МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

ТЕРЕЩЕНКО Т. М.

МОДЕЛІ, ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ ТА УПРАВЛІННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Конспект лекцій

Одеса
Одеський державний екологічний університет
2017

УДК 681.518
Т35

Рекомендовано методичною радою Одеського державного екологічного університету Міністерства освіти і науки України як конспект лекцій (протокол №9 від 29.06. 2017 р.)

Терещенко Т.М.
Моделі, технології проектування та управління інформаційних систем: конспект лекцій. Одеса, Одеський державний екологічний університет, 2017. 70 с.

В конспекті лекцій викладено питання щодо використання складних моделей в процесах проектування та управління інформаційних систем. Розглянуті в конспекті теоретичні матеріали дозволяють отримати знання в галузі моделювання бізнес-процесів та використовувати їх для побудови сучасних інформаційних систем. Конспект лекцій орієнтований на студентів, які навчаються на програмі підготовки магістрів за спеціальністю "Комп'ютерні науки" освітньо-професійної програми "Інформаційні управляючі системи та технології"

ISBN 978-966-186-097-0

ЗМІСТ

ВСТУП 5
1 СТРУКТУРНІ МЕТОДОЛОГІЇ ІС 7
1.1 Основні поняття та методологія моделювання 7
1.2 Методи управління вимогами 14
1.3 Методи моделювання потоків даних 17
2 ОБ'ЄКТНО-ОРІЄНТОВАНА МЕТОДОЛОГІЯ ПРОЕКТУВАННЯ ІС 23
2.1 Концептуальна модель UML 23
2.2 Моделювання стану інформаційних систем 29
2.3 Моделювання видів діяльності інформаційних систем 33
3 ІНСТРУМЕНТАЛЬНІ ЗАСОБИ ПРОЕКТУВАННЯ. ТЕХНОЛОГІЯ SAP 37
3.1 Технологія впровадження CASE-засобів 37
3.2 Технологія SAP 49
4 УПРАВЛІННЯ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ 56
4.1 Поняття інфраструктури IT-підприємства 56
4.2 Управління інформаційних систем 59
4.3 Бізнес-орієнтоване управління системами 60
4.4 Ефективність функціонування інформаційної інфраструктури 62
4.5 Безпека функціонування інформаційної інфраструктури 67
ПЕРЕЛІК ПОСИЛАНЬ 69

ВСТУП

Дисципліна МОДЕЛІ, ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ ТА УПРАВЛІННЯ ІС» є обов’язковою дисципліною магістерської підготовки за спеціальністю "Комп’ютерні науки та інформаційні технології" та належить до циклу професійної та практичної підготовки.

МЕТОЮ дисципліни МОДЕЛІ, ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ ТА УПРАВЛІННЯ ІС є забезпечення профілюючої підготовки за фахом, формування теоретичних знань та практичних навичок у галузі використання складних моделей в процесах проектування та управління інформаційних систем.

Вивчення навчальної дисципліни дозволяє студентам оволодіти знаннями в галузі проектування та управління ІС: уміло використати теорію та практику проектування та управління ІС в процесах побудови сучасних інформаційних систем за результатами моделювання бізнес-процесів.

Після вивчення дисципліни студенти повинні бути здатними до проектної діяльності в професійній сфері, вміти будувати і використовувати моделі для опису об’єктів та процесів, здійснювати їх якісний аналіз, застосовувати їх під час розробки та інтеграції систем, продуктів і сервісів інформації.

Засвоєння цього курсу можливе, якщо студенти отримали необхідні знання з дисциплін "Вища математика", "Дискретна математика", "Теорія ймовірності, ймовірнісні процеси та математична статистика", "Системний аналіз".

Загальний обсяг навчального часу, що припадає на вивчення дисципліни, визначається робочим навчальним планом.

За результатами вивчення дисципліни студент повинен ЗНАТИ:

- технології моделювання та проектування SSADM, SADT, IDEF0;
- методи виявлення та управління вимогами (стандарти JAD, RAD);
- структура моделі потоків даних, планування розробки системи;
- правила побудови діаграми послідовностей, розробка моделей стану, правила побудови діаграм кооперацій;
- критерії вибору CASE-засобів, функціональні можливості сучасних CASE-засобів;
- призначення та основи SAP, інструментальні засоби в SAP Query.

- структуру функціональних об’єктів ІС, основи управління ІТ-ресурсами, інструментальні засоби управління ІТ-інфраструктурою;
- особливості оцінки ефективності ІТ-інфраструктури, основи забезпечення інформаційної безпеки ІТ.

За результатами вивчення дисципліни студент повинен ВМІТИ:

- проводити збір даних та аналізувати предметну область при проектування інформаційної системи;
- будувати структурні діаграми, використовувати діаграми для проектування інформаційних систем, обирати методи виявлення та управління вимогами;
- будувати структуру моделі потоків даних, побудовати концептуальні моделі для заданої предметної області, складати план розробки інформаційної системи;
- будувати діаграми послідовностей та використовувати їх в процесах проектування ІС, розробляти моделі стану та використовувати їх в процесах аналізу функціонування ІС, будувати діаграми кооперацій;
- обирати CASE-засоби з урахуванням реальних виробничих умов;
- використовувати SAP-засоби в процесах управління ІС;
- визначати умови для забезпечення якості ІТ-сервісу;
- будувати структуру функціональних об’єктів ІС, використовувати сучастні інс рументальні засоби для управління ІТ-ресурсами;
- планувати та виконувати оцінку ефективності ІТ-інфраструктури;
- розробляти заходи забезпечення інформаційної безпеки ІТ.

1 СТРУКТУРНІ МЕТОДОЛОГІЇ ІС

1.1 Основні поняття та методологія моделювання

У структурному аналізі і проектуванні використовуються різні моделі, що описують функціональну структуру системи. Функціональна структура системи – це ієрархія діаграм потоків даних, що описують асинхронний процес перетворення інформації від її введення в систему до видачі споживачеві. На практиці будь-який клас систем успішно моделюється за допомогою DFD-орієнтованих методів. Вони із самого початку створювалися як засіб проектування інформаційних систем, тоді як SADT – як засіб моделювання систем взагалі, і мають багатший набір елементів, що адекватно відображають специфіку таких систем.

З іншого боку, ці різновиди засобів структурного аналізу приблизно однакові з погляду функціональних можливостей засобів моделювання. При цьому одним з основних критеріїв вибору того чи іншого методу є ступінь володіння ним з боку консультанта або аналітика.

Найбільш поширеним засобом моделювання даних є модель "сутність - зв'язок" (Entity-Relationship Model – ERM). Вона вперше була введена П. Ченом у 1976 р. Ця модель традиційно використовується у структурному аналізі і проектуванні, проте, по суті, це підмножина об'єктної моделі предметної області. Один з різновидів моделі "сутність – зв'язок" використовується в методі IDEF1-X, що належить сімейству стандартів IDEF, і реалізується у низці поширених CASE-засобів (зокрема, AH Fusion ERwin Data Modeler).

Концептуальною основою об'єктно-орієнтованого аналізу і проектування ПЗ (ООАП) є об'єктна модель. Її основні принципи (абстрагування, інкапсуляція, модульність та ієрархія) і поняття (об'єкт, клас, атрибут, операція, інтерфейс тощо) найчіткіше сформульовані Г. Бучем у його фундаментальних працях.

Більшість сучасних методів ООАП базуються на використанні мови UML. Уніфікована мова моделювання UML (Unified Modeling Language) є мовою для визначення, подання, проектування і документування програмних систем, організаційно-економічних систем, технічних систем та інших систем різної природи. UML містить стандартний набір діаграм і нотацій найрізноманітніших видів.

UML – це наступник того покоління методів ООАП, які з'явилися в кінці 1980-х і на початку 1990-х років. Створення UML фактично розпочалося в кінці 1994 р., коли Граді Вуч і Джеймс Рамбо почали роботу щодо об'єднання їх методів Booch і OMT (Object Modeling Technique) під егідою компанії Rational Software. До кінця 1995 р. вони створили першу специфікацію об'єднаного методу, який було названо Unified Method. Тоді ж у 1995 р. до них приєднався автор методу OOSE (Object Oriented Software Engineering) Івар Якобсон. Таким чином, UML є прямим об'єднанням і уніфікацією методів Г. Буча, Д. Рамбо і Г. Якобсона, проте доповнює їх новими можливостями.

Головними при розробці UML були такі цілі:

- надати користувачам готову до використання виразну мову візуального моделювання, що дозволяє їм розробляти осмислені моделі й обмінюватися ними;
- передбачити механізми розширюваності і спеціалізації для розширення базових концепцій;
- забезпечити незалежність від конкретних мов програмування і процесів розробки;
- забезпечити формальну основу для розуміння цієї мови моделювання (мова має бути одночасно точною і доступною для розуміння, без зайвого формалізму);
- стимулювати зростання ринку об'єктно орієнтованих інструментальних засобів;
- інтегрувати кращий практичний досвід.

UML прийнятий на озброєння практично всіма найбільшими компаніями - виробниками ПЗ (Microsoft, Oracle, IBM, Hewlett-Packard, Sybase тощо). Крім того, практично всі світові виробники CASE-засобів, крім IBM Rational Software, підтримують UML у своїх продуктах (Oracle Designer, Together Control Center (Borland), AllFusion Component Modeler (Computer Associates), Microsoft Visual Modeler).

Стандарт UML версії 1.1, прийнятий OMG у 1997 р., містить такий набір діаграм (структурні моделі (structural)):

- діаграми класів (class diagrams) – для моделювання статичної структури класів системи і зв'язків між ними;
- діаграми компонентів (component diagrams) – для моделювання ієрархії компонентів (підсистем) системи;
- послідовність виконуваних дій;

– передачу інформації між функціональними процесами;

– відношення між даними.

Поширеними моделями проектування програмного забезпечення є:

1) Функціональна модель SADT (Structured Analysis and Design Technique);

2) Модель IDEF3;

3) DFD (Data Flow Diagrams) – діаграми потоків даних.

Метод SADT є сукупністю правил і процедур, призначених для побудови функціональної моделі об'єкта певної предметної області. Функціональна модель SADT відображає функціональну структуру об'єкта, тобто його дії і зв'язки між цими діями. Метод SADT розроблений Дугласом Россом у 1969 р. для моделювання штучних систем середньої складності. Цей метод успішно використовувався у військових, промислових і комерційних організаціях США для вирішення широкого кола завдань, таких як довгострокове і стратегічне планування, автоматизоване виробництво і проектування, розробка ПЗ для оборонних систем, управління фінансами і матеріально-технічним постачанням тощо. Метод SADT підтримується Міністерством оборони США, яке було ініціатором розробки сімейства стандартів IDEF (Icam DEFinition), які є основною частиною програми ICAM (інтегрована комп'ютеризація виробництва), що проводиться за ініціативою ВВС США.

DEF-0 – це методологія функціонального моделювання. За допомогою наочної графічної мови система представляється у вигляді набору взаємопов'язаних функцій. IDEF-1 - методологія моделювання інформаційних потоків, що дозволяє відображати та аналізувати їх структуру і взаємозв'язки. IDEF-їх - методологія побудови реляційних структур. IDEF-2 - методологія динамічного моделювання розвитку систем. IDEF-3 - методологія документування процесів, що відбуваються в системі і використовуються, наприклад, при дослідженні технологічних процесів.

Метод SADT реалізовано саме в одному зі стандартів цього сімейства – IDEF-0, який був затверджений як федеральний стандарт США в 1993 р.

Моделі SADT (IDEF0) традиційно використовуються для моделювання організаційних систем (бізнес-процесів). Слід зазначити, що метод SADT успішно функціонує тільки при описі добре специфікованих і стандартизованих бізнес-процесів у зарубіжних корпораціях, тому він і прийнятий у США як типовий. Перевагами застосування моделей SADT для опису бізнес-процесів є:

- повнота опису бізнес-процесу (управління, інформаційні і матеріальні потоки, зворотні зв'язки);
- жорсткі вимоги методу, що забезпечують отримання моделей стандартного вигляду;
- відповідність підходу до опису процесів стандартам ISO 9000.

На вітчизняних підприємствах бізнес-процеси почали формуватися і розвиватися порівняно недавно. Вони слабо типізуються, тому розумніше орієнтуватися на менш жорсткі моделі.

Метод моделювання IDEF-3, що є частиною сімейства стандартів IDEF, розроблено у 1980 р. для закритого проекту Міноборони США. Цей метод призначений для таких моделей процесів, у яких важливо зрозуміти послідовність виконання дій і взаємозалежності між ними. Хоча IDEF-3 і не досяг статусу федерального стандарту США, він набув значного поширення серед системних аналітиків як доповнення до методу функціонального моделювання IDEF-0 (моделі IDEF-3 можуть використовуватися для деталізації функціональних блоків IDEF-0, що не мають діаграм декомпозиції). Основою моделі IDEF-3 слугує сценарій процесу, що виділяє послідовність дій і під-процесів аналізованої системи.

Діаграми потоків даних (Data Flow Diagrams – DFD) є ієрархією функціональних процесів, пов'язаних потоками даних. Мета такого представлення – показати, як кожен процес перетворює свої вхідні дані у вихідні, а також виявити відношення між цими процесами.

Для побудови DFD традиційно використовуються дві різні нотації, відповідні методам Йордона – ДеМарко і Гейна – Сер-сона. Ці нотації відрізняються одна від одної графічним зображенням символів. Ця технологія забезпечує створення на ранніх стадіях діючої інтерактивної моделі системи-прототипу, що дає змогу демонструвати користувачам майбутню систему, уточнювати їх вимоги, оперативно модифікувати елементи інтерфейсів: форми введення повідомлень, меню, вихідні документи, склад функцій, структуру діалогу.

Підхід RAD-технології передбачає наявність трьох складових:

- невеликих груп розробників, що виконують роботи з проектування окремих підсистем ПЗ. Це зумовлено вимогою максимального управління колективом;
- короткого, але ретельно пророблено виробничого графіка;

– циклів повторення, при яких розробники в міру того, як програми починають працювати, вносять зауваження, отримані в результаті взаємодії із замовником.

Команда – це група професіоналів, які мають досвід у проектуванні, програмуванні та тестуванні ПЗ, і здатні добре взаємодіяти з користувачами, трансформуючи їх пропозиції в робочі прототипи.

Життєвий цикл ПЗ за підходом RAD включає чотири стадії:

1) аналіз і планування вимог;

2) проектування;

3) реалізація;

4) введення в дію.

На стадії ***аналізу і планування вимог*** користувачі здійснюють такі дії:

а) визначають функції, що має виконувати система;

б) виділяють найважливіші функції, що вимагають про-робки в першу чергу;

в) описують інформаційні потреби, список вимог до системи складається на основі пояснень користувачів під керівництвом фахівців-розробників. Крім того, на цій стадії:

– обмежується масштаб проекту;
– встановлюються часові рамки для кожної з наступних стадій;
– визначається сама можливість реалізації проекту в заданих розмірах фінансування.

У результаті має бути складено список функцій, що задані пріоритетним шляхом, майбутнього ПЗ ІС; спроектовано моделі ПЗ.

На стадії ***проектування*** частина користувачів бере участь у технічному проектуванні системи під керівництвом фахівців-розробників. Для швидкого одержання прототипів застосувань використовуються відповідні інструментальні засоби (CASE-засоби). Користувачі, взаємодіючи з розробниками, уточнюють і доповнюють вимоги до ІС, що не були виявлені на попередніх стадіях. На цій стадії виконуються такі дії:

– детальніше розглядаються процеси ІС;
– за необхідності для кожного процесу створюється частковий прототип: екранна форма, діалог, звіт. При цьому усуваються неоднозначності;
– встановлюються вимоги розмежування доступу до даних;
– визначається склад необхідної документації.

Після детального визначення складу процесів оцінюється кількість функціональних точок (function point) проектної ІС і приймається рішення про поділ ІС на підсистеми, що має реалізовуватися однією командою розробників за певний час для RAD-проектів (до 3 місяців). Функціональною точкою може бути кожнен з таких елементів ІС:

- вхідний елемент застосування (вхідний документ або екранна форма);
- вихідний елемент застосування (звіт, документ, екранна форма);
- запит (пари "питання/відповідь");
- логічний файл (сукупність записів даних, що використовуються всередині застосування);
- інтерфейс застосування (сукупність записів даних, переданих іншому застосуванню).

Далі проект розподіляється між різними командами розробників. Досвід розроблення великих ІС показує, що для підвищення ефективності робіт необхідно розбити проект на окремі підсистеми. Реалізація підсистем має виконуватися окремими групами фахівців. При цьому необхідно забезпечити координацію ведення загального проекту і виключити дублювання.

Моделі поведінки (behavioral):

- діаграми розміщення (deployment diagrams) – для моделювання фізичної архітектури системи.
- діаграми варіантів використання (use case diagrams) – для моделювання функціональних вимог до системи (у вигляді сценаріїв взаємодії користувачів з системою);
- діаграми взаємодії (interaction diagrams);
- діаграми послідовності (sequence diagrams) і кооперативні діаграми (collaboration diagrams) – для моделювання процесу обміну повідомленнями між об'єктами;
- діаграми станів (statechart diagrams) – для моделювання поведінки об'єктів системи при переході з одного стану в інший;
- діаграми діяльності (activity diagrams) – для моделювання поведінки системи в рамках різних варіантів використання або потоків управління.

UML має механізм розширення, призначений для того, щоб розробники могли адаптувати мову моделювання до своїх конкретних потреб, не змінюючи при цьому його метамодель. Наявність механізмів розширення

принципово відрізняє UML від таких засобів моделювання, як IDEF-0, IDEF-IX, IDEF-3, DFD і ERM. Перераховані мови моделювання можна визначити як сильно типізовані (аналогічно з мовами програмування), оскільки вони не допускають довільної інтерпретації семантики елементів моделей. UML, допускаючи таку інтерпретацію (в основному за рахунок стереотипів), є мовою, що слабо типізується. До її механізмів розширення відносять: стереотипи; тегування (іменовані) значення; обмеження.

Стереотип – це новий тип елементу моделі, який визначається на основі вже існуючого елементу. Стереотипи розширюють нотацію моделі і можуть застосовуватися до будь-яких елементів моделі. Стереотипи класів – це механізм, що дає змогу розділяти класи на категорії. Розробники ПЗ можуть створювати свої власні набори стереотипів, формуючи тим самим спеціалізовані підмножини UML. Такі підмножини (набори стереотипів) у стандарті мови UML мають назву профілів мови.

Іменоване значення – це пара рядків "тег - значення", або "ім'я - вміст", у яких зберігається додаткова інформація про який-небудь елемент системи, наприклад час створення, статус розробки або тестування, час закінчення роботи над ним тощо.

Обмеження – це семантичне обмеження, що має вид текстового виразу природною або формальною мовою (OCL – Object Constraint Language), який неможливо представити за допомогою нотації UML.

Основою взаємозв'язку між структурним і об'єктно орієнтованим підходами є спільність ряду категорій і понять обох підходів (процес і варіант використання, суть і клас тощо). Цей взаємозв'язок може проявлятися в різних формах. Так, одним з можливих варіантів є використання структурного аналізу як основи для об'єктно орієнтованого проектування. При цьому структурний аналіз слід припиняти, як тільки структурні моделі почнуть відображати не тільки діяльність організації, а і систему ПЗ. Після виконання структурного аналізу можна різними способами приступити до визначення класів та об'єктів. Іншою формою прояву взаємозв'язку можна вважати інтеграцію об'єктної і реляційної технологій. Реляційні СУБД є на сьогодні основним засобом реалізації великомасштабних баз даних і сховищ даних. Причини цього очевидні: реляційна технологія використовується досить довго, освоєна величезною кількістю користувачів і розробників, стала промисловим стандартом, у неї вкладені значні засоби і створена множина корпоративних БД у найрізноманітніших галузях, реляційна модель проста і має суто математичне подання; є велика різноманітність

промислових засобів проектування, реалізації та експлуатації реляційних БД. Внаслідок цього реляційні БД здебільшого використовуються для зберігання і пошуку об'єктів у так званих об'єктно реляційних системах.

1.2 Методи управління вимогами

RAD-технологія – кодування. Одночасність створення клієнтських і серверних місць ІС та активне залучення користувачів до процесу розроблення прикладного ПЗ зумовили поширення технологи швидкого розроблення застосувань RAD (Rapid Application Development) у рамках спіральної моделі ЖЦ.

У разі використання CASE-засобів це означає поділ функціональної моделі системи (наприклад за допомогою діаграм потоків даних для структурного підходу або діаграм варіантів використання для об'єктно орієнтованого підходу). В результаті має бути створено:

- загальну інформаційну модель ІС;
- функціональні моделі системи в цілому і підсистеми, що реалізовані окремими командами розробників;
- інтерфейси між автономно працюючими підсистемами;
- прототипи екранних форм, звітів, діалогів.

Усі моделі і прототипи мають бути отримані із застосуванням саме тих CASE-засобів, які будуть використовуватися далі, при побудові системи. Ця вимога зумовлюється необхідністю уникнути неконтрольованого перекручування даних при передачі інформації про проект з однієї стадії на іншу.

У підході RAD кожен прототип розвивається таким чином, що наступна стадія наслідує попередню.

На ***стадії реалізації*** відбувається безпосереднє швидке розроблення застосування:

- розробники здійснюють ітеративну побудову реальної системи на основі отриманих на попередній стадії моделей, а також вимог нефункціонального характеру (вимог до надійності, продуктивності тощо);
- користувачі оцінюють результати і вносять коригування, якщо у процесі розроблення система перестає відповідати визначеним раніше вимогам.

Тестування системи здійснюється у процесі розроблення. Після закінчення робіт кожної окремої команди розробників здійснюється поступова інтеграція однієї частини системи з іншими, формується повний програмний код, проводиться тестування спільної роботи окремої частини застосування, а потім – тестування системи в цілому. Реалізація системи завершується такими процесами:

- аналізується використання даних і визначається необхідність їхнього розподілу;
- розробляється фізичне проектування бази даних;
- формулюються вимоги до апаратних ресурсів;
- установлюються способи підвищення продуктивності;
- завершується розроблення документації проекту.

На ***стадії упровадження*** здійснюється навчання користувачів та організаційні зміни, і паралельно із введенням нової системи продовжується експлуатація старої. Стадія реалізації займає небагато часу, тому планування і підготовка до впровадження мають починатися заздалегідь, ще на стадії проектування системи. Конкретна реалізація стадії залежить від умов, у яких починалось розроблення ІС, тобто потрібно:

- розробити нову систему "з нуля";
- створити модель діяльності підприємства, за якою можна розробити ІС;
- розробити ІС на основі старої.

Варто зазначити, що підхід RAD не претендує на універсальність. Він придатний тільки для невеликих проектів. Крім того, підхід RAD недоцільно застосовувати для побудови складних розрахункових програм, операційних систем чи програм управління складними об'єктами в реальному масштабі часу, тобто програм, що містять великий обсяг програмного коду.

Основна проблема процесу розробки ІС через підхід RAD полягає у визначенні моменту переходу на наступний етап, тому вводяться часові обмеження на кожен етап життєвого циклу. Після формування технічного завдання та декомпозиції системи здійснюється незалежне розроблення підсистем з наступним збиранням, тестуванням, упровадженням ІС.

Використовують два основних варіанти організації технологічного процесу проектування з використанням систем-прототипів. Перший з них використовують для специфікації вимог до ІС, після розроблення яких прототип стає непотрібним.

Основний недолік цього варіанта – неефективне використання системи-прототипу; після усування недоліків у проекті та вдосконалення постановки задачі прототипи не використовують.

Другий варіант передбачає ітераційний розвиток системи-прототипу в готовий для експлуатації програмний продукт. Ітерації розроблення системи-прототипу включають створення та модифікацію системи-прототипу, її демонстрацію користувачу та узгодження, розроблення нових специфіка цій-вимог до системи, нову модифікацію доти, доки не буде створено готовий продукт. При цьому підході різко скорочуються час, ресурси на проектування, розроблення і впровадження ІС. Основні принципи технології RAD такі:

- розроблення застосувань ітераціями;
- необов'язковість повного завершення робіт на кожній стадії ЖЦПЗ;
- обов'язковість залучення користувачів у процес розроблення ІС;
- доцільність застосування CASE-засобів, що забезпечують цілісність проекту і генерацію коду застосувань;
- доцільність застосування засобів управління конфігурацією, що полегшують внесення змін у проект і супровід готової системи;
- використання прототипування, що дає змогу повніше з'ясувати і задовольнити потреби користувачів;
- тестування й розвиток проекту, що здійснюються одночасно з розробленням;
- ведення розробки завдяки нечисленній команді професіоналів;
- грамотне управління розробленням ІС, чітке планування і контроль виконання робіт.

У 1977 році Тоні Кроуфорд (Tony Crawford) з IBM звернув увагу на труднощі досягнення взаєморозуміння між бізнес клієнтами та IT фахівцями. Для вирішення цих труднощів він винайшов новий підхід. Ідея полягала у тому, щоб зібрати всіх учасників і досягти консенсусу. Перші наради щодо спільної розробки додатків відбулася в офісі IBM в Релайе (Raleigh, штат Північна Кароліна). Так виникла нова методика JAD (Joint Application Development).

JAD (Joint Application Development – спільна розробка додатків) – методика управління проектами, яка передбачає тісну взаємодію замовників і виконавців, з метою домогтися взаєморозуміння в питаннях, що стосуються розроблюваної системи.Зокрема, цю методологію згадує Ерік Спірлі (Eric Sperly) у книзі «Корпоративные хранилища данных. Планирование,

разработка и реализация» (Enterprise Data Warehouse. Planning, Building, and Implementation).

JAD процес забезпечує для розробки комп'ютерних систем те, що Генрі Форд зробив для автомобільного виробництва (метод організації механізмів, матеріалів, та робочої сили таким чином, що машину можна зібрати набагато швидше і дешевше, ніж будь-коли раніше – лінія по збірці). Цілі в області розробки систем полягає в тому, щоб визначити , що користувачам дійсно потрібно, а потім створити систему або процес, який забезпечить це. Традиційні методи мають ряд вбудованих уповільнюють факторів, які тим сильніше, чим більше людей залучено в проект.

Метою JAD є визначення меж проекту, розробка рішення і контроль проекту до його завершення.

Учасниками спільної розробки додатків є:

Спонсор (спонсор) – керівник, власник системи. Вони повинні бути достатньо високопоставленими в організації, щоб бути ув змозі прийняти рішення таі забезпечити необхідні ресурси і підтримку проекту .

Бізнес-користувачі – передбачувані користувачі розроблюваної системи.

Системні аналітики – забезпечують нетехнічні пояснення, які допомагають іншим членам JAD – групи розуміти і повністю використовувати доступну технологію. Контролюють дизайн щодо простоти використання/обслуговування та відповідно доості стандартівам. Забезпечують розробку апаратних засобів і програмного забезпечення.

JAD зосереджений навколо конструктивних робочих нарад. Всі збираються разом у кімнаті для обговорень. Кожен чує, що говорить інша частина групи. Немає затримок між питаннями і відповідями, ні "зайнятих телефонів" або очікування відповіді на записку. Спільна розробка додатків усуває багато проблем традиційних зустрічей. JAD перетворює зустрічі в наради, які є більш конструктивними й продуктивними. Порядок денний забезпечує конструктивізм, помічник спрямовує процес, наочні посібники уточнюють обговорювані поняття, і активність групи, з постійним зворотним зв'язком.

1.3 Методи моделювання потоків даних

Основою даної методології (методології Gane/Sarson) є побудова моделі аналізованої ІС – проектованої або реально існуючої. Відповідно з

методологією модель системи визначається як ієрархіядіаграм потоків даних (DFD), що описують асинхронний процес перетворення інформації від її введення в систему до видачі користувачеві. Діаграми верхніх рівнів ієрархії (контекстні діаграми) визначають основні процеси або підсистеми ІС із зовнішніми входами і виходами. Вони деталізуються за допомогою діаграм нижнього рівня. Така декомпозиція триває, створюючи багаторівневу ієрархію діаграм, до тих пір, поки не буде досягнутий такий рівень декомпозиції, наякому процеси стають елементарними і деталізувати їх далі неможливо.

Джерела інформації (зовнішні сутності) породжують інформаційні потоки (потоки даних), які переносять інформацію до підсистем або процесів. Ті, в своючергу, перетворюють інформацію і породжують нові потоки, які переносять інформацію до інших процесів або підсистем, накопичувачів даних, або зовнішнім сутностей – споживачам інформації. Таким чином, основними компонентами діаграм потоків даних є:

- зовнішні сутності;
- системи (підсистеми);
- процеси;
- накопичувачі даних;
- потоки даних.

Зовнішня сутність – матеріальний предмет або фізична особа, що представляє собою джерело або приймач інформації, наприклад, замовники, персонал, постачальники, клієнти, склад. Визначення деякого об'єкта або системи в якості зовнішньої суті вказує на те, що вона знаходиться за межами кордонів аналізованої ІС. У процесі аналізу деякі зовнішні суті можуть бути перенесені всередину діаграми аналізованої ІС, якщо це необхідно, або, навпаки, частина процесів ІС може бути винесена за межі діаграми і представлена як зовнішня сутність.

Системи і підсистеми. При побудові моделі складної ІС вона може бути представлена в найзагальнішому вигляді на так званій контекстної діаграмі у вигляді однієї системи як єдиного цілого або може бути декомпозирована на ряд підсистем. Номер підсистеми виконує функцію її ідентифікації. В полі імені вводиться найменування підсистеми у вигляді пропозиції з підметом і відповідними визначеннями і доповненнями.

Процес є перетворенням вхідних потоків даних у вихідні відповідно до певного алгоритму. Фізично процес може бути реалізований різними способами: це може бути підрозділ організації (відділ), яке виконує обробку

вхідних документів і випуск звітів; програма; аппаратно реалізоване логічне пристрій та ін.

Накопичувач даних є абстрактним пристроєм для зберігання інформації, яку можна в будь-який момент помістити в накопичувач і через деякий час витягнути, при цьому способи розміщення і вилучення можуть бути будь-якими. Накопичувач даних може бути реалізований фізично у вигляді ящика в картотеці, таблиці в оперативній пам'яті, файла та ін.

Накопичувач даних в загальному випадку є прообразом майбутньої бази даних, опис даних який зберігається в ньому, має бути ув'язано з інформаційною моделлю.

Потік даних визначає інформацію, передану через деяке з'єднання від джерела до приймача. Реальний потік даних може бути інформацією, що передається по кабелю між двома пристроями, пересилаються електронною поштою, магнітними носіями, які переносяться з комп'ютера на комп’ютер та ін.

Ієрархія діаграм потоків даних (ДПД). Першим кроком при побудові ієрархії ДПД є побудова контекстних діаграм. Зазвичай при проектуванні простих ІС будується єдина контекстна діаграма із зіркоподібною топологією, в центрі якої знаходиться так званий головний процес, сполучений з приймачами і джерелами інформації, за допомогою яких з системою взаємодіють користувачі та інші зовнішні системи. Якщо для складної ІС обмежитися єдиною контекстної діаграмою, то вона буде містити занадто велику кількість джерел і приймачів інформації, які важко розташувати на аркуші паперу нормального формату, і, крім того, єдиний головний процес не розкриває структури розподіленої системи.

Ознаками складності (в сенсі контексту) можуть бути:

- наявність великої кількості зовнішніх сутностей (більше десяти);
- розподілена природа системи;
- багатофункціональність системи з сформованим або виявленим угрупованням функцій в окремі підсистеми.

Для складних ІС будується ієрархія контекстних діаграм. При цьому контекстна діаграма верхнього рівня містить не єдиний головний процес, а набір підсистем, що з'єднуються потоками даних. Контекстні діаграми наступного рівня деталізують контекст і структуру підсистем.

Ієрархія контекстних діаграм визначає взаємодію основних функціональних підсистем ІС як між собою, так і з зовнішніми вхідними і вихідними потоками даних та зовнішніми об'єктами (джерелами і

приймачами інформації), з якими взаємодіє ІС. Розробка тематичних діаграм вирішує проблему суворого визначення функціональної структури ІС на ранній стадії її проектування, що особливо важливо для складних багатофункціональних систем, в розробці яких беруть участь різні організації та колективи розробників.

Після побудови контекстних діаграм отриману модель слід перевірити на повноту вихідних даних про об'єкти системи і на ізольованість об'єктів (відсутність інформаційних зв'язків з іншими об'єктами). Для кожної підсистеми, яка міститься на контекстних діаграмах, виконується її деталізація за допомогою ДПД. Кожен процес на ДПД, в свою чергу, може бути деталізований за допомогою ДПД або мініспецификації. При деталізації їх необхідно виконувати такі правила:

- правило балансування – означає, що при деталізації підсистеми або процесу діаграма в якості зовнішніх джерел (приймачів) даних може мати тільки ті компоненти (підсистеми, процеси, зовнішні сутності, накопичувачі даних), з якими має інформаційну зв'язок підсистема або процес на батьківській діаграмі;
- правило нумерації – означає, що при деталізації процесів повинна підтримуватися їх ієрархічна нумерація. Наприклад, процеси, які деталізують процес з номером 12, отримують номери 12.1, 12.2, 12.3.

Мініспецификація (опис логіки процесу) повинна формулювати його основні функції таким чином, щоб надалі фахівець, що виконує реалізацію проекту, зміг їх виконати або розробити відповідну програму.

Мініспецификація є кінцевою вершиною ієрархії ДПД. Рішення про завершення деталізації процесу і використання мініспецификації приймається аналітиком виходячи з таких критеріїв:

- наявність у процесу відносно невеликої кількості вхідних і вихідних потоків даних (2 – 3 потока), можливості опису перетворення даних процесом у вигляді послідовного алгоритму;
- виконання процесом єдиної логічної функції перетворення вхідної інформації у вихідну, можливість опису логіки процесу за допомогою мініспецификації невеликого обсягу (не більше 20 – 30 рядків).

При побудові ієрархії ДПД переходити до деталізації процесів слід тільки після визначення змісту всіх потоків і накопичувачів даних, які описуються за допомогою структур даних. структури даних конструюються з

елементів даних і можуть містити альтернативи, умовні входження і ітерації. Умовне входження означає, що даний компонент може бути відсутнім в структурі. Альтернатива означає, що в структуру може входити один з перелічених елементів. Ітерація означає входження будь-якого числа елементів в зазначеному діапазоні. Для кожного елемента даних може вказуватися його тип (безперервні або дискретні дані); для безперервних даних –одиниця виміру (кг, см і т.п.), діапазон значень, точність представлення та форма фізичного кодування; для дискретних даних – таблиця допустимих значень. Діаграми потоків даних будуються за ієрархічним принципом (рис. 1.1).

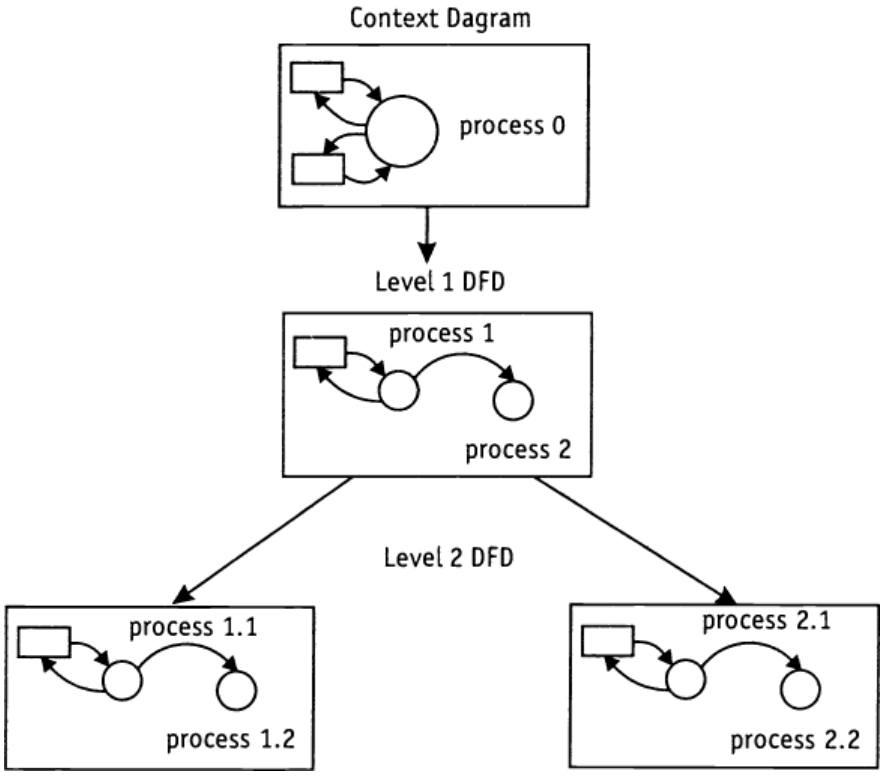

Рисунок 1.1 − Структура ієрархії DFD-діаграм

Контекстна діаграма верхнього рівня визначає межі моделі. Як правило, вона має зіркоподібну топологію, в центрі якої знаходиться головний процес, який з'єднаний з приймачами і джерелами інформації, що є зовнішнім оточенням інформаційної системи (рис. 1.2).

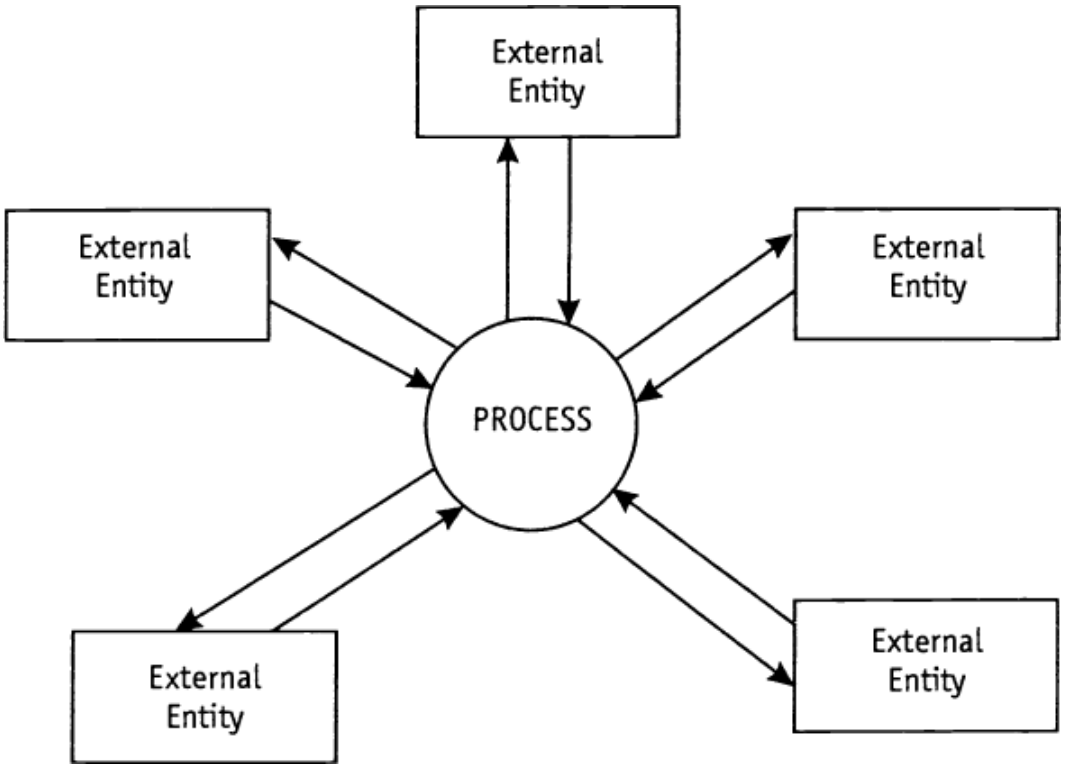

Рисунок 1.2 – Контекстна діаграма потоків даних

Для кожного процесу діаграми першого рівня може бути проведена декомпозиція, яка, в свою чергу, також може бути розкрита більш докладно. Декомпозиція процесів закінчується, коли досягнутий необхідний ступінь деталізації або процеси на черговому рівні діаграм є елементарними і не можуть бути розбиті на більш дрібні.

2 ОБ’ЄКТНО-ОРІЄНТОВАНА МЕТОДОЛОГІЯ ПРОЕКТУВАННЯ ІС

2.1 Концептуальна модель UML

Для розуміння UML необхідно засвоїти його концептуальну модель, яка включає в себе три складові частини:

- основні будівельні блоки мови;
- правила їх поєднання;
- деякі загальні для всієї мови механізми.

Словник мови UML включає три види будівельних блоків:

- сутності;
- відносини;
- діаграми.

Сутності – це абстракції, які є основними елементами моделі. ***Відносини*** пов'язують різні сутності. ***Діаграми*** представляють інтерес сукупності сутностей.

В UML є чотири типи сутностей:

- структурні;
- поведінкові;
- сутності, що групують;
- анотаційні.

Сутості є основними об'єктно-орієнтованими блоками мови. З їх допомогою можна створювати коректні моделі. ***Структурні сутності*** – це імена іменники в моделях на мові UML. Як правило, вони представляють собою статичні частини моделі, які відповідають концептуальним або фізичним елементам системи. Існує кілька різновидів структурних сутностей.

- клас (Class) – це опис сукупності об'єктів із загальними атрибутами, операціями, відносинами і семантикою. Клас реалізує один або кілька інтерфейсів.
- інтерфейс (Interface) – це сукупність операцій, які визначають сервіс (набір послуг), що надається класом або компонентом.
- прецедент (Use case) – це опис послідовності дій, які виконує система.
- кооперація (Collaboration) визначає взаємодію; вона являє собою сукупність ролей і інших елементів, які спільно працюють і роблять деякий кооперативний ефект.

– компонент (Component) – це фізична замінна частина системи, яка відповідає деякому набору інтерфейсів і забезпечує його реалізацію.

Поведінкові сутності (Behavioral things) є динамічними складовими моделі UML. Вони описують поведінку моделі в часі і просторі. Існує всього два основних типи поведінкових сутностей.

Взаємодія (Interaction) – це поведінка, суть якого полягає в обміні повідомленнями (Messages) між об'єктами в рамках конкретного контексту для досягнення певної мети.

Сутності, що групують – це організаційні частини моделі UML. Це блоки, на які можна розкласти модель. Пакети (Packages) представляють собою універсальний механізм організації елементів в групи. У пакет можна помістити структурні, поведінкові і навіть інші сутності, що групують.

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

Переваги використання UML:

1) UML об'єктно-орієнтований, в результаті чого методи опису результатів аналізу і проектування семантично близькі до методів програмування на сучасних об'єктно-орієнтованих мовах.

2) UML дозволяє описати систему практично з усіх можливих точок зору і різні аспекти поведінки системи.

3) Діаграми UML порівняно прості для читання після досить швидкого ознайомлення з його синтаксисом.

4) UML розширює і дозволяє вводити власні текстові та графічні стереотипи, що сприяє його застосування не тільки в сфері програмної інженерії.

5) UML набув широкого поширення і динамічно розвивається.

Для підтримки моделювання різних етапів життєвого циклу ІС мова UML пропонує цілу сукупність діаграм. У нотації мови UML визначені наступні види канонічних діаграм:

– варіантів використання (use case diagram);
– класів (class diagram);
– кооперації (collaboration diagram);
– послідовності (sequence diagram);
– станів (statechart diagram);
– діяльності (activity diagram);

– компонентів (component diagram);
– розгортки (deployment diagram).

Перелік цих діаграм і їх назв є канонічними в тому сенсі, що являють собою невід'ємну частину графічної нотації мови UML. Більше того, процес об'єктно-орієнтованого проектування нерозривно пов'язаний із процесом побудови цих діаграм. Сукупність побудованих у такий спосіб діаграм є самодостатньою в тому сенсі, що в них міститься вся інформація, яка необхідна для реалізації проекту складної системи (рис. 2.1).

Рисунок 2.1 − Класифікація груп діаграм UML 2.0

Діаграма класів (Class diagram) – статична структурна діаграма, що описує структуру системи, що демонструє класи системи, їх атрибути, методи і залежності між класами.

Існують різні точки зору на побудову діаграм класів в залежності від цілей їх застосування:

Концептуальна точка зору – це діаграма класів, яка описує модель предметної області, в ній присутні тільки класи прикладних об'єктів.

Точка зору специфікації – діаграма класів застосовується при проектуванні інформаційних систем.

Точка зору реалізації – діаграма класів, яка містить класи, що використовуються безпосередньо в програмному коді (при використанні об'єктно-орієнтованих мов програмування).

Діаграма компонентів (Component diagram) – статична структурна діаграма, яка показує розбиття програмної системи на структурні компоненти та зв'язки (залежності) між компонентами. Як фізичні компоненти можуть виступати файли, бібліотеки, модулі, файли виконання, пакети та ін.

Діаграма композитної (складової) структури (Composite structure diagram) – статична структурна діаграма, яка демонструє внутрішню структуру класів і взаємодію елементів (частин) внутрішньої структури класу.

Підвидом діаграм композитної структури є ***діаграми кооперації*** (Collaboration diagram, введені в UML 2.0), які показують роль і взаємодію класів в рамках кооперації. Кооперації зручні при моделюванні шаблонів проектування.

Діаграми композитної структури можуть використовуватися спільно з діаграмами класів.

Діаграма розгортання (Deployment diagram, діаграма розміщення) виконують функцію моделювання працюючих вузлів (апаратних засобів) і артефактів, що розгорнуті на них. В UML 2 на вузлах розгортаються артефакти (artifact), в той час як в UML 1 на вузлах розгорталися компоненти. Між артефактом і логічним елементом (компонентом), який він реалізує, встановлюється залежність маніфестації.

Діаграма об'єктів (Object diagram) демонструє повний або частковий знімок системи в заданий момент часу. На діаграмі об'єктів відображаються екземпляри класів (об'єкти) системи із зазначенням поточних значень їх атрибутів і зв'язків між об'єктами.

Діаграма пакетів (Package diagram) – структурна діаграма, основним змістом якої є пакети і відносини між ними. Жорсткого поділу між різними структурними діаграмами не проводиться, тому дана назва пропонується виключно для зручності і не має семантичного значення (пакети і діаграми пакетів можуть бути присутні на інших структурних діаграмах). Діаграми пакетів служать, в першу чергу, для організації елементів в групи за будь-якою ознакою з метою спрощення структури і організації роботи з моделлю системи.

Діаграма діяльності (Activity diagram) – діаграма, на якій показано розкладання деякої діяльності на її складові частини. Під діяльністю (activity) розуміється специфікація поведінки у вигляді координованого послідовного і паралельного виконання підлеглих елементів (вкладених видів діяльності і окремих дій), які з'єднані між собою потоками, що йдуть від виходів одного вузла до входів іншого.

Діаграми діяльності використовуються при моделюванні бізнес-процесів, технологічних процесів, послідовних і паралельних обчислень.

Аналогом діаграм діяльності є схеми алгоритмів по ГОСТ 19.701-90 і дракон-схеми.

Діаграма станів (State Machine diagram, діаграма кінцевого автомата, діаграма станів) – діаграма, на якій представлено кінцевий автомат з простими станами, переходами і композитними станами.

Діаграма кінцевих станів (State machine) – специфікація послідовності станів, через які проходить об'єкт або взаємодія у відповідь на події свого життя, а також відповідні дії об'єкта на ці події. Кінцевий автомат прикріпляється до вихідного елементу (класу, кооперації або методу) і служить для визначення поведінки його примірників.

Аналогом діаграм станів (діаграм автомата) є дракон-схеми.

Діаграма варіантів використання (Use case diagram, діаграма прецедентів) – діаграма, на якій відображені відносини, що існують між акторами і варіантами використання.

Основне завдання – представляти собою єдиний засіб, що дає можливість замовнику, кінцевому користувачеві і розробнику спільно обговорювати функціональність і поведінку системи.

Діаграма комунікації (Communication diagram, в UML 1.x – діаграма кооперації, collaboration diagram) – діаграма, на якій зображуються взаємодії між частинами композитної структури або ролями кооперації. На відміну від діаграми послідовності, на діаграмі комунікації явно вказуються відносини між елементами (об'єктами), а час, як окремий вимір, не використовується (застосовуються порядкові номери викликів).

Діаграма послідовності (Sequence diagram) – діаграма, на якій показані взаємодії об'єктів, впорядковані за часом їх прояву, на якій зображено впорядкована в часі взаємодія об'єктів. Зокрема, на ній зображуються об’єкти, що беруть участь у взаємодії, і послідовність повідомлень, якими вони обмінюються.

Діаграма співпраці – це діаграма, що дозволяє описати взаємодії об'єктів, абстрагуючись від послідовності передачі повідомлень. На цьому типі діаграм в компактному вигляді відображаються всі прийняті і передані повідомлення конкретного об'єкта і типи цих повідомлень.

У зв'язку з тим, що діаграми Sequence і Collaboration є різними поглядами на одні і ті ж процеси, програмне забезпечення дозволяє створювати з Sequence діаграми діаграму Collaboration і навпаки, а також робить автоматичну синхронізацію цих діаграм.

Діаграма огляду взаємодії (Interaction overview diagram) – різновид діаграми діяльності, що включає фрагменти діаграми послідовності і конструкції потоку управління.

Цей тип діаграм включає в себе діаграми Sequence diagram (діаграми послідовностей дій) і Collaboration diagram (діаграми співробітництва). Ці діаграми дозволяють з різних точок зору розглянути взаємодія об'єктів в системі.

Діаграма синхронізації (Timing diagram) – альтернативне подання діаграми послідовності, явно показує зміни стану на лінії життя із заданою шкалою часу. Може бути корисна в додатках реального часу.

Незважаючи на те, що UML досить широко поширений стандарт, його часто критикують через наступні недоліки:

1) Надмірність мови. UML часто критикується, як невиправдано великий і складний. Він включає багато надлишкових або практично невикористовуваних діаграм і конструкцій. Найчастіше це можна почути по відношенню до UML 2.0, ніж UML 1.0, так як більш нові версії включають більше «розроблених комітетом» компромісів.

2) Неточна семантика. Так як UML є комбінацією абстрактного синтаксису, OCL (мова опису обмежень) і англійської мови (докладна семантика), то він позбавлений скутості, що властива мовам формального опису. У деяких випадках абстрактний синтаксис UML, OCL і англійська суперечать один одному, в інших випадках вони неповні. Неточність опису самого UML однаково впливає на користувачів і постачальників інструментів, що приводить до несумісності інструментів через унікальні трактування специфікацій.

3) Проблеми при вивченні і впровадженні. Вказані вище проблеми роблять досить складним вивчення та впровадження UML, особливо коли керівництво насильно змушує використовувати UML інженерів за відсутності у них попередніх навичок.

4) Тільки код відображає код. Існує потреба в кращому способі написання ПЗ; UML цінується при підходах, які компілюють моделі для генерування вихідного коду. Однак цього все ж може бути недостатньо, так як UML не має повноти можливостей і будь-який згенерований код буде мати обмеження, що пов’язані з особивостями інтерпретації UML інструментів.

5) Кумулятивне навантаження (Cumulative Impedance/Impedance mismatch). Неузгодженість навантаження – це термін з теорії системного

аналізу для позначення нездатності входу однієї системи сприйняти вихід іншої. Як у будь-якій системі позначень UML може представити одні системи більш коротко і ефективно, ніж інші. Таким чином, розробник схиляється до рішень, які більш комфортно підходять до використання сильних сторін UML і мов програмування. Проблема стає більш очевидною, якщо мова розробки не дотримується принципів ортодоксальної об'єктно-орієнтованої доктрини (не відповідає традиційним принципам ООП).

6) Намагається бути всім для всіх. UML – це мова моделювання загального призначення, яка намагається досягти сумісності з усіма можливими мовами розробки. У контексті конкретного проекту, для досягнення командою проектувальників визначеної мети, повинні бути обрані інструменти UML, що мають можливість подальшого застосування.

2.2 Моделювання стану інформаційних систем

Діаграма стану призначена для опису можливих послідовностей станів і переходів, які в сукупності характеризують поведінку елемента моделі протягом його життєвого циклу.

Найчастіше діаграми станів використовуються для опису поведінки окремих екземплярів класів (об'єктів), але вони також можуть бути застосовані для специфікації функціональності інших компонентів моделей, таких як варіанти використання, актори, підсистеми, операції і методи.

Діаграма станів по суті є графом спеціального виду, який представляє певний автомат. Вершинами цього графа є стани і деякі інші типи елементів автомата (псевдостани), які зображуються відповідними графічними символами. Дуги графа служать для позначення переходів зі стану в стан. Для розуміння семантики конкретної діаграми станів необхідно представляти не тільки особливості поведінки сутності, а і знати загальні відомості з теорії автоматів.

Автомат (state machine) в мові UML є певний формалізм для моделювання поведінки елементів моделі і системи в цілому. Автомат описує поведінку окремого об'єкта в формі послідовності станів, які охоплюють всі етапи його життєвого циклу, починаючи від створення об'єкта і закінчуючи його знищенням. Кожна діаграма станів представляє певний автомат.

Найпростішим прикладом візуального представлення станів і переходів на основі формалізму автоматів може служити ситуація з справністю технічного пристрою, такого як комп'ютер. В цьому випадку вводяться в

розгляд два найзагальніших стану: "справний" і "несправний" і два переходу: "вихід з ладу" і "ремонт". Графічно ця інформація може бути представлена у вигляді зображеної нижче діаграми станів комп'ютера (рис.2.2).

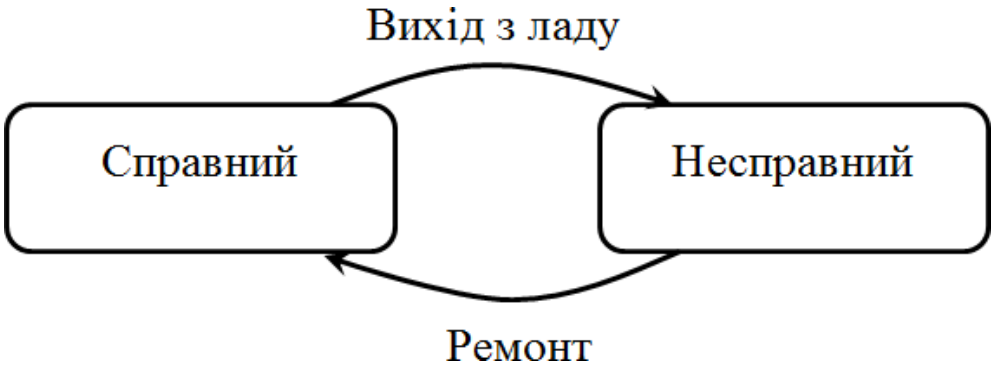

Рисунок 2.1 − Приклад діаграми стану для технічного пристрою (комп’ютера)

Основними поняттями, що входять в формалізм автомата, є стан і перехід. Головна відмінність між ними полягає в тому, що тривалість перебування системи в окремому стані істотно перевищує час, який витрачається на перехід з одного стану в інший. Передбачається, що в межі час переходу з одного стану в інший дорівнює нулю (якщо додатково нічого не сказано). Іншими словами, перехід об'єкта зі стану в стан відбувається миттєво.

У мові UML під станом розуміється абстрактний метаклас, що використовується для моделювання окремої ситуації, протягом якої має місце виконання деякої умови. Стан може бути задано у вигляді набору конкретних значень атрибутів класу або об'єкта, при цьому зміна їх окремих значень буде відображати зміну стану класу або об'єкта, що моделюється.

Стан на діаграмі зображується прямокутником із закругленими вершинами (рис. 2.2). Цей прямокутник, в свою чергу, може бути розділений на дві секції горизонтальною лінією. Якщо вказана лише одна секція, то в ній записується тільки ім'я стану. В іншому випадку в першій з них записується ім'я стану, а в другій – список деяких внутрішніх дій чи переходів в даному стані (рис. 2.3).

Початковий стан є окремим випадком стану, який не містить ніяких внутрішніх дій. У цьому стані перебуває об'єкт за умовчанням в початковий

момент часу. Графічно початковий стан в мові UML позначається у вигляді закрашеної окружності, з якої може тільки виходити стрілка, що відповідає переходу.

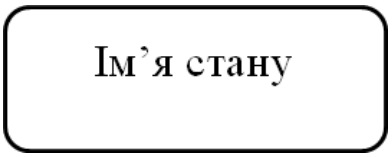

Рисунок 2.2 – Графічне зображення стану

Рисунок 2.3 − Графічне зображення стану із списком дій

Кінцевий (фінальний) стан є окремим випадком стану, який також не містить ніяких внутрішніх дій. В цьому стані буде перебувати об'єкт за умовчанням після завершення роботи автомата в кінцевий момент часу. Графічно кінцевий стан в мові UML позначається у вигляді закрашеної окружності, яка міститься в колі, в неї також може тільки входити стрілка, що відповідає переходу.

Простий перехід (simple transition) – це відношення між двома послідовними станами, яке вказує на факт зміни одного стану іншим. Перебування об'єкта в першому стані може супроводжуватися виконанням деяких дій, а перехід в другий стан буде можливий після завершення цих дій, а також після задоволення деяких додаткових умов. На діаграмі станів перехід зображується суцільною лінією зі стрілкою, яка направлена в цільовий стан (рис. 2.4).

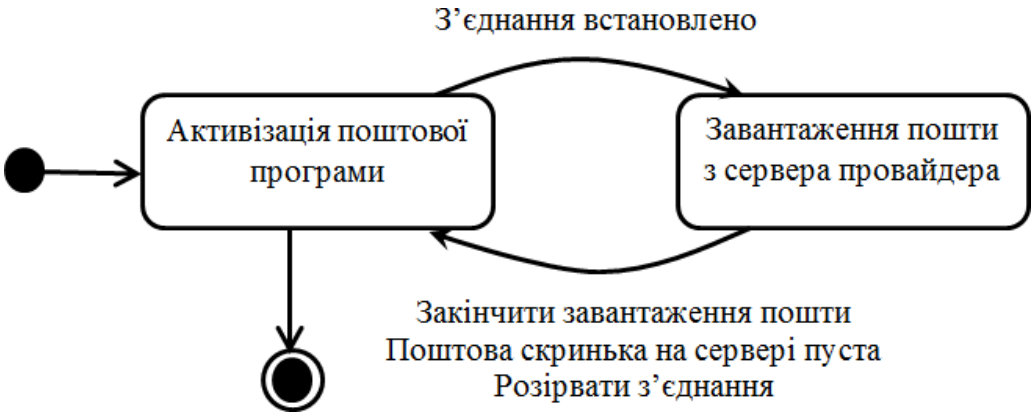

Рисунок 2.4 – Діаграма стану моделювання поштової програми

Розглянемо діаграму стану для моделювання поведінки банкомату (рис. 2.5).

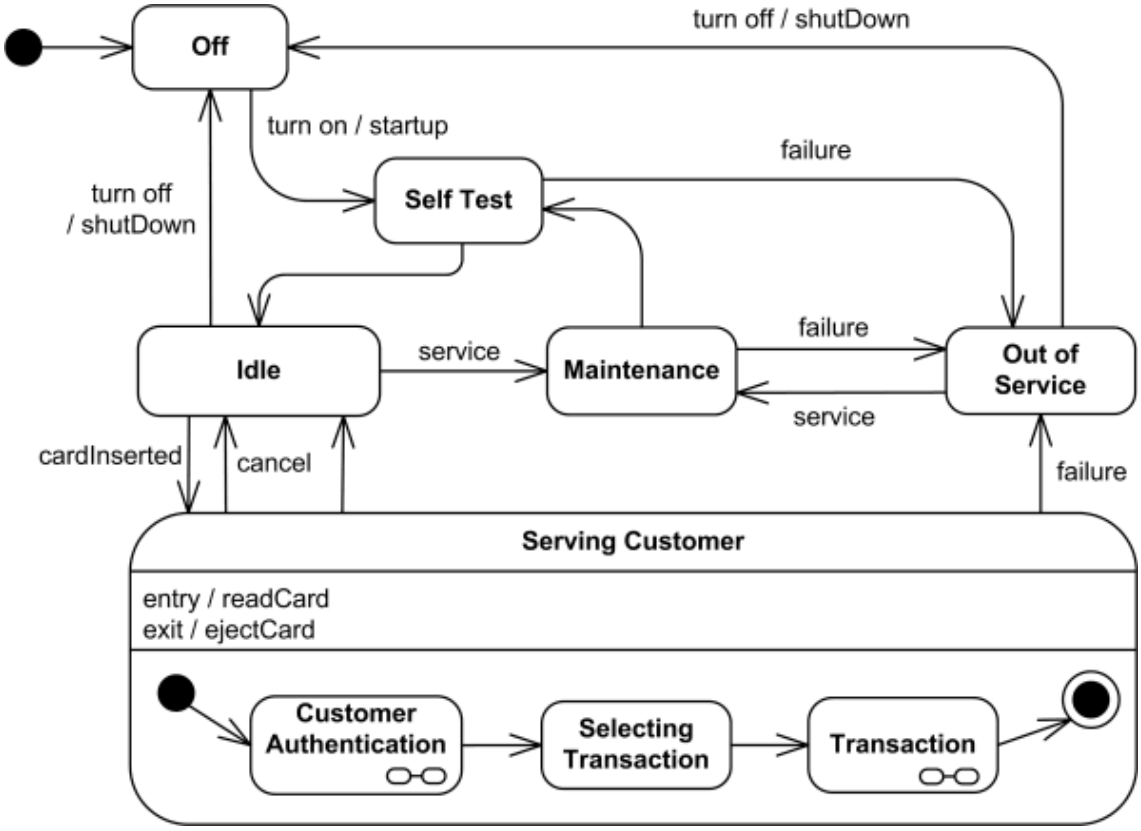

Рисунок 2.5 – Діаграма стану моделювання банкомату

Слід зауважити, що в представленій моделі діаграма станів є єдиною і описує поведінку системи управління банкоматом в цілому. Головна

перевага даної діаграми станів – можливість моделювати умовний характер реалізації всіх варіантів використання в формі зміни окремих станів системи, що розробляється. Іноді розробку діаграми станів, особливо в умовах дефіциту часу, запланованого на виконання проекту, опускають, тому що часто відбувається дублювання інформації, представленої на діаграмах кооперації і послідовності.

2.3 Моделювання видів діяльності інформаційних систем

При моделюванні поведінки системи при аналізі або проектуванні виникає необхідність не тільки представити процес зміни її станів, а і деталізувати особливості алгоритмічної і логічної реалізації операцій, які виконує система.

Для моделювання процесу виконання операцій в мові UML використовуються так звані діаграми діяльності. Графічна нотація таких діаграм багато в чому схожа на нотацію діаграми станів, оскільки на діаграмах діяльності також присутні позначення станів і переходів. Відмінність полягає в семантиці станів, які використовуються для уявлення не діяльностей, а дій, і у відсутності на переходах сигнатури подій. Кожний стан на діаграмі діяльності відповідає виконанню деякої елементарної операції, а перехід в наступний стан спрацьовує тільки при завершенні цієї, операції в попередньому стані. Графічно діаграма діяльності видається в формі графа діяльності, вершинами якого є стани дії, а дугами – переходи від одного стану дії до іншого.

В контексті мови UML діяльність (activity) – це деяка сукупність окремих обчислень, що виконуються автоматом. При цьому окремі елементарні обчислення можуть призводити до деякого результату або дії (action). На діаграмі діяльності відображається логіка або послідовність переходу від однієї діяльності до іншої, при цьому увагу фіксується на результаті діяльності. Результат може привести до зміни стану системи або повернення деякого значення.

Стан дії (action state) є спеціальним випадком стану з деякою вхідною дією і принаймні одним станом переходу, який виходить. Графічно стан дії зображується фігурою, що нагадує прямокутник. Усередині цієї фігури записується вираз дії (action-expression), який повинен бути унікальним в межах однієї діаграми діяльності.

Дія може бути записана на природній мові, деякому псевдокоді або мовою програмування. Ніяких додаткових або неявних обмежень при записі дій не накладається. При побудові діаграми діяльності використовуються тільки ті переходи, які переводять діяльність в подальший стан відразу, як тільки закінчиться дія в попередньому стані (нетригерні). На діаграмі такий перехід зображується суцільною лінією зі стрілкою. Розгалуження на діаграмі діяльності позначається невеликим ромбом, всередині якого немає ніякого тексту.

Розглянемо фрагмент алгоритму знаходження коренів квадратного рівняння. У загальному випадку після приведення рівняння другого ступеня до канонічного вигляду: a * x * x + b * x + c = 0 необхідно обчислити його дискримінант. Причому, в разі негативного дискримінанту рівняння не має рішення на множині дійсних чисел, і подальші обчислення повинні бути припинені. При позитивному дискримінанті рівняння має рішення, коріння якого можуть бути отримані на основі конкретної розрахункової формули.

Процедуру обчислення коренів квадратного рівняння можна представити у вигляді діаграми діяльності з трьома станами дії і розгалуженням (рис. 2.6).

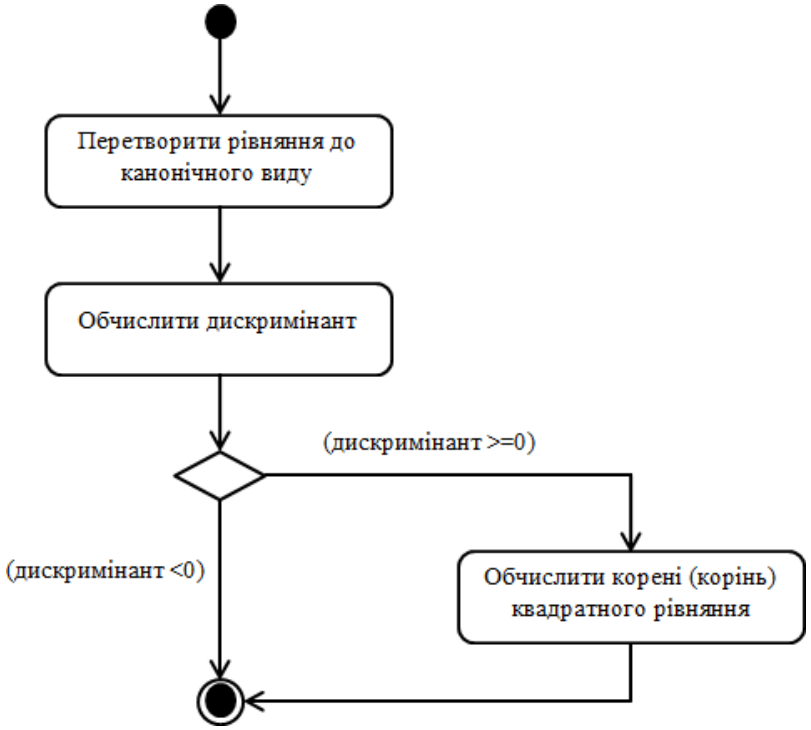

Рисунок 2.6 − Фрагмент діаграми діяльності для алгоритма визначення коренів квадратного рівняння

Кожен з переходів, що виходять зі стану "Обчислити дискримінант", має умову, що визначає єдину гілку, по якій може бути продовжено процес обчислення коренів в залежності від знака дискримінанта.

У мові UML використовується спеціальний символ для розділення і злиття паралельних обчислень або потоків управління (рис. 2.7).

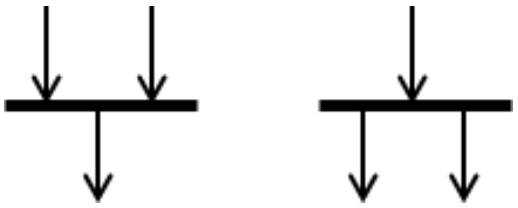

Рисунок 2.7 − Графічне зображення управління паралельними потоками

Діаграми використовують не тільки для специфікації алгоритмів обчислень або потоків управління в програмних системах. Не менш важлива область їх застосування пов'язана з моделюванням бізнес-процесів. Для моделювання цих особливостей в мові UML використовується спеціальна конструкція, яка отримала назву доріжки (swimlanes). Мається на увазі візуальна аналогія з плавальними доріжками в басейні, якщо дивитися на відповідну діаграму. При цьому всі стани дії на діаграмі діяльності діляться на окремі групи, які відокремлюються одна від одної вертикальними лініями. Дві сусідні лінії і утворюють доріжку, а група станів між цими лініями виконується окремим підрозділом (відділом, групою, відділенням, філією) компанії (рис. 2.8).

У загальному випадку дії на діаграмі діяльності виконуються над тими чи іншими об'єктами. Ці об'єкти або ініціюють виконання дій, або визначають деякий результат цих дій. При цьому дії специфікують виклики, які передаються від одного об'єкта графа діяльності до іншого. Оскільки в такому ракурсі об'єкти грають певну роль в розумінні процесу діяльності, іноді виникає необхідність явно вказати їх на діаграмі діяльності.

Для графічного представлення об'єктів використовуються прямокутник класу, з тією відмінністю, що ім'я об'єкта підкреслюється. Далі після імені може вказуватися характеристика стану об'єкта в прямих дужках. Такі прямокутники об'єктів приєднуються до станів дії пунктирною лінією зі стрілкою. Відповідна залежність визначає стан конкретного об'єкта після виконання попередньої дії.

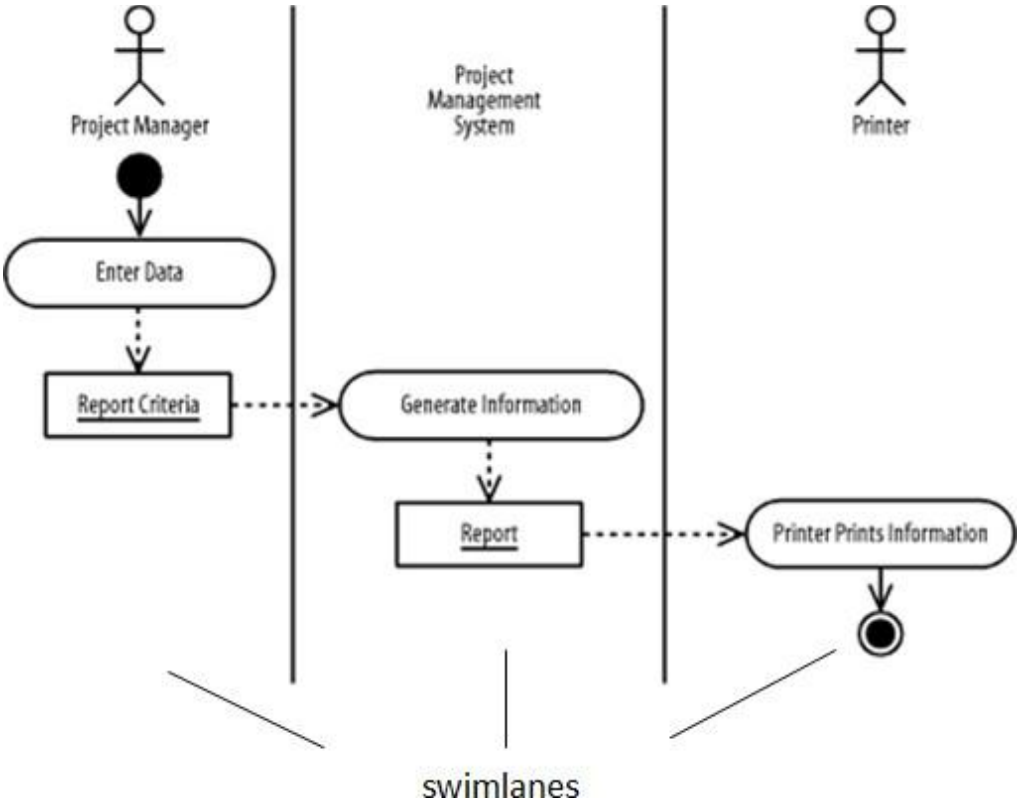

Рисунок 2.8 – Варіант діаграми діяльності з доріжками

3 ІНСТРУМЕНТАЛЬНІ ЗАСОБИ ПРОЕКТУВАННЯ. ТЕХНОЛОГІЯ SAP

3.1 Технологія впровадження CASE-засобів

Для успішної реалізації проекту об'єкт проектування (ІС) повинен бути перш за все адекватно описаний, повинні бути побудовані повні і несуперечливі функціональні та інформаційні моделі ІС. Накопичений до теперішнього часу досвід проектування ІС показує, що це логічно складна, трудомістка і тривала за часом робота, що вимагає високої кваліфікації фахівців, які беруть участь в ній. Однак до недавнього часу проектування ІС виконувалося в основному на інтуїтивному рівні із застосуванням неформалізованих методів, заснованих на практичному досвіді, експертних оцінках і дорогих експериментальних перевірках якості функціонування ІС. Крім того, в процесі створення і функціонування ІС інформаційні потреби користувачів можуть змінюватися чи уточнюватися, що ще більше ускладнює розробку і супровід таких систем.

Все це сприяло появі програмно-технологічних засобів спеціального класу – CASE-засобів, які реалізують CASE-технологію створення і супроводу ІС. Термін CASE (Computer Aided Software Engineering) використовується в даний час в досить широкому сенсі. Первісне значення терміна CASE, що обмежене питаннями автоматизації розробки лише програмного забезпечення (ПЗ), в даний час набуло нового змісту, який охоплює процес розробки складних ІС в цілому. Тепер під терміном CASE-засобу розуміються програмні засоби, що підтримують процеси створення і супроводу ІС, що включають аналіз і формулювання вимог, проектування прикладного ПЗ і баз даних, генерацію коду, тестування, документування, забезпечення якості, конфігураційне управління і управління проектом, а також інші процеси. CASE-засоби разом із системним ПЗ і технічними засобами утворюють повну середу розробки ІС.

CASE-технологія є методологію проектування ІС, а також набором інструментальних засобів, які дозволяють у наочній формі моделювати предметну область, аналізувати цю модель на всіх етапах розробки і супроводу ІС і розробляти додатки відповідно до інформаційних потреб користувачів. Більшість існуючих CASE-засобів засновано на методологіях структурного або об'єктно-орієнтованого аналізу і проектування, використовують специфікації у вигляді діаграм або текстів для опису

зовнішніх вимог, зв'язків між моделями системи, динаміки поведінки системи та архітектури програмних засобів (рис. 3.1).

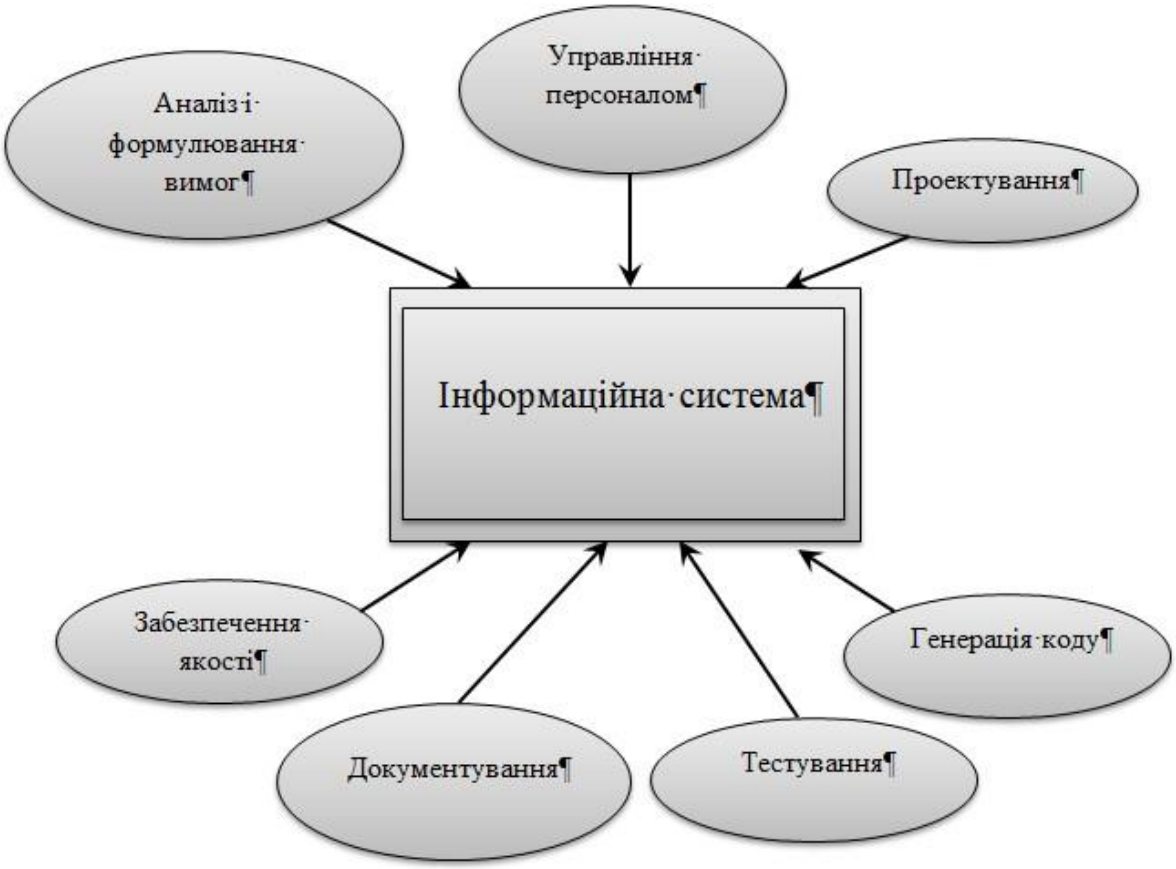

Рисунок 3.1 – Схема використання CASE-засобів

Основна мета CASE полягає в тому, щоб відокремити початкові етапи (аналіз і проектування) від наступних етапів розробки, а також не обтяжувати розробників усіма деталями середовища розробки і функціонування системи. Чим більший обсяг робіт буде винесено на етапи аналізу і проектування, тим краще. У більшості сучасних CASE-систем застосовуються методології структурного або об'єктно-орієнтованого аналізу і проектування, що засновані на наочних діаграмах, при цьому для опису моделі використовуються графи, діаграми, таблиці та схеми.

При використанні CASE змінюються всі фази ЖЦ, при цьому найбільші зміни стосуються фаз аналізу і проектування. У табл. 3.1 наведені оцінки трудовитрат за фазами ЖЦ, в табл. 3.2 зведені основні зміни в ЖЦ при використанні CASE в порівнянні з традиційної розробкою.

Таблиця 3.1 – Переваги використання CASE-засобів

Спосіб розробки	**Аналіз, %**	**Проекту-вання, %**	**Коду-вання, %**	**Тесту-вання, %**
Традиційна розробка	20	15	20	45
Використання структурної методології	30	30	15	25
Використання CASE-технологій	40	40	5	15

Таблиця 3.2 – Зміни в життєвому циклі інформаційної системи

Традиційна розробка	**Використання CASE-технології**
Основні ресурси направлені на кодування і тестування	Основні ресурси направлені на аналіз і проектування
Паперові специфікації	Швидке ітеративне прототипування
Ручне кодування	Автоматична кодогенерація
Ручне документування	Автоматична генерація документації
Тестування кодів	Автоматичний контроль проекту
Супровід кодів	Супровід специфікацій проектування

Найбільш трудомісткими етапами розробки ІС є етапи аналізу та проектування, в процесі яких CASE-засоби забезпечують якість прийнятих технічних рішень і підготовку проектної документації. При цьому велику роль відіграють методи візуального подання інформації. Це передбачає побудову структурних чи інших діаграм у реальному масштабі часу, використання різноманітної кольорової палітри, наскрізну перевірку синтаксичних правил. Графічні засоби моделювання предметної області дозволяють розробникам в наочному вигляді вивчати існуючу ІС, перебудовувати їх у відповідності з поставленими цілями і наявними обмеженнями.

До CASE-засобів відносять будь-яке програмний засіб, який автоматизує ту чи іншу сукупність процесів життєвого циклу ПЗ і володіє наступними характерними рисами:

- потужні графічні засоби для опису і документування ІС, які забезпечують зручний інтерфейс з розробником і розвивають його творчі можливості;
- інтеграція окремих компонентів CASE-засобів, яка забезпечує керованість процесом розробки ІС;
- використання спеціальним засобом організованого сховища проектних метаданих (репозиторія).

Інтегрований CASE-засіб (або комплекс засобів, які підтримують повний ЖЦ ПЗ) містить наступні компоненти:

репозиторій, що є основою CASE-засобу, він повинен забезпечувати зберігання версій проекту і його окремих компонентів, синхронізацію надходження інформації від різних розробників при груповий розробці, контроль метаданих на повноту і несуперечливість;

- графічні засоби аналізу і проектування, забезпечують створення і редагування ієрархічно пов'язаних діаграм (DFD, ERD та ін.), що утворюють моделі ІС;
- засоби розробки додатків, генератори кодів;
- засоби конфігураційного управління;
- засоби документування;
- засоби тестування;
- засоби управління проектом;
- засоби реінжинірингу.

Всі сучасні CASE-засоби класифікуються за типами та категоріями. Класифікація за типами відбиває функціональну орієнтацію CASE-засобів на

ті чи інші процеси ЖЦ. Класифікація за категоріям визначає ступінь інтегрованості з урахуванням функцій і включає окремі локальні засоби, які вирішують невеликі автономні завдання (tools), набір частково інтегрованих засобів, що охоплюють більшість етапів життєвого циклу ІС (toolkit) і повністю інтегровані засоби, що підтримують весь ЖЦ ІС і пов'язані спільним репозиторієм. Крім цього, CASE-засоби можна класифікувати за такими ознаками:

- застосування методологій і моделей систем і БД;
- ступінь інтегрованості з СУБД;
- доступні платформи.

Класифікація за типами переважно збігається з компонентним складом CASE-засобів і включає наступні основні типи:

- засоби аналізу, які призначені для побудови і аналізу моделей предметної області (Design/IDEF, BPwin, ARIS);
- засоби аналізу та проектування, що підтримують найбільш поширені методології проектування, які використовуються для створення проектних специфікацій (Vantage Team Builder, Designer/2000, Silverrun, PRO-IV, CA8E.Аналітік, ARIS). Готовим продуктом таких засобів є специфікації компонентів і інтерфейсів системи, архітектури системи, алгоритмів і структур даних;
- засоби проектування баз даних, що забезпечують моделювання даних і генерацію схем баз даних для найбільш поширених СУБД. До них відносяться ERwin, S-Designor і DataBase Designer, ARIS Toolset;
- засоби розробки додатків, до них відносяться засоби 4GL (Uniface, JAM, PowerBuilder, Developer/2000, SQL Windows, Delphi та ін.), а також генератори кодів, що входять до складу Vantage Team Builder, PRO-IV, ARIS Easy Design;
- засоби реінжинірингу, що забезпечують аналіз програмних кодів і схем баз даних і формування на їх основі різних моделей і проектних специфікацій; засоби аналізу схем БД і формування ERD входять до складу Vantage Team Builder, PRO-IV, Silverrun, Designer/2000, ARIS Toolset, ERwin і S-Designor; в області аналізу програмних кодів найбільшого поширення отримують об'єктно-орієнтовані CASE-засоби, що забезпечують реінжиніринг програм на мові С ++ (Rational Rose, Object Team).

Допоміжні типи включають:

- засоби планування і управління проектом (SE Companion, Microsoft Project та ін.);
- засоби конфігураційного управління (PVCS);
- засоби тестування (Quality Works);
- засоби документування (SoDA).

CASE-засоби охоплюють різноманітну діяльність, від аналізу бізнес-структур і бізнес-вимог до підтримки життєвого циклу розробки і супроводу інформаційних систем, і є нерозривним зв'язком систем управління організаціями і ІС. У вузькому сенсі CASE-засоби – це засоби візуального моделювання, а в широкому – засоби, що максимально автоматизують всі процеси життєвого циклу проекту розробки та реалізації (як в організаційній, так і в програмній інженерії). Крім того, це засоби правильного ведення робіт, що підтримують комунікації учасників проекту на різних етапах і з різних позицій (як між командами замовника і розробника, так і всередині робочої групи).

Моделі, що використовуються в CASE-засобах (візуальна складова CASE-інструментів) – це спільна мова для всіх учасників процесу, що забезпечує можливість задавати ті чи інші аспекти предметної області за допомогою загальної термінології, загальних графічних зображень (нотацій). Завдання проектування ІС і завдання оптимізації бізнес-процесів дуже тісно пов’язані між собою. Тому все має бути пов'язано методологічно, а ще краще – технологічно (це може бути єдина CASE-система, може бути кілька інтегрованих CASE-інструментів).

Сьогодні важливі не тільки зручність і швидкість роботи в тому чи іншому середовищі розробки. На перший план виходять аспекти забезпечення якості готових програмних продуктів, ступінь їх документованості (як з точки зору користувачів, так і ІТ-фахівців та розробників), легкість супроводу і, звичайно, можливість розширення їх функціональності відповідно до вимог користувачів.

Якщо метою роботи з CASE-засобами є програмний продукт, то він, як правило, створюється на деякій мові програмування в певному середовищі розробки. Такі можливості CASE-засобів, як автоматичне створення коду і зворотний процес – побудова діаграм на підставі вихідного коду, методи аналізу якості коду, вимагають сумісності додатку, мов і середовища програмування на рівні компілятора цієї мови. CASE-інструментарій забезпечує розуміння і коректну взаємодію аналітиків, що визначають вимоги бізнесу (описують бізнес-процеси), і розробників, що відповідають за

структуру даних і об'єктно-орієнтований аналіз, проектування та програмування.

Існує більше 20 технологій проектування інформаційних систем і кілька сотень інструментів, що призначені для автоматизації цього процесу. Порівняльний аналіз найбільш популярних засобів проектування: BPwin/ERwin, Rational Rose і ARIS наведений в табл. 3.3 (1 – підтримується, 0,5 – частково підтримується, 0 – не підтримується).

Таблиця 3.3 – Порівняльний аналіз інструментальних засобів

Характеристика	**Rational Rose**	**ARIS**	**BPwin/ ERwin**
1	**2**	**3**	**4**
Можливість використання ПЗ цього класу для різних категорій користувачів			
Спеціалісти з організаційного управління (бізнес-аналітики, бізнес-проектувальники)	0,5	1	1
Розробники ІС (розробники завдань для програмування, бізнес-аналітики)	1	1	1
Системні архітектори ІС	1	0,5	1
Програмісти ІС	1	0	1
Менеджери та керівники проектів	1	1	1
Склад та функціональність продуктів (повнота за життєвим циклом)			
Управління вимогами	1	1	0,5
Моделювання			
Моделі даних (ERD, логічні, фізичні)	1	1	1
Модклі процесів, функцій, робіт (BPM)	1	1	1
Моделі ООП (UML та ін.)	1	1	1
Оргструктури, потоки та інші класи діаграм та моделей предметної області	0	1	1

Характеристика	Rational Rose	ARIS	BPwin/ ERwin
1	**2**	**3**	**4**
Перевірка моделей	1	1	1
Аналіз (динамічний та вартісний аналіз процесних моделей)	1	1	1
Розробка технічного завдання	1	0,5	0,5
Оптимізація бізнес-процесів	1	0	0
Імітаційне моделювання, подієво-кероване моделювання	1	0,5	0
Генерація кода додатку	0	1	0,5
Зберігання моделей діяльності підприємств	1	0,5	0,5
Документування результатів моделювання, аналіз моделей	1	1	1
Проектування, розробка технічних аспектів ІС (архітектура, модулі, екранні форми)	1	0,5	1
Створення ІС на основі моделей (формування БД, коду, екраних форм)	1	0	1
Тестування	1	0	0,5
Обратний зв'язок кода з моделями ІС	1	0	1
Документування ІС	1	1	0
Автоматизоване навчання користувачів ІС	0	1	0
Автоматизований аудит ІС після впровадження	0	1	1
Зв'язок процедурних моделей з WorkFlow-системами	0	1	0

Характеристика	Rational Rose	ARIS	BPwin/ ERwin
1	**2**	**3**	**4**
Управління груповою роботою			
Можливість роботи з моделями та підсистемами декількома користувачами	1	1	1
Управління взаємодією користувачів (сайт, повідомлення, пошта, форуми, конференції)	1	1	1
Бібліотеки термінів, моделей, об’єктів, додатків, модулей, компонентів, готових рішень, функцій та ін.	1	1	1
Методологія ведення робіт	1	1	0,5
Засоби управління бізнес-процесами та документообігом	1	1	1
Управління проектом розробки	0,5	0,5	1
Розмежування доступу користувачів системи	1	1	1

На підставі даних таблиці 3.3 можна зробити висновок про те, що жоден із засобів проектування не задовольняє повною мірою вимогам комплексного проектування ІС з метою реінжинірингу бізнес-процесів. Наприклад, при проектуванні ІС для невеликих організацій з метою регламентації і документування робіт, а також реінжинірингу може бути використано засіб BPwin. На великих підприємствах при проведенні реінжинірингу з подальшим впровадженням ІС буде краще ARIS, що має більш розширені можливості.

Існують допоміжні засоби підтримки життєвого циклу програмного забезпечення, к ним відносять:

1. ***Засоби планування та управління проектом***. Найбільш поширеним засобом планування і управління процесом проектування ІС є Microsoft Project.

2. ***Засоби конфігураційного управління***. Мета конфігураційного управління (КУ) – забезпечити керованість і контрольованість процесів розробки і супроводу ПЗ. Для цього необхідна точна і достовірна інформація про стан ПЗ і його компонент в кожен момент часу, а також про всі передбачувані і виконані зміни. Для вирішення завдань КУ застосовуються методи і засоби, які забезпечують ідентифікацію стану компонентів, облік номенклатури всіх компонентів і модифікацій системи в цілому, контроль над змінами, що вносяться в компоненти, структуру системи і її функції, а також координоване управління розвитком функцій і поліпшенням характеристик системи.

3. ***Засоби документування***. Для створення документації в процесі розробки ІС використовуються різноманітні засоби формування звітів, а також компоненти видавничих систем. Зазвичай кошти документування вбудовані в конкретні CASE-засоби. Винятком є деякі пакети, що надають додатковий сервіс при документуванні. З них найбільш активно використовується SoDA (Software Document Automation).

4. ***Засоби тестування***. Під тестуванням розуміється процес виконання програми з метою виявлення помилок. Регресійне тестування – це тестування, що проводиться після удосконалення функцій програми або внесення до неї змін. Одне з найбільш розвинених засобів тестування QA є інтегрований набір інструментів для розробки автоматизованих тестів будь-якого рівня, включаючи тести регресії для додатків з графічним інтерфейсом користувача. QA дозволяє починати тестування на будь-якій фазі ЖЦ, планувати процес тестування і керувати ним, відображати зміни в додатку і повторно використовувати тести для більш ніж 25 різних платформ.

Основними компонентами QA є:

- QA Partner – середовище для розробки, компіляції та виконання тестів;
- QA Planner – модуль для розробки планів тестування і обробки результатів; для створення і виконання тестів в процесі роботи QA Planner викликається QA Partner;
- Agent – модуль, що підтримує роботу в мережі.

Процес тестування складається з декількох етапів, таких як:

- створення плану тестування;
- зв'язування плану з тестами;
- позначка та виконання тестів;
- отримання звітів про тестування і управління результатами.

Процес впровадження CASE-засобів складається з наступних етапів:

- визначення потреб у CASE-засобах;
- оцінка та вибір CASE-засобів;
- виконання пілотного проекту;
- практичне впровадження CASE-засобів.

Визначення потреб в CASE-засобах. Даний етап включає досягнення розуміння потреб організації і технології подальшого процесу впровадження CASE-засобів. Він повинен привести до виділення тих областей діяльності організації, в яких застосування CASE-засобів може бути дісно корисним. Результатом даного етапу є документ, що визначає стратегію впровадження CASE-засобів.

Аналіз можливостей організації. Першою дією даного етапу є аналіз можливостей організації щодо її технологічної бази, персоналу та існуючого ПЗ. Такий аналіз може бути формальним або неформальним. Формальні підходи визначаються моделлю оцінки зрілості технологічних процесів організації CMM (Capability Maturity Model), розробленої SEI (Software Engineering Institute), а також стандартами ISO 9001: 1994, ISO 9003-3: 1991 і ISO 9004-2: 1991. У центрі уваги цих підходів знаходиться аналіз різних аспектів процесів, що відбуваються в організації. Для отримання інформації щодо стану та потреб організації можуть використовуватися неформальні оцінки та анкетування. Питання, що включаються в анкету, повинні бути керівництвом зі збору інформації, необхідної для визначення ступеня готовності організації до впровадження CASE-технології.

Оцінка готовності організації до впровадження CASE-технології повинна бути відвертою і ретельною, оскільки в разі відсутності такої готовності всі зусилля по впровадженню будуть марними.

Визначення організаційних потреб. Організаційні потреби походять безпосередньо з проблем організації і цілей, яких вона прагне досягти. Проблеми та цілі можуть бути пов'язані з управлінням, виробництвом продукції, економікою, персоналом або технологією. Визначення потреб повинно виконуватися в поєднанні з аналізом ринку CASE-засобів, оскільки інформація про технології, які доступні на ринку в певний момент часу, може вплинути на потреби.

Визначення потреб організації, що пов'язані з використанням CASE-технології, включає аналіз цілей і існуючих можливостей. Після того як всі потреби організації визначені, кожній з них повинен бути присвоєно певний пріоритет, що відображає її значимість для успішної діяльності організації.

Якщо потреби, що пов'язані з CASE-технологією, не володіють вищим пріоритетом, має сенс відмовитися від її впровадження і зосередитися на потребах із найвищим пріоритетом. Має чітко простежуватися вплив кожної функції або можливості на задоволення конкретних потреб. Результатом даної дії є формулювання потреб з їх пріоритетами, яка використовується на етапі оцінки і вибору в якості «призначених для користувача потреб ».

Метою процесу ***оцінки і вибору CASE-засобів*** є визначення функціональності і якості CASE-засобів для подальшого вибору. Оцінка виконується поступово за кожним конкретним критерієм, її результати включають як об'єктивні, так і суб'єктивні дані по кожному засобу.

Процес оцінки включає наступні дії:

- формулювання завдання оцінки;
- визначення критеріїв оцінки, що походять з визначення завдання;
- визначення засобів-кандидатів шляхом перегляду списку кандидатів і аналізу інформації про конкретні засоби;
- оцінка засобів-кандидатів в контексті обраних критеріїв; необхідні для цього дані можуть бути отримані шляхом аналізу самих засобів і їх документації, опитування користувачів, роботи з демо-версіями, виконання тестових прикладів, експериментального застосування засобів і аналізу результатів попередніх оцінок;
- підготовка звіту за результатами оцінки.

Процеси оцінки і вибору тісно пов'язані. Коли аналізуються остаточні результати оцінки і до них застосовуються критерії вибору, може бути рекомендовано придбання CASE-засобу або набору CASE-засобів. Альтернативою може бути відсутність адекватних CASE-засобів, в цьому випадку рекомендується розробити новий CASE-засіб, модифікувати існуючий або відмовитися від впровадження.

Пілотний проект є початковим реальним використанням CASE-засобу в призначеному для цього середовищі і зазвичай має на увазі більш широкий масштаб використання CASE-засобу по відношенню до того, який був досягнутий під час оцінки. Пілотний проект повинен відповідати високим вимогам характеристик реальних проектів, для яких призначений даний засіб. Він має наступні цілі:

- підтвердити достовірність результатів оцінки та вибору;
- визначити, чи дійсно CASE-засіб придатний для використання в даній організації, і якщо так, то визначити приоритетну область його застосування;

- зібрати інформацію, яка необхідна для розробки плану практичного впровадження;
- придбати власний досвід використання CASE-засобів.

Важливою функцією пілотного проекту є прийняття рішення щодо придбання або відмови від використання CASE-засобу.

Процес переходу до ***практичного використання CASE-засобів*** починається з розробки та подальшої реалізації плану переходу. Цей план може відображати поетапний підхід до переходу, починаючи з ретельно обраного пілотного проекту до проектів з різноманітними характеристиками. План переходу повинен включати наступне:

- інформацію щодо цілей, критеріїв оцінки, можливих ризиків, пов'язаних з реалізацією плану;
- інформацію щодо придбання, установки та настройки CASE-засобів;
- інформацію щодо інтеграції кожного засобу з існуючими засобами, включаючи як інтеграцію CASE-засобів один з одним, так і їх інтеграцію в процеси розробки та експлуатації ПЗ, існуючі в організації;
- очікувані потреби в навчанні і ресурси, що використовують протягом і після завершення процесу переходу;
- визначення стандартних процедур використання засобів.

Результатом даного етапу є впровадження CASE-засобів в повсякденну практику організації, при цьому більше не потрібно будь-якого спеціального планування. Крім того, підтримка CASE-засобів включається в план поточної підтримки ПЗ в даній організації.

3.2 Технологія SAP

Прогроамне забезпечення компанії SAP SE є лідером ринку в області ERP (англ. Enterprise Resource Planning, планирование ресурсов предприятия) рішень і послуг. Планування ресурсів підприємства (ERP) являє собою програмне забезпечення, яке будується для організацій, що належать до різних галузей промисловості, незалежно від їх розміру та міцності.

Пакет ERP використовується для підтримки і інтеграції практично в усіх функціональних областях бізнес-процесу, таких як закупівлі товарів і послуг, продажу та розподілу, фінансів, accountings, людських ресурсів,

виробництва, планування виробництва, логістики та управління складом (рис. 3.2).

Рисунок 3.2 – Модель програмного забезпечення процесу реінжинірингу

Кожне підприємство, незалежно від галузі, вимагає наявність системи, якиа забезпечує ефективність обробки потоків інформації, що рухаються від одного бізнес-процесу до іншого. Business Process Integration (BPI) виконує функції автоматизації бізнес-процесів, інтеграції системи і послуг, забезпечує обмін даними між численними додатками та системою автоматизації оперативного управління, підтримки процесу. На рис. 3.3 показано бізнес-процеси, які працюють на підприємстві та іхня інтеграція в загальну систему.

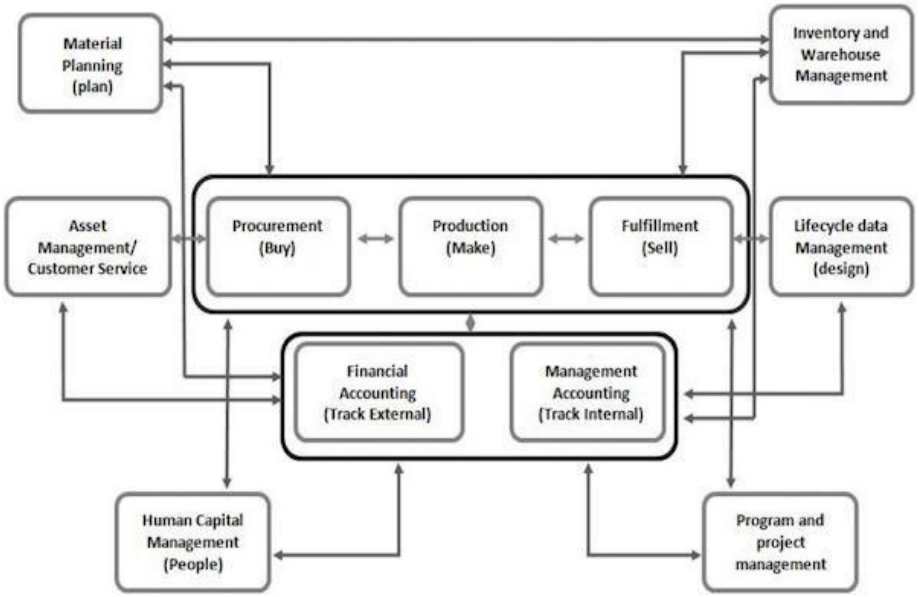

Рисунок 3.3 – Модель інтеграції бізнес-процесів

Система ERP, як правило, виконує наступні функції:

1) Підтримка інтегрованого бізнес-процесу всередині організації.

2) Підвищення ефективності планування капітальних витрат і допомога в здійсненні організаційних планів і стратегій.

3) Забезпечення прискорення процесу прийняття рішень з приводу аналізу точних даних.

4) Розширення бізнес-мережі для більш широких областей, розширення продуктів і послуг з метою залучення нових клієнтів, постачальників і партнерів.

5) Визначення операційних ризиків, підвищення ефективності управління.

6) Забезпечення захисту організаційних даних від порушень і загроз безпеки витоку інформації.

7) Адаптація організації до швидких змін в бізнес-процесах відповідно до потреб.

8) Забезпечення довгострокового прибутку за рахунок надходження коштів для збільшення клієнтської бази.

ERP є програмним забезпеченням для управління бізнесом. Це набір інтегрованих програм, які компанія може використовувати для збору, зберігання, управління і інтерпретування даних, що надходять з багатьох функціональних областей, в тому числі:

1) Фінансовий облік – обробка інформації з фінансовими операціями і даними.

2) Людський ресурс – робота з інформацією, що пов'язана з працівниками компанії (організації).

3) Управління взаємовідносинами з клієнтами.

4) Збут – розміщення замовлення, постачання, відвантаження та виставлення рахунків-фактур.

5) Логістика і управління складом – зберігання продукції і відвантаження.

6) Виробництво і управління матеріальними потоками виробничої діяльності.

7) Business Intelligence – аналіз даних і перетворення інформації.

Рішення SAP включають в себе ряд функціональних модулів (рис. 3.4), які підтримують транзакції для виконання наступних ключових бізнес-процесів:

1) FI – фінансовий облік;
2) FSCM – фінансове управління ланцюгами поставок;
3) CO – модуль забезпечення контролю;
4) MM – управління матеріальними потоками;
5) SD – збут;
6) LE – оперативна логістика;
7) PP – планування виробництва;
8) QM – управління якістю;
9) PM – технічне обслуговування та ремонт обладнання;
10) PS – система проектів;
11) HR – управління персоналом.

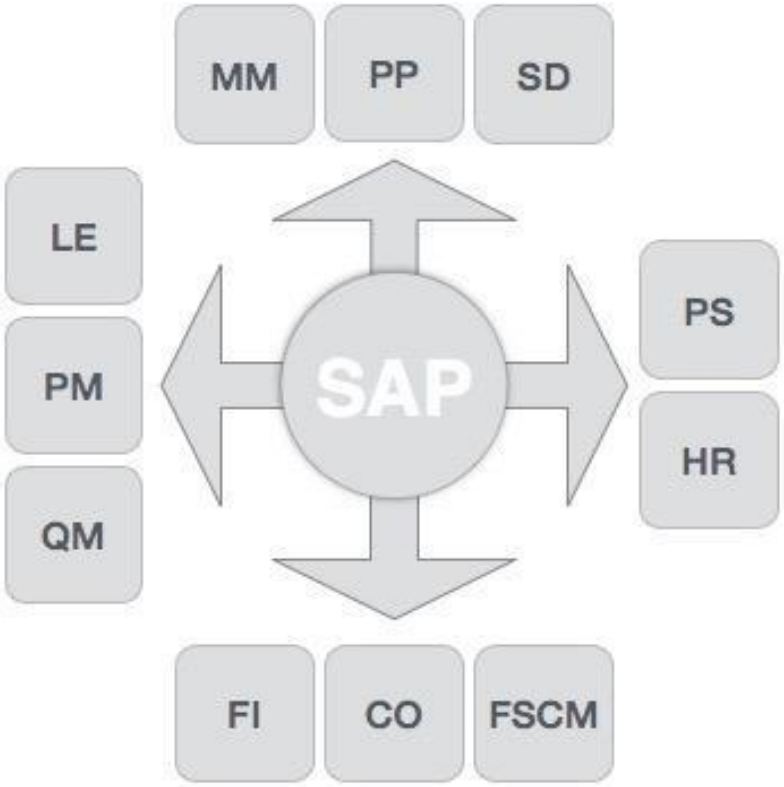

Рисунок 3.4 – Структурна схема модулів SAP

Розглянемо архітектуру і графічний інтерфейс користувача SAP. Корпоративне програмне забезпечення SAP нового покоління представлено різними архітектурами, від мейнфреймів обчислень (клієнт-серверної архітектури) до трирівневої архітектури бази даних, додатік і інтерфейсу користувача, схема другої архітектури представлена на рис. 3.5.

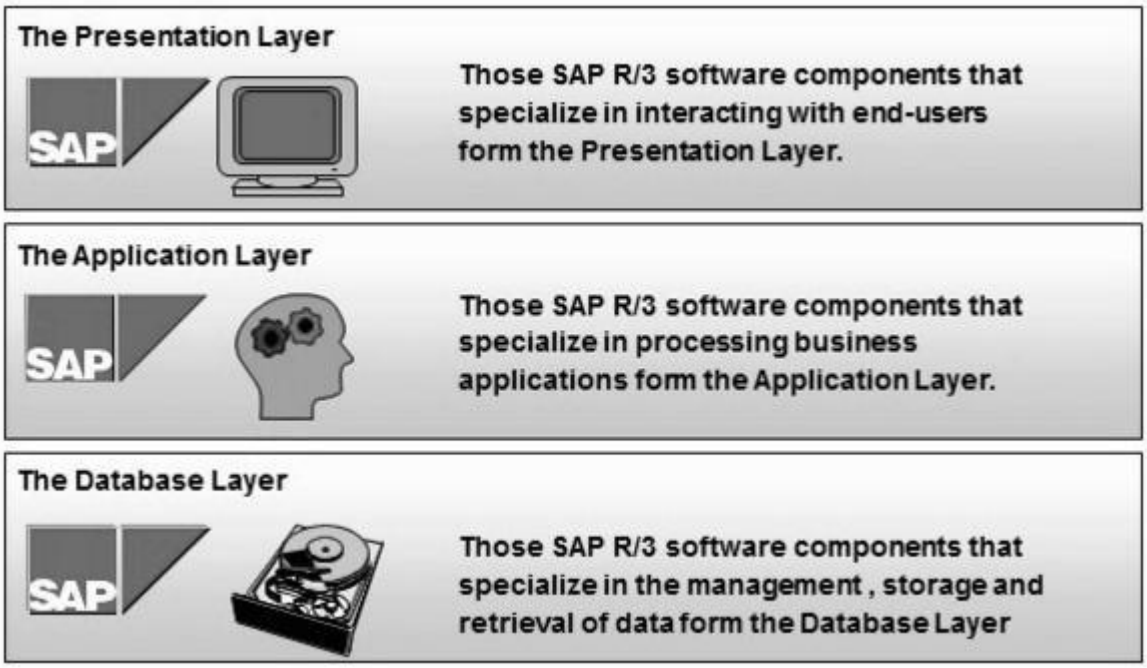

Рисунок 3.5 – Трирівнева архітектура SAP R/3

Клієнт є логічною частиною фізичної бази даних в SAP R/3. З точки зору бізнесу, клієнт може бути предсьавлен як логічна група компаній (рис. 3.6).

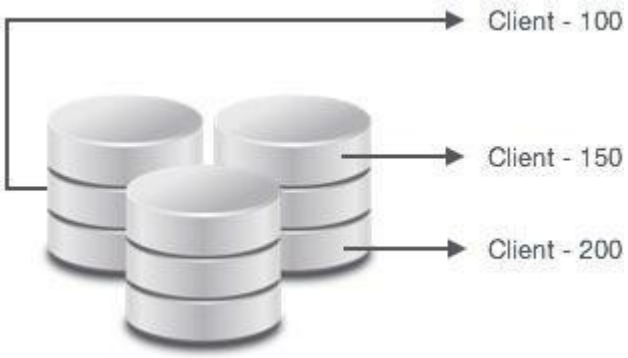

Рисунок 3.6 – Схема взаємодії клієнтів в SAP R/3

Всі функції налаштування, конфігурування і розширення (ABAP) в системі SAP R/3 виконуються на боці клієнта. Дані в кожному клієнті можуть бути відокремлені від інших клієнтів. Є два основних типи даних в системі SAP R/3 – клієнт-залежні і клієнт-незалежні дані. При цьому дані налаштування можуть бути збережені в межах окремого клієнта (клієнт-залежні) або для всіх клієнтів (клієнт-незалежні) з урахуванням потреб самої системи.

Клієнт-залежні дані визначаються як специфічні дані для індивідуального клієнта. Наприклад, діапазони номерів, варіанти ABAP, налаштування майстрів користувачів, а також дані, створений або оновлений через SAP R/3 угоди. Клієнт-незалежні дані визначаються як дані, що містяться у всіх клієнтів в системі. Наприклад, словники даних об'єктів (таблиць, уявлень), вихідний код ABAP, екрани і меню. Дані зберігаються в таблицях. Для того, щоб визначити, чи є конкретна таблиця клієнт-залежною або клієнт-незалежною, необхідно переглянути структуру таблиці за допомогою словника даних (SE11). Якщо першим ключовим полем таблиці є MANDT, то таблиця є клієнт-залежною, в іншому випадку, таблиця клієнт-незалежна.

Кожена система SAP R/3 містить три клієнта 000, 001 і 066 (рис. 3.7). Ці клієнти забезпечують різні функції і не повинні бути видалені. Клієнт 000 виконує спеціальні функції розширення функціональних можливостей під час оновлення. Клієнт 001 є копією 000 і може бути використаний в якості основи для нового клієнта в процесі налаштування користувача. Клієнт 066 є спеціальним клієнтом для оперативного моніторингу системи. Він використовується в SAP R/3 для раннього спостереження в процесах контролю продуктивності.

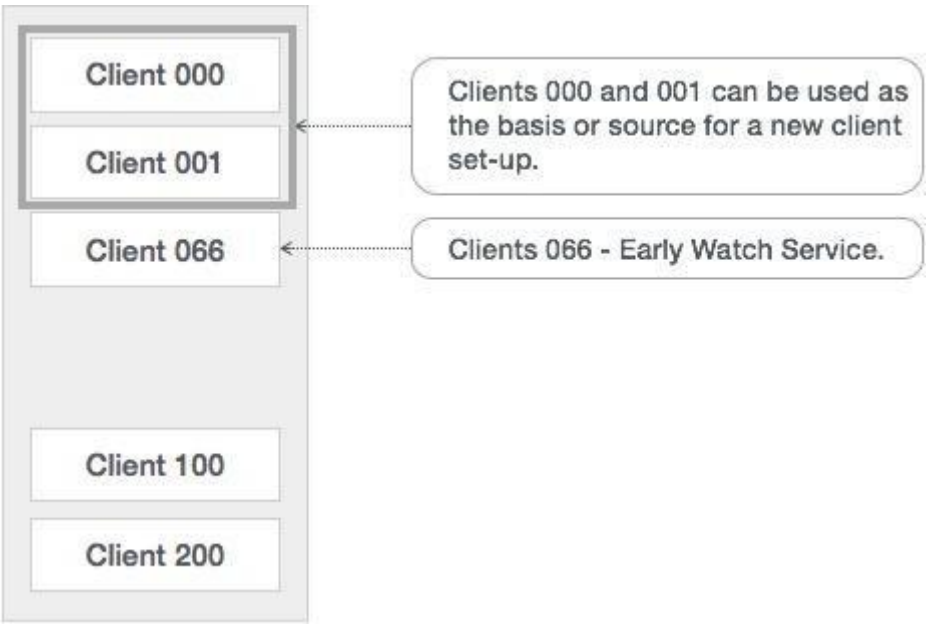

Рисунок 3.7 – Структурна схема клієнтів в SAP R/3

В різних версіях системи клієнти мають різне наповнення. Наприклад, у версіях до 3.0 клієнт 000 містить модель компанії, у версії 4.0 клієнти 000 і

001 ідентичні. В останніх версіях клієнт 000 більше не містить модель компанії і може бути використаний в якості основи для конфігурації за допомогою копії клієнта. Більшість проектів починають з копіювання клієнта 000, щоб почати будувати конфігурації. Але слід пам’ятати, що робота з клієнтами ніколи не повинна проходити в трьох клієнтах одночасно.

Система SAP R/3 досить вимоглива до конфігурації комп’ютерного обладнання. Вона працює під управлінням Windows 7, Windows Vista або Windows XP (Service Pack 3). Для комп’ютерів з Apple Mac необхідно встановити програмне забезпеченням Virtual Machine (VMWare, Fusion, Parallels) під керуванням Windows 7, Windows Vista або XP (Service Pack 3). Системна пам'ять (ОЗУ) для Windows XP має бути не менш 1 Гб (рекомендується 2 Гб), Windows 7 – не менш 2 Гб (рекомендується 4 Гб). Для установника програми SAP GUI має бути 145 МБ вільного місця на диску, для повного додатку – 250 МБ. Необхідно додатково встановити програмне забезпечення Java Platform Enterprise Edition 7 SDK з ліцензійною угодою.

4 УПРАВЛІННЯ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

4.1 Поняття інфраструктури IT-підприємства

Під IT-інфраструктурою організації розуміється вся сукупність наявних у ній сервісів і систем, мереж, технічних і програмних засобів, даних, автоматизованих процесів.

Між різними частинами IT-інфраструктури існують численні взаємозв'язки: один процес може забезпечуватися декількома автоматизованими системами, системи можуть обмінюватися один з одним даними, системи більш низького рівня служать механізмами реалізації систем більш високого рівня та ін. При цьому взаємозв'язки можуть бути як явними, так і опосередкованими, але при цьому дуже важливими. Наприклад, якщо і в бухгалтерській, і в CRM-системах зберігаються дані про контрагентів, виникає проблема їх узгодження. Якщо автоматизована система отримує нормативно-довідкову інформацію з іншої системи по протоколу HTTP, то виходить, що її працездатність залежить від працездатності HTTP-сервера, який може формально не входити до складу жодної з цих систем.

Таким чином, IT-інфраструктура організації – це не просто набір IT-рішень, випадковим чином зібраних в одному місці. Вона являє собою велику (на порядки перевершує масштабом кожну зі своїх частин) інтегровану систему, що забезпечує діяльність організації в цілому.

Об’єкти, що є невідємною частиною IT-інфраструктури:

1) Сервера та робочі станції. Кількість та конфігурація цього обладнання визначається набором та вимогами програмного забезпечення, яке використовується в процесах управління інформаційними системами.

2) Перефірійне обладнання (наприклад, принери, багатофункціональні пристрої, факси та ін.).

3) Програмне забезпечення (системне та прикладне).

4) Мережі для передачі даних. Структурована кабельна мережа – це універсальна телекомунікаційна інфраструктура будівлі чи комплексу будівель, що забезпечує передачу сигналів усіх типів, включаючи мовні, інформаційні та відео. СКМ може бути встановлена перед тим, як стануть відомими вимоги користувачів, швидкість передачі даних, тип мережевих протоколів.

5) Засоби зв’язку.

Приклади різних видів IT-інфраструктур наведені на рис. 4.1 – рис. 4.3.

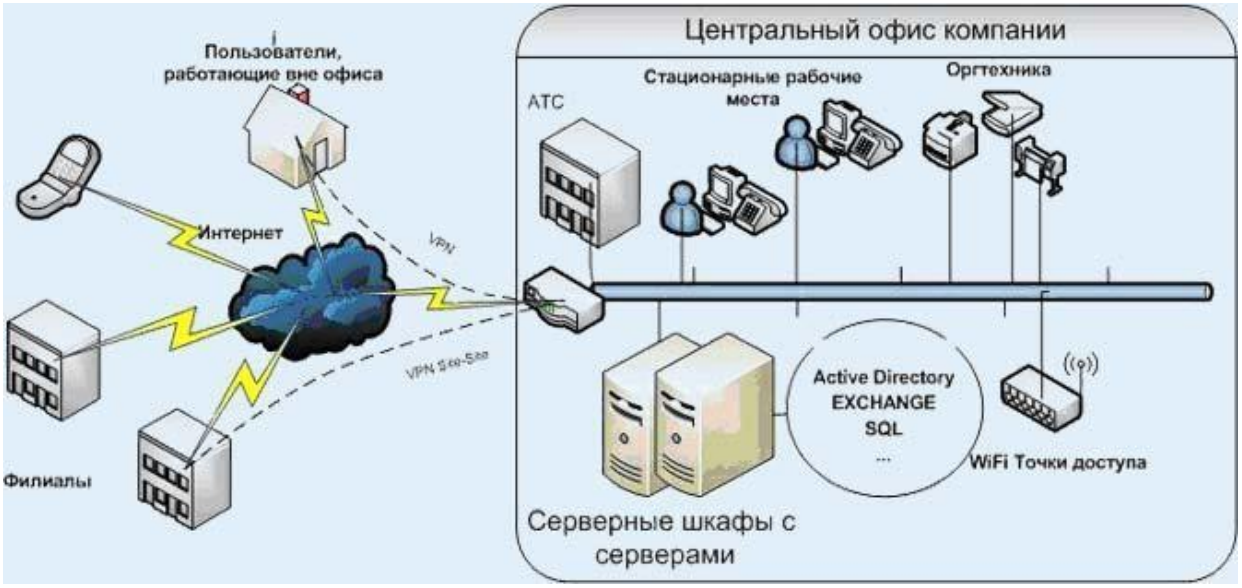

Рисунок 4.1 – Схема IT-інфраструктури підприємства
з розподіленою структурою

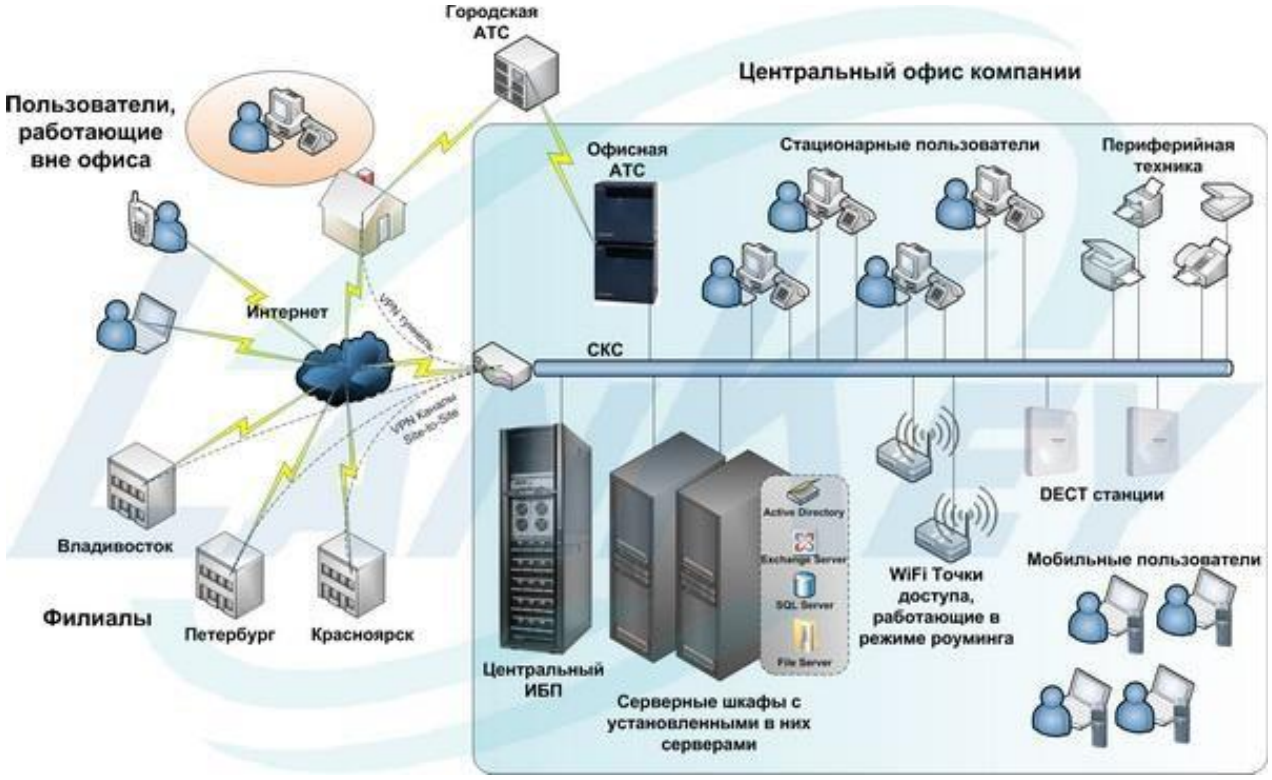

Рисунок 4.2 – Схема IT-інфраструктури підприємства
з надання послуг

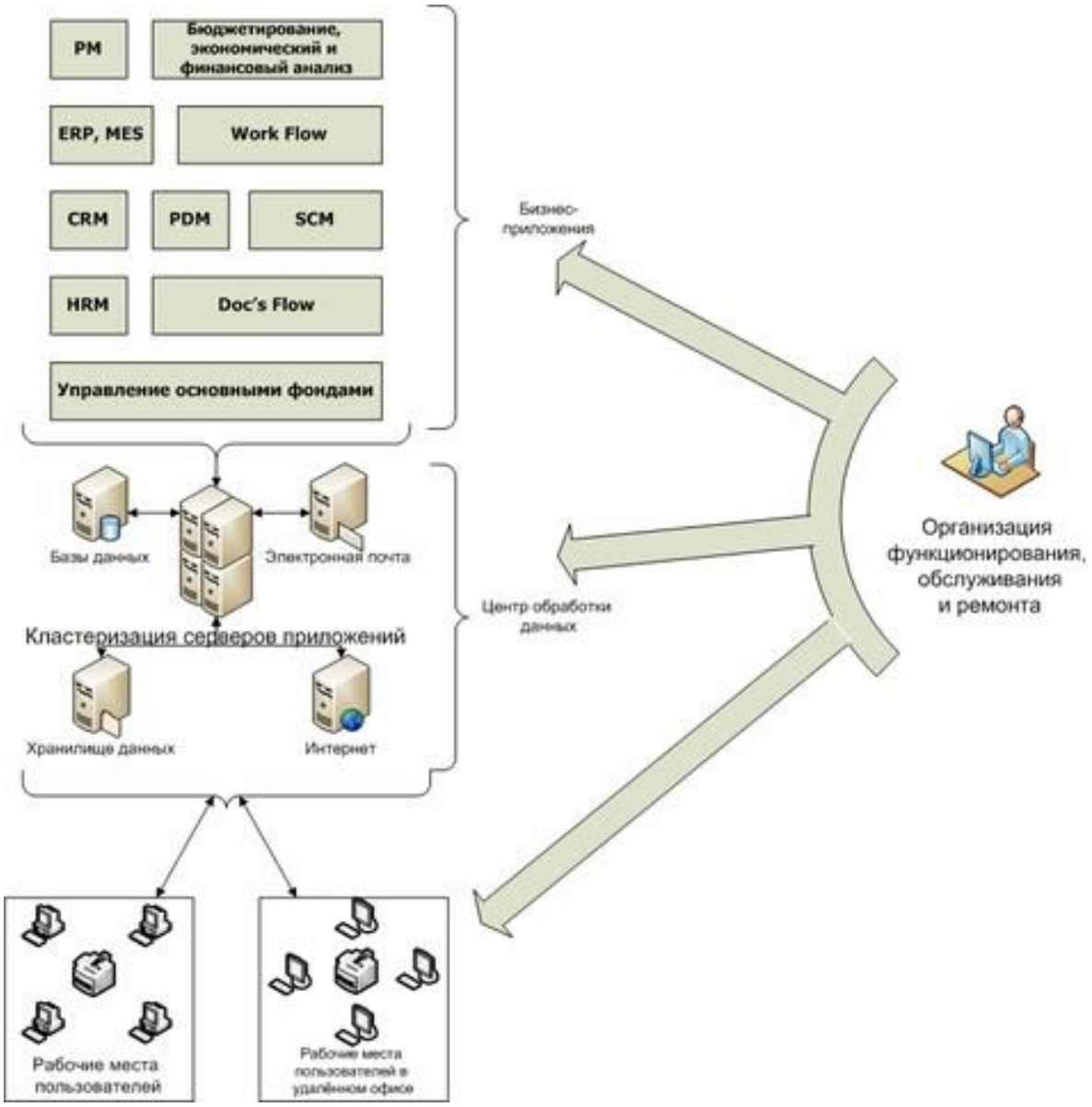

Рисунок 4.3 – Схема IT-інфраструктури виробничого підприємства

Перелік скорочень, що використані на рис. 4.3:
ERP – планування ресурсів підприємства.
MES - система управління виробничими процесами.
Work Flow – управління бізнес-процесами.
CRM - система управління відносинами з клієнтами.
PDM - система управління даними про вироби.
SCM – системи управління поставками.
HRM – управління персоналом
Doc's Flow – управління документами

4.2 Управління інформаційних систем

ІТ-служба підприємства є першою ланкою в структурі управління ІТ-інфрвструктурою. Вона виконую чотири основних функції:

- планування та організація;
- розробка, придбання і впровадження нових інформаційних систем;
- надання і супровід ІТ-сервісу;
- моніторинг (аудит процесів служби ІС).

Функція планування та організації реалізується за наступними напрямками:

- розробка стратегії в області ІТ;
- координація розвитку ІТ в організації;
- планування ресурсів служби ІС (бюджет, людські ресурси, зовнішні послуги та ін.);
- управління ризиками, управління якістю.

Функція надання і супроводу ІТ-сервісу реалізується за наступними напрямками:

- формалізація вимог підрозділів-замовників до ІТ-сервісів;
- погодження розбіжностей між вимогами до сервісів з відповідними ресурсами служби ІС;
- надання кінцевим користувачам ІТ-сервісів, що відповідають узгодженим вимогам.

Наступні фактори впливають на організаційну структуру ІТ-служби підприємства:

- масштаб служби ІС – більші служби ІС зазвичай мають більш складну і розгалужену організацією структуру;
- галузева приналежність, з якою пов'язано наявність або, навпаки, відсутність певних структурних підрозділів;
- розподіл організації по території – наявність територіально віддалених підрозділів і філій істотно змінює організаційну структуру служби ІС;
- склад ІС, які використовуються в організації.

Для підвищення ефективності функціонування ІТ-служби в організаціях з плоскою структурою використовують процесний підхід до управління. Для цього необхідно визначення мети, критеріїв результату, наявних ресурсів та розробка послідовності робіт (кроків процесу). Стосовно

до процесів служби ІТ метою є надання замовнику ІТ-сервісу високого рівня якості. Ця спільне завдання можна розділити на два етапи:

- визначення та узгодження параметрів ІТ-сервісу;
- забезпечення відповідності фактичних параметрів ІТ-сервісу досягнутим угодам.

4.3 Бізнес-орієнтоване управління системами

В залежності від функцій та масштабу розрізняють наступні типи ІТ-інфраструктур:

- базовий;
- стандартизований;
- раціональний;
- динамічний.

Структурні та функціональні ознаки ***базової*** інфраструктури:

- відсутність координації;
- ручний супровід;
- розрізнені робочі місця.

Рекомендації щодо підвищення рівня управління базової ІТ-інфраструктури:

- побудова серверної інфраструктури;
- введення служби каталогів для аутентифікації;
- налагодження сервісів для автоматичного оновлення;
- застосування антивірусного захисту;
- забезпечення захист трафіку;
- реалізація базових сценаріїв мережевої технології (DNS, DHCP).

Структурні та функціональні ознаки ***стандартизованої*** інфраструктури:

- централізоване управління ІТ-інфраструктурою;
- наявністю автоматизованих базових процесів;
- служба каталогів для аутентифікації;
- автоматизація поновлення;
- забезпечення антивірусного захисту на робочих місцях;
- реалізація системи резервного копіювання для критично важливих серверів;
- наявність центрального міжмережевого екрану;

– наявність внутрішніх DNS, DHCP.

Рекомендації щодо підвищення рівня управління стандартизованої IT-інфраструктури:

– забезпечення оновлення ПЗ на робочих місцях для останніх версій операційної системи (ОС) і пакету офісних додатків;
– активне застосування System Management Server;
– застосування рішень з централізованого резервного копіювання і відновлення після збоїв;
– організація віддаленого доступу VPN-мереж;
– ізоляція критично важливих серверів за допомогою застосування протоколу IPSec (для Active Directory / Exchange).

Структурні та функціональні ознаки ***раціональної*** інфраструктури:

– централізована і консолідована IT-інфраструктура;
– використання служби каталогів і групових політик для централізованого адміністрування;
– автоматизація контролю та моніторингу функціонування ПЗ і апаратного забезпечення;
– моніторинг серверів;
– резервне копіювання і відновлення для всіх серверів і робочих станцій;
– віддалений доступ (VPN, Remote Desktop);
– ізоляція серверів за допомогою IPSec.

Рекомендації щодо підвищення рівня управління раціональної IT-інфраструктури:

– впровадження технологій автоматизації управління ідентифікацією;
– використання System Management Server для управління серверами;
– перевірка додатків на сумісність;
– управління образами робочих станцій;
– розгортання та управління міжмережевими екранами на робочих місцях;
– організація захищеного бездротового мережевого доступу до служби Internet Authentication Service (IAS) та служби каталогів Active Directory.

Структурні та функціональні ознаки ***динамічної*** інфраструктури:

– повна автоматизація управління IT-інфраструктурою,

- повне забезпечення потреб користувача в умовах гетерогенних середовищ;
- автоматичне керування оновленнями для серверів;
- автоматичне тестування сумісності додатків і автоматичне керування образами робочих станцій;
- міжмережеві екрани на серверах і робочих місцях;
- захищені бездротові підключення.

Рекомендації щодо підвищення рівня управління динамічної IT-інфраструктури:

- впровадження рішення для автоматичного розповсюдження образів серверів;
- використання рішення для визначення рівня навантаження;
- забезпечення підтримки карантину робочих місць;
- постійний моніторинг продуктивності робочих місць;
- забезпечення готовності до переходу на нову версію операційної системи;
- впровадження інструментарію для ефективного переходу на нові версії ПЗ;
- забезпечення можливості ізоляції доменів Active Directory з використанням IPSec.

4.4 Ефективність функціонування інформаційної інфраструктури

Стрімкий розвиток IT-технологій, а також збільшення витрат компанії на підтримку IT-інфраструктури вимагає нових підходів і рішень для ведення успішного бізнесу. Одним з напрямків є оптимізація IT-ресурсів. При цьому необхідно застосування комплексного підходу. Необхідний аудит інфраструктури та процесів, аналіз поточного стану, оцінка можливих варіантів переходу в бажаний стан, планування та впровадження змін в інфраструктурі і процесах, оцінка кінцевого результату. При цьому в процесі впровадження змін зазвичай відбувається і модернізація інфраструктури, і інтеграція в IT-системи компаній нових елементів. Переважна більшість робіт з оптимізації інфраструктури є частиною повного комплексу інтеграційних робіт.

Аудит IT – це процес отримання незалежної оцінки стану IT-інфраструктури та процесів управління IT кращим світовим практикам і

цілям бізнесу. Об'єктами IT аудиту є компоненти IT інфраструктури компанії:

1) Обладнання і програмне забезпечення:

- інфраструктурне обладнання (сервери, системи зберігання даних);
- обладнання користувача (робочі станції, ноутбуки, оргтехніка);
- активне мережеве обладнання (комутатори, маршрутизатори, обладнання Wi-Fi);
- системне програмне забезпечення (операційні системи, СУБД та ін.);
- бізнес-додатки (1С, CRM, ERP, BI);
- офісні програми (MS Office, OpenOffice, кабельні мережі, електроживлення, безперебійне електроживлення).

2) Системи передачі даних:

- зовнішні канали зв'язку;
- доступ в Інтернет;
- телефонія;
- хостинг;
- електронна пошта;
- інші системі передачі даних (месенджери та ін.).

3) Безпека інформації:

- системи забезпечення безпеки інформації;
- антивірусний захист;
- антиспам-захист;
- системи захисту від зовнішніх атак;
- системи захисту від внутрішніх атак;
- системи шифрування інформації;
- системи резервного та архівного копіювання інформації.

Аудит IT- інфраструктури проводиться відповідно до плану та включає наступні етапи:

1) Планування:

- формування робочої групи;
- підписання початково-дозвільної документації;
- визначення цілей, завдань і меж аудиту;
- інтерв'ю з керівництвом організації;
- ідентифікація ключових ресурсів ІС;
- підготовка та узгодження плану аудиту.

2) Збір інформації:

- збір документованої інформації про ІС;
- анкетування та інтерв'ювання співробітників Замовника;
- ідентифікація існуючих механізмів управління ІТ;
- документування процедур;
- підготовка документів, що описують поточний стан ІТ.

3) Аналіз отриманих даних:

- оцінка механізмів управління ІТ;
- оцінка відповідності механізмів управління ІТ рекомендаціям COBIT (стандарт в галузі управління ІТ, аудиту та ІТ-безпеки.);
- аналіз ІТ-ризиків.

4) Розробка рекомендацій:

- аналіз ризиків, вибір засобів контролю для ключових ризиків;
- підготовка плану дій з мінімізації ризиків;
- розробка організаційних рекомендацій;
- визначення технічних рекомендацій;
- написання методологічних рекомендацій;
- підготовка рекомендацій на внесення змін.

5) Підготовка і підписання звітних документів:

- підготовка звіту про поточний стан ІТ-інфраструктури;
- узгодження і підписання звітних документів.

Існують різні варіанти проведення ІТ-аудиту. Структура складових ІТ-аудиту залежить від цілей аудиту, ІТ-інфраструктури підприємства, аудитора тощо. Далі представлені два варіанта ІТ-аудиту для підприємств різних галузей господарської діяльності.

Аудит ІТ-інфраструктури (варіант 1).

Основні складові аудиту ІТ-інфраструктури:

1. Аудит СКС (структурованої кабельної системи):

- «прозвон» всіх кабельних ліній і подальше маркування кожного кабелю, кожної патч-панелі;
- складання схеми комп'ютерної та телефонної мереж із зазначенням розміщення розеток на планах приміщень і кабельних трас (у випадку, якщо використовується IP-телефонія, то тільки комп. мережі).

2. Аудіт парку техніки (компьтери, ноутбуки, сервери, ДБЖ та ін.).

2.1. Робочі станції, ноутбуки (для кожної одиниці техніки):

– перевірка технічного стану обладнання;
– перевірка встановленого ПЗ, налаштувань операційної системи, перевірка на віруси.

2.2. Сервери і ДБЖ:

– перевірка технічного стану;
– перевірка адекватності налаштувань серверного ПЗ завданням, що виконуються;
– перевірка стану батарей ДБЖ.

3. Аудіт загальної структури ІТ на підприємстві:

– IP-адресація всіх мережевих пристроїв у мережі;
– складання схеми всього обладнання на підприємстві, із зазначенням розміщення і IP-адрес кожного мережевого пристрою (в т.ч. принтери, факси та інше, що теж слід нанести на схему);
– отримання всіх логінів-паролів до всіх мережевих пристроїв (комутатори, маршрутизатори) і їх налаштувань.

Аудит ІТ-інфраструктури (варіант 2).

Даний процес полягає в складанні докладної документації про кількісні складові ІТ-інфраструктури компанії і систематизації даних за категоріями адміністративної моделі і мережевої інфраструктури:

– телекомунікаційне та мережеве забезпечення;
– обладнання робочих місць;
– телефонія та канали передачі даних;
– інформаційна безпека.

Перелік структурних елементів ІТ-інфраструктури, що підлягають аудиту наведено в табл. 4.1.

Таблиця 4.1 – Варіант змісту складових аудиту ІТ-інфраструктури

Об'єкти аудиту	**Роботи в ході аудиту**
1	**2**
Аудит апаратного забезпечення комп'ютерної мережі	1. Інвентаризація мережевого і обладнання користувача. 2. Опис топології мережі та надання рекомендацій щодо її оптимізації. 3. Моніторинг морального і фізичного зносу апаратних складових мережі. 4. Оцінка відповідності приміщень вимогам умов експлуатації обладнання.

Об'єкти аудиту	Роботи в ході аудиту
1	**2**
Аудит програмного забезпечення	1. Інвентаризація ПЗ, що використовується в комп'ютерній мережі. 2. Оцінка правомірності його використання. 3. Надання рекомендацій по ліцензуванню (включаючи варіанти використання вільно поширюваного ПЗ).
Аудит роботи окремих серверів, мережевих ресурсів	1. Обстеження програмної та апаратної частини сервера. 2. Оцінка його надійності. 3. Аналіз відмовостійкості. 4. Моніторинг ефективності роботи. 5. Оцінка відповідності конфігурації сервера вирішуваним завданням.
Аудит мережі зберігання і передачі даних	1. Опис схеми зберігання та обміну даними в комп'ютерній мережі. 2. Складання політики доступу до окремих категорій даних. 3. Оцінка відповідності існуючої схеми політикам конфіденційності Замовника. 4. Надання рекомендацій.
Аудит корпоративної електронної пошти	1. Складання актуального списку електронних адрес і політик переадресацій та зіставлення їх з поточною кадровою політикою компанії Замовника. 2. Приведення схеми переадресацій у відповідність з політикою конфіденційності Замовника. 3. Оцінка системи збереження даних поштових скриньок ключових персон компанії Замовника, надання рекомендацій щодо поліпшення.
Аудит схем резервного копіювання	1. Складання актуальною схеми резервного копіювання даних. 2. Оцінка вразливостей існуючої схеми. 3. Надання рекомендацій щодо поліпшення.

Об'єкти аудиту	Роботи в ході аудиту
1	**2**
Аудит систем моніторингу та управління ІТ інфраструктурою	1. Оцінка існуючої схеми моніторингу та попередження збоїв у роботі ІТ інфраструктури. 2. Виявлення вразливих місць. 3. Складання інструкцій дії у разі різних збоїв у роботі ІТ інфраструктури. 4. Складання схем швидкого відновлення ІТ інфраструктури. Надання рекомендацій з їх впровадження.
Аудит захисту інформації	1. Оцінка зовнішніх вразливостей ІТ інфраструктури. 2. Надання рекомендацій з організації захисту важливих даних.
Аудит телефонії	1. Опис поточної схеми роботи телефонії. Складання актуальної схеми переадресацій, розподілів та скидів дзвінків. 2. Аналіз витрат на зв'язок, враховуючи особливості роботи компанії Замовника. 3. Надання рекомендацій щодо зниження витрат на зв'язок.
Аудит роботи внутрішніх сервісів	1. Опис роботи ключових внутрішніх ІТ сервісів. 2. Оцінка їхньої продуктивності. 3. Надання рекомендацій щодо поліпшення.

4.5 Безпека функціонування інформаційної інфраструктури

Безпека мережі – це основа працездатної ІТ-інфраструктури. Кожен елемент мережі використовує передачу даних, а це одна з причин вразливості мережі до зловмисних атак. Розташовуючи елементи безпеки у критичних точках мережі, можна надавати належним користувачам доступ до ресурсів, які їм потрібні, й одночасно захищати корпоративні дані ІТ-системи.

Визнання загроз безпеки та розгортання рішень для боротьби з ними – це основна сфера відповідальності сучасних ІТ-відділів. Нижче перелічені деякі з проблем, які постають перед ІТ-фахівцями під час забезпечення захисту та надійності роботи служб.

Запобігання проникненню вірусів і зловмисного вмісту. У разі проникнення до корпоративної мережі комп'ютерні віруси можуть пошкодити роботу комп'ютерних систем, оскільки вони заважають операціям, заражають дані та навіть сприяють викраданню. Незалежно від того, як комп'ютерні віруси потрапляють до мережі, вони в результаті спричиняють пошкодження та збитки.

Забезпечення функціонування та доступності мережі. Оскільки це служба основної IT-інфраструктури, мережа має бути доступною. Сучасні мережі включають велику кількість різноманітних систем і пристроїв, тому керування доступом може бути складною нестандартною задачею. Надійні інструменти безпеки можуть зменшити складність і покращити безпеку, не порушуючи доступність.

Баланс доступу та безпеки. Розрізнення загроз для мережі та припустимих запитів на доступ є основною задачею. Найбільш безпечна система у світі – це та, до якої немає доступу, але на практиці вона не приносить користі. Таким чином, загрози безпеці мають розпізнаватися і усуватися без порушення роботи служб.

Визначення загроз, їх нейтралізація та запровадження надійних методів керування може забезпечити доступність служб мережі. Більш захищена інфраструктура мережі підтримує бізнес-служби та забезпечує надійну основу. Ключові елементи більш захищених служб мережі:

1) ***Брандмауери***. Централізовані брандмауери та брандмауери окремих комп'ютерів можуть запобігати проникненню зловмисного трафіку до мережі, яка підтримує діяльність компанії.

2) ***Антивірусні засоби***. Більш захищена мережа може виявляти загрози, що створюють віруси та інше зловмисне програмне забезпечення, і боротися з ним попереджувальними методами, перш ніж вони зможуть заподіяти шкоду.

3) ***Надійність мережі***. Знаряддя, які відстежують стан мережі, грають важливу роль під час визначення загроз.

4) ***Захищений віддалений доступ і обмін даними***. Безпечний доступ для всіх типів клієнтів із використанням різноманітних механізмів доступу грає важливу роль для забезпечення доступу користувачів до потрібних даних, незалежно від їх місцезнаходження та пристроїв.

5) ***Керування конфігурацією***. Хоча це і не елемент безпеки, у строгому значенні цього слова, керування конфігурацією дає змогу визначити відповідну конфігурацію систем на основі політик безпеки.

ПЕРЕЛІК ПОСИЛАНЬ

1. Гвоздева Т.В., Баллод Б.А. Проектирование информационных систем: Учебное пособие. – Ростов н/Д: Феникс, 2009. – 512 с.
2. Маклафлин Б. Объектно-ориентированный анализ и проектирование: пер. с англ. / Поллайс Г., Уэст Д. . – П.: Питер, 2013. – 602 с.
3. Кречмер Р. Разработка приложений SAP R/3 на языке ABAP/4: пер. с англ. / Вейс В. – М.: Лори, 2013. – 334 с.
4. О'Лири Д.ERP системы. Современное планирование и управление ресурсами предприятия. Выбор, внедрение, эксплуатация / Д. 'Лири; пер. с англ. – М.: ООО"Вершина", 2004. –272 с.
5. Гвоздева В.А. Основы построения автоматизированных информационных систем: учебник/ В.А.Гвоздева, Ю.И.Лаврентьева. –М.: ИД "Форум": ИНФРА-М, 2007. –320 с.
6. Вилл Р. Системное администрирование SAP R/3: пер. с англ.. – М.: Лори, 2013. – 360 с.
7. Федотова Д.Э. CASE-технологии: практикум/ Д.Э.Федотова, Ю. Д. Семенов, К. Н. Чижик. –М. :Горячая линия-Телеком, 2005. –160 с.
8. Мацяшек Л. А. Анализ требований и проектирование систем. Разработка нформационных систем с использованием UML/ Л. . Мацяшек; пер. с англ. – М.: Издательский дом "Вильямс", 2002. – 432 с.
9. Дубаков А. А. Проектирование информационных систем / А. А. Дубаков. –Томск: Изд.Томского политехнического университета, 2011. –258 с.
10. Елиферов В.Г. Бизнес-процессы: регламентация и управление:учебник / В.Г.Елиферов, В.В.Репнин. –М. :ИНФРА-М, 2004. –320 с.
11. Информационные системы в экономике: учебник под ред. Г. А. Титоренко. –2-е изд., перераб. и доп.–М. : Юнити-Дана, 2008. –463 с.
12. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: учебник/ А. М.Вендров. –2-е изд., перераб. и доп. –М.:Финансы и статистика, 2006. –544с.
13. Грекул В. И. Проектирование информационных систем: учебн. пособ./В. И. Грекул, Г. Н.Денищенко, Н. Л. Коровкина. –М.: БИНОМ. Лаборатория знаний, 2008. –300 с.
14. Избачков Ю. С. Информационные системы: учебник/ Ю. С. Избачков, В.Н. Петров. – 2-е изд. –СПб.: Питер, 2006. –656 с.

15. Методи структурного аналізу і проектування ПЗ [Электроний ресурс]. – Режим доступа: http://pidruchniki.com/1707032647730/informatika/ metodi_ strukturnogo_analizu_proektuvannya.
16. JAD – методика спільної розробки ком`пютерних інформаційних систем [Электроний ресурс]. – Режим доступа: http://dss-bi.com.ua/System/jad.
17. Безпека інформаційних систем [Электроний ресурс]. – Режим доступа: https://www.microsoft.com/ukraine/infrastructure/capabilities/securityandnetworking.mspx.

Навчальне електронне видання

ТЕРЕЩЕНКО ТЕТЯНА МИХАЙЛІВНА

МОДЕЛІ, ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ ТА УПРАВЛІННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Конспект лекцій

Видавець і виготовлювач
Одеський державний екологічний університет
вул. Львівська, 15, м. Одеса, 65016
тел./факс: (0482) 32-67-35
E-mail: info@odeku.edu.ua
Свідоцтво суб'єкта видавничої справи
ДК № 5242 від 08.11.2016