток його учасників.

Ефективність – це відношення отриманого результату до проведених витрат. Не можна розглядати ефективність, виходячи тільки з результативності: чим більше працюєш, тим вище твоя ефективність. З таким підходом можна «зарізати на вечерю курку що несе золоті яйця».

Витрати не слід плутати з інвестиціями. Оплата оренда, електроенергії, комунальні платежі – витрати. Створення і закріплення ефективної команди - це стратегічне придбання компанії.

Навчання учасників проекту – інвестиції. Вкладення в людей – це збільшення чисельника у формулі ефективності.

Відхід з компанії всіх професіоналів після проекту, виконаного за принципом «за всяку ціну», – витрати, причому дуже важко заповнювані.

Наростаюча конкуренція указує на абсолютно чіткий тренд в світовій економіці: персонал – це форма інвестицій, активів, які потрібно уміти нарощувати, управляти і зберігати.

Сьогодні люди – це капітал.

Сучасне підприємство зобов'язане відноситися до своїх працівників так само, як до своїм кращих клієнтів.

Головний капітал сучасної компанії – це знання. Велика частина цих знань невід'ємна від їх носія – людини. Ті підприємства, які цього не зрозуміли, не виживуть тому, що не зможуть бути ефективними.

Таким чином, ефективні процеси ініціації програмного проекту багато в чому визначають його майбутню успішність. Недостатня увага цій фазі проекту неминуче приводить до істотних проблем при плануванні реалізації і завершенні. Концепція проекту – це ключовий документ, який використовується для ухвалення рішень в ході всього проекту, а також на фазі приймання – для підтвердження результату.

Таким чином, пріоритет проекту визначається на основі оцінки трьох показників:

- фінансова цінність;
- стратегічна цінність;
- рівень ризиків.

7 МОДЕЛІ АНАЛІЗУ ТА ПРОЕКТУВАННЯ ОБ'ЄКТНО-ОРІЄНТОВАНИХ ПРОГРАМНИХ СИСТЕМ

7.1 Уніфікована мова моделювання UML

Для створення моделей аналізу і проектування програмних систем використовують мови візуального моделювання. З'явившись порівняно недавно, в період з 1989 по 1997 рік, ці мови вже мають славну історію розвитку.

В даний час розрізняють три покоління мов візуального моделювання. І якщо перше покоління склали 10 мов, то чисельність другого покоління вже перевищила 50 мов.

Серед найбільш популярних мов 2-го покоління можна виділити: мову Буча (G. Booch), мову Рамбо (J. Rumbaugh), мову Джекобсон (I. Jacobson), мову Коад-Йордона (Coad-Yourdon), мову Шлеер-Меллора (Shlaer-Mellor) тощо [41], [64], [69]. Кожна мова мала свої виразні засоби, власний синтаксис і семантику. В результаті розробники (і користувачі цих мов) перестали розуміти один одного. Виникла гостра необхідність уніфікації мов.

Ідея уніфікації привела до появи мов 3-го покоління. В якості стандартного мови третього покоління був прийнятий Unified Modeling Language (UML), що створювався в 1994-1997 роках (основні розробники Г. Буч, Дж. Рамбо, І. Джекобсон).

Визначимо базові поняття мови UML.

Визначення 7.1 Уніфікована мова моделювання UML – це стандартна мова для написання моделей аналізу, проектування і реалізації об'єктно-орієнтованих програмних систем. Моделі UML прямо транслюються в текст на мовах програмування (Java, C ++, Visual Basic, Ada 95, Object Pascal) і навіть в таблиці для реляційної БД.

Словник UML утворюють три основні блоки: предмети, відносини, діаграми.

Визначення 7.2. Предмети – це абстракції, які є основними елементами в моделі, відносини пов'язують ці предмети, діаграми групують колекції предметів.

В UML є чотири різновиди предметів:

a) структурні предмети;

b) предмети поведінки;

c) предмети, що групують;

d) пояснюючі предмети.

Ці предмети є базовими об'єктно-орієнтованими будівельними блоками. Вони використовуються для написання моделей.

Структурні предмети є іменниками в UML-моделях. Вони представляють статичні частини моделі – понятійні або фізичні елементи.

Існує кілька різновидів структурних предметів.

1. Клас – опис множини об'єктів, які поділяють однакові властивості, операції, відносини і семантику (сенс). Клас реалізує один або кілька інтерфейсів. Як показано на рис. 7.1, графічно клас відображається у вигляді прямокутника, зазвичай включає секції з ім'ям, властивостями (атрибутами) і операціями.

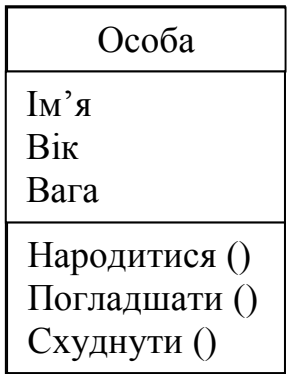
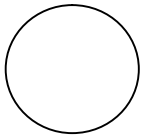


Рисунок 7.1 – Приклад класу

2. Інтерфейс – набір операцій, які визначають послуги класу або компонента. Інтерфейс описує поведінку елемента, яка є видимою ззовні. Інтерфейс може представляти повні послуги класу або компонента або частину таких послуг. Інтерфейс визначає набір специфікацій операцій (їх сигнатури), а не набір реалізацій операцій.

Графічно інтерфейс зображується у вигляді кола з ім'ям, як показано на рисунку 7.2. Ім'я інтерфейсу починається з букви «I». Інтерфейс приєднують до класу або компоненту, який реалізує інтерфейс.



IНавчання

Рисунок 7.2 – Інтерфейс

3. Кооперація (співробітництво) визначає взаємодію і є сукупністю ролей і інших елементів, які працюють разом для забезпечення колективної поведінки більш складного компонента, ніж проста сума всіх елементів.

Таким чином, кооперації мають як структурний, так і поведінковий виміри. Конкретний клас може брати участь в декількох кооперациях. Ці кооперації представляють реалізацію патернів (зразків), які формують систему.

Графічно кооперація зображується як пунктирний еліпс, в який вписується її ім'я (рис. 7.3).

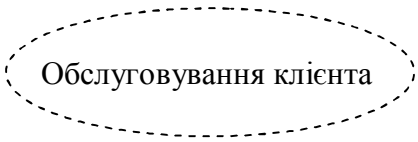


Рисунок 7.3 – Кооперації

4. Актор – набір узгоджених ролей, які можуть грати користувачі при взаємодії з системою (її елементами Use Case). Кожна роль вимагає від системи певної поведінки. Як показано на рисунку 7.4, актор зображується як дротяний чоловічок з ім'ям.

Замовник

Рисунок 7.4 – Актори

5. Елемент Use Case (Прецедент) – опис послідовності дій (або декількох послідовностей), що виконуються системою в інтересах окремого актора і виробляють видимий для актора результат. У моделі елемент Use Case застосовується для структурування предметів поведінки. Елемент Use Case реалізується кооперацією. Як показано на рисунку 7.5, елемент Use Case зображується як еліпс, в який вписується його ім'я.

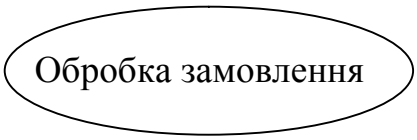


Рисунок 7.5 – Елементи Use Case

6. Активний клас – клас, чиї об'єкти мають один або кілька процесів (або потоків) і тому можуть ініціювати керуючу діяльність. Активний клас схожий на звичайний клас за винятком того, що його об'єкти діють одночасно з об'єктами інших класів. Активний клас зображується як заокруглений прямокутник, включає ім'я, властивості (атрибути) і операції (рис.7.6).

Рисунок 7.6 – Активні класи

7. Компонент – фізична і замінна частина системи, яка відповідає набору інтерфейсів і забезпечує реалізацію цього набору інтерфейсів. У систему входять як компоненти, які є результатами процесу розробки (файли вихідного коду), так і різні різновиди використовуваних компонентів (COM + -компоненти, Java Beans). Компонент - це фізична упаковка різних логічних елементів (класів, інтерфейсів і кооперацій). Як показано на рис.7.7, компонент зображується як прямокутник з вкладками, що зазвичай включає ім'я.

Рисунок 7.7 – Компоненти

8. Вузол – фізичний елемент, який існує в період роботи системи і являє ресурс, зазвичай має пам'ять і можливості обробки. У вузлі розміщується набір компонентів, який може переміщатися від вузла до вузла. Як показано на рис. 7.8, вузол зображується як куб з ім’ям.

Предмети поведінки – динамічні частини UML-моделей. Вони є дієсловами моделей, поданням поведінки в часі і просторі. Існує дві основні різновиди предметів поведінки.

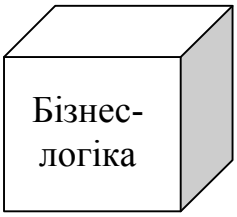


Рисунок 7.8 – Вузли

1. Взаємодія – поведінка, яка є набором повідомлень, якими обмінюється набір об'єктів в конкретному контексті для досягнення певної мети. Взаємодія може визначати динаміку як сукупності об'єктів, так і окремої операції. Елементами взаємодії є повідомлення, послідовність дій (поведінка, що викликається повідомленням) і зв'язку (з'єднання між об'єктами). Повідомлення зображується у вигляді спрямованої лінії з ім'ям її операції (рис.7.9).

Рисунок 7.9 – Повідомлення

2. Скінчений автомат – поведінка, яка визначає послідовність станів об'єкта або взаємодії, що виконуються в ході його існування у відповідь на події (і з урахуванням обов'язків за цими подіями). За допомогою скінченого автомата може визначатися поведінка індивідуального класу або кооперації класів. Елементами скінченого автомата є стани, переходи (від стану до стану), події (предмети, що викликають переходи) і дії (реакції на перехід).
Як показано на рис.7.10, стан зображується як закруглений прямокутник, зазвичай включає його ім'я і його підстани (якщо вони є).

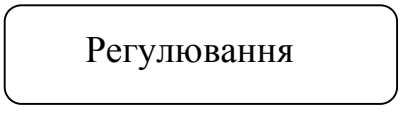


Рисунок 7.10 – Стани

Ці два елементи – взаємодії і скінчені автомати – є базисними предметами поведінки, які можуть включатися в UML-моделі. Семантично ці елементи асоціюються з різними структурними елементами (перш за все з класами, кооперaціями і об’єктами).

Предмети, що групуються – організаційні частини UML-моделей. Це скриньки, за якими може бути розкладена модель. Передбачений один різновид предметів, що групуються – це пакет.

Пакет – загальний механізм для розподілу елементів по групах. У пакет можуть поміщатися структурні предмети, предмети поведінки і навіть інші угруповання предметів. На відміну від компонента (який існує в період виконання), пакет – чисто концептуальне поняття. Це означає, що пакет існує тільки в період розробки. Як показано на, пакет зображується як папка з закладкою, на якій позначено його ім’я і, іноді, його зміст (рис. 7.11).

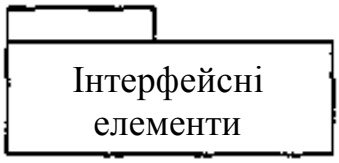


Рисунок 7.11 – Пакети

Пояснюючі предмети – це роз’яснюючі частини UML-моделей. Вони є зауваженнями, які можна застосувати для опису, пояснення і коментування будь-якого елементу моделі.

Передбачений один різновид пояснюючого предмета - примітка.

Примітка – символ для відображення обмежень і зауважень, що приєднуються до елементу або сукупності елементів. Примітка зображується у вигляді прямокутника з загнутим кутом, в який вписується текстовий або графічний коментар (рис. 7.12).

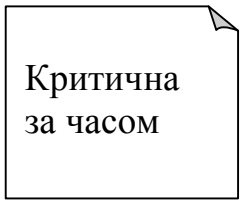


Рисунок 7.12 – Примітки

Відносини в UML

У UML є чотири різновиди відносин:

– залежність;

– асоціація;

– узагальнення;

– реалізація.

Ці відносини є базовими будівельними блоками відносин. Вони використовуються при написанні моделей.

1. Залежність – семантичне відношення між двома предметами, в якому зміна в одному предметі (незалежному предмет) може впливати на семантику іншого предмета (залежного предмета). Як показано на рисунку 7.13, залежність зображується у вигляді пунктирної лінії, можливо спрямованої на незалежний предмет і такої, що іноді має мітку.

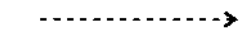
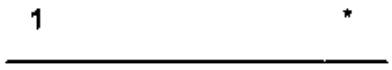


Рисунок 7.13 – Залежності

2. Асоціація – структурне відношення, яке описує набір зв'язків, які є з'єднанням між об'єктами. Агрегація – це спеціальний різновид асоціації, що представляє структурне відношення між цілим і його частинами. Асоціація зображується у вигляді суцільної лінії, можливо спрямованої, іноді має мітку і часто включає інші «прикраси», такі як потужність і імена ролей (рис. 7.14).



Замовник Замовлення

Рисунок 7.14 – Асоціації

3. Узагальнення – відношення спеціалізації / узагальнення, в якому об'єкти спеціалізованого елемента (нащадка, дитини) можуть замінювати об'єкти узагальненого елемента (предка, батька). Інакше кажучи, нащадок розділяє структуру і поведінку батьків. Як показано на рисунку 7.15, узагальнення зображується у вигляді суцільної стрілки з порожнистим наконечником, що вказує на батька.

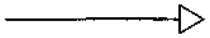


Рисунок 7.15 – Узагальнення

4. Реалізація – семантичне відношення між класифікаторами, де один класифікатор визначає контракт, який інший класифікатор зобов'язується виконувати (до класифікаторів відносять класи, інтерфейси, компоненти, елементи Use Case, кооперації). Відносини реалізації застосовують в двох випадках: між інтерфейсами і класами (або компонентами), що реалізують їх; між елементами Use Case і кооперації, які реалізують їх. Як показано на рисунку 7.16, реалізація зображується як щось середнє між узагальненням і залежністю.

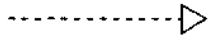


Рисунок 7.16 – Реалізації

7.2 Діаграми в UML

Діаграма – графічне представлення множини елементів, найбільш часто зображується як зв'язний граф з вершин (предметів) і дуг (відносин).

Діаграми розробляються для візуалізації системи з різних точок зору, потім їх буде запропоновано в систему. Теоретично діаграма може містити будь-яку комбінацію предметів і відносин, на практиці обмежуються малою кількістю комбінацій, які відповідають п’яти представленням архітектури ПС. З цієї причини UML включає дев'ять видів діаграм:

1) діаграми класів;
2) діаграми об'єктів;
3) діаграми Use Case (діаграми прецедентів);
4) діаграми послідовності;
5) діаграми співробітництва (кооперації);
6) діаграми схем станів;
7) діаграми діяльності;
8) компонентні діаграми;
9) діаграми розміщення (розгортання).

Діаграма класів показує набір класів, інтерфейсів, кооперацій і їх відносин. При моделюванні об'єктно-орієнтованих систем діаграми класів використовуються найбільш часто. Діаграма класів забезпечують статичне проектне подання системи. Діаграми класів, що включають активні класи, забезпечують зріз процесів системи.

Діаграма об'єктів показує набір об’єктів і їх відносини. Діаграма об'єктів представляє статичний «моментальний знімок» з примірників предметів, які знаходяться в діаграмах класів. Як і діаграми класів, ці діаграми забезпечують статичне проектне подання або зріз процесів системи (але з точки зору реальних або фототипічних випадків).

Діаграма Use Case (діаграма прецедентів) показує набір елементів Use Case, акторів і їх відносин. За допомогою діаграм Use Case для системи

створюється зріз Use Case. Ці діаграми особливо важливі при організації і моделюванні поведінки системи, завданні вимог замовника до системи.

Діаграми послідовності і діаграми співпраці – це різновиди діаграм взаємодії.

Діаграма взаємодії показує взаємодію, що включає набір об'єктів і їх відносин, а також повідомлення, що пересилаються між об'єктами. Діаграми взаємодії забезпечують динамічне представлення системи.

Діаграма послідовності – це діаграма взаємодії, яка виділяє впорядкування повідомлень за часом.

Діаграма співпраці (діаграма кооперації) – це діаграма взаємодії, яка виділяє структурну організацію об'єктів, що посилають і приймають повідомлення. Діаграми послідовності і діаграми співпраці ізоморфні, що означає, що одну діаграму можна трансформувати в іншу діаграму.

Діаграма схем станів показує скінчений автомат, являє стани, переходи, події і дії. Діаграми схем станів забезпечують динамічне представлення системи. Вони особливо важливі при моделюванні поведінки інтерфейсу, класу або співпраці. Ці діаграми виділяють таку поведінку об'єкта, яким керує подіями, що особливо корисно при моделюванні реактивних систем.

Діаграма діяльності – спеціальний різновид діаграми схем станів, яка показує потік від дії до дії всередині системи. Діаграми діяльності забезпечують динамічне представлення системи. Вони особливо важливі при моделюванні функціональності системи і виділяють потік управління між об'єктами.

Компонентна діаграма показує організацію набору компонентів і залежності між компонентами. Компонентні діаграми забезпечують зріз реалізації системи. Вони пов'язані з діаграмами класів в тому сенсі, що в компонент зазвичай відображається один або кілька класів, інтерфейсів або кооперацій.

Діаграма розміщення (діаграма розгортання) показує конфігурацію обробних вузлів періоду виконання, а також компоненти, що живуть в них.

Діаграми розміщення забезпечують зріз розміщення системи. Вони пов'язані з компонентними діаграмами в тому сенсі, що вузол включає один або кілька компонентів.

7.3 Механізми розширення в UML

UML – розвинена мова, що має великі можливості, але навіть він не може відобразити всі нюанси, які можуть виникнути при створенні різних моделей. Тому UML створювався як відкрита мова, що допускає контрольовані розширення.

Механізмами розширення в UML є:

- обмеження;
- тегові величини;
- стереотипи.

Обмеження (constraint) розширюють семантику UML-блоку, дозволяючи додати нові правила або модифікувати існуючі. Обмеження показують як текстовий рядок, укладений в фігурні дужки {}. Наприклад, на рисунку 7.17 введено просте обмеження на властивість сума класу Сесія Банкомату – його значення має бути кратне 20.

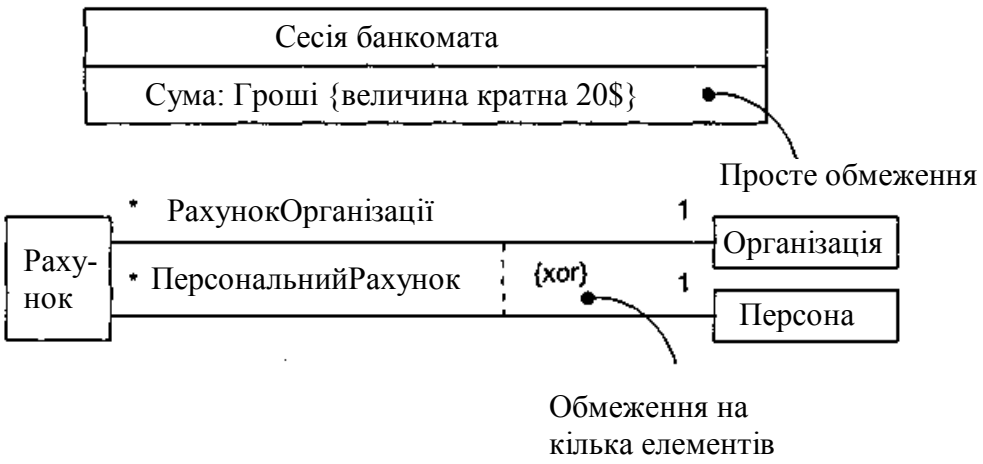


Рисунок 7.1 – Обмеження

Крім того, показано обмеження на два елементи (дві асоціації), воно розташовується біля пунктирної лінії, що з'єднує елементи, і має наступний сенс – власником конкретного рахунку е може бути і організація, і персона одночасно.

Тегова величина (tagged value) розширює характеристики UML-блоку, дозволяючи створити нову інформацію в специфікації конкретного елемента.

Тегові величини показують як рядок в фігурних дужках {}. Рядок має вигляд

ім'я тегової величини = значення.

Іноді (в разі зумовлених тегів) вказується тільки ім'я тегової величини. Відзначимо, що при роботі з продуктом, який має багато реалізацій, корисно відстежувати версію і автора певних блоків. Версія і автор не належать до основних понять UML. Вони можуть бути додані до будь-якого будівельного блоку (наприклад, до класу) введенням в блок нових тегових величин.

Наприклад, на рис. 7.18 клас ТекстовийПроцессор розширено шляхом явної вказівки його версії і автора.

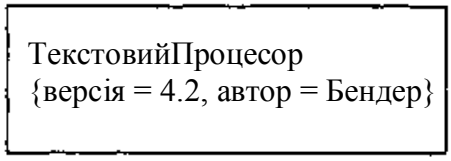


Рисунок 7.18 – Розширення класу

Стереотип (stereotype) розширює словник мови, дозволяє створювати нові види будівельних блоків, похідні від існуючих і враховує специфіку нової проблеми. Елемент зі стереотипом є варіацією існуючого елемента, що має таку ж форму, але відрізняється по суті. У нього можуть бути додаткові обмеження і тегові величини, а також інше візуальне подання. Він інакше

обробляється при генерації програмного коду. Відображають стереотип як ім'я, визначене в подвійних кутових дужках (або в кутових лапках).

Приклади елементів зі стереотипами наведені на рисунку 7.19. Стереотип exception говорить про те, що клас ВтратаЗначущості тепер розглядається як спеціальний клас, якому, між іншим, дозволяється тільки генерація і обробка сигналів винятків. Особливі можливості метакласу отримав клас ЕлементМоделі. Крім того, тут показано застосування стереотипу «call» до відношенню залежності (у нього з'явився новий зміст).

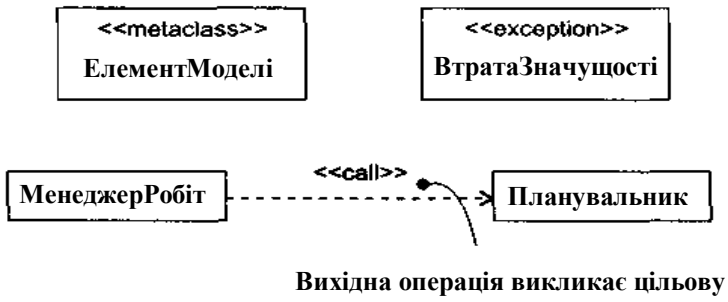


Рисунок 7.19 – Стереотипи

Таким чином, механізми розширення дозволяють адаптувати UML під потреби конкретних проектів і під нові програмні технології. Можливе додавання нових блоків, модифікація специфікацій існуючих блоків і навіть зміна їх семантики.

8. ЗАХИСТ ІНФОРМАЦІЇ В РОЗПОДІЛЕНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

8.1 Основи комплексних систем захисту інформації

Визначення 8.1. Захист інформації в ІС (information protection, information security, computer system security) – діяльність, яка спрямована на гарантування безпеки інформації в ІС і дає змогу запобігти загрозам або ускладнити можливість їх реалізації, а ще зменшити величину потенційних збитків внаслідок реалізації загроз.

Визначення 8.2. Безпека інформації (information security) – стан інформації, в якому забезпечується збереження визначених політикою безпеки властивостей інформації.

Властивостями інформації є конфіденційність, цілісність та доступність.

Конфіденційність інформації (information confidentiality) – властивість інформації, яка полягає в тому, що інформація не може бути отримана неавторизованим користувачем і/або процесом.

Цілісність інформації (information integrity) – властивість інформації, яка полягає в тому, що інформація не може бути видозмінена неавторизованим користувачем і/або процесом.

Доступність (availability) – властивість ресурсу системи (інформації), яка полягає в тому, що користувач і/або процес, який має відповідні повноваження, може використовувати ресурс відповідно до правил, установлених політикою безпеки, не очікуючи довше заданого (малого) проміжку часу, тобто коли цей ресурс має вигляд, прийнятний для користувача, в місці, яке йому підходить, і в той час, коли він йому потрібен.

Визначення 8.3. Об'єкт захисту – це такий структурний складник системи, у якому розміщується або може розміщуватися інформація, яка

підлягає захисту.

Згідно з об’єктом комп’ютерної системи (product object, system object) є елемент ресурсу комп’ютерної системи, що перебуває під керуванням КЗЗ і характеризується певними атрибутами і поведінкою. Крім того, пасивний об’єкт (passive object) – це об’єкт комп’ютерної системи, який в конкретному акті доступу виступає як пасивний компонент системи, з яким виконується дія і/або який служить джерелом чи приймальником інформації.

Комплексна система захисту інформації; КСЗІ – це сукупність організаційних і інженерних заходів, програмно-апаратних засобів, які забезпечують захист інформації в ІС.

Політика безпеки інформації (information security policy) – це сукупність законів, правил, обмежень, рекомендацій, інструкцій тощо, які регламентують порядок обробки інформації.

Усі засоби захисту і керування доступом об'єднують у комплекс засобів захисту – КЗЗ (trusted computing base; TCB), який є сукупністю програмно-апаратних засобів, що забезпечують реалізацію політики безпеки інформації

У сучасних умовах спостерігається цілеспрямований всебічний вплив на інформаційні ресурси, тому в складі АС треба передбачити комплексну систему захисту інформації, яка має містити:

– структурні органи (із визначеною ієрархією), що здійснюють розробку нормативних і керівних документів щодо забезпечення захисту інформації і контроль їхнього виконання;

– сукупність різних методів (фізичних, організаційних, криптографічних, і ін.) і засобів (програмних, апаратних, апаратно-програмних), що здійснюють повний захист апаратного і програмного забезпечення інформаційних систем, а також безпеку і контроль власне систем захисту.

Питання організації захисту інформації слід вирішувати вже на проектній стадії розробки.

8.2 Вибірковий і обов’язковий підходи до убезпечення даних

У сучасних СКБД застосовується один із двох відомих підходів до убезпечення даних, а саме вибірковий підхід (discretionary access control) або обов'язковий підхід (mandatory access control). В обох випадках одиницею даних або „об'єктом даних”, для яких має бути створена система безпеки, може бути як уся база даних цілком або якийсь набір відношень, так і певне значення даних для заданого атрибута усередині деякого кортежу у визначеному відношенні. Ці підходи відрізняються такими властивостями:

У випадку вибіркового керування користувач має різні права (привілеї або повноваження) при роботі з різними об'єктами. Мало того, різні користувачі звичайно мають і різні права доступу до одного об'єкта. Тому вибіркові схеми характеризуються значною гнучкістю.

У випадку обов'язкового керування, навпаки, кожному об'єктові даних надається певний класифікаційний рівень, а кожен користувач має власний рівень допуску. Отже, за такого підходу доступ до визначеного об'єкта даних мають лише користувачі з відповідним рівнем допуску. Тому обов'язкові схеми не гнучкі й статичні.

Усі рішення щодо допуску користувачів до виконання тих чи інших операцій приймаються на стратегічному, а не технічному рівні. Тож, СКБД тільки реалізує уже прийняті раніше рішення. З огляду на це, можна відзначити наступне:

1. Стратегічні рішення мають бути відомі системі (тобто виконані на основі тверджень, заданих за допомогою деякої мови програмування) і зберігатися в ній у вигляді правил безпеки, що також називаються повноваженнями.

2. Очевидно, мають бути певні засоби регулювання запитів доступу стосовно відповідних правил безпеки. (Тут під «запит доступу» – це комбінація операції, що запитується, об’єкта, що запитується і користувача, що запитує). Така перевірка виконується підсистемою безпеки СКБД, що

також називається підсистемою повноважень.

3. Для того щоб розібратися, які правила безпеки до яких запитів доступу застосовуються, у системі повинні бути передбачені способи упізнання джерела цього запиту, тобто упізнання користувача, що запитує. Тому в момент входу в систему користувачеві звичайно потрібно ввести не тільки його ідентифікатор (наприклад, ім’я або посаду), але ще і пароль (щоб підтвердити свої права на заявлені раніше ідентифікаційні дані). Звичайно передбачається, що пароль відомий лише системі і певним особам з особливими правами.

Щодо останнього пункту варто зауважити, що різні користувачі можуть мають один ідентифікатор певної групи. Таким чином, у системі можуть підтримуватися групи користувачів і забезпечуватися однакові права доступу для користувачів однієї групи, наприклад, для усіх осіб із розрахункового відділу. Крім того, операції додавання окремих користувачів до групи або видалення користувачів з групи можуть виконуватися незалежно від операції надання повноважень цій групі.

Методи обов'язкового керування доступом застосовуються до баз даних, у яких дані мають досить статичну і не гнучку структуру, властиву, наприклад, урядовим або військовим організаціям. Як уже відзначалося, основна ідея полягає в тому, що кожний об’єкт має певний *рівень* класифікації, наприклад: таємно, абсолютно таємно, для службового користування і т.д., а кожний користувач має *рівень допуску* з тими самими градаціями, що й рівні класифікації. Ці рівні утворюють суворий ієрархічний порядок, наприклад: абсолютно таємно > таємно > для службового користування і так далі. Тоді на основі цих відомостей можна сформулювати два дуже простих правила безпеки.

1. Користувач і має доступ до об’єкта j, тільки якщо його рівень допуску більше або дорівнює рівню класифікації об’єкта j.

2. Користувач і може модифікувати об’єкт j, тільки якщо його рівень допуску дорівнює рівню класифікації об’єкта j.

Правило 1 досить очевидне, а правило 2 вимагає додаткових роз'яснень. Насамперед, слід зазначити, що по-іншому правило 2 можна сформулювати так: будь-яка інформація, записана користувачем *i*, автоматично набуває рівня класифікації *i*. Таке правило потрібно, наприклад, для того, щоб запобігти записові таємних даних, який виконується користувачем із рівнем допуску «таємно», у файл із меншим рівнем класифікації, що порушує всю систему таємності.

Останнім часом методи обов'язкового керування доступом набули широкого поширення. Ці методи насправді є частиною більш загальної класифікації рівнів безпеки, що будуть викладені тут у дуже короткій формі. Насамперед, визначаються чотири класи безпеки (security classes) – D, C, B і A. Серед них клас D найменш безпечний, клас C – безпечніший, ніж клас D, і т.д. Отже, клас D забезпечує найменший захист, клас C – вибірковий, клас B – обов'язковий, а клас A – перевірений захист.

Вибірковий захист. Клас C ділиться на два підкласи – C1 і C2 (де підклас C1 менш безпечний, ніж C2), які підтримують вибіркове керування доступом, тобто керування доступом здійснюється власником даних.

Відповідно до вимог класу C1 слід застосувати роздільне використання даних користувачами.

Відповідно до вимог класу C2 слід додатково організувати облік на основі процедур входу в систему, аудиту та ізоляції ресурсів.

Обов'язковий захист. Клас B містить вимоги до методів обов'язкового керування доступом і ділиться на три підкласи – B1, B2 і B3 (де B1 є найменш, а B3 – найбільш безпечним підкласом).

Відповідно до вимог класу B1 необхідно забезпечити «відзначений захист» (це означає, що кожний об'єкт даних повинен містити оцінку про його рівень класифікації, наприклад: таємно, для службового користування і т.д.), а також неформальне повідомлення про діючу стратегію безпеки.

Відповідно до вимог класу B2 необхідно додатково забезпечити *формальне* твердження про діючу стратегію безпеки, а також виявити і

виключити *погано захищені канали передачі інформації.*

Відповідно до вимог класу В3 необхідно додатково забезпечити підтримку аудиту і відновлення даних, а також призначення *адміністратора режиму безпеки.*

Перевірений захист. Клас А є найбезпечнішим і відповідно до його вимог потрібен математичний доказ того, що даний метод забезпечення безпеки сумісний і адекватний заданій стратегії безпеки.

Введемо наступні означення.

Повноваження (privilege) – право користувача або процесу на виконання певних дій, зокрема на одержання певного типу доступу до об’єктів.

Правила розмежування доступу (ПРД) (access mediation rules) – частина політики безпеки, що регламентує правила доступу користувачів і процесів до пасивних об’єктів;

Санкціонований доступ до інформації (authorized access to information) – це доступ до інформації, що не порушує ПРД;

Несанкціонований доступ до інформації (unauthorized access to information) – це доступ до інформації, який здійснюється з порушенням ПРД.

8.3 Основні аспекти інформаційної безпеки

Серед численних аспектів проблеми безпеки інформації слід відзначити такі:

– правові, суспільні та етичні аспекти (чи має право певна особа дістати інформацію, що запитується ,наприклад, про кредит клієнта);

– фізичні умови (наприклад, чи закриті або захищені якось інакше даний комп'ютер і термінальна кімната);

– організаційні питання (наприклад, як у межах підприємства, яке експлуатує інформаційну систему, організовано доступ до даних);

– питання реалізації керування (наприклад, якщо використовується пароль як метод доступу, чи часто змінюються паролі);

– апаратне забезпечення (чи забезпечуються заходи безпеки на апаратному рівні, наприклад, за допомогою захисних ключів або привілейованого режиму керування);

– безпека операційної системи (наприклад, чи затирає базова операційна система зміст структури збереження і файлів із даними у разі припиненні роботи з ними).

Файли і бази даних як інформаційні об'єкти захисту. Інформаційні об'єкти ІС – це будь-яка інформація (повідомлення, відомості, файли бази даних тощо) у будь-яких формах її подання (аналогова, цифрова, віртуальна, уявна та ін.)

Коли розглядають процедури захисту мережних баз даних, то дані та їх логічні структури представляють двома способами. Окремі об'єкти даних самі можуть бути об'єктами захисту, але можуть бути організовані в структури бази даних (сегменти, відношення, каталоги і т.п.).

Колективне використання файлів потребує організації їх захисту від несанкціонованого використання, а також від фізичної руйнації.

Проблема ще ускладнюється тим, що користувачі можуть давати свої файли іншим користувачам. Таким чином, усі файли, що потребують захисту, можна умовно класифікувати як: загальні, групові, особисті.

Для забезпечення цілісності файлів можна використати апаратні й програмні засоби захисту, а також сукупність організаційних заходів, що уможливлюють проведення обліку, збереження й використання файлів.

Організація збереження й використання інформації в базах даних має специфічні особливості. Якщо до інформації, що міститься в базах даних, звертається багато користувачів, то надто важливо, щоб елементи даних і з в'язки між ними не руйнувалися. Потрібно ще враховувати можливість виникнення помилок і різного роду випадкових збоїв. Збереження, відновлення і процедури включення даних мають бути такими, щоб система

у разі виникнення збоїв могла відновлювати дані без утрат.

Захист баз даних означає захист власне даних і їх контрольоване використання на робочих місцях мережі, а також захист будь-якої супутньої інформації, що генерується з цих даних. Керування даними при організації захисту інформаційних баз, що застосовують різні механізми захисту і криптографічні ключі в якості даних, ввійшли в процедури захисту об'єктів АС. У перебігу передачі між вузлами мережі дані захищають за допомогою процедур захисту ліній зв'язку.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

Базова:

1. Боггс У. UML и Rational Rose / У. Боггс, М. Боггс. – М. : ЛОРИ, 2000.– 582 с.

2. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++ / Г. Буч. – М. : Бином, 1999. – 560 с.

3. Буч Г. Язык UML. Руководство пользователя / Г. Буч, Дж. Рамбо, А. Джекобсон. – М. : ДМК, 2000. – 432 с.

4. Калбертсон Р. Быстрое тестирование / Р. Калбертсон, К. Браун, Г. Кобб. – СПб. : Вильямс, 2002. – 374 с.

5. Канер С. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений / С. Канер. – Киев: ДиаСофт, 2001. – 544 с.

6. Ларман К. Применение UML и шаблонов проектирования / К. Ларман – М. : Вильямс, 2001. – 496 с.

7. Леоненков А. В. Самоучитель UML / А. В. Леоненков. – [2-ге вид., перероб. і доп.]. – СПб. : БХВ-Петербург, 2004. – 432 с.

8. Леффингуэлл Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход / Д. Леффингуэлл, Д. Уидриг. – М. : Вильямс, 2002. – 448 с.

9. Маклаков С.В. BPWin и ERWin. CASE – средства разработки информационных систем / С.В. Маклаков. – М.: Диалог-МИФИ, 1999. – 256 с.

10. Мандел Т. Разработка пользовательского интерфейса / Т. Мандел [Пер. с англ.]. – М. : ДМК Пресс, 2001. – 416с.

11. Розенберг Д. Применение объектного моделирования с использованием UML и анализ прецедентов / Д. Розенберг, К. Скотт. – М. : ДМК Пресс, 2002. – 160 с.

12. Торрес Р. Дж. Практическое руководство по проектированию и разработке пользовательского интерфейса / Р. Дж. Торрес [Пер. с англ.]. – СПб. : Вильямс, 2002. – 390 с.

13. Титенко С. В. Моделювання спеціалізованих інформаційних об’єктів в універсальних системах керування Web-контентом / С. В. Титенко // Вісник Східноукраїнського національного університету імені Володимира Даля – 2012. – № 8 (179). – Ч.2. – С. 235–239.

14. Фаулер М. UML. Основы / М. Фаулер, К. Скотт. – СПб. : Символ-Плюс, 2002. – 192 с.

15. Шаллоуей А. Шаблоны проектирования. Новый подход к объектно-ориентированному анализу и проектированию / А. Шаллоуей, Дж.Р. Тротт. – М. : Вильямс, 2002. – 288 с.

16. Якобсон А. Унифицированный процесс разработки программного обеспечения / А. Якобсон, Г. Буч, Дж. Рамбо. – СПб. : Питер, 2002. – 496 с.

17. Гамма Э., Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб. : Питер, 2001. – 368 с.

18. Грехем И. Объектно-ориентированные методы. Принципы и практика / И. Грехем. – М. :Вильямс, 2004. – 880 с.

19. Коналлен Дж. Разработка Web-приложений с использованием UML / Дж. Коналлен [Пер. с англ.]. – М. : Вильямс, 2001. – 288 с.

20. Кьоу Дж. Объектно-ориентированное программирование / Дж. Кьоу, М. Джеанини. – СПб. : Питер, 2005. – 238 с.

21. Нейбург Э. Дж. Проектирование баз данных с помощью UML / Э. Дж. Нейбург, Р. А. Максимчук. – М. : Вильямс, 2002. – 288 с.

21. Санблэд С. Разработка масштабируемых приложений для Microsoft Windows. Мастер-класс / С. Санблэд. – М. : ИТД «Русская редакция», 2002. – 416 с.

Навчальне видання

ЧУБ Ігор Андрійович,
НОВОЖИЛОВА Марина Володимирівна,
ПАН Микола Павлович

ІННОВАЦІЙНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

КОНСПЕКТ ЛЕКЦІЙ

(для студентів денної і заочної форм навчання освітнього рівня «магістр»
за спеціальністю 122 – Комп'ютерні науки,
освітньо-професійна програма «Комп'ютерні науки. Управління проектами»)

Відповідальний за випуск *М. В. Новожилова*

За авторською редакцією

Комп'ютерне верстання *М. В. Новожилова*

План 2018 р., поз. 183 Л
Підп. до друку 10.12.2018. Формат 60 х 84 1/16.
Надруковано на ризографі. Умов. друк. арк. 4,3
Тираж 50 пр. Зам. №

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