МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Євгенія Левус
Наталія Мельник

ВСТУП ДО ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Навчальний посібник

Львів
Видавництво Львівської політехніки
2017

Р е ц е н з е н т и:

Яковина В.С. – д-р техн. наук, доц., завідувач кафедри програмного забезпечення Національного університету "Львівська політехніка",

Савула Я.Г. – д-р фіз.-мат. наук, проф., завідувач кафедр прикладної математики Національного університету імені І. Франка,

Сеник Т.Д. – канд. фіз.-мат. наук, директор ТзОВ «ДАТА ПРО»

Рекомендувала Науково-методична рада
Національного університету "Львівська політехніка"
як навчальний посібник для студентів спеціальності
«Інженерія програмного забезпечення»
(протокол № від _____ р.)

Левус Є.В., Мельник Н.Б.

Вступ до інженерії програмного забезпечення: навч. посібник / Є.В. Левус, Н.Б. Мельник – Львів: Видавництво Львівської політехніки, 2017. – 280 с.

Посібник призначений для вивчення основного матеріалу навчальної дисципліни «Вступ до інженерії програмного забезпечення». У посібнику містяться теоретичні матеріали, що стосуються означення інженерії програмного забезпечення, життєвого циклу програмного забезпечення, його моделей. Також наведено матеріали для самостійного вивчення розділу «Групова динаміка і комунікації» навчальної дисципліни.

Важливим засобом підготовки студентів для поточного й семестрового контролю є наведені у посібнику більше 100 тестових завдань.

Видання призначено для студентів спеціальності «Інженерія програмного забезпечення», а також може використовуватися для інших спеціальностей галузі знань «Інформаційні технології» для вивчення дисциплін, пов’язаних з розробленням програмного забезпечення.

УДК

Передмова

Інженерія програмного забезпечення, як і будь-яка ІТ-галузь, інтенсивно розвивається – з'являються нові технології, платформи й середовища програмування, вдосконалюються методології розроблення програмного забезпечення, розробляються новітні методи й засоби автоматизації процесів виробництва програмних продуктів. Що 5-10 років стаються події, які іменуються черговими інформаційними революціям. Відповідно ця динамічна галузь знань постійно доповнюється накопиченим досвідом фахівців індустрії програмного забезпечення. Тому вивчення фундаментальних основ інженерії програмного забезпечення є важливим для розуміння динаміки і тенденцій розвитку інженерії програмного забезпечення.

Навчальна дисципліна «Вступ до інженерії програмного забезпечення» має за мету сформувати у студентів інженерні погляди на процеси у галузі інженерії програмного забезпечення, допомогти здобути теоретичні знання та виробити базові практичні навички з виконання основних процесів життєвого циклу програмного забезпечення.

Навчальна дисципліна «Вступ до інженерії програмного забезпечення» містить розділи: «Базові поняття інженерії програмного забезпечення», «Життєвий цикл програмного забезпечення», «Моделі життєвого циклу програмного забезпечення», «Групова динаміка і комунікації».

У ході вивчення першого розділу перед студентами ставляться такі задачі:

- ознайомитися з історією становлення і розвитку інженерії програмного забезпечення як складової галузі інформаційних технологій;
- засвоїти основні поняттями інженерії програмного забезпечення;
- ознайомитися з базовими методами розробки програмного забезпечення, які були сформовані в ході етапів розвитку інженерії програмного забезпечення.

У ході вивчення другого розділу перед студентами ставляться такі задачі:

- засвоїти основний принцип інженерії програмного забезпечення – життєвий цикл програмного забезпечення;
- навчитися проводити ідентифікацію, аналіз та контроль різних етапів життєвого циклу програми;
- вміти складати супроводжувальну проектну документацію до результатів основних етапів життєвого циклу програмного забезпечення.

У ході вивчення третього розділу перед студентами ставляться такі задачі:

- ознайомитися з основними моделями життєвого циклу програмного забезпечення та умовами їх застосування;
- вміти обґрунтовано обирати, комбінувати основні моделі життєвого циклу для конкретних умов програмних проектів.

У ході вивчення четвертого розділу перед студентами ставляться такі задачі:

- ознайомитися з загальною характеристикою груп в системі суспільної взаємодії при підготовці програмного забезпечення;
- засвоїти моделі поведінки в групі та механізми їх реалізації;
- ознайомитися з принципами ділового спілкування як основою механізму професійного взаєморозуміння, засадами організації продуктивної групової роботи та документування результатів діяльності інженерів з програмного забезпечення;
- вміти налагоджувати комфортні взаємовідносини у колективі.

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

Загальна інформація

Навчальний посібник призначений для студентів першого курсу спеціальності «Інженерія програмного забезпечення» усіх форм навчання для вивчення основних розділів дисципліни «Вступ до інженерії програмного забезпечення». Може також використовуватися для студентів суміжних спеціальностей «Комп'ютерні науки», «Інформаційні технології і системи», «Системний аналіз», «Прикладна математика», «Комп'ютерна інженерія».

Дисципліна «Вступ до інженерії програмного забезпечення» належить до циклу професійної і практичної підготовкою та є базовою для вивчення інженерних основ процесу розроблення програмного забезпечення.

Пререквізитом дисципліни «Вступ до інженерії програмного забезпечення» є дисципліна «Основи програмування».

Компетентності, які забезпечуються вивченням розділів дисципліни

Внаслідок вивчення представлених у посібнику розділів студент повинен бути здатним продемонструвати такі **результати навчання** :

- знання принципів та методів інженерії ПЗ;
- вміння створювати документи, що описують основні результати етапів ЖЦ ПЗ;
- вміння застосовувати базові поняття інженерії ПЗ та інших дисциплін комп'ютингу у процесах життєвого циклу програмного забезпечення

Вивчення представлених у посібнику розділів передбачає формування та розвиток у студентів **фахових компетентностей**:

- здатність застосовувати знання і вміння інженерії вимог для аналізу предметної області та розроблення, верифікації і атестації специфікації вимог до програмного забезпечення;
- розуміння принципів людино-машинної взаємодії та здатність створювати програмні продукти з ефективним інтерфейсом користувача;
- здатність створювати технічну документацію на програмні системи відповідно до прийнятих стандартів та практик.

Результати деталізують такі програмні результати навчання:

- здатність демонструвати знання та розуміння основ інформаційних технологій та математики як теоретичної бази для розроблення програмного забезпечення;
- здатність демонструвати знання інженерних основ у галузі розроблення програмного забезпечення, зокрема процесів життєвого циклу програмного забезпечення, їх моделей, стандартів інженерії програмного забезпечення;
- здатність демонструвати знання методів виявлення та аналізу вимог користувачів для визначення вимог до програмної системи, що розробляється;
- застосувати знання та розуміння для отримання та документування результатів основних складових етапів життєвого циклу програмного забезпечення;
- знання та розуміння для аналізу та специфікації вимог, а також їх верифікації.

Місце навчальної дисципліни «Вступ до інженерії програмного забезпечення» серед інших дисциплін навчального плану спеціальності «Інженерія програмного забезпечення»

Дисципліна «Вступ до інженерії програмного забезпечення» є ввідною до спеціальності та забезпечує знання фундаментальних понять галузі розроблення програмного забезпечення, а саме інженерних аспектів цієї діяльності. Дисципліна є пререквізитом для багатьох інших дисциплін спеціальності.

Пререквізит навчальної дисципліни	**Кореквізити навчальної дисципліни**
Основи програмування	Людино-машинна взаємодія
	Аналіз вимог до програмного забезпечення
	Моделювання й аналіз програмного забезпечення
	Основи командної розробки програмного забезпечення
	Якість і тестування програмного забезпечення
	Менеджмент програмних проектів

Використані скорочення

АОП –аспектно--орієнтоване програмування

ЕОМ – електронно-обчислювальна машина

ІПЗ – інженерія програмного забезпечення

ІС – інформаційна система

ІТ – інформаційні технології

ЖЦ ПЗ – життєвий цикл програмного забезпечення

КІ – комп’ютерна інженерія

КН – комп’ютерні науки

ООП – об’єктно-орієнтоване програмування

ОС – операційна система

ПЗ – програмне забезпечення

СУБД – система управління базами даних

ТЗ – технічне завдання

РОЗДІЛ 1. БАЗОВІ ПОНЯТТЯ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У розділі розглянуто питання виникнення та формування галузі інженерії програмного забезпечення. Наведено означення інженерії програмного забезпечення. Описано методи розроблення програмного забезпечення – модульне, структурне, об'єктно-орієнтоване програмування і пост-об'єктні методи програмування. Пояснено базові інженерні поняття – програмний продукт, проект, процес, персонал, які складають так звану еталонну модель інженерії програмного забезпечення.

Тема 1.1. ОЗНАЧЕННЯ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЯК ГАЛУЗІ ІТ

Передумови формування інженерії програмного забезпечення

У кінці 20 століття на зміну індустріальному суспільству прийшло суспільство інформаційне, тобто засноване на знаннях. У провідних країнах зайнятість населення в інформаційній сфері становить 60%, а у сфері матеріального виробництва – 40%. Саме тому спеціальності галузі інформаційних технологій гарантують набуття найбільш престижних, дефіцитних і високооплачуваних професій. Так вважають в усіх розвинутих країнах світу. Адже не дарма стверджують: «Хто володіє інформацією – той володіє світом».

Сучасне виробництво програмного забезпечення (ПЗ) – це найбільша галузь світової економіки, в якій задіяно мільйони фахівців. Уперше термін software увів відомий статистик Джон Т'юкей (John Tukey) у 1958 р. для позначення відмінностей апаратного забезпечення ЕОМ (hardware) від засобів обробки даних. Саме приголомшливий прогрес в області ПЗ допоміг впоратися з інформаційним бумом кінця 20 століття.

У даний час практично немає жодної сфери людської діяльності (економіка, комерція, промисловість, освіта, медицина, наука, сфера розваг, кіноіндустрія і т.д.), де б не застосовувалися комп'ютерні програми. При цьому багато компаній, що працюють у різних областях матеріального виробництва й «традиційного» бізнесу, починають усвідомлювати, що забезпечити більш високі темпи зростання, домогтися конкурентної переваги вони можуть як за рахунок апаратно-орієнтованих, так і за рахунок програмно-орієнтованих рішень, і все частіше віддають перевагу останнім.

Звичайно ж, професійних розробників ПЗ не може не тішити той факт, що світова економіка стає все більш залежна від програмного забезпечення. Завдяки розвитку комп’ютерної техніки й технологій розроблення програмного забезпечення стало можливим створення програмних систем, які постійно ускладнюються і знаходять все більше поширення. Крім того, зростає і суспільний попит на ці системи.

Зняті практично всі апаратні обмеження на вирішення завдань. Решту обмежень припадають на частку ПЗ.

Надзвичайно актуальними стали такі проблеми:

- апаратна складність випереджає наше вміння будувати ПЗ, що може використовувати потенційні можливості апаратури;
- вміння будувати нові програми відстає від вимог до нових програм;
- нашим можливостям експлуатувати існуючі програми загрожує низька якість їх розроблення.

Ключем до вирішення цих проблем є грамотна організація процесу створення ПЗ, реалізація технологічних принципів промислового конструювання програмних систем. Інтуїтивний похід до розроблення програмних систем, заснований на знаннях, уміннях і талантах окремих програмістів-одинаків, не дозволяє розробляти складні системи і суперечить принципам їх командної розробки.

Чим програмування відрізняється від інженерії програмного забезпечення? Програмування є деякою абстрактної діяльністю і може виконуватися в багатьох

різних контекстах. Можна програмувати для власних потреб, щоб автоматизувати виконання різноманітних завдань у процесі навчання в школі, можна програмувати в рамках наукових розробок. А можна займатися промисловим програмуванням. Як правило, це відбувається *в команді, і для замовника,* який платить за роботу розробників *гроші* та очікує потрібний результат за певний період *часу.* При цьому необхідно точно розуміти, що потрібно замовнику, виконати роботу, використавши наявні ресурси й результат повинен бути потрібної якості. Щоб задовольнити цим умовам, традиційне програмування «обростає» різними додатковими видами діяльності: розробленням вимог до програмної системи, плануванням робіт, тестуванням програмного забезпечення, проектним менеджментом, створенням різної документації та ін.

Будь-яка програма створюється з метою забезпечення визначених потреб її користувачів. Отже, для побудови успішної програмної системи необхідно знати, чого потребують і чого хочуть її майбутні користувачі. Як наслідок, повинні бути регламентовані і коректно сформульовані вимоги до системи. Тому можна вважати, що процес розроблення програмного забезпечення – це сума різних видів діяльності, необхідних для перетворення вимог користувачів в програмну систему.

Інженерія програмного забезпечення – це наукова і інженерна дисципліни виробництва програмних продуктів.

Рис. 1.1. Зміст інженерії ПЗ.

Необхідність в інженерії програмного забезпечення як у спеціальній галузі знань були усвідомлені світовою спільнотою в кінці 60-х років минулого століття. Це сталося більш ніж на 20 років пізніше народження самого програмування, якщо вважати таким знаменитий звіт фон Неймана «First Draft of a Report on the EDVAC», оприлюднений ним в 1945 році. Народженням інженерії програмного забезпечення є 1968 – конференція NATO Software Engineering, м Гарміш (Німеччина).

Інженерія програмного забезпечення (промислове програмування) зазвичай асоціюється з виробництвом великих і складних програм колективами розробників. становлення і розвиток цієї галузі діяльності було викликано низкою проблем, пов'язаних з високою вартістю програмного забезпечення, складністю його створення, необхідністю управління і прогнозування процесів розроблення.

У сферу інженерії програмного забезпечення потрапляють, крім безпосередньо процесів розроблення програмного забезпечення, всі питання і теми, пов'язані з організацією і вдосконаленням процесів розроблення ПЗ, управлінням колективу розробників, економікою та фінансами питання.

Інженерія програмного забезпечення використовує досягнення інформатики. Інформатика (Computer Science, англ.) – це зведення теоретичних наук, заснованих на математиці і присвячених формальним основам обчислюваності. До інформатики відносять математичну логіку, теорію граматик, методи побудови компіляторів, математичні формальні методи, використовувані в верифікації та тестуванні і т.д. На перш погляд, важко строго відокремити інженерію програмного забезпечення від інформатики, але в цілому спрямованість цих дисциплін різна. Інженерія програмного забезпечення націлена на вирішення проблем виробництва, інформатика – на розроблення формальних, математизованих підходів до програмування.

Інженерія програмного забезпечення (Software Engineering) –це наука побудови програмних систем, що містить у собі теоретичні концепції, методи і засоби програмування, технології програмування, системи та інструменти їхньої

підтримки, сучасні стандарти, зокрема, процеси життєвого циклу, вимірювання, оцінювання якості програмних систем. Головне призначення інженерії програмного забезпечення – побудова програмних систем, починаючи з аналізу предметної області й закінчуючи виготовленням вихідного коду для виконання на комп'ютері. Фундаментальну основу побудови програмних систем складає теорія алгоритмів, математична логіка, теорія обчислень, теорія керування й ін.

Процеси в інженерії програмного забезпечення практично вже відпрацьовані й за своєю сутністю близькі до інженерної діяльності у промисловості, де інженерія – це спосіб застосування наукових результатів у виготовленні технічних виробів на основі технологічних правил і процедур, методик виміру, оцінки і сертифікації в цілях задовольняння й отримання користі від виготовленого продукту або товару.

Інженерія програмного забезпечення прагне поєднати принципи математики та комп’ютерних наук разом з інженерною практикою. Інженерна діяльність у програмуванні за своєю сутністю дуже близька до інженерної конвеєрної діяльності в промисловості, тільки тут готовими «деталями» виступають компоненти багаторазового використання, яких достатньо створено у різних областях, і вони подібні до готових деталей в промисловості. Це є фундаментальним для становлення конвеєрного виробництва програмних продуктів, як продуктів промислово-технічного призначення.

Характерною ознакою виробництва програмних продуктів стала поява нових категорій фахівців, крім програмістів, а саме, менеджерів, керівників команди розробників, інженерів служби ведення бібліотек, технологів, тестувальників і різного роду контролерів проміжних результатів проектування на процесах ЖЦ.

Інженерія програмного забезпечення як напрям виникла і формувалась під впливом росту вартості створюваного програмного забезпечення. Головна ціль цієї області знань – зменшення вартості й термінів розроблення програм.

Місце інженерії ПЗ серед інших комп'ютерних дисциплін

Для означення області знань комп'ютерних дисциплін часто використовують термін комп'ютинг (computing, англ.), який є комплексно узагальнюючим для виокремлення галузі знань, науки, виробництва, надання відповідних послуг та сервісів.

Уперше термін комп'ютинґ уведений 1998 року Яном Фостером з арагонської національної лабораторії Чікагського університету та Карлом Кесельманом з інституту інформатики штату Каліфорнія (США) і запропонований для означення комплексної галузі знань, яка включає проектування та побудову апаратних і програмних систем для широкого кола застосувань; вивчення процесів, структур і керування інформацією різних видів; виконання наукових досліджень із застосування комп'ютерів та їх інтелектуальності; створення і використання комунікаційних та демонстраційних засобів пошуку та збирання інформації для конкретної мети і т. ін.

Прийнято вважати, що комп'ютинг формують п'ять комп'ютерних галузей (напрямків), а саме: комп'ютерна інженерія, комп'ютерні науки, інформаційні системи, інформаційні технології та інженерія програмного забезпечення.

Історично формування цих напрямків відбувалося з поступовим виявлення їх особливостей серед спільної області знань. Можна говорити про тенденцію виокремлення на потребу часу.

Рис. 1.2. Розвиток комп’ютерних дисциплін:
Е – електроніка ; КІ – комп’ютерна інженерія; КН – комп’ютерні науки;
ІС – інформаційні системи; ІПЗ – інженерія програмного забезпечення;
ІТ – інформаційні технології.

Комп’ютерна інженерія (КІ) пов’язана з розробленням і конструюванням комп’ютерів й автоматизованих систем. Вона включає вивчення складових комп’ютера, програмного забезпечення, елементів комунікацій та взаємодії між ними. КІ базується на теоріях та принципах традиційної радіотехніки і математики й застосовує їх для вирішення проблем розроблення комп'ютерів. Цей напрям

вивчає проектування цифрових комплектуючих, включаючи комунікаційні системи, комп'ютери та їх складові. На сьогодні домінуючою областю в КІ є розроблення пристроїв, які потребують інтеграції апаратного та вбудованого програмного забезпечення. Наприклад, такі пристрої як мобільні телефони, цифрові програвачі, мікроконтролери, системи сигналізації, медичне устаткування використовують досягненням комп'ютерної інженерії.

Історія виникнення комп'ютерних наук сягає 40-х років XX століття. Як наукова дисципліна комп'ютерні науки (КН) виникли в результаті злиття теорії алгоритмів і математичної логіки, а також виникнення електронно-обчислювальних машин (ЕОМ). КН – це обширна галузь теоретичних і прикладних знань, пов'язаних з отриманням, збереженням, обробкою, передаванням і використанням інформації.

Робота фахівців у галузі комп'ютерних наук поділяється на такі категорії:

• Розроблення та впровадження програмного забезпечення.

• Розроблення нових шляхів для використання комп'ютерів. Наприклад, дослідники КН співпрацюють з науковцями інших галузей, для того щоб створити роботів, які б стали практичними та розумними допоміжними засобами, використовувати бази даних для створення нових знань та за допомогою комп'ютерів розкрити секрети нашого ДНК.

• Розроблення ефективних шляхів для розв'язання комп'ютерних проблем. Наприклад, фахівці у галузі комп'ютерних наук розробляють найкращий спосіб для зберігання інформації у базі даних.

На перетині областей ІПЗ, КН, КІ формується багато перспективних напрямків застосування інформаційних технологій. Таким є Інтернет речей (Internet of Things, IoT*, англ.*) — це мережа, що складається із взаємозв'язаних фізичних речей або пристроїв, які мають вбудовані датчики, а також програмне забезпечення, що дозволяє здійснювати передачу й обмін даними між фізичним світом і комп'ютерними системами. Ці взаємопов'язані об'єкти мають можливість зчитування та приведення в дію, функцію програмування та ідентифікації, а також

дозволяють виключити необхідність участі людини, за рахунок використання інтелектуальних інтерфейсів.

Інформаційна технологія – це процес, що використає сукупність засобів і методів збору, обробки й передачі даних (первинної інформації) для одержання інформації нової якості про стан об'єкта, процесу або явища. Ця отримана нова інформація називається інформаційним продуктом.

Основними принципами застосування нової (комп'ютерної) інформаційної технології є:

- ✓ інтерактивний (діалоговий) режим роботи з комп'ютером;
- ✓ інтегрованість (стикування, взаємозв'язок) з різними програмними продуктами;
- ✓ гнучкість процесу зміни як даних, так і постановок завдань.

Широка інформатизація всіх сфер життєдіяльності суспільства принципово змінює роль інформації й інформаційних технологій у соціальному й економічному розвитку країни. Від масштабів й якості використання інформаційних технологій у професійній діяльності фахівців різних галузей залежать рівень економічного й соціального розвитку суспільства, його інтеграція у світову економічну систему. Тому інформаційні системи й технології використаються у всіх сферах діяльності людини.

Інформаційні технології як комп’ютерний напрям має за мету виробництво інформації для її аналізу людиною й прийняття на його основі рішення з виконання якої-небудь дії.

Інформаційна система – це організовані людиною системи збору, збереження, обробки і видачі інформації, необхідної для ефективного функціонування суб’єктів і об’єктів управління. Такі системи є засобом задоволення потреб управлінської діяльності в інформації. До компонентів інформаційної системи належать:

1) інформація, необхідна для виконання одної чи декількох функцій управління;

2) персонал, який забезпечує функціонування інформаційної системи;

3) технічні засоби;

4) методи і процедури збору і переробки інформації.

Інформаційна система як комп'ютерний напрям вивчає методи та засоби як у потрібний момент з відповідних джерел отримувати інформацію, яка повинна бути попередньо систематизована і певним чином опрацьована.

Інженерія програмного забезпечення – це дисципліна розвитку систем програмного забезпечення, які надійно та раціонально працюють, можуть розвиватися, а також задовольняти усім вимогам, які для них визначають клієнти.

Інженерія програмного забезпечення відрізняється за своєю сутністю від інших інженерних дисциплін як завдяки невиразній природі програмного забезпечення, так і дискретній природі роботи програмного забезпечення. Вона прагне поєднати принципи математики й комп'ютерних наук разом з інженерною практикою.

Інженерія програмного забезпечення як напрям виникла і формувалась під впливом росту вартості створюваного програмного забезпечення. Потреба контролювати процес розроблення ПЗ, прогнозувати і гарантувати вартість розробки, терміни й якість результатів привела в кінці 70-х років 20 століття до необхідності переходу від кустарних до індустріальних методів створення ПЗ і появи сукупності інженерних методів і засобів створення ПЗ, об'єднаних загальною назвою «інженерія програмного забезпечення». Головна ціль цієї області знань – зменшення вартості і термінів розроблення програм.

Інженерія програмного забезпечення пройшла декілька етапів розвитку, в процесі яких були сформульовані фундаментальні принципи і методи розроблення програмних продуктів. Основний принцип інженерії програмного забезпечення полягає в тому, що програми створюються в результаті виконання декількох взаємозв'язаних етапів.

Інженерія програмного забезпечення – це інженерна дисципліна, яка пов’язана зі всіма аспектами виробництва ПЗ від початкових стадій створення специфікації до підтримки системи після здачі в експлуатацію.

Означення терміну «Інженерія програмного забезпечення»

На сьогоднішній день можна навести декілька варіантів означення поняття «Інженерія програмного забезпечення». Ці означення сформовано фахівцями в цій галузі та зафіксовані в документах провідних організацій й міжнародних стандартах. Не дивлячись на різноманіття означень ІПЗ, всі вони задають інженерну діяльність в галузі програмного забезпечення, спрямовану на створення програмного продукту, а не просто програмного забезпечення.

Інженерія програмного забезпечення – це

- установлення і використання обґрунтованих інженерних принципів (методів) для економного створення ПЗ, яке надійно працює на реальних машинах (Bauer, 1972).
- та форма інженері, яка застосовує принципи інформатики (computer science) і математики для рентабельного вирішення проблем ПЗ (CMU/SEI-90-TR-003).
- застосування систематичного, дисциплінованого, виміряного підходу до розроблення, використанню і супроводженню ПЗ (IEEE 1990).
- дисципліна, ціллю якої є створення якісного ПЗ, яке завершаються вчасно, не перевищує виділених бюджетних засобів і задовольняє сформованим вимогам (Schach, 1999).

Сам термін software engineering (інженерія програмного забезпечення) вперше був озвучений у жовтні 1968 року на конференції підкомітету НАТО з науки і техніці (місто Гарміш, Німеччина). Були присутні 50 професійних розробників ПЗ з 11 країн. Розглядалися проблеми проектування, розроблення,

використання та супроводу програм. Там вперше і прозвучав термін «інженерія програмного забезпечення» як деяка дисципліна, яку треба створювати і якою слід керуватися у вирішенні зазначених проблем.

Незабаром після цього в Лондоні відбулася зустріч керівників проектів з розроблення програмного забезпечення. На зустрічі аналізувалися проблеми і перспективи розвитку ПЗ. Відзначалася зростаючий вплив ПЗ на життя людей. Вперше серйозно заговорили про кризу у галузі розроблення ПЗ. Зрозумілим стає, що принципи та методи розроблення ПЗ вимагають постійного вдосконалення. Саме на цій зустрічі була запропонована концепція життєвого циклу ПЗ (SLC - Software Lifetime Cycle) як послідовності кроків-стадій, які необхідно виконати в процесі створення та експлуатації ПЗ.

Спочатку навколо цієї концепції було багато суперечок. В 1970 р. У.У. Ройс (W.W. Royce) здійснив ідентифікацію декількох стадій у типовому циклі ПЗ та було висловлено припущення, що контроль виконання стадій призведе до підвищення якості ПЗ і скорочення вартості розробки.

Таким чином, виникнення інженерії ПЗ як дисципліни розроблення програмних систем визначено такими важливими факторами:

- значним обсягом накопичених знань в області створення ПЗ;
- появою нових методів аналізу, моделювання й проектування ПЗ;
- удосконалюванням методів виявлення помилок у ПЗ;
- ефективною організацією колективів розробників ПЗ й оцінки їхньої трудової діяльності;
- використанням готових програмних компонентів, високотехнологічних засобів й інструментів розроблення ПЗ;
- необхідністю еволюційного розвитку компонентів і систем, а також їхньою адаптацією до нових умов, що змінюються, операційних середовищ і комп'ютерних мереж.

Особливості інженерії програмного забезпечення

Інженерія програмного забезпечення відрізняється від традиційної промислової інженерії природою свого продукту, який не матеріалізується в наочний фізичний предмет, а постійно змінюється під час супроводження та при стрімких темпах розвитку комп'ютерних платформ і середовищ. Серед особливих властивостей інженерії ПЗ можна виокремити такі, що стосуються абстрактності й нематеріальності ПЗ, відсутності фізичних законів, неможливості опрацювання виробничими процесами.

Інженерія ПЗ займається не тільки технічними питаннями виробництва ПЗ, але й питанням взаємодії замовників і розробників, управлінням програмними проектами, включаючи питання планування, фінансування, управління колективом і т.д.

Частково теоретичні основи інженерії ПЗ складає КН. Інженер з ПЗ повинен знати інформатику. Так же, як інженер з електроніки повинен знати фізику. В ідеалі, інженерія ПЗ повинна бути підтримана теоріями КН, але це не завжди так. Програмні інженери часто використовують прийоми, які застосовуються тільки у конкретних умовах, вони не можуть бути узагальнені на інші випадки, а теорії КН не завжди можуть бути застосовані до реальних великих систем. КН – це не єдиний теоретичний фундамент ПІ, так як коло проблем перед програмним інженером значно ширше просто написання програм. Доповнює теоретичну основу ПІ управління фінансами, організація робіт в колективі, взаємодія з замовником, управління проектом і т.д. Рішення цих проблем вимагає фундаментальних знань, які виходять за рамки інформатики.

Комп'ютерна програма – це, на відміну від об'єктів інших інженерій, нематеріальний об'єкт. Звідси слідують принципові відмінності. Фаза виробництва полягає в копіюванні зразка на інші носії збереження інформації. Вартість фази дуже мала. Якщо кодування вважати елементом проектування, то відсутня також і фаза створення зразка.

Комп’ютерна програма – штучний об’єкт. Для програми немає об’єктивних законів, яким би підкорялася її поведінка. Наприклад, у інженера-будівельника є об'єктивні закони будівельної механіки: рівноваги моментів і сил, стійкості механічних систем і т.д. Інженер-будівельник може перевірити свої архітектурні рішення на відповідність до цих законів і тим самим забезпечити успіх проекту. Ці закони об'єктивні, вони будуть діяти завжди. У програмного інженера на перший погляд також є типові, перевірені часом архітектурні рішення (наприклад, клієнт-серверна архітектура). Але ці рішення визначаються рівнем розвитку обчислювальної техніки і адекватним їм рівнем вимог. З появою техніки з принципово новими можливостями програмного інженерові доведеться шукати нові рішення.

Відсутність «теоретичного» контролю проекту зумовлює те, що тестування програмного продукту – це єдиний спосіб впевнитися в його якості. Вартість тестування складає істотну частину вартості ПЗ**.** До речі, інженер-будівельник, як правило, позбавлений можливості такого «тестування» свого продукту перед здачею його в експлуатацію.

Інженерія ПЗ – молода дисципліна, досвід якої нараховує всього декілька десятків років. У порівнянні з досвідом будівельної інженерії, для якої – тисячоліття, це звичайно дуже мало. Інженерію програмного забезпечення іноді порівнюють з ранньою будівельною інженерією, коли закони будівельної механіки ще не були відомі й інженери-будівельники діяли методом спроб і помилок, накопичуючи безцінний досвід. Незважаючи на молодий вік, інженерія програмного забезпечення також накопичила певний досвід, який дозволяє при раціональному його застосуванні робити вдалі проекти. Цей досвід виражений в основних принципах інженерії програмного забезпечення. Щоб стати наукою, розроблення ПЗ повинне піддатися зрушенню парадигми від методу спроб і помилок до системи основних принципів.

SWEBOK (Software Engineering Body of Knowledge) – це документ, що готує комітет Software Engineering Coordinating Committee зі спільнотою IEEE Computer

Society. Призначення SWEBOK – це об'єднання знань з інженерії програмного забезпечення.

У документі редакції 2004 року визначаються десять областей знань в контексті інженерії програмного забезпечення (Додаток А):

- Вимоги до ПЗ (Software requirements, англ.);
- Проектування (Software design, англ.);
- Конструювання (Software construction, англ.);
- Тестування (Software testing англ.);
- Супровід (Software maintenance, англ.);
- Керування конфігурацією (Software configuration management, англ.);
- Керування проектами (Software engineering management, англ.);
- Процеси інженерії програмного забезпечення (Software engineering process, англ.);
- Засоби та інструменти (Software engineering tools and methods, англ.).
- Якість ПЗ (Software quality, англ.).

Також SWEBOK визначає дисципліни, що відіграють велику роль в інженерії програмного забезпечення: комп'ютерна інженерія, комп'ютерні науки, менеджмент, математика, контроль якості, ергономіка ПЗ, системне адміністрування.

Контрольні питання

1. Дайте означення, що таке інженерія програмного забезпечення.
2. Які обмеження існують на розроблення й використання ПЗ?
3. Які ознаки інформаційного суспільства?
4. Які задачі розв’язують фахівці комп’ютерних наук?
5. З чого складається інформаційна система?
6. Що таке інформаційна технологія?
7. Які задачі розв’язують фахівці комп’ютерної інженерії?

8. Що спільного й відмінного між інженерією програмного забезпечення і комп'ютерними науками?

9. Що спільного й відмінного між інженерією програмного забезпечення і комп'ютерною інженерією?

10. Що є теоретичною основою інженерії програмного забезпечення?

11. Наведіть приклад застосування інформаційних технологій у повсякденному житті?

12. Поясніть чим програмування відрізняється від інженерії ПЗ?

13. Коли вперше було використано термін Software Engineering?

14. Як комп'ютерні науки пов'язані з інформатикою?

15. Які характерні риси має інженерна діяльність?

16. Які основні принципи застосування інформаційної технології?

17. Що таке інформаційний продукт? Чим інформаційний продукт принципово відрізняється від просто інформації?

18. Яке місце інженерії ПЗ серед інших комп'ютерних дисциплін? Назвіть базові комп'ютерні дисципліни?

19. Які принципові відмінності інженерії ПЗ від інших інженерій?

20. Що є на заміну «теоретичному» контролю розроблення ПЗ у програмного інженера?

21. У чому полягає фаза виробництва «зразка» у галузі розроблення ПЗ?

22. Порівняйте виробництво мікросхем з виробництвом ПЗ у розрізі інженерної діяльності. Які є принципові відмінності?

23. У чому полягає необ'єктивність законів, типових рішень, що використовуються в інженерії ПЗ?

24. Наведіть приклади апаратно-програмних комплексів.

25. Охарактеризуйте галузь Internet of Things (IoT). Які завдання вирішуються в цій галузі фахівцями ІПЗ, КН і КІ?

26. Наведіть приклад розробки галузі IoT.

27. Порівняйте найвідоміші означення терміну «інженерія ПЗ».

28. Що таке SWEBOK? Яка його мета?
29. Які області знань ІПЗ є головними згідно SWEBOK?
30. Які області знань ІПЗ належать до організаційних згідно SWEBOK?
31. Назвіть дисципліни, які є важливими для ІПЗ згідно SWEBOK.
32. Що є результатом систематизації знань галузі ІПЗ?
33. Що зумовило виокремлення інженерії ПЗ з комп'ютерних дисциплін?
34. У чому суть концепції SLC в інженерії ПЗ?

Тема 1.2. СТАНОВЛЕННЯ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Інженерія програмного забезпечення пройшла достатньо динамічний розвиток. Вона розвивалася й розвивається, в основному, у відповідності до постановок нових задач побудови складних програмних систем для промисловості, урядового управління, військової галузі, фінансової й банківської справи, комерції, освіти, індустрії розваг. Етапи розвитку інженерії програмного забезпечення можна виокремити по-різному. Кожен етап пов'язаний з появою чи усвідомленням чергової проблеми в галузі розроблення програм і знаходженням шляхів та способів її вирішення. Проблеми пов'язані, в першу чергу, із зростанням вартості розроблення ПЗ, і як наслідок – збільшення термінів виконання проектів, зниження якості програмних продуктів.

У кінці 60-х – на початку 70-х років минулого століття відбулася подія, яка ввійшла в історію як перша криза програмування. Вартість ПЗ стала наближатися до вартості апаратного забезпечення. Тоді і заговорили про інженерію програмного забезпечення (або технології програмування) як про деяку дисципліну, ціллю якої є скорочення вартості програм.

Для вирішення кожної чергової проблеми було розроблено новий метод, що стосувався технології програмування. Ці методи й на сьогодні складають основу підходів до розроблення програмних систем.

У ракурсі технологій програмування можна виділити в історії розвитку інженерії ПЗ такі етапи:

- ✓ перший етап (60-70 –ті роки 20 століття) стосується становлення модульного методу розроблення ПЗ (технології модульного програмування);

- ✓ другий етап (70-80 –ті роки 20 століття) пов'язаний зі становлення структурного методу розроблення ПЗ (технологій структурного програмування);
- ✓ третій етап (80-90 –ті роки 20 століття) стосується становлення об’єктно-орієнтованого методу розроблення ПЗ (об’єктно-орієнтованого програмування);
- ✓ четвертий етап (наші часи) стосується розвитку об’єктно-орієнтованого й пост-об’єктного програмування.

Проблема повторного використання коду

На початку становлення інженерії ПЗ програмування характеризувалося як «стихійне». Накопичувався досвід виконання реальних проектів, не застосовувався системний підхід до процесу розроблення ПЗ.

Проте вже на перших етапах становлення інженерії ПЗ було помічено, що висока вартість програм пов’язана з розробленням однакових чи подібних фрагментів коду в різних програмах. Викликано це було тим, що в різних програмних системах як частини вирішувалися одинакові або подібні задачі. Наприклад, для задач математичного моделювання різних об’єктів і процесів: знаходження розв’язку нелінійних рівнянь, обчислення визначеного інтегралу і т.п.; для підприємств і організацій розрахунок заробітної плати працівників, обчислення доходів і т.п. Використання при створенні нових програм раніше написаних фрагментів обіцяло істотне зниження термінів і вартості розробки. Таким чином, відбулося зародження модульного методу розроблення ПЗ як методу уникнення дублювання однакового програмного коду.

Модуль – програмний фрагмент, який є будівельним блоком для побудови великих програм. Як правило, модуль складається з інтерфейсної частини та частини реалізації. Він є функціонально завершеним і логічно незалежним. Головний принцип модульного програмування полягає у виділенні однакових чи подібних фрагментів і оформленні їх у вигляді модулів.

Основні концепції модульного програмування:

— Кожен модуль реалізує єдину незалежну функцію (перелік регламентованих операцій);

— Кожен модуль має єдину точку входу і виходу;

— Розмір модуля за можливості повинен бути мінімізований;

— Кожен модуль може бути розроблений і закодований різними членами команди програмістів і може бути окремо протестований;

— Вся система побудована з модулів так, щоб обмін інформацією між модулями був за можливістю мінімізований;

— Модуль не повинен давати побічних ефектів;

— Кожен модуль не залежить від того, як реалізовані інші модулі.

Незалежність модулів є важливою основою модульного програмування. Програма має бути поділена на модулі таким чином, щоб зв'язки між модулями були слабкими, а всередині – сильними. Найпростішим модулем є функція-підпрограма, яка багаторазово використовується через відповідні виклики з фактичними параметрами.

Кожний модуль супроводжується описом, в якому встановлюються правила його використання – інтерфейс модуля. Інтерфейс задавав зв'язки модуля з основною програмою – зв'язки за даними і зв'язки за управлінням. При цьому можливість повторного використання модулів визначалась кількістю і складністю цих зв'язків, або наскільки ці зв'язки вдалось узгодити з організацією даних і управлінням основної програми. Найбільш простими в цьому відношенні виявились модулі розв'язування математичних задач: розв'язування рівнянь, систем рівнянь, задач оптимізації.

Для багатьох інших типів модулів можливість їх повторного використання виявилась проблематичною внаслідок складності їх зв'язків з основною програмою. Повторне використання модулів з складними інтерфейсами є достатньо актуальною і по цей день. Для її вирішення розробляються спеціальні форми (структури) представлення модулів і організації їх інтерфейсів.

Слід зазначити, що технології модульного програмування постійно розвивалися й розвиваються далі на потребу інженерії ПЗ. Модуль пройшов шлях від найпростішого тлумачення як функція до найскладнішого системної компоненти.

Закономірність розвитку модуля як незалежної частини системи можна представити від функції до аспекту (Рис)

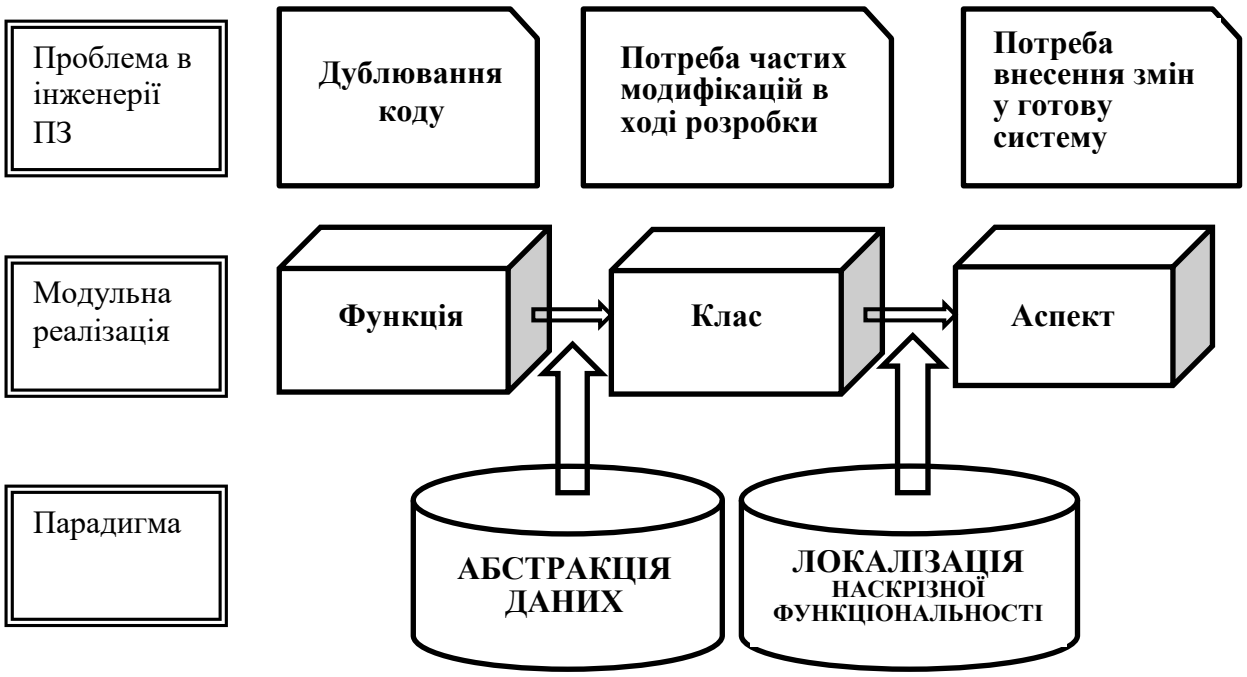


Рис. 1.3. Розвиток поняття модуля в інженерії ПЗ.

Кінцева мета модульного програмування – дати інструменти для конструювання довільної системи з готових вже частин. Як готовий автомобіль сходить з конвеєра, так і програмна система має автоматизовано конструюватися з готових компонентів.

Отже, проблема дублювання фрагментів коду була вирішена **модульним програмуванням**, яке полягає у виділенні фрагментів програмного коду в модулі, їх повторному використанні і створенні бібліотеки модулів.

Проблема росту складності програм

Наступний етап зростання вартості ПЗ був пов’язаний з переходом від розроблення відносно простих програм до розроблення складних програмних комплексів. До числа таких складних програм відносяться: системи управління космічними об’єктами, управління оборонним комплексом, автоматизації великого фінансового закладу і т.д. Складність таких комплексів оцінювалась такими показниками:

1. Великий об’єм коду (мільйони рядків);
2. Велика кількість зв’язків між елементами коду;
3. Велика кількість розробників (сотні людей);
4. Велика кількість користувачів (тисячі);
5. Тривалий час використання (роки).

Для таких складних програм тривалого використання виявилось, що значна частина їх вартості припадає не тільки на створення безпосередньо програмного коду, а на їх впровадження і експлуатацію. За аналогією до промисловості стали говорити про життєвий цикл програмного продукту, як про послідовність визначених етапів: аналізу вимог, етапу проектування, розроблення, тестування, впровадження і супровід.

Етап супроводу програмного комплексу включає дії з виправлення дефектів у роботі програми, а також і внесенню змін у відповідності із новими вимогами користувачів. Основна причина високої вартості етапу супроводу пояснювалася тим, що програми були погано спроектовані – документація була не зрозуміла і не відповідала програмному коду, а сам програмний код був дуже складний, заплутаний, погано структурований. Потрібна була технологія, яка б забезпечила «правильне» розроблення (проектування і кодування) програмних систем.

Основні принципи структурного методу розроблення ПЗ можна виокремити так:

1. Низхідне функціональне проектування системи.

2. Застосування спеціальних мов проектування і засобів автоматизації використання цих мов.
3. Дисципліна розроблення ПЗ.
4. Структурне кодування без goto

Принцип «розділяй і володарюй» у галузі ІПЗ вказує на потребу виділяти у системі основні функціональні підсистеми, які потім розбиваються на підсистеми й т.д. Така функціональна (алгоритмічна в той час) декомпозиція є важливим принципом для застосування модульних технологій програмування.

Щоб автоматизувати процес розроблення програм почали використовувати CASE-засоби (Computer-Aided Software Engineering, англ.) – засоби інженерії ПЗ для автоматизованого розроблення програм. За допомогою CASE-засобів щонайменше автоматизують процеси проектування інтерфейсів, документування й генерування структурованого коду бажаною мовою програмування. Сучасні CASE-системи у повній мірі реалізують інженерний підхід до розроблення ПЗ й/або його технічного обслуговування.

Типовими CASE інструментами є інструменти моделювання даних; інструменти аналізу й проектування; інструменти перетворення моделей; інструменти редагування програмного коду; інструменти рефакторингу коду; генератори коду; інструменти для побудови UML-діаграм та інші.

Дисципліна розроблення ПЗ передбачає планування робіт і документування результатів проекту, контроль поточних результатів, підтримку відповідності коду проектній документації. Поняття життєвого циклу для промислового виробництва, адаптоване до галузі ІПЗ, допомагало дотримуватись системного підходу у розробці програмних систем.

Структурне кодування (програмування) передбачало для організації обчислювального процесу використання чітко визначених конструкцій (лінійний блок, розгалуження, цикл). Використання безумовного переходу не

віталося. Код завдяки цьому не був заплутаним, а ставав зрозумілим і не для його авторів.

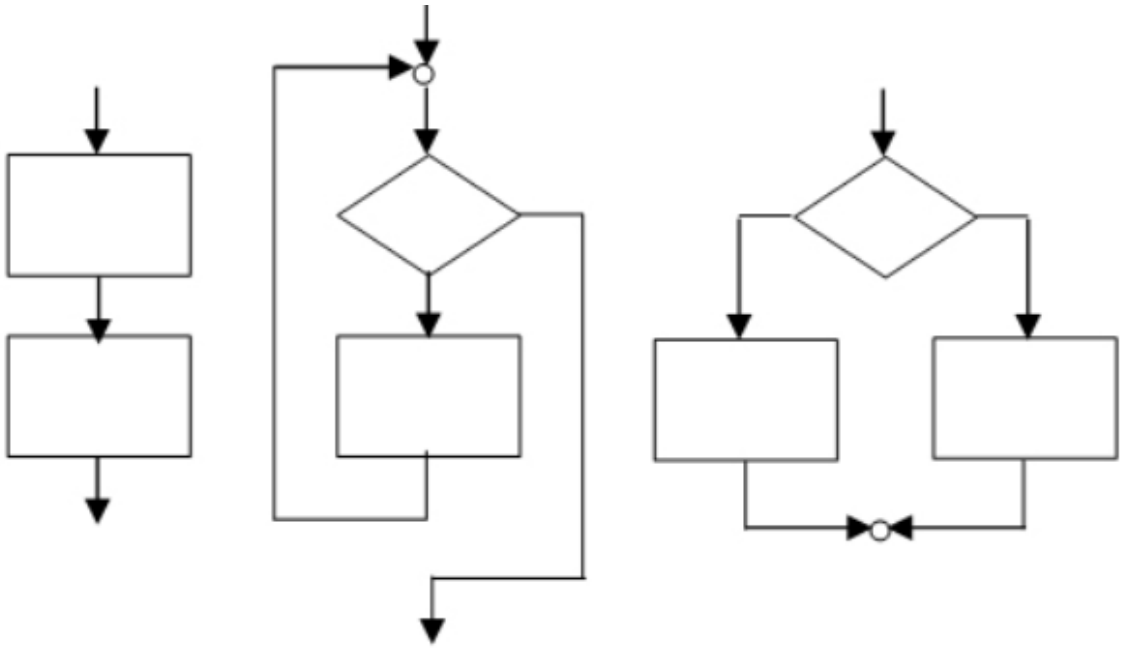


Рис. 1.4. Блоки «послідовність», «цикл», «розгалуження».

Отже, проблема складності програмних систем була вирішена **структурним методом розроблення ПЗ**, що базується на принципі життєвого циклу, та полягає у «правильному» проектуванні і кодуванні, а саме: функціональна декомпозиція , застосування спеціальних засобів для автоматизації розроблення ПЗ, документування проекту, структурне кодування.

Проблема потреби модифікації програм ще до її завершення

Наступна проблема росту вартості програм була зумовлена тим, що зміни вимог до програми стали виникати не тільки на стадії супроводу, але і вже на етапі розроблення. Це так звана проблема замовника, який не знає, що він хоче. Виникло питання, як проектувати і створювати програми, щоб забезпечити можливість внесення змін в програму, не змінюючи кардинально раніше написаний код.

Вирішенням цієї проблеми стало використання метода, який отримав назву об’єктно-орієнтоване проектування й програмування (ООП). Суть підходу полягає в тому, що вводиться поняття класу як розвиток поняття модуля з визначеними властивостями і поведінкою, які характеризують клас. Об'єкти і класи – основні абстракції предметної області. Об’єкт є екземпляром (представником) класу, клас, в свою чергу, є об’єднанням однотипних елементів.

Абстракцією є будь-яка модель, яка включає найбільш важливі, істотні або відмітні характеристики деякого об'єкту, і яка ігнорує менш важливі або незначні деталі. Абстрагування дозволяє управляти складністю системи, концентруючись на істотних властивостях об'єкту. Абстракція залежить від предметної області і точки зору. Те, що важливе в одному контексті, може бути не важливе в іншому. Вибір правильного набору абстракцій для заданої предметної області є головним завданням об'єктно-орієнтованого проектування. Кожний клас може породжувати об’єкти – екземпляри даного класу. Програма при такому підході – це взаємодія об’єктів. Таким чином, на зміну функціональній декомпозиції системи приходить об’єктна декомпозиція.

Об’єктно-орієнтована парадигма доводить до логічної завершеності принцип моделювання реального світу, а точніше – тієї його частини, абстракцією якої є програма. За такого підходу програма складається з об’єктів, які відповідають реальним поняттям або предметам, а її виконання зводиться до взаємодії цих об’єктів, що є абстракцією реальної взаємодії їхніх прототипів.

При цьому працюють основні принципи ООП:

1. Інкапсуляція – об’єднання в класі даних (властивостей) і методів (процедур обробки) з приховання деталей їхьреалізації.
2. Успадкування – можливість виведення нового класу зі старого з частковою зміною властивостей і методів
3. Поліморфізм – визначення властивостей і методів об’єкта за контекстом.

Інкапсуляція – локалізація властивостей і поведінки у рамках єдиної абстракції (яка розглядається як «чорний ящик»), що приховує реалізацію за загальнодоступним інтерфейсом. При інкапсуляції відділяється внутрішній устрій об'єкта від його зовнішньої поведінки. Об'єктний підхід припускає, що внутрішні ресурси об'єкта, приховані від зовнішнього середовища. Абстрагування й інкапсуляція є взаємодоповнюючими принципами.

Саме ієрархії виявилися головною рисою новітніх технологій програмування. По-перше, ієрархічні структури дають змогу керувати складністю програмних проектів, відділяючи внутрішню складність конструкції від її зовнішнього використання. По- друге, ієрархічні структури забезпечують ефективні будівельні блоки для конструювання програм, надаючи ємніші програмні конструкції, ніж окремі процедури або функції. І, нарешті, ієрархічні структури забезпечують можливість пристосування до нових умов вже наявних програмних кодів, не втручаючись у їхню внутрішню будову.

Поняття поліморфізму може бути інтерпретоване, як здатність об'єкту належати більш ніж одному типу. Поліморфізм – це здатність приховувати множину різних реалізацій під єдиним загальним інтерфейсом. Інтерфейс – це сукупність операцій, які визначають набір послуг класу або компоненти. Інтерфейс не визначає внутрішню структуру, а всі відкриті операції.

Все це разом забезпечує об’єктно-орієнтованому підходу беззаперечне лідерство в галузі розроблення програм.

Отже , проблема внесення змін у проект і програму без переписування раніше створеного коду, вирішувалося методом **об’єктно-орієнтованого проектування**, основною ідеєю якого є абстракція даних у вигляді класу, а принципами інкапсуляція, успадкування, поліморфізм.

Таким чином, інженерія програмного забезпечення як напрям інформаційних технологій виникла і формувалася під впливом зростання

вартості створюваного програмного забезпечення. Головна мета цієї галузі – скорочення вартості і термінів розробки програм.

Інженерія програмного забезпечення пройшла кілька етапів розвитку, в процесі яких були сформульовані фундаментальні принципи та методи розроблення програмних продуктів.

Основний принцип інженерії програмного забезпечення полягає в тому, що програми створюються в результаті виконання декількох взаємозв'язаних етапів (аналіз вимог, проектування, розроблення, впровадження, супровід). Ці етапи складають життєвий цикл програмного забезпечення. Фундаментальними методами розроблення ПЗ є модульний, структурний і об'єктно-орієнтований методи.

Пост-об’єктні методи програмування

Мета усіх пост-об’єктних методів програмування розвинути ООП для полегшення розроблення ПЗ, давши розробнику на замін класу інше поняття модуля, зокрема аспект, сервіс, агент тощо.

Аспектно-орієнтоване програмування (АОП) – парадигма програмування, в основі якої лежить ідея виділення наскрізний функціональсті в окремі сутності – аспекти (aspects). Наскрізною називають функціональсть, розосереджену по різних частинах програми. Як приклади наскрізної функціональності можна привести завдання логування, трасування, обробки виключних ситуацій, перевірки прав доступу. АОП може так само використовуватися для вирішення завдань захисту і багатопотоковості. Тобто під аспектом розуміють деяку програмну одиницю (модуль), яка забезпечує деяку ціль функціонування системи. Таким чином, АОП забезпечує оптимальну модульність програми, що досягається локалізацією наскрізних вимог у аспектах. АОП спрощує супровід програмної системи та внесення змін, дає нові можливості повторного використання коду.

Сервісно-орієнтоване програмування дозволяє по-новому представити функціонал програмної системи – у вигляді набору сервісів, які є окремо розташовані, самостійні, повноцінні застосунки і можуть перебувати як на одній і тій же машині, так і в мережі. Сервіс – це ресурс, що реалізує деяку функцію і має технологічно незалежний інтерфейс з іншими ресурсами.

Агентно-орієнтоване програмування дозволяє представити програмну систему через поведінку агентів (спеціальних модулів, які є самодостатніми програмами), причому поведінка залежить від середовища, в якому вони знаходяться.

Сучасний стан ІПЗ – продовження кризи розроблення ПЗ

Незважаючи на те, що інженерія ПЗ досягла певних успіхів, криза розроблення ПЗ триває. Пов'язано це з інтенсивним розвитком інформаційних технологій, зокрема появою нових апаратних засобів і потребою у відповідному ПЗ.

Межа 80-90-х років минулого століття відзначається як початок інформаційно-технологічної революції, викликаної вибуховим зростанням використання персонального комп'ютера, мережі Internet, локальних та глобальних обчислювальних мереж, мобільного зв'язку. Через десятки років новий етап у розвитку ІТ пов'язують з використанням різних платформ, гіпер-, медіатехнологій, великих обсягів даних, хмарних технологій, необхідністю в інтелектуальності ПЗ. Все це зумовлює постійну потребу в нових технологіях розроблення ПЗ.

Науково-дослідна компанія Standish Group International, що надає консультативні послуги в сфері розроблення ПЗ, аналізує впродовж багатьох років дані американських компаній стосовно програмних проектів. Така статистика стосовно 50000 проектів з розроблення ПЗ показує розподіл між:

- успішними – вчасно і в рамках бюджету був виконаний весь запланований фронт робіт;

- відхиленими (проблемними) – порушення термінів, перевитрата бюджету та / або зробили не все, що було потрібно;
- провалениими – не були доведені до кінця з-за перевитрати коштів, часу.

Останній звіт Standish Group International Inc. свідчить, що ситуація в індустрії ПЗ має й надалі ознаки кризи (Рис.1.5).

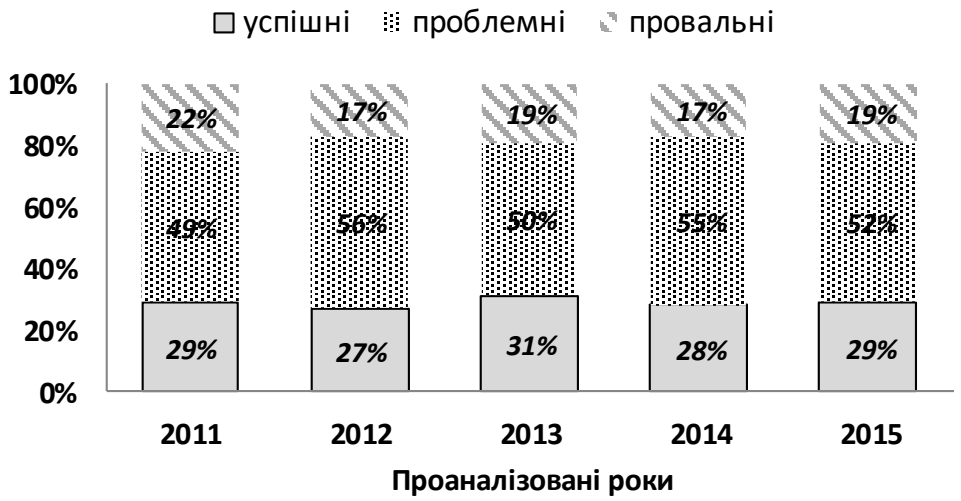


Рис. 1.5. Дані про проекти зі звіту компанії Standish Group International.

Причинами неуспішного завершення проектів найчастіше вважається:

- Недостатнє планування робіт;
- Низький рівень залучення замовника та користувачів до процесу розроблення;
- Бюджет і терміни проекту визначаються без достатнього урахування об’єму робіт, необхідних для виконання проекту;
- Складність опису реальних бізнес-вимог на старті проекту;
- Відсутність чітких вимог до ПЗ з боку замовника чи майбутнього користувача.

Для перетворення інженерії програмного забезпечення в спеціальність світова комп'ютерна громадськість створила професійні комітети, що регламентують аспекти процесу програмування: ядро знань SWEBOK, етичний кодекс програміста, навчальні курси (Curricula -2001, 2004) з підготовки фахівців в області інженерії програмного забезпечення, навчання спеціальності й сертифікація фахівців.

Інженерія програмного забезпечення наголошує на підвищенні якості й продуктивності ПЗ за рахунок застосування: нових й удосконалених методів проектування ПЗ; готових компонентів і методів їхньої генерації; методів еволюції, верифікації й тестування ПЗ; інструментальних засобів; методів керування проектами, оцінки якості й вартості.

Розроблення й використання комп'ютерних програм у наш час стали масовою діяльністю. Більше 7 мільйонів людей займаються їхнім розробленням, а сотні мільйонів активно їх використають. Практично немає жодної сфери діяльності людини (промисловість, економіка, медицина, бізнес, комерція, освіта і т.д.), де б ПЗ не використалося, як засіб автоматизації й покращення робіт. Попит на ПЗ постійно збільшується, складність розробок росте, і кількість помилок, що виникають в ході розроблення, не зменшується.

Знання розробників ПЗ відрізняються розмаїтістю й, як правило, вони є неповними, неузгодженими й різнорідними, орієнтованими на реалізацію різних предметних областей, починаючи від ОС і закінчуючи прикладними бізнесами-системами. І найголовніше, знання в процесі інженерної діяльності поступово уточнюються, видозмінюються й поповнюються, і їх необхідно враховувати розробникам нового ПЗ.

Приблизно кожні 10 років відбувається зміна мов програмування й операційних середовищ для опису й функціонування програм, що припускає переклад раніше виготовлених і функціонуючих програм на нові мови й операційні середовища. На це витрачаються величезні людські й фінансові ресурси. Так, в 2000 році в зміні формату дати в програмах і мікросхемах на

десятках млн. комп'ютерів брало участь більше 2 мільйонів програмістів, а витрати склали сотні мільйонів доларів.

У зв'язку з постійним оновленням персональних комп'ютерів, цифрових пристроїв, відповідно, операційних середовищ і систем програмування, виникають складності, пов'язані з адаптацією діючих програм і прикладних систем до нових умов. У практиці програмування не раз робилися спроби полегшити працю з написання програм, перейшовши до "програмування без програмістів". У зв'язку із цим з'являлися програмні проекти, які ставили своєю метою замінити постановки завдань математичним описом і змусити обчислювальні машини їх опрацьовувати. До них відносяться комп'ютеризація математичних знань (машини серії "Мир", японський проект "ЕОМ 5-го покоління"), фабрики програм з виготовлення програмної продукції по типу складального конвеєра з готових "деталей"-програм (завод для зборки АСУ, система АПРОП, ПРИЗ) і багато інших проектних рішень, спрямовані на зміну концепції програмування. Наприклад, формальні специфікації й математичне доведення правильності програм (1980 р.), систематизація знань в області інженерії програмного забезпечення й створення загального ядра знань SWEBOK (2001, 2003р.), теорія побудови й верифікації програм (проект 2005 р.) і ін.

Переворот у програмуванні з вивільнення армії користувачів комп'ютерів від розроблення й написання програм так і не відбувся, навпаки, методи й засоби програмування постійно розвиваються. Особливо це пов'язане з новими можливостями платформ і розподілених середовищ, у яких компоненти розташовуються на різних вузлах мережі й взаємодіють між собою через мережні протоколи. Крім того, з'явилися нові методи й підходи до розроблення ПЗ: структурний, об'єктно-орієнтований, компонентний, аспектний, візуальний, агентно-орієнтований, сервісний й ін.

Для підтримки нових методів розроблена величезна кількість різноманітних інструментальних засобів і методів оцінки якості,

продуктивності, вартості й т.п. Процес розроблення ПЗ й методи оцінювання продуктів стандартизовані (ISO/IEC 12207, 15504, 9126 й ін.). Все це сприяє підвищенню ефективності проектування, тестування, прогнозування надійності й оцінки якості ПЗ.

Новий програмний проект масштабного рівня розробляється 1-2 року, а еволюціонує 6-7 років. На супровід проекту витрачається 61% проти 39% засобів на його розроблення. На сьогодні ядро стабільних знань з інженерії програмного забезпечення становить близько 75% від тих знань, якими користуються в практичній діяльності. У зв'язку із цим проведена систематизація накопичених знань у програмуванні й ряді інших областей інформатики. Міжнародним комітетом при американському об'єднанні комп'ютерних фахівців ACM (Association for Computing Machinery) і інституті інженерів з електроніки й електротехніки IEEE Computer Society було створене ядро знань SWEBOK. У цьому ядрі були систематизовані різнорідні знання в області програмування, планування й керування, сформульоване поняття інженерії програмного забезпечення й десяти областей, які відповідають процесам розроблення ПЗ (Додаток А).

Основними перспективами розвитку інженерії ПЗ є такі напрямки:

- Розроблення єдиної теорії створення та аналізу програм;
- Побудова інтегрованого набору інструментів верифікації для всіх етапів ЖЦ;
- Створення репозиторію формальних специфікацій для верифікованих програм;
- Мовно-орієнтоване програмування;
- Комп’ютеризація математичних знань тощо.

Контрольні питання

1. Які відомі етапи розвитку ІПЗ?

2. Які ознаки кризи ІПЗ?
3. Яку проблему вирішує метод модульного програмування?
4. Охарактеризуйте модульний метод розроблення ПЗ.
5. Опишіть основні концепції модульного програмування.
6. Що таке інтерфейс модуля? Наведіть приклад.
7. Яку проблему вирішує метод структурного програмування?
8. Охарактеризуйте структурний метод розроблення ПЗ.
9. Які є показники складності програмних систем?
10. Як принцип життєвого циклу застосовується в галузі розроблення ПЗ?
11. У чому полягає складність етапу супроводу програмних систем?
12. Що таке функціональна декомпозиція програмної системи? Наведіть приклад застосування цього принципу.
13. Що і для яких конкретних задач використовується з метою автоматизації розроблення ПЗ?
14. Яку проблему вирішує метод об’єктно-орієнтованого програмування?
15. Охарактеризуйте об’єктно-орієнтований метод розроблення ПЗ.
16. Чому розробники не мають відразу готових сформованих вимог до ПЗ?
17. Наведіть приклади принципів інкапсуляції, успадкування, поліморфізму з реального світу (ручка, вікно, студент тощо).
18. Що є базовою ідеєю ООП, яка робить цю парадигму людино-зорієнтованішою?
19. Як модульне програмування пов’язане з об’єктно-орієнтованим програмуванням?
20. Наведіть приклади пост-об’єктних методів розроблення ПЗ.
21. З чим пов'язаний розвиток пост-об’єктних методів розроблення ПЗ?
22. Що є базовою ідеєю АОП? Які є мови програмування, що підтримують АОП?
23. Що таке клас? Що таке аспект?
24. Наведіть приклади модулів програмної системи.

25. Яка тенденція розвитку методів розроблення програмних систем?
26. Наведіть дані, що засвідчують продовження кризи інженерії ПЗ.
27. Охарактеризуйте сучасний стан ІПЗ.
28. Як відбувається систематизація знань в галузі ІПЗ?
29. Охарактеризуйте поняття інформаційно-технологічної революції.
30. З якими галуззями пов'язаний розвиток ІПЗ?
31. Наведіть приклади науково-технічних проектів, які мали за мету змінити принципи програмування, переклавши «відповідальність» від людини на комп’ютер.
32. Які є типові причини неуспішного виконання проектів з розроблення ПЗ?
33. Які дані останнього звіту Standish Group International Inc. свідчать, що ситуація в індустрії ПЗ має й надалі ознаки кризи?
34. Що представлено у SWEBOK?

Тема 1.3. СКЛАДОВІ ЕЛЕМЕНТИ ЕТАЛОННОЇ МОДЕЛІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Промислове програмування як інженерна діяльність характеризується такими складовими частинами:

- **Продукт**
- **Проект**
- **Процес**
- **Персонал**

Також для позначення важливості і взаємозв'язку цих елементів використовують термін «еталонна модель інженерії програмного забезпечення у вигляді «чотирьох П»

Рис.1.6. Зв'язок елементів еталонної моделі інженерії ПЗ.

Проект складається з елементарних частин – процесів, виконавцями проекту є персонал, а результатом виконання проекту є програмний продукт.

Програмний продукт

Чим «програмний продукт» відрізняється від «просто програми»?

«Просто програма» придатна для використання своїм автором в системі, де й була розроблена. «Програмний продукт» – програма (програмний комплекс), яку будь-яка людина може запускати, тестувати, виправляти і розвивати.

Програмний продукт – набір комп'ютерних програм, процедур і, можливо пов'язаних з ними документації та даних. Поняття продукту не обмежується програмним кодом. Продукт – це артефакти, створювані протягом усього життя проекту, такі як моделі, тексти програм, виконувані файли і документація.

Програмний продукт (ПП) повинен

- бути досить універсальним (в рамках розв'язуваних їм завдань),
- стійко працювати в різних ситуаціях (аварійних, некоректної роботи користувача),
- мати достатньо повну документацію.

За оцінкою Фредеріка Брукса, ПП зазвичай коштує як мінімум втричі дорожче, ніж просто налагоджена програма з такою ж функціональністю.

До базових характеристик якості програмного продукту належать:

1. *функціональність (functionality);*
2. *надійність (reliability);*
3. *зручність застосування (usability);*
4. *ефективність (efficiency);*
5. *супровожуваність (maintainability);*
6. *переносність (portability).*

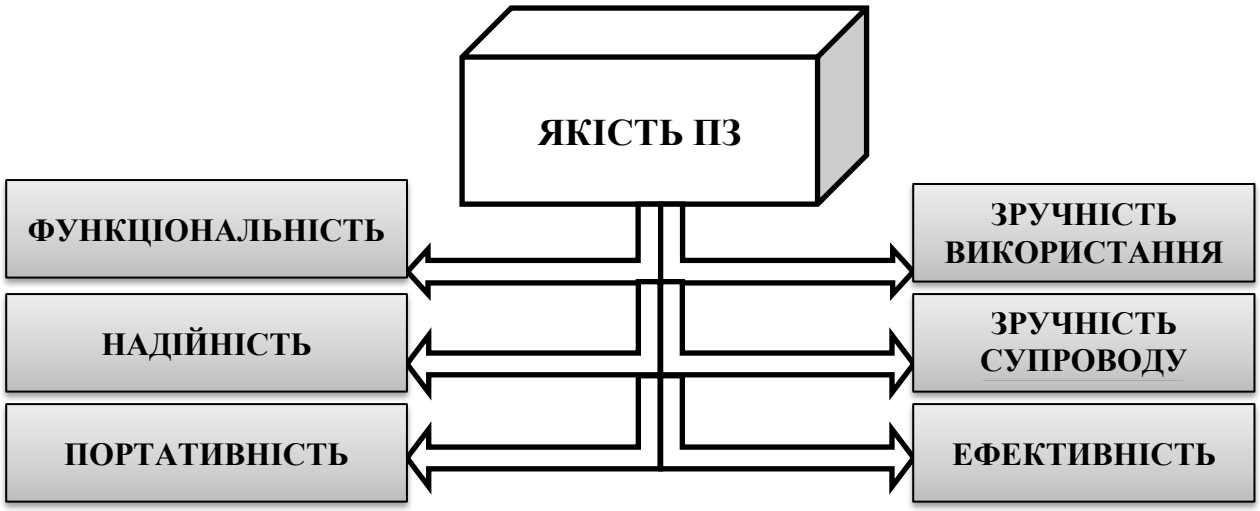


Рис. 1.7. Складові якості ПЗ.

Функціональність – перелік задач, які вирішуються програмною системою, це корисність програми.

Надійність ПЗ включає такі елементи як:

- Відмовостійкість – можливість виконання функцій у випадку нестандартних, непередбачених ситуацій чи збоїв у роботі
- Безпека – збої в роботі програми не повинні призводити до небезпечних наслідків (аварій)
- Захищеність від випадкових або навмисних зовнішніх впливів (захист від «нерозумного» користувача, вірусів, спаму)

ПЗ повинно бути легким й простим у використанні, причому саме тим типом користувачів, на яких розраховано програму. Це поняття стосується інтерфейсу користувача і адекватної документації. Причому, користувацький інтерфейс повинен бути не інтуїтивно, а професійно зрозумілим користувачеві.

ПЗ не має даремно витрачати системні ресурси, такі як пам'ять, процесорний час, канали зв'язку. *Ефективність ПЗ* оцінюється наступними показниками: час виконання коду, завантаженість процесора, обсяг необхідної пам'яті, час відповіді і т.п

Здатність до розвитку ПЗ – це критична властивість системи, тому що зміни ПЗ неминучі внаслідок зміни бізнесу. Супровід програми виконують, як правило, не ті люди, які її розробляли. Супровід включає такі елементи як наявність і зрозумілість проектної документації, відповідність проектної документації вихідному коду, зрозумілість вихідного коду, простота змін вихідного коду, простота додавання нових функцій.

Переносність (портативність) – здатність використовувати ПЗ на різних платформах – операційних системах, середовищах, пристроях.

Проект

Єдиний спосіб розробити програмне забезпечення – це ініціювати проект з його створення.

Вважалося і вважається досі, що проекти в середовищі високих технологій, до яких відноситься і розроблення ПЗ, дуже важко планувати і виконувати, і що доказ цього – низькі показники успішності таких проектів.

Ще в 70-роках Фредерік Брукс стверджував, що незважаючи на всі досягнення в області створення апаратних і програмних засобів, не було ніякого відповідного розвитку ні в технології, ні в методиці управління, який дозволив би виконувати програмні проекти вчасно, в межах узгодженого бюджету та вимог до якості кінцевого продукту. Здавалося очевидним, що методи, які довели свою ефективність в будівництві або кінематографії абсолютно точно не спрацьовують в умовах змінних вимог, технологій, що швидко розвиваються, та нерегламентованої жорсткими рамками творчого процесу створення ПЗ.

Проект – сукупність дій, обмежена в часі і спрямована на отримання унікального результату.

Потрійне обмеження проекту («залізний трикутник») означає, що необхідна якість результату отримується в умовах визначених ресурсів, а саме

коштів і часу. Відповідно підвищення якості може здійснюватися за рахунок додаткового залучення коштів і/або часу. Зменшення ресурсів, відповідно, впливає негативно на якість очікуваного результату.

Рис. 1.8. «Залізний» трикутник проекту.

Закономірності, що проявляються у багаточисленних проектах з розроблення ПЗ, є основою для прийняття рішень в умовах конкретних проектів.

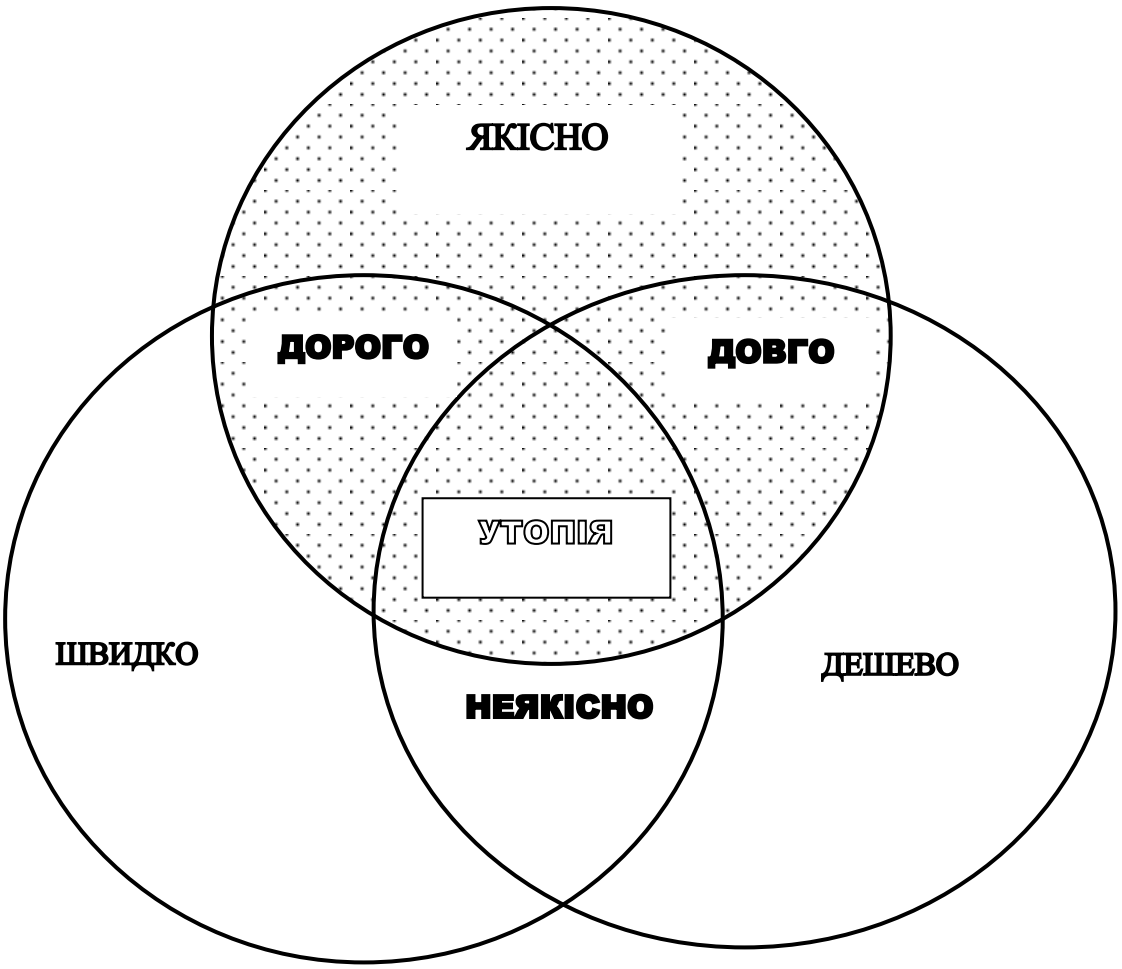


Рис. 1.9. Обмеження в проекті з розроблення ПЗ.

У 1988 році вийшла стаття Баррі Боема «Список 10 правил промислового створення ПЗ», які найбільш повно відображають основні принципи традиційного управління програмними проектами.

10 правил промислового створення ПЗ

1. Пошук і виявлення помилки в ПЗ після його здачі в експлуатацію обходиться в 100 разів дорожче, ніж пошук і виявлення помилки на ранніх стадіях розроблення.

2. Можна скоротити термін розробки ПЗ на 25% від номінального, але не більше.

3. На кожен долар, вкладений в розроблення, доводиться витрачати два долари на супровід.

4. Вартість розробки і супроводу програмного забезпечення є, насамперед, функцією кількісті рядків вихідного коду

5. Відмінності в рівні кваліфікації розробників призводять до величезної різниці в продуктивності при створенні програмного забезпечення

6. Загальне відношення вартості програмного забезпечення до вартості апаратного забезпечення продовжує зростати. У 1955 р воно становило 15:85; в 1985 г. – 85:15

7. При створенні ПЗ всього лише близько 15% зусиль витрачається власне на програмування.

8. Програмні системи і продукти зазвичай коштують в три рази дорожче в перерахунку на один рядок вихідного коду, ніж окремі програми. Продукти, що складаються з програмних систем (тобто системи систем), коштують дорожче в дев'ять разів.

9. Наскрізний контроль дозволяє виявити 60% помилок.

10. 80% роботи виконують 20% виконавців проекту.

Характеристики проекту

- *Мета проекту* – призначення, що та навіщо буде створено, тобто що отримає споживач у результаті реалізації проекту.
- *Вартість проекту* - кошторисні витрати, необхідні для виконання робіт проекту.
- *Обсяги робіт проекту* – кількісні показники робіт.
- *Терміни виконання проекту* – дати початку, закінчення, тривалість
- *Якість проекту* – відповідність характеристик проекту та його результатів встановленим стандартам якості
- *Ресурси проекту* – сукупність ресурсів, потрібних для здійснення проекту: обладнання, матеріали, персонал, програмне забезпечення, виробничі площі і т.д.
- *Виконавці проекту* – фахівці та організації, залучені до виконання робіт проекту, їх кількість, склад і кваліфікація

- *Ризики проекту* – визначення ризикованих подій в проекті, ймовірності їх звершення і збитку від їхнього впливу на проект

Програмний проект має за мету розробку одного з видів ПЗ:

- *споживчого ПЗ,* під яким розуміються програми, розроблені для реалізації в роздріб (наприклад, комп'ютерні ігри, навчальні програми);
- *виробничого ПЗ*, тобто великих систем, які розроблені для придбання компаніями та організаціями без будь-яких модифікацій (САПР, видавничі системи);
- *замовленого ПЗ* – систем, розроблених відповідно до того, що визначено контрактом;
- *програмних компонентів*, виконаних за спеціальним замовленням, для побудови патентованих систем, які складаються з багаторазово використовуваних компонентів;
- *вбудованого ПЗ,* яке є складовою апаратно-програмних комплексів.

Ключовими учасниками проекту є замовник та виконавець.

У межах проекту можуть функціонувати такі схеми взаємодії між замовником і розробником ПЗ

1) «Свій» замовник

Це означає доступність замовника в будь-який час, стійке уявлення про потреби замовника, яке сформувалося за роки співпраці, невисокі вимоги до якості програмного забезпечення, постійний супровід своїх продуктів, їх доопрацювання і вдосконалення. Фінансові відносини між замовником і розробниками є стійкі, плани щодо співпраці є довгострокові.

2) Продукт під замовлення

Це означає доступність замовника є «періодичною», уявлення про потреби замовника формується на базі обстеження та аналізу предметної області, заздалегідь обговорюються вимоги до програмного забезпечення, згідно з якими проводиться приймання готового продукту. Розробник

протягом заздалегідь обумовленого періоду часу підтримує продукт. Фінансові відносини оформляються у вигляді разового контракту.Плани щодо подальшої співпраці значною мірою залежать від якості продукту, який розробник поставив замовнику.

3) Тиражований продукт

Це означає , що конкретного замовника, який формує вимоги до продукту, не існує, уявлення про потреби теперішніх і потенційних користувачів формуються на основі аналізу ринку і контактів з типовими представниками користувачів. Зворотній зв'язок з користувачами продукту здійснюється в основному за допомогою електронної пошти або «гарячої лінії», широко використовується бета-тестування. Розробник протягом заздалегідь обумовленого періоду часу підтримує поставлений продукт. Фінансові відносини існують у вигляді фіксованої ціни на продукт.

4) Аутсорсинг

Це означає, що розробник не контактує з кінцевим споживачем своєї продукції, а вся інформація надається у вигляді технічного завдання великою компанією-підрядником. Тобто уявлення про потреби замовника формуються тільки на основі інформації, представленої фірмою-підрядником. Підрядник надає свої вимоги до процесу створення ПЗ і його якості. Підтримку продукту, як правило, здійснює підрядник. Фінансові відносини існують у вигляді індивідуальних контрактів між членами команди розробника і підрядником. Варіантом є укладення договору між підрядником та юридичною особою, що діє від імені команди.

Процес.

Процес – сукупність взаємопов'язаних дій, що перетворюють деякі вхідні дані у вихідні.

Набір процесів складають ЖЦ ПЗ. Процеси згруповані за категоріями – основні процеси, процеси підтримки чи допоміжні, організаційні процеси.

Тобто процеси орієнтовані не лише на розроблення програмної системи, але і на організацію і керування проектом

 Основні процеси пов’язані безпосередньо зі створенням програмних продуктів (наприклад, аналіз вимог, конструювання, тестування і т.д.)

 Окрім основних, існує багато додаткових і допоміжних процесів, зв'язаних не зі створенням продукту, а з організацією та супроводом робіт (нефункціональні процеси): створення інфраструктури, управління конфігурацією, управління якістю, навчання, вирішення протиріч, документування тощо .

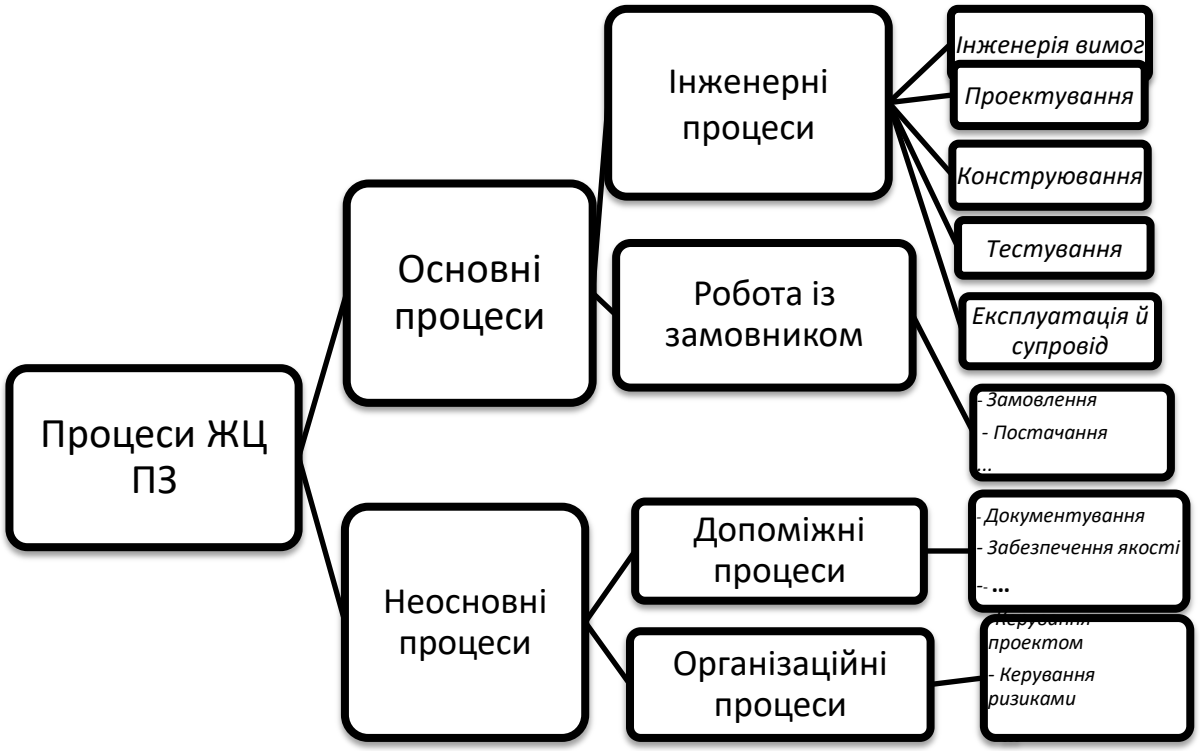


Рис.1.10. Класифікація процесів ЖЦ ПЗ

Процес в інженерній діяльності має бути встановлений. Повне встановлення процесу передбачає:

 Опис процесу – детальний опис дій і операцій процесу

 Навчання процесу – проведення занять з персоналом з освоєння процесу

 Введення метрик – встановлення кількісних показників ходу виконання процесу

- Контроль виконання – вимір метричних показників і оцінка ходу виконання
- Удосконалення – зміна процесу за змінних умов застосування

У сучасних умовах жорсткої конкуренції, дуже важливо гарантувати високу якість процесу. Для оцінки процесів використовується модель зрілості процесів – Capability Maturity Model (CMM) (дослівно «модель зрілості можливостей»).

У 1987 року Software Engineering Institute (SEI) випустив короткий огляд процесів розроблення ПЗ з описом їх рівнів зрілості, а також опитувальник, що призначався для виявлення проблем в компаніях. Більшість компаній розглядало даний опитувальник як готову модель, внаслідок чого через декілька років була створена реальна модель, Capability Maturity Model for Software. Перша версія CMM (Version 1.0), яка вийшла в 1991 році, в 1992 році була переглянута учасниками робочої зустрічі, в якій брали участь близько 200 фахівців в галузі програмного забезпечення, і членами співдружності розробників.

У компаніях, що виробляють ПЗ діють корпоративні стандарти на процеси взаємодії з замовником, процеси аналізу, проектування, програмування, тестування і впровадження програмних продуктів. Все це створює середовище, яке забезпечує якісну розробку програмного забезпечення.

Модель CMM фіксує критерії для оцінки зрілості компанії і пропонує рецепти для поліпшення існуючих в ній процесів. Іншими словами, в ній не тільки сформульовані умови, необхідні для досягнення мінімальної організованості процесу, а й даються рекомендації щодо подальшого вдосконалення процесів.

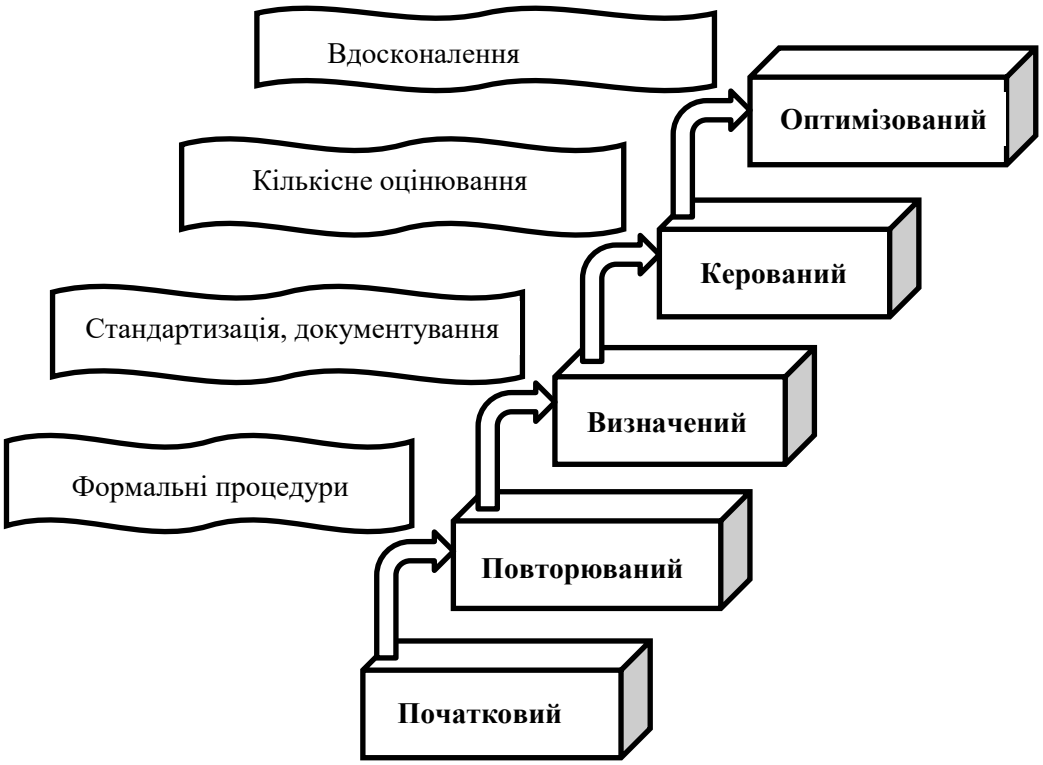


Рис. 1.11. Рівні зрілості процесів IT-компанії.

Початковий рівень (рівень 1) означає, що процес у компанії не формалізований. Він не може строго плануватися і відслідковуватися, його успіх носить випадковий характер. Результат роботи цілком і повністю залежить від особистих якостей окремих співробітників. При звільненні таких співробітників проект зупиняється.

Для переходу на повторюваний рівень (рівень 2) необхідно впровадити формальні процедури для виконання основних елементів процесу конструювання. Результати виконання процесу відповідають заданим вимогам і стандартам. Основна відмінність від рівня 1 полягає в тому, що виконання процесу планується і контролюється. Застосовувані засоби планування і управління дають можливість повторення раніше досягнутих успіхів.

Наступний, визначений рівень (рівень 3) вимагає, щоб всі елементи процесу були визначені, стандартизовані і задокументовані. Основна відмінність від рівня 2 полягає в тому, що елементи процесу рівня 3

плануються і управляються на основі єдиного стандарту компанії. Якість розроблюваного ПЗ вже не залежить від здібностей окремих особистостей.

З переходом на керований рівень (рівень 4) в компанії використовуються кількісні показники якості як програмних продуктів, так і процесу. Це забезпечує більш точне планування проекту та контроль якості його результатів. Основна відмінність від рівня 3 складається в більш об'єктивною, кількісній оцінці продукту та процесу.

Вищий, оптимальний рівень (рівень 5) має на увазі, що головним завданням компанії стає постійне покращення і підвищення ефективності існуючих процесів, введення нових технологій. Основна відмінність від рівня 4 полягає в тому, що технологія створення та супроводження програмних продуктів планомірно і послідовно вдосконалюється.

Персонал

У створенні програмного продукту бере участь безліч різних фахівців, тобто персонал. Персонал – це реальні люди, які виконують різноманітні процеси в ході розроблення програмного забезпечення протягом усього його життєвого циклу.

Крім програмістів, що займаються безпосередньо розробленням ПЗ, у проекті учасниками є й інші IT-фахівці. Загалом існує близько 200 різнопланових професій в галузі IT. У першу чергу, звичайно говорячи про «IT-шника» мають на увазі програміста. Розглянемо основні категорії IT-фахівців.

Розробник, програміст (Developer, англ.*)* конструює безпосередньо систему, вже маючи готові проектні рішення. Він володіє певною технологією програмування і відповідно спеціалізується у конкретних задачах . Наприклад, C++ *Developer*, C# *Developer*, Ruby *Developer*, Python *Developer*, Java *Developer*, PHP *Developer*, веб-розробник, Front-end- розробник, Back-end-розробник і т.д.

Архітектор програмної системи (*Software Architect*, англ.) – це IT-фахівець, який приймає рішення щодо внутрішнього устрою і зовнішніх інтерфейсів програмної системи з урахуванням проектних вимог і наявних ресурсів. Ві розробляє високорівневу архітектуру системи, визначає технічні стандарти, в тому числі стосовно кодування, інструментів і платформ. Завданням є розроблення «креслень» (ескізів) системи, які представляють складові системи і алгоритми роботи. У невеликих командах функція архітектора розподіляється між менеджером і розробниками. У великих проектах це може бути цілий відділ.

Проектувальник інтерфейсів (*UI/UX-Designer*, англ.) займається розробленням інтерфейсів користувача з точки зору зовнішнього вигляду (User Interface) та розробленням зручності користування з точки зору експлуатаційних характеристик системи (User Experience -досвід користування). Ці діяльності можуть бути розділено, і тоді говорять окремо про фахівців *UI-Designer* і *UX-Designer*. Тобто UI-Designer забезпечує гарний дизайн системи (кнопки, іконки, меню, вікна…). За аналогією до будівництва, це відповідає задачі «Чи має гарний вигляд будинок?». А UX-Designer забезпечує продуманість і зручність роботи в системі. За аналогією до будівництва, це відповідає задачі «Чи зручно користуватися усім у будинку?». Наприклад, чи є зручними виходи на балкони?

Архітектор бази даних (*DB-Architect*, англ.) проектує бази даних, що є складовою програмної системи. Для цього використовує відомі системи управління базами даних (СУБД).

Перспективними спеціальностями є інженер даних (*Big Data Engineer*, англ.), інженер з машинного навчання (*Machine Learning Engineer*, англ.), дослідник даних (Data Scientist, *англ.*) , які розробляють методи, алгоритми, пов'язані з аналізом даних, необхідні для інтелектуальних систем.

Інженер із забезпечення якості (QA-Engineer, **Q**uality **A**ssurance,) – це фахівець із забезпечення якості, діяльність якого спрямована на поліпшення

процесу розроблення ПЗ, запобігання дефектам і виявлення помилок в роботі продукту. Вузька спеціальність у рамках QA – тестувальник ПЗ, який перевіряє готовий продукт на наявність помилок (багів) і невідповідність вимогам, і потім документує знайдені дефекти й шляхи їх усунення. Є багато різних спеціалізацій фахівців в сфері забезпечення якості ПЗ. Наприклад, QA-Automation Engineer, QA- Manual Engineer, Test Analyst, UI/UX-Tester, Security-Tester, Test Executor, Verifier, Validator.

Керівник команди (*Team Lead*, англ.) робить керівництво командою окремих фахівців в процесі виконання проекту. Наприклад, QA Team Lead керує командою фахівців з якості. Для великих проектів можливе залучення декількох керівників підкоманд, що відповідають за вирішення окремих завдань. За аналогією до будівництва, це як прораб будівельників.

Керівник проекту (*Project Manager*, англ.) відповідальний за «все», в першу чергу, за оперативне вирішення проблем. Менеджери проектів створюють план розроблення, , організовують команду, налаштовують процес роботи над проектом, забезпечують зворотній зв’язок між командами та замовником, усувають перешкоди для команд, контролюють якість, витрати та виконання термінів.

Devops-інженер (Devops Engineer, Devops=**Dev**elopment+**Op**eration**s,** , англ.) автоматизує процес розроблення, узгоджує результати проектування, конструювання, тестування; бере участь в розгортанні системи, моніторить процес експлуатації. Це за аналогією на будівництві технолог, який допомагає будівельникам використовувати техніку, щоб пришвидшити процес розроблення і покращити якість кінцевого продукту.

Системний аналітик (System Analyst/Business-Analyst, англ.) аналізує потреби замовника у застосуванні ІТ. Розробляє вимоги до системи. Аналізує бізнес (діяльність) з точки зору автоматизації процесів.

Експерт предметної області (*Domain Expert*, англ.) консультує розробників як знавець предметної області, для якої створюється програмна система, допомагає розробникам зрозуміти, що має робити система.

Технічний письменник/редактор (*Technical Writer*, англ.) створює технічну документацію до системи у зрозумілому вигляді, не тільки IT-фахівцям. У першу чергу, це посібники/керівництва з користування системою.

Серед відповідальних за успішність IT-компанії є такі фахівці:

• Business Development Executive (відповідальний за розвиток бізнесу шукає клієнтів, партнерів…)

• Researcher (дослідник ринку трудових ресурсів, ринку IT-продуктів…)

• Chief Technology Officer (головний технічний директор).

У малочисельних командах ролі фахівців можуть поєднуватися. У великих – навпаки, виділятися групи або відділи (відділ проектування, відділ тестування, відділ контролю якості, відділ підготовки документації, ...).

Склад команди визначається також типом виконуваних робіт: під замовлення або коробкове виробництво (продукт на ринок).

Незалежно від сфери типовий склад учасників проекту має такий вигляд:

- замовники,
- інвестори,
- користувачі,
- постачальники ресурсів,
- консультанти,
- ліцензіари,
- фінансові інститути – банки,
- команда проекту,
- менеджер проекту.

Інженерна діяльність обов’язково планується та ґрунтується на розподілі робіт у проекті між різними категоріями виконавців.

Команда проекту - це група фахівців, які безпосередньо працюють над здійсненням проекту та підпорядковані керівнику проекту.

Серед важливих задач зі створення команди проекту та організації її ефективної роботи виділяють в першу чергу таку:

 визначити потреби, чисельний і кваліфікаційний склад персоналу на весь період здійснення проекту;

 провести пошук і відбір кандидатур, оформити їх прийом на роботу та звільнення;

 здійснити планування і розподіл працівників по робочих місцях; організувати навчання та підвищення кваліфікації;

 розподілити ролі і відповідальності в команді;

 створити умови і робочу атмосферу для колективної роботи, попереджати і вирішувати виникаючі конфлікти; розробити механізми мотивації і оплати та ін.

Контрольні питання

1. Що таке еталонна модель інженерії ПЗ?
2. Чим програмний продукт відрізняється від програмного забезпечення?
3. Які властивості обов’язково має програмний продукт?
4. Яка властивість програмного продукту є необхідною, щоб він вважався якісним?
5. Які властивості програмного продукту є достатніми, щоб він вважався якісним?
6. Що таке процес в інженерії ПЗ? Наведіть приклади.
7. Що передбачає встановлення процесу?
8. Як поділяють процеси в інженерії ПЗ?
9. Які відомі рівні зрілості процесів в ІТ-компаніях?
10. Чим оптимізований рівень CMM відрізняється від керованого рівня?

11. Чим повторюваний рівень CMM відрізняється від визначеного рівня?
12. Що таке проект з розроблення ПЗ?
13. Які розрізняють види ПЗ як визначені результати проектів з розроблення ПЗ?
14. Які є важливі характеристики проекту з розроблення ПЗ?
15. Прокоментуєте основні принципи управління проектом за Б.Боемом.
16. Що є ресурсами проекту з розроблення ПЗ?
17. Поясніть принцип «залізного трикутника» в проектах з розроблення ПЗ,
18. Які є схеми взаємодії замовника і розробників ПЗ?
19. Хто є учасниками типового проекту з розроблення ПЗ?
20. Які задачі інженерії ПЗ виконує експерт предметної області?
21. Що входить в обов'язки архітектора програмної системи?
22. Що є результатом роботи системного аналітика в проекті з розроблення ПЗ?
23. Які IT-фахівці беруть участь в процесі тестування ПЗ?
24. Чим відрізняються функції менеджера проекту від керівника команди?
25. Які IT-фахівці займаються безпосередньо розробленням ПЗ?
26. Які є спеціалізації серед розробників ПЗ?
27. Які є IT-фахівці, що спеціалізуються на розробці інтелектуальних систем?
28. Які задачі виконує технічний письменник в проекті з розроблення ПЗ?
29. Чи залучається замовник до процесів розроблення ПЗ? Відповідь поясніть.
30. Назвіть типових учасників проекту.
31. Які задачі необхідно вирішити для організації команди та її ефективної роботи?
32. Які фахівці безпосередньо відповідальні за успішність IT-компанії?
33. Наведіть приклади проектів з розроблення ПЗ з високими ризиками.

34. Які є спеціалізації фахівців з розроблення інтерфейсу користувача?

РОЗДІЛ 2. ЖИТТЄВИЙ ЦИКЛ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ – ОСНОВНИЙ ПРИНЦИП ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗБЕЗПЕЧЕННЯ.

У розділі наведено означення життєвого циклу програмного забезпечення. Описано типові складові життєвого циклу програмного забезпечення. Проведено ідентифікацію основних етапів життєвого циклу програмного забезпечення – аналіз і визначення вимог, проектування, програмна реалізація, тестування, експлуатація та супровід.

Тема 2.1. ОЗНАЧЕННЯ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Виникнення поняття про ЖЦ ПЗ.

Перехід від ручних засобів розроблення ПЗ до промислового виробництва програм потребував розвитку теоретичних основ розроблення ПЗ. Постійна необхідність внесення змін у програми як спричинена помилками, так і розвитком вимог до них, є принциповою властивістю програмного забезпечення.

Поява поняття життєвого циклу ПЗ пов'язана із першою кризою програмування. В кінці 60-х – початку 70-х років минулого століття сталася подія, яка ввійшла в історію як перша криза програмування. Суть кризи полягала в тому, що програмні проекти всі частіше стали виходити з-під контролю: порушувалися терміни, перевищувалися заплановані обсяги фінансування, результати не відповідали необхідним критеріям. Багато проектів взагалі не були завершені. Крім того, виявилося, що недостатньо розробити програму, а треба її ще супроводжувати й цей етап часто вимагає більше засобів, чим розроблення. Ситуація була викликана ростом складності проектів. Масштаби кризи наростали. Необхідно було вживати заходів для

радикального вдосконалення принципів і методів розроблення ПЗ з урахуванням його розвитку й супроводу.

Поняття ЖЦ виникло під впливом потреби у систематизації робіт у процесі розроблення ПЗ. Систематизація була першим етапом на шляху до автоматизації процесу розроблення ПЗ.

Провідні фахівці заговорили про те, що треба звернутися до досвіду промислового проектування й виробництва, де був накопичений великий досвід успішного розроблення складних проектів. Методологічну основу промислової інженерії становить поняття життєвого циклу виробу (продукту) як сукупності всіх дій, які треба виконати протягом всього «життя» виробу.
Отже, життєвий цикл промислового виробу – це набір етапів (фаз, стадій), а саме: проектування, виготовлення зразка, організація виробництва, серійне виробництво, експлуатація, ремонт, висновок з експлуатації тощо. Тобто життєвий цикл складається з дій, технологічних процесів , операцій.

Уперше про життєвий цикл ПЗ заговорили в 1968 р. у Лондоні, де відбулася зустріч керівників проектів з розроблення ПЗ. На зустрічі аналізувалися проблеми й перспективи проектування, розроблення, поширення й підтримки програм. Принципи і методи розроблення ПЗ вимагали постійного вдосконалення. Саме на цій зустрічі була запропонована концепція життєвого циклу ПЗ (SLC - Software Lifetime Cycle) як послідовності кроків-стадій, які необхідно виконати в процесі створення й експлуатації ПЗ.

Ця концепція не було сприйнята одностайно схвально. Суперечки, в деякій мірі, припинилися, коли у 1970 році У. Ройс ідентифікував стадії типового життєвого циклу ПЗ, а також висловив припущення, що контроль над виконанням цих стадій приведе до підвищення якості ПЗ та скороченню вартості та термінів виконання проектів з розроблення ПЗ.

ІПЗ пройшла декілька етапів розвитку, в процесі яких були сформульовані фундаментальні принципи і методи розроблення програмних

продуктів. Основний принцип інженерії ПЗ полягає в тому, що програми створюються в результаті виконання декількох взаємозв'язаних етапів (аналіз вимог, проектування, розроблення, впровадження, супровід), Ці етапи складають життєвий цикл програмного продукту. Характер взаємозв'язаності окремих етапів життєвого циклу розробки визначається моделлю життєвого циклу. Тобто життєвий цикл програмного забезпечення (ЖЦ ПЗ) вказує на всі процеси розроблення ПЗ, а модель ЖЦ ПЗ – на необхідні для заданого проекту і, що важливо, на порядок виконання цих процесів.

Думка провідних фахівців галузі ПІ зводиться до того, що ключове завдання ІПЗ – не використання технологій, а комбінування технологічних знань, людей і процесів для досягнення бізнес цілей організації. Це завдання вирішується на основі поняття ЖЦ ПЗ.

Короткий зміст ЖЦ ПЗ

Життєвий цикл – безперервний період, що починається з моменту ухвалення рішення про створення ПЗ і що закінчується зняттям його з експлуатації. Життєвий цикл розбивається на окремі процеси.

Процес – сукупність дій і завдань, що мають на меті досягнення значимого результату. Важливі процеси інколи називають етапами або фазами життєвого циклу.

Окрім основних, існує багато додаткових і допоміжних процесів, зв'язаних не створенням продукту, а з організацією робіт (нефункціональні процеси): створення інфраструктури, управління конфігурацією, управління якістю, навчання, вирішення протиріч .

Використання повних процесів вимагає додаткових ресурсів (часто істотних) і далеко не завжди окупається отриманим результатом. Тому, вибір складу процесів, міри їх установленності (повнота установленності) у кожному

конкретному випадку можуть бути зроблені по-різному, відповідно до вибраної моделі процесу.

Життєвий цикл розбивається на окремі складові – процеси, а процеси – на підпроцеси, роботи, дії. Тобто відбувається ієрархічна декомпозиція життєвого циклу ПЗ.

Процес – сукупність дій і завдань, що мають на меті досягнення значимого результату. Процес – це важлива складова будь-якої інженерної діяльності. Кожен процес має чіткий опис, визначені вхідні дані та результати.

Важливі процеси інколи називають етапами або фазами життєвого циклу.

До основних етапів життєвого циклу ПЗ належать (рис. 1.1):

1. етап аналізу та специфікації вимог (важливий і фундаментальний етап, на якому виконується збір та аналіз вимог замовника виконавцем і подання їх у нотації, яка є зрозумілою як для замовника, так і для виконавця. Окрім визначення вимог, на даному етапі здійснюється планування якості, оцінка майбутніх ризиків і т.д.);
2. проектування (перетворення вимог до розробки в послідовність проектних рішень, які визначають структуру й поведінку системи, а також інтерфейс користувача);
3. кодування (власне розроблення програмного забезпечення);
4. тестування (перевірка програмного продукту на наявність помилок);
5. експлуатація (дії з обслуговування системи під час її використання — консультації користувачів, вивчення їхніх побажань тощо);
6. супровід (дії з керування модифікаціями, підтримки актуального стану та функціональної придатності, інсталяцію нових та вилучення попередніх версій програмних систем у користувача).
7. зняття системи з експлуатації (дії з поступового припинення використання програми, перехід до нової програми, навчання персоналу новій програмі).

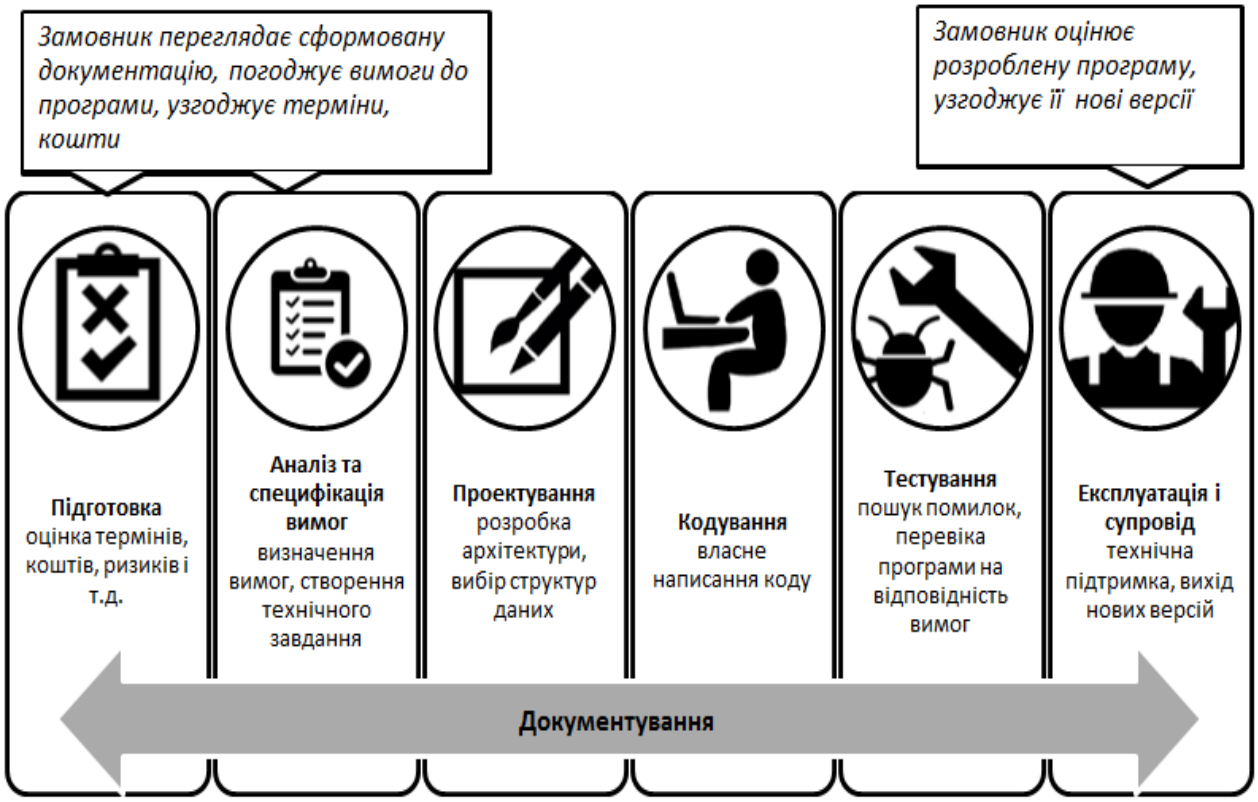


Рис. 2.1. Етапи життєвого циклу ПЗ.

Окрім основних, існує багато додаткових, допоміжних процесів, зв'язаних не безпосередньо зі створенням продукту, а з організацією робіт та підтримкою основних чи й неосновних процесів. До таких процесів, наприклад, належать створення інфраструктури, документування, управління якістю, навчання, вирішення протиріч і т.п.

Основні процеси життєвого циклу звертаються до допоміжних процесів, в той час, як організаційні процеси діють протягом життєвого циклу і пов'язані з основними процесами.

Узагальнена класифікація етапів життєвого циклу ПЗ наведена на рис.1.2. Детальна класифікація процесів ЖЦ ПЗ описана у міжнародних стандартах інженерії програмного забезпечення.

Контрольні питання

1. Що таке життєвий цикл програмного забезпечення?
2. Яка історія виникнення терміну ЖЦ в інженерії ПЗ?
3. Що є складовими ЖЦ ПЗ? Наведіть приклади.
4. Чим ЖЦ ПЗ відрізняється від моделі ЖЦ ПЗ?
5. Назвіть основні етапи ЖЦ ПЗ?
6. Чим основні процеси відрізняються від неосновних процесів ЖЦ ПЗ?
7. З чого складаються процеси ЖЦ ПЗ?
8. Охарактеризуйте коротко етап аналізу і специфікації вимог до ПЗ.
9. Охарактеризуйте коротко етап проектування ПЗ.
10. Охарактеризуйте коротко етап тестування ПЗ.
11. Охарактеризуйте коротко етап експлуатації і супроводу ПЗ.
12. У чому суть завершального етапу ЖЦ ПЗ?
13. Як допоміжні етапи пов’язані з основними?
14. Що означає декомпозиція ЖЦ ПЗ?
15. Наведіть приклади етапів ЖЦ для довільного промислового виробу.
16. Де наведена детальна класифікація процесів життєвого циклу ПЗ?

Тема 2.2. АНАЛІЗ ТА СПЕЦИФІКАЦІЯ ВИМОГ – ПОЧАТКОВИЙ ЕТАП ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Етап аналізу та специфікації вимог є фундаментальним, адже без чіткого визначення вимог не можливо перейти на наступні кроки розроблення програмної системи.

Основні поняття інженерії вимог

Саме правильно визначені вимоги дають уявлення про те, що хоче замовник і як кінцевий користувач буде використовувати майбутній програмний продукт. Вимоги – це те вихідне розуміння завдання розробниками, яке є основою всієї розробки.

Вимога (requirement) – бажана властивість, характеристика, умова щодо якості та функцій програмного забезпечення, яким має відповідати програмна система в ході експлуатації.

Вимоги визначаються в процесі аналізу вимог та фіксуються в специфікації вимог. Специфікація вимог до ПЗ (Software Requirements Specification, *англ.*) – це структурований технічний документ, в якому записується функціонал (що буде робити система) та важливі властивості системи. Крім того, загальна інформація концептуального рівня.

Важливість цього етапу полягає в тому, що

- по-перше, ціна програми у першу чергу встановлюється відповідно до об’єму виконуваних нею функцій та її властивостей,
- по-друге, відповідно до функцій та властивостей майбутньої програми розробляють стратегію роботи над проектом.

Тому невірне визначення цих функцій може привести або до непередбачуваного збільшення вартості програми, або до зміни великого об’єму коду внаслідок додання додаткових функцій.

Визначення вимог є нетривіальною задачею і проводиться, як правило, шляхом обговорення поглядів замовника на систему з майбутніми її розробниками.

Часто для виявлення потреб замовника, а також для виявлення змісту висловлених бажань користувача перед програмуванням доводиться проводити дуже велику додаткову роботу, яка називається аналізом предметної галузі або бізнес-моделюванням.

Існують труднощі взаєморозуміння замовника і розробників через великий розрив між програмістами та іншими людьми. По перше, тому, що щоб добре розібратися, якою має бути система інформатизації певної діяльності чи бізнесу треба попрацювати у відповідній області достатній час або якось іншим способом навчитися бачити проблеми даної предметної області зсередини. По-друге, позначається специфічність програмування як сфери діяльності. Для більшості користувачів і замовників вкрай непросто сформулювати точне знання, яке необхідно програмістам.

Ще одним важливим поняттям в інженерії ПЗ є управління вимогами (Software Requirements Management, *англ.*) – це безперервний процес, який включає в себе визначення, документування, аналіз, відстеження, пріорітизацію вимог та контроль над їхніми змінами. Мета управління вимогами полягає в тому, щоб переконатися, що програмне забезпечення відповідає потребам і очікуванням своїх клієнтів.

У 2008 році консалтингова компанія IAG Consulting провела дослідження понад 100 IT-компаній і надала звіт про вплив визначення вимог на успіх технічних проектів. У дослідженні брали участь лише великі фірми із прибутком понад 250 тис. доларів США. Середня вартість проектів, які розробляли дані компанії, становила 3 млн. доларів США. Згідно звіту, компанії, які чітко не визначили вимоги до проектів, затратили на 50% більше часу та коштів для їхньої реалізації.

Отже, важливо чітко та своєчасно ідентифікувати всі вимоги до програмного продукту, а також постійно слідкувати за їхньою зміною та уточненнями зі сторони замовника.

У зводі знань з інженерії програмного забезпечення SWEBOK визначаються такі види діяльності при роботі з вимогами (Додаток Б):

• Видобування вимог (requirements elicitation, *англ.*), націлене на виявлення всіх можливих джерел вимог й обмежень на роботу системи і витяг вимог з цих джерел.

• Аналіз вимог (requirements analysis, *англ.*), метою якого є усунення протиріч і неоднозначностей у вимогах, їх уточнення та систематизація, зокрема, виявлення взаємозв'язків між вимогами.

• Опис вимог (requirements specification, *англ.*). У результаті цієї діяльності вимоги повинні бути оформлені у вигляді структурованого набору документів і моделей, який може систематично аналізуватися, оцінюватися з різних позицій і в підсумку повинен бути затверджений як офіційне формулювання вимог до системи.

• Валідація вимог (requirements validation, *англ.*), яка вирішує завдання оцінки зрозумілості сформульованих вимог і їх характеристик, необхідних, щоб розробляти ПЗ на їх основі, в першу чергу, несуперечності і повноти, а також відповідності корпоративним стандартам на технічну документацію.

Є багато різних методів, що дозволяють виявити вимоги. Їх різноманіття пояснюється неформальністю процесів, що пов'язані з виявленням вимог і різним типом розроблюваного ПЗ. Серед найвідоміших методів є такі:

- Інтерв'ю, опитування, анкетування,
- Мозковий штурм, семінар,
- Спостереження за діяльністю,
- Аналіз нормативної документації,
- Аналіз моделей діяльності,
- Аналіз продуктів-конкурентів,

- Аналіз статистики використання попередніх версій програми.

Типи і характеристики вимог

Всі вимоги до програмного забезпечення поділяються на функціональні та нефункціональні.

Функціональні вимоги (functional or behavioral requirements, *англ.*) регламентують функціонування або поведінку системи. Вони визначають те, що система повинна вміти робити. Функціональні вимоги відповідають на питання «що повинна робити система в тих чи інших ситуаціях». Формуються функціональні вимоги як перелік задач, які вирішуються програмною системою.

Функціональні вимоги визначають основний «фронт робіт» команди розробників і встановлюють цілі, завдання та сервіси, що надаються системою замовнику.

Щодо виявлення функціональних вимог, то варто акцентувати увагу саме на властивості програми виконувати корисну дію, розв’язувати конкретну задачу. Тому «вихід з програми», «відкриття інтерфейсного вікна» і т.п. не є варіантами використання програми й не можуть фіксуватися як функціональні вимоги.

Крім того, також важливі нефункціональні вимоги. Нефункціональні вимоги не є описом функцій системи (рис.). Такі вимоги стосуються в першу чергу атрибутів якості (надійність і безпека, ефективність/продуктивність, зручність користування, супроводжуваність, переносимість, адаптованість і т.д.), а також системи – як апаратного забезпечення (ПК, мобільний телефон, оперативна пам'ять, графічний процесор тощо), так і системного чи спеціального ПЗ (необхідні драйвери, операційна система тощо). Розрізняють мінімальні та рекомендовані системні вимоги.

Додатково можуть існувати інші нефункціональні вимоги – до інсталяції, документації, відповідності стандартам, постачання тощо. Вимоги цього виду часто ставляться до всієї системи в цілому.

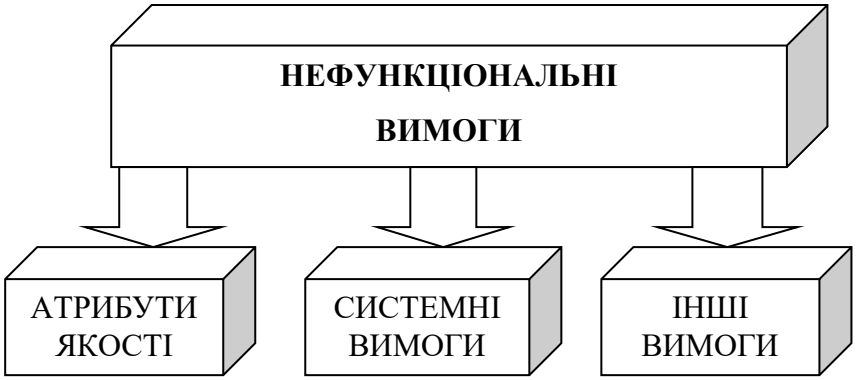


Рис.2.2. Групи нефункціональних вимог.

Зазвичай вимоги використовуються як засіб комунікації між різними зацікавленими сторонами (розробники, замовник, користувачі). Тому вимоги повинні бути простими та зрозумілими. Крім цього, усі вимоги програмного забезпечення повинні мати такі властивості:

1. коректність (без вживання технічного жаргону, вимоги повинні виражати об’єктивні факти);
2. необхідність (вимоги повинні відповідати певній потребі замовника або користувача);
3. завершеність (вимоги повністю визначені в одному місці документу і вся інформація для розуміння присутня);
4. верифікованість (здатність протестувати програмне забезпечення, щоб впевнитись чи воно відповідає даним вимогам);
5. недвозначність (можливість тільки однієї інтерпретації вимоги, зрозумілість);
6. послідовність (вимоги не суперечать одна одній).

Таблиця 2.1.

Приклад «хороших» та «поганих» вимог

«Погана» вимога	**«Хороша» вимога**
Основний фон головного вікна програми має бути синього кольору. (відтінків кольору є багато. Який код синього кольору?)	Код кольору основного фону головного вікна має бути #000080.
Після відкриття головної сторінки сайту повинна на деякий короткий час відобразитись зображення-реклама. («на трошки» – це на скільки?)	Після відкриття головної сторінки сайту повинна відображатись зображення-реклама на протязі 10 сек.
Програмна система повинна відображати точні показники температури кожних 15 сек. (наскільки точно?)	Програмна система повинна відображати показники температури з точністю ±0.5°C кожних 15 сек.

Відзначимо помилки, що зустрічаються при складанні технічних завдань та інших документів з вимогами.

- Опис можливих рішень замість вимог. Користувачеві не потрібно знати, яким чином отримується необхідний йому результат. Важливо описати, що отримується в результаті роботи програми, а не як програма працює. Наприклад, не потрібно описувати сам метод сортування замість вимоги «сформувати відсортований список студентів за рейтингом».
- Нечіткі вимоги, які не допускають однозначної перевірки, залишають недомовленості, мають відтінок порад, обговорень,

рекомендацій. Наприклад, невірно використовувати такі фрази «Можливо, що має сенс реалізувати також ... ..», «і т.д. ».

- Ігнорування аудиторії, для якої призначене представлення вимог. Наприклад, якщо специфікацію складає інженер замовника, то часто зустрічається надлишок інформації про обладнання, з яким повинна працювати програмна система, відсутній глосарій термінів і визначень основних понять, використовуються численні синоніми і т.д. Або допущений занадто великий ухил в бік програмування, що робить дану специфікацію незрозумілою всім непрограмістів.

Документування вимог

Вимоги до ПЗ можуть документуватися в текстовому або графічному вигляді. Документ опису вимог є основним документом, який визначає вимоги та порядок створення програмного забезпечення та прийняття його при введенні в експлуатацію.

Для опису вимог часто використовують такі типи документів:

- ✓ список вимог – найпростіший тип документу, в якому коротко описуються усі основні вимоги до програмного продукту;
- ✓ прототип – макет системи, документ, який здебільшого використовують для визначення вимог до дизайну програмного продукту;
- ✓ сценарії використання – текстовий опис усіх способів взаємодії користувача з системою через деталізацію усіх кроків чи дій, необхідних для досягнення мети (приклад наведено у Додатку В);
- ✓ діаграми прецедентів – діаграми, що зображають зв’язки між користувачами системи та її основними функціями (приклад наведено у Додатку Г);

- ✓ технічне завдання (специфікація вимог) – вихідний документ для розроблення нового програмного забезпечення, в якому формулюються основні цілі розробки, список принципових вимог до продукту, визначаються терміни та етапи розроблення і регламентуються характеристики готового до експлуатації програмного продукту (приклад структури документа наведено у Додатку Д);.

Зазвичай, найчастіше використовують для опису вимог саме технічне завдання (ТЗ) , в розробці якого беруть участь як представники замовника, так і виконавці. Аналіз й визначення вимог, здебільшого, є ітеративним процесом. Кожна ітерація для масштабного програмного проекту фіксується окремим документом.

Фахівці вважають, що грамотно складене ТЗ – це більше 50% успіху у вирішенні завдання, а час, витрачений на його підготовку, – одне з кращих вкладень, які фірма може зробити в період аналізу та специфікації вимог. Недарма складання ТЗ доручається провідним фахівцям – керівникам проектів і робіт, системним аналітикам, головним проектувальникам і т.п.

Згідно стандарту IEEE 830 описані такі переваги добре складеного технічного завдання:

1. Технічне завдання – це основа для угоди між замовниками та виконавцями щодо розроблення програмного забезпечення. Визначення повного опису функцій, які повинні бути реалізовані, допоможе потенційним користувачам визначити, чи програмне забезпечення відповідає їхнім потребам, чи воно повинне бути зміненим, щоб задовольнити їхні потреби. [Примітка: технічне завдання використовується в ролі основи договору з клієнтами протягом усього часу розроблення програмної системи].
2. Скорочення зусиль, необхідних на розроблення програмного забезпечення. Підготовка чіткого технічного завдання змушує усі

зацікавлені сторони розглянути ретельно всі вимоги, перш ніж розпочати етап розроблення дизайну та кодування. Такий аналіз вимог може виявити упущення, непорозуміння і протиріччя ще на перших етапах життєвого циклу програми, коли ці проблеми легше виправити.

3. Технічне завдання – базовий документ, на основі якого здійснюють розрахунок витрат і часу для реалізації програмного забезпечення.
4. Технічне завдання є основою валідації та верифікації програмного продукту (верифікація і валідація – це методи аналізу, перевірки специфікацій і правильності виконання програм відповідно до заданих вимог і формального опису програми). [Примітка: технічне завдання використовується для створення плану тестування].
5. Оскільки у технічному завданні описано характеристики продукту, тому воно служить основою для подальшого вдосконалення програмного забезпечення. [Примітка: постійно завжди слідкувати за оновленням технічного завдання згідно побажань клієнтів або розробників].

Після формування ТЗ на його основі здійснюють аналіз поставленої задачі на розроблення ПЗ, визначають структуру вхідних і вихідних даних, оцінюють альтернативні методи розв’язування поставлених завдань, а також обирають оптимальний алгоритм для вирішення певної задачі.

Висновок. Аналіз і визначення вимог – один з основних етапів ЖЦ ПЗ. Трудність цього етапу полягає у поєднанні формальних й неформальних методів ІПЗ, потребі врахуванні знань предметної області, необхідності спілкування із замовником та майбутніми користувачами. Існують різні типи вимог – функціональні й нефункціональні. Основним результатом цього етапу є задокументовані узгоджені із замовником вимоги у формі спеціального документу. Аналіз і визначення вимог виконують досвідчені фахівці – аналітики, керівники проектів.

Контрольні питання для самоперевірки

1. Що таке вимога до програмного забезпечення?
2. Які є типи вимог до ПЗ?
3. Що таке функціональні вимоги до програми? Наведіть приклади.
4. Які документи використовують для опису вимог?
5. Які характеристики правильної вимоги?
6. Що таке специфікація вимог?
7. Наведіть приклади нефункціональних вимог.
8. Як поділяються нефункціональні вимоги?
9. Наведіть приклад вимоги, що стосується якості ПЗ.
10. Що включає в себе управління вимогами?
11. Що означає властивість вимоги – завершеність?
12. Яка вимога вважається верифікованою?
13. Назвіть форми документування вимог до ПЗ.
14. Назвіть основні методи виявлення вимог.
15. Який найпростіший документ може представити вимоги до ПЗ?
16. Назвіть графічну форму представлення вимог.
17. Наведіть приклади апаратних вимог до ПЗ.
18. Чи у ТЗ описано як буде працювати ПЗ? Відповідь поясніть.
19. Який етап ЖЦ ПЗ слідує за визначенням вимог? Яка його основна суть?
20. Чи допускається використання у специфікації вимог термінів-синонімів? Відповідь поясніть.
21. Що зображується на діаграмі прецедентів? Як ця діаграма пов’язана із ТЗ?
22. Які фахівці ІПЗ беруть участь у специфікації вимог?
23. Яка роль замовника у визначенні вимог до програмної системи?
24. Що є вхідними даними та результатами етапу визначення вимог до ПЗ?
25. Як визначення вимог до програмної системи пов’язане з її тестуванням?

26. У чому полягає складність отримання результатів аналізу й визначення вимог до ПЗ?
27. У чому полягає ітеративність визначення вимог?
28. Які є методи виявлення вимог?
29. Назвіть типові розділи технічного завдання чи специфікації вимог.
30. Складіть технічне завдання на розроблення комп’ютерної програми для вивчення студентами навчальної дисципліни «Основи програмування».
31. Як найпростіше можна зафіксувати вимоги до ПЗ?
32. Назвіть типові помилки, що виникають при визначенні вимог.

Тема 2.3. ПРОЕКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ – КОНСТРУКТОРСЬКІ ЕТАПИ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Отже, спочатку визначають “Що має робити програмна система та якими володіти характеристиками?” Далі треба знайти відповідь на питання “Яким чином буде реалізувати програмна система визначені раніше вимоги?”

Зміст етапу проектування

Проектування є невід’ємною частиною будь-якого виробництва. У загальному випадку мається на увазі створення сукупності документів, розрахунків, креслень, необхідних для виготовлення виробу; а також планування, задум заснування, організації будь-чого.

Під час проектування виконується трансформація вимог у проектні рішення. При цьому складається план робіт, необхідних для отримання результату, обираються методи і технології вирішення проблеми.

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

Даний етап життєвого циклу програмного забезпечення є найбільш трудоємким. Але, на відміну від попереднього етапу, перевагою є формалізація процесів, використання готових інструментів та зразків і засобів автоматизації.

На етапі проектування важливо описати з яких частин складатиметься ПЗ і як воно буде працювати, тобто задається структура й поведінка ПЗ. Отримані проектні рішення у подальшому переводяться в програмні коди.

Проектування полягає в створенні представлення про:

- ✓ складові програми;
- ✓ модульну структуру програмного продукту;
- ✓ алгоритмічну структуру;
- ✓ структури даних;

- ✓ вхідний і вихідний інтерфейс (вхідні і вихідні формати даних).

У строгому значенні архітектура ПЗ (software architecture, *англ.*) – опис підсистем і компонентів програмної системи, а також зв'язків між ними. Архітектура, в першу чергу, визначає внутрішню структуру системи, задаючи спосіб, яким система буде організована або конструюється.

Проектування архітектури програмного продукту включає такі завдання:

- ✓ трансформація вимог до програмного забезпечення в архітектуру (визначення структури програмної системи та її проектування);
- ✓ розбиття програмної системи на окремі компоненти та їх проектування з визначенням ключових елементів структури даних
- ✓ розроблення і документування інтерфейсів програми і баз даних.

Будь-яка програмна система може розглядатися з різних точок зору – наприклад, *структурної* (статичної), *поведінкової* (динамічної), *логічної* (задоволення функціональним вимогам), *фізичної* (розподіленість за обчислювальними вузлами), *реалізації* (як деталі архітектури представляються в коді) і т.п. У результаті, отримуються різні архітектурні подання (view, *англ.*). Архітектурне подання може бути визначене, як частковий аспект програмної архітектури, що відображає специфічні властивості програмної системи. У свою чергу, дизайн системи можна розглядати як комплекс архітектурних подань, достатній для реалізації системи й задоволення вимог, заданих до системи.

Для спрощення візуалізації процесу проектування використовуються так звані нотації – схематичне зображення характеристик розроблюваної програмної системи (блок-схеми, макети, діаграми і т.д.).

У зводі знань з інженерії програмного забезпечення SWEBOK визначаються основні розділи області знань, пов’язані з проектуванням ПЗ (Додаток Е).

Перших розробників програмних систем можна було назвати вільними творцями, художниками від програмування, які створювали все з нуля. Проте проектування в сучасній інженерії ПЗ – це здебільшого використання готових рішень, шаблонів проектування, так званих "патернів" (pattern, *англ.* – шаблон). "Винахід велосипеда" в розробці архітектури ПЗ є справою невдячною, і будь-який проектувальник зараз зобов'язаний знати як мінімум базовий набір стандартних рішень.

Шаблони проектування (патерн, design pattern, *англ.*) – це багато разів застосовувана архітектурна конструкція, яка надає рішення загальної проблеми проектування в рамках конкретного контексту. Патерн не є закінченим зразком проекту, який може бути прямо перетворений в код, це опис або зразок для того, як вирішити завдання таким чином, щоб це можна було використовувати в різних ситуаціях (Додаток Ж). Об'єктно-орієнтовані шаблони часто показують відношення і взаємодії між класами або об'єктами, без визначення того, які кінцеві класи чи об'єкти додатку будуть використовуватися. Шаблони проектування незалежні від застосовуваної мови програмування.

Патерни проектування в програмуванні розглядаються як стандартні будівельні блоки для архітекторів програмного забезпечення і дозволяють їм легко знаходити спільну мову.

Тому проектувальник повинен мати достатній досвід програмування і підходити до кожної нової задачі, спираючись на хороший досвід та встановлену методику проектування. На жаль, через ріст складності програмних систем дуже тяжко здійснити проектування, щоб уникнути помилок під час майбутнього розроблення програм.

Тому для подолання складності проектування розбивається на два етапи (рівні):

1. *архітектурне проектування або високорівневий дизайн* (software architectural design/top-level design, *англ.*), протягом якого

обумовлюються так звані компоненти високого рівня: зв’язок між найбільшими і загальними частинами програмного забезпечення;

2. *детальне проектування компонентів* (software detailed design, *англ.*), протягом якого визначається деталізована архітектура, що описує кожну компоненту у тому обсязі, який необхідний для конструювання.

Крім того, у окремо виділяють інтерфейсне проектування, мета якого - сформувати графічний інтерфейс користувача (GUI).

Розроблення інтерфейсу користувача

Інтерфейс користувача – сукупність засобів для обробки та відображення інформації, які максимально пристосовані для зручності користувача.

У графічних системах інтерфейс користувача реалізовується багатовіконним режимом, змінами кольору, розміру, видимості (прозорість, напівпрозорість, невидимість) вікон, їхнім розташуванням, сортуванням елементів вікон, гнучкими налаштовуваннями як самих вікон, так і окремих їхніх елементів (піктограми, ярлики, вкладки, кнопки, шрифти тощо), доступністю багатокористувацьких налаштувань.

Графічний інтерфейс користувача (GUI, Graphical User Interface, *англ.*) – інтерфейс між комп’ютером і його користувачем, що використовує піктограми, меню, і вказівний засіб для вибору функцій та виконання команд. Зазвичай, можливе відкриття більше, ніж одного вікна на одному екрані.

GUI – система засобів для взаємодії користувача з комп’ютером, заснована на представленні всіх доступних користувачеві системних об’єктів і функцій у вигляді графічних компонентів екрану (вікон, значків, меню, кнопок, списків і т. п.). При цьому, на відміну від інтерфейсу командного рядка, користувач має довільний доступ (за допомогою клавіатури або пристрою координатного введення) до всіх видимих екранних об’єктів.

Вперше концепція GUI була запропонована вченими з дослідницької лабораторії Xerox PARC в 1970-х, але отримала комерційне втілення лише в продуктах корпорації Apple Computer.

Зручність використання користувацького інтерфейсу (Usability, *англ.*) – показник його якості, що визначає кількість зусиль, необхідні для вивчення засад роботи з програмною системою з допомогою даного інтерфейсу, її використання, підготовки вхідних даних, і інтерпретації вихідних. Тобто, зручність використання виявляє міру простоти доступу користувача до функцій системи, наданих через людино-машинний (користувацький) інтерфейс.

На зручність використання користувацького інтерфейсу впливають такі чинники:

- легкість навчання (чи швидко людина навчається використовувати систему);

- ефективність навчання (чи швидко людина працює після навчання);

- запам’ятовуваність навчання (чи легко запам’ятовується все, чому людина навчилася);

- наявність помилки (чи часто людина припускається помилок у роботі);

- загальна задоволеність (чи є загальне враження від співпраці з системою позитивним).

Усі ці фактори, незважаючи на неформальність, можуть бути виміряні. Для таких вимірів вибирається група типових користувачів системи, і під час її роботи вимірюються показники роботи і системи (наприклад, кількість допущених помилок, витрачений час на виконання завдання), а також користувачам пропонується висловити враження від системи з допомогою опитувальних карток.

Найбільш цитованими у книгах з людино-машинного інтерфейсу є так звані евристичні правила, розроблені спільно відомим американським фахівцем у галузі проектування інтерфейсів Якобом Нільсеном (Jakob Nielsen)

та дослідником Рольфом Молічем (Rolf Molich). Евристичні правила не мають строгого доведення, вони виведені з практичного досвіду, відображають мінімальні критерії, яким повинен відповідати будь-який інтерфейс.

Фактично, це «десять заповідей» будь-якого розробника інтерфейсів комп'ютерних програм:

1. Сповіщення про поточний стан (Visibility of system status, *англ.*).
2. Близькість до реального світу (Match between system and real world, *англ.*).
3. Управління свободою дій користувача (User control and freedom, *англ.*).
4. Цілісність та стандарти (Consistency and standards, *англ.*).
5. Допомога користувачам в розпізнаванні, діагностиці та усуненні помилок (Help users recognize, diagnose and recover from errors, *англ.*).
6. Запобігання помилок (Error prevention, *англ.*).
7. Розпізнавання, а не згадування (Recognition rather than recall, *англ.*).
8. Гнучкість і ефективність використання (Flexibility and efficiency of use, *англ.*).
9. Естетичний і мінімально необхідний дизайн (Aesthetic and minimalistic design, *англ.*).
10. Допомога та документація (Help and documentation, *англ.*).

Наведені принципи є базовими і розширюються в документах, присвячених розробці інтерфейсу користувача для певних класів систем. Написані цілі томи про те, як треба створювати інтерфейс для застосунків на платформі Android, для застосунків на платформі iOS і т.д.

Отже, на етапі проектування системні архітектори, дизайнери й розробники, керуючись вимогами, розробляють високорівневий дизайн та детальну архітектуру системи. Під архітектурою системи мається на увазі

внутрішня структура продукту (компоненти програми і зв’язки між ними), а також основи користувацького інтерфейсу програмного продукту.

Зміст етапу реалізація

Після того як вимоги та архітектура програмного продукту затверджені, відбувається перехід до наступного етапу життєвого циклу – програмна реалізація (розроблення, кодування). Зміст цього етапу – конструювання програмної системи за розробленими проектними документами (ескізами) та кодування проектних рішень вибраною мовою програмування. Термін конструювання вказує на елемент автоматизації цього етапу, зокрема, що програмна система може збиратися з компонентів багаторазового використання, а програмні коди можуть автоматизовано генеруватися за проектними рішеннями.

Основна суть етапу – написання програмістами коду програми відповідно до раніше визначених вимог.

Основними складовими даного етапу є:

- ✓ створення алгоритмів і кодів окремих модулів вибраною мовою програмування;
- ✓ створення вихідного тексту програми;
- ✓ налагодження вихідного тексту.

Основний результат цього етапу – вихідний код і файли конфігурації.

Конфігурація системи - склад функцій, програмного і технічного забезпечень системи, можливі їх комбінації в залежності від наявності обладнання, загальносистемних засобів, зазначених в технічній документації системи.

Правильно оформлений текст програми є якісною характеристикою, що дозволяє легко вносити зміни, виправляти помилки та супроводжувати

програмне забезпечення. Тому прийнято писати тексти програм, зважаючи на загальноприйняті вимоги до коду.

Стандарт оформлення коду (coding conventions, *англ.*) – це сукупність вказівок, що стосуються певної конкретної мови програмування, які встановлюють правила стильового оформлення коду, практики та методи написання програм цією мовою. Метою прийняття та використання стандарту є спрощення сприйняття програмного коду людиною, мінімізація навантаження на пам’ять та зір при читанні програми.

Код програми повинен бути на стільки простим, на скільки це можливо. Досвідчені розробники дотримуються правил "Не треба ускладнювати завдання", "Keep it simple, stupid! (Правило KISS)","Вибирайте найпростіше з працюючих рішень", "Не треба робити того, що не передбачено технічним завданням".

Простий код зазвичай коротше і зрозуміліше, а значить, містить менше помилок.

Висновок. Розроблення ПЗ у загальному випадку передбачає два основних етапи ЖЦ ПЗ – проектування й створення програмного коду за проектними рішеннями. У ході розроблення ПЗ використовуються вже готові рішення й зразки, автоматизовані засоби, що значно полегшує виконання типових задач цього етапу ЖЦ. Проектні рішення стосуються, в першу чергу, структури системи й її поведінки. Проектування ПЗ виконують спеціальні фахівці з інженерії ПЗ – архітектори систем, проектувальники, дизайнери. Програмною реалізацією займаються розробники, програмісти, кодувальники.

Контрольні питання

1. Які завдання вирішуються на етапі проектування?
2. У чому полягає етап проектування для найпростішої програми?
3. Що розуміють під архітектурою програмної системи?

4. Які є рівні проектування ПЗ? Чим вони відрізняються?
5. Що є вхідними даними для етапу проектування ПЗ? Для якого етапу ці вхідні дані є результатом?
6. Для якого етапу ЖЦ ПЗ проектні рішення є вхідними? Наведіть приклади проектних рішень.
7. Які фахівці галузі ІПЗ задіяні на етапі проектування ПЗ?
8. Що таке шаблони проектування? Яка мета їх використання?
9. Що таке інтерфейсне проектування?
10. Дайте означення інтерфейсу користувача.
11. Яким евристичним характеристикам має відповідати інтерфейс користувача?
12. Наведіть приклад неграфічного інтерфейсу користувача?
13. Чи є переваги у використанні неграфічного інтерфейсу?. Відповідь обґрунтуйте.
14. Що є складовими типового графічного інтерфейсу користувача?
15. Які переваги/недоліки модульної структури програми над монолітною?
16. Що таке проектування виробу (у загальному випадку)?
17. Назвіть типові вигляди програмної системи (view), які формуються на етапі проектування.
18. Що таке стандарт оформлення коду? Наведіть приклад.
19. Які основні результати отримуються на етапі програмної реалізації системи?
20. Які фахівці галузі ІПЗ задіяні на етапі розроблення ПЗ?

Тема 2.4. ТЕСТУВАННЯ – ЕТАП КОНТРОЛЮ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Випробування будь-якої програми є одним з найбільш відповідальних етапів її розроблення й часто буває пов’язано з найбільшими труднощами і найбільшими втратами часу. У сучасних проектах на тестування відводиться понад 50% загальних витрат.

Означення етапу тестування.

Можна визначити такі основні цілі тестування програмного забезпечення:

- ✓ оцінка якості програмного забезпечення;
- ✓ підвищення якості програми;
- ✓ запобігання появи помилок в майбутньому.

Цілі тестування можуть відрізнятися, в залежності від етапу розроблення ПЗ, на якому воно проводиться. Наприклад, на етапі кодування метою тестування буде пошук як можна більшої кількості помилок в роботі програми, що дозволить їх локалізувати і виправити. У той же час, під час прийому програми замовником, тестують програму з метою показу, що система працює правильно. У період супроводу, тестування в основному необхідно для того, щоб переконається у відсутності нових помилок, що з’явилися під час внесення змін до нових її версій.

Головне ж завдання тестування – пошук помилок. Для виконання заходів забезпечення якості ПЗ розробники часто використовують спеціальні групи контролю якості, які мають назву QA. Група QA всередині компанії фактично виконує роль вимогливого користувача. У деяких компаніях заборонено неформальне спілкування між групою розробників та групою тестувальників. Від відповідальності групи QA залежить успіх продукту.

На ранніх етапах інженерії ПЗ етап тестування програм не розглядався окремо від реалізації. Вважалося, що проведення перевірки програми може відбуватися на інтуїтивному рівні і є схожим до мистецтва. Сьогодні

безперечно тестування є одним з найважливіших етапів життєвого циклу програмного забезпечення. Підхід до тестування лише як до мистецтва є неактуальним, оскільки тестування – це наука зі своїми моделями, методами та інструментами.

У зводі знань з інженерії програмного забезпечення SWEBOK визначаються основні розділи області знань, пов'язані з тестуванням ПЗ (Додаток З).

Згідно міжнародної термінології інженерії ПЗ *тестування програмного забезпечення (Software testing, англ.)* – це процес перевірки готової програми в статиці (перегляди, інспекції, налагодження вихідного коду) і в динаміці способом прогону кінцевого набору тестових даних для різних шляхів виконання програми, та порівняння отриманих результатів із заздалегідь запланованими.

Даний етап тестування варто розпочинати якомога раніше, адже згідно досліджень Б. Боема, вартість виправлення помилок, знайдених на ранніх етапах життєвого циклу ПЗ є значно дешевшою, в порівнянні із вартістю її виправлення, наприклад, на етапі експлуатації (рис.3.1).

Рис. 2.3. Графік вартості виправлення помилок на різних етапах життєвого циклу програмного забезпечення

Варто пам’ятати, що тестування – процес ітераційний. Після виявлення і виправлення кожної помилки обов’язково слід повторити виконання тестів, щоб переконатися в працездатності програми.

Тестування ПЗ – це техніка контролю його якості. У більш вузькому сенсі, тестування – це перевірка відповідності між фактичною та очікуваною поведінкою програми, що здійснюється на обмеженому наборі перевірочних даних, які були вибрані певним чином.

Якість ПЗ включає ряд характеристик і коректність є лише однією з них. Програму, яка задовольняє специфікаціям, тобто видає очікувані відповіді на визначені комбінації значень вхідних даних, називають *коректною*.

З-поміж інших характеристик, включених у поняття якості, є такі:

Ефективність – міра використання системних ресурсів (зокрема, обсягу пам’яті);

Надійність – здатність системи функціонувати в наперед визначених умовах, а також у випадку нестандартних, непередбачених ситуацій чи збоїв у роботі, а одним з показників надійності є середній інтервал між відмовами;

Практичність – простота у вивченні та застосуванні системи;

Цілісність – здатність системи запобігати неавторизованому або некоректному доступу до програм і даних;

Адаптованість – можливість використання системи без її зміни у середовищах, на які вона не орієнтувалася початково.

Живучість – здатність системи функціонувати при вводі недопустимих даних або ж у напружених умовах.

Як бачимо, надійність програми – це властивість програми, більш строга, ніж коректність, оскільки програма може бути коректною, але не надійною. Програма є *надійною*, якщо вона коректна, прийнятно реагує на неточні вхідні дані і задовільно функціонує в незвичних умовах.

Програму не можна здавати в експлуатацію до тих пір, поки не буде впевненості в її надійній роботі.

Етап тестування має за мету визначення коректності та надійності програми. Тобто тестування дає відповідь на запитання, чи програма вирішує дійсно поставлену перед нею задачу і видає правильний результат при будь-яких умовах. Варто пам’ятати, що тестування доводить наявність помилок, але не їхню відсутність.

Серед основних принципів тестування є такі:

1. Про тестування необхідно думати протягом усього періоду розроблення програми.
2. Повнота тестування забезпечується не кількістю тестів, а правильним вибором тестових даних для перевірки все можливих варіантів роботи програми.
3. У кожному наступному тесті повинен використовуватися клас даних, відмінний від попереднього.

Тестування ПЗ включає такі діяльності: планування тестування, проектування тестів, проведення тестування та аналіз результатів тестування.

Результатом планування тестування є створення плану (тест-плану) ПЗ. План тестування (тест-план) – це документ, що описує весь об’єм робіт з тестування, починаючи від опису об’єкта, стратегій, розкладу, критеріїв початку та закінчення тестування, до необхідного в процесі роботи обладнання, спеціальних знань, а також оцінки ризиків з варіантами їх вирішення.

Метою проектування тестових варіантів є систематичне виявлення різних класів помилок при мінімальних витратах часу та вартості.

Тестові набори розробляються так, щоб вони охопили якомога більше ситуацій щодо зміни даних для програми, або хоча б по одному набору тестових даних для кожної вітки розгалуження алгоритму.

Кожен тест визначає:

- свій набір вхідних даних і умов для запуску програми;
- набір очікуваних результатів роботи програми.

Повну перевірку програми гарантує вичерпне тестування.

Вичерпне тестування вимагає перевірити всі набори вихідних даних, всі варіанти їх обробки і включає велику кількість тестових варіантів.

Хорошим вважають тестовий варіант з високою імовірністю виявлення ще не розкритою помилки. Успішним називають тест, який виявляє досі не розкриту помилку.

Виконуються тести, всі отримані результати оцінюються.

Реальні результати тестів порівнюються з очікуваними результатами.

Коли виявляється розбіжність, фіксується помилка – починається відлагодження.

Різновиди тестування

Виокремлюють різні види тестування залежно від обраного критерію.

На ранніх етапах розроблення ПЗ виявляти помилки допомагає статичний аналіз коду. *Статичний аналіз коду* – це уважна перевірка коду (найчастіше – вихідного коду програми) з метою виявлення в ній семантичних помилок, без виконання програми. Найбільш ефективною і найбільш дорогою є перевірка коду командою програмістів (техніка перегляду, code review, *англ.*). Вартість такої перевірки зумовлена значними затратами часу і тим, що увага людини досить швидко притупляється. Частково працю людей заміняє спеціальне ПЗ (одним з прикладів такого ПЗ є PCLint). Статичні аналізатори коду допомагають зекономити час, вказуючи розробникові на ділянки коду з потенційними помилками. Зокрема, статичні аналізатори допомагають “виловити” помилки, що виникають при неуважному редагуванні скопійованих фрагментів коду, наприклад:

```
dist1 = sqrt(x1^2 + y1^2 + z1^2);
```

dist2 = sqrt(x2^2 + y2^2 + **z1^2**);

На відміну від статичного, *динамічний аналіз коду* – перевірка працездатності при виконанні програмного коду на реальному чи віртуальному процесорі.

За об'єктом тестування можна виділити функціональне та нефункціональне тестування. У першому випадку мова йде про перевірку правильності виконання функцій програмою, у іншому – про перевірку нефункціональних характеристик програми.

Наприклад, тестування продуктивності (Performance testing) – перевірка стабільності роботи програми або її частини при заданому навантаженні (Load testing), при перевантаженні (Stress testing) та тривалому середньому навантаженні (Stability testing); юзабіліті-тестування (Usability testing) – перевірка зручності використання, ергономічності ПЗ; перевірка безпеки (Security testing) – тестування роботи системи з точки зору безпеки інформації; тестування сумісноті (Compatibility testing) – нефункціональне тестування, метою якого є перевірка коректної роботи ПЗ у певному оточенні.

За *часом проведення* розрізняють димове тестування, санітарне, регресійне тестування, альфа- та бета-тестування. Димовим тестом є перевірка, чи програма запускається взагалі (якщо вона, наприклад, працює з базою даних, то чи є правильне підключення). Тобто, димовий тест дає відповідь на питання, чи є сенс переходити до подальшого тестування, але не дає змогу зробити висновок про правильність роботи програми. Термін походить з інженерного середовища: якщо ввімкнення приладу не призводить до появи диму, можна вважати, що першочергове випробовування прилад пройшов успішно. Змістом димового тестування ПЗ є елементарна, дуже поверхнева перевірка, чи ключові модулі програми виконують свої основні функції. Санітарне тестування або перевірка узгодженості – вузьконаправлене тестування, достатнє для доведення того, що конкретна функція працює згідно заявлених у специфікації вимог. Таке тестування є підмножиною регресійного

тестування і використовується для визначення працездатності певної частини програми після змін, вироблених у ній або навколишньому середовищі. Регресійне тестування орієнтоване на повторне вибіркове тестування системи або її компонентів після внесення в них змін на тих самих тестах, що і до модифікації. Альфа- тестування – це тестування системи групою тестування організації-розробника, а бета-тестування системи – «зовнішніми» користувачами.

Згідно з тим, як враховується *інформація про програмну систему* розглядають тестування «чорної скриньки» (black box, *англ.*), тестування «білої скриньки» (white box, *англ.*). Тестування «чорної скриньки» – це перевірка роботи програми без врахування внутрішньої структури програми (тільки як працюють функції), а тестування на підставі аналізу структури програми – тестування «білої скриньки». Тому тестування «чорної скриньки» – це функціональне тестування, а тестування «білої скриньки» отримало назву структурного тестування. Є ще тестування «сірої» скриньки – це комбінація методів «білої» і «чорної» скриньки. При цьому тестувальникам відомі деякі (але не всі) деталі реалізації системи, що є об’єктом тестування.

За рівнем тестування розрізняють модульне, інтеграційне, системне та приймальне тестування. Рівні відображають компонентний підхід у розробленні ПЗ. Модульне тестування (unit testing, *англ.*) – це метод тестування програмного забезпечення, який полягає в окремому тестуванні кожного модуля коду програми, зазвичай в ізольованому середовищі (тобто ізольовано від решти всіх модулів).. Під модулем розуміється логічно замкнутий фрагмент програми, який може бути викликаний через його інтерфейс. Модуль перевіряється на відповідність своїм специфікаціям і внутрішню логіку. У процедурному програмуванні модулем вважають окрему функцію, в об’єктно-орієнтованому програмуванні – клас. Інтеграційне тестування (англ. integration testing) – це фаза тестування програмного забезпечення, під час якої окремі модулі програми комбінуються та

тестуються разом, у взаємодії. Системне тестування (system testing, англ.) передбачає перевірку інтегрованої системи на її відповідність всім вимогам. Тобто підтверджується, що доступ до всіх компонентів системи і взаємодія з ними несуперечливі і передбачені згідно специфікацій систем. Приймальне тестування (acceptance testing, *англ.*) – це фінальний етап тестування програми перед здачею в експлуатацію, який проводять для перевірки на відповідність до вимог замовника.

За методом тестування розрізняють ручне, автоматизоване та напівавтоматизоване тестування. Ручне тестування – це процес пошуку дефектів у роботі програми, коли тестувальник перевіряє працездатність програми, якщо б він був користувачем. Автоматизоване тестування використовує програмні засоби для виконання тестів і перевірки коректності результатів виконання, що спрощує тестування і скорочує його тривалість. Головна перевага автоматизованого тестування полягає в можливості повторного прогону тестів без участі людини. Очевидно, напівавтоматизоване тестування – це поєднання ручного та автоматизованого тестування.

Наведена вище класифікація не є вичерпною. Найбільш часто використовувані види тестування наведено на рис.2.4.

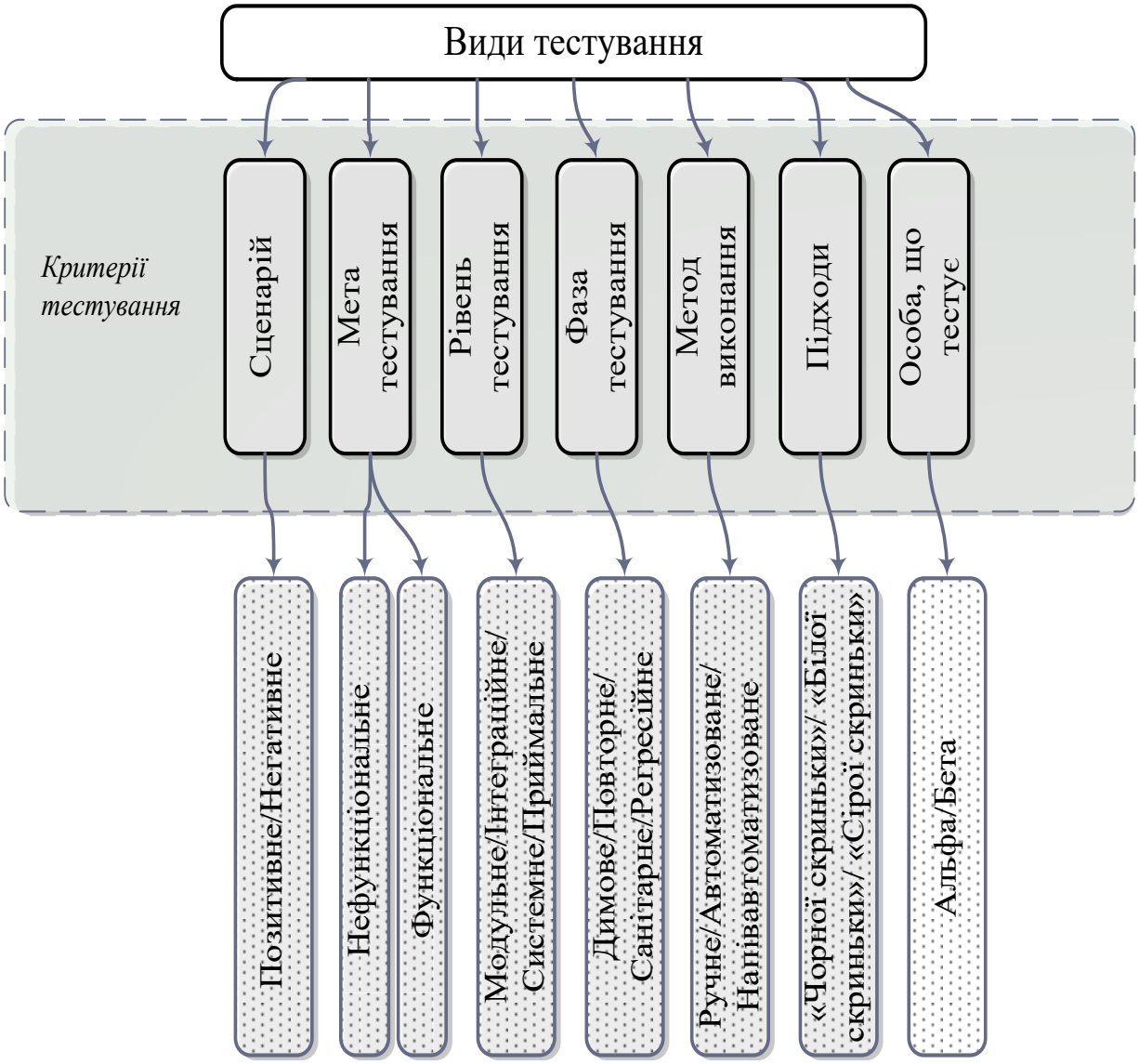


Рис. 2.4. Класифікація видів тестування

Функціональне тестування

Функціональне тестування – це перевірка того, чи програма правильно виконує усі свої функції, корисні з точки зору кінцевого споживача. Функціональне тестування ще називають тестуванням “чорної скриньки”, оскільки для його проведення не потрібно знати внутрішньої організації коду, а тільки як взаємодіяти з програмним продуктом. Функціональне тестування тісно пов’язане з поняттям верифікації.

Верифікація (verification, *англ.*) – перевірка, чи програмне забезпечення відповідає технічному завданню (специфікації вимог), тобто, чи насправді програма реалізована саме так, як описано у документації на неї. Крім того, процес верифікації містить перевірку правильності специфікацій вимог на їх відповідність, несуперечність, повноту і здійсненність, а також на дотримання вимог стандартів.

Перший тест має бути максимально простий, оскільки початкова мета тестування полягає в тому, щоб перевірити, чи працює програма взагалі. Таку перевірку часто називають *димовий тест.*

Для складних систем неможливо перебрати усі комбінації вхідних даних, особливо якщо вхідні дані не є дискретними. Часом використовують випадкове (стохастичне) тестування, коли програму перевіряють на наборі випадково згенерованих даних. Для забезпечення надійності ПЗ з відмовою 10^{-5} і помилкою, не більшою за 5 %, потрібно згенерувати 299572 тестів. Тести при цьому мають бути *незалежними.* Як бачимо, випадкове тестування потребує надто багато зусиль.

Для зменшення кількості тестів було запропоновано поняття клас еквівалентності (термін уведено Майєрсом).

Клас еквівалентності – набір даних з загальними властивостями. Обробляючи різні елементи класу, програма має вести себе однаково.

Розрізняють “правильні” та “неправильні” класи еквівалентності. Нехай, наприклад, програма обробляє оцінки студентів. “Правильним” класом еквівалентності будуть значення від 0 до 100 (у стобальній шкалі) включно, а двома “неправильними” класами еквівалентності – 1) від’ємні значення і 2)значення >100. Значення на границях підлягають окремій перевірці.

Є наступні правила формування класів еквівалентності:

1) якщо умова введення задає діапазон n … m, то визначається один допустимий і два неприпустимих класи еквівалентності.

2) якщо умова введення задає конкретне значення, то визначається один допустимий і два неприпустимих класи еквівалентності.

3) якщо умова введення задає множину значень {a, b, c}, то визначається один допустимий і один неприпустимий клас еквівалентності.

4) якщо умова введення задає логічне значення, наприклад true, то визначається один допустимий і один неприпустимий клас еквівалентності.

Отже, для функціонального тестування необхідно визначити “правильні” та “неправильні” класи еквівалентності та виділити граничні (екстремальні) значення.

У контексті перевірки функціонування програми в різних умовах етап тестування можна розділити на три частини:

1. Тестування програми в нормальних умовах.
2. Тестування програми в екстремальних умовах.
3. Тестування програми у виняткових ситуаціях.

Перевірка в нормальних умовах передбачає тестування на основі даних, які характерні для реальних умов функціонування програми. Випадки, коли програма має працювати зі всіма можливими даними, надзвичайно рідкісні. Звичайно мають місце конкретні обмеження на область зміни даних, в якій програма має зберігати свою працездатність.

Перевірка в екстремальних умовах має слідувати відразу за перевіркою програми в нормальних умовах. Тестові дані цього етапу містять граничні значення області зміни вхідних даних, які мають сприйматися програмою як правильні дані.

Для числових даних тестують програму при початковому та кінцевому значеннях допустимої області зміни даних, при зміні довжини відповідного поля від мінімальної до максимальної. На практиці перевіряють не тільки самі граничні значення, але й такі, що незначно від них відхиляються. Наприклад, якщо програма приймає дискретне число від 0 до 100, то потрібно перевірити її роботу при введених значеннях 0, 1, 99, 100. Якщо ж очікуване число є

неперервним у діапазоні від 0 до 100, то перевіряють поведінку програми при введених значеннях в околі 0 та 100 відповідно.

Контроль граничних вхідних значень дає змогу виявити проблеми переповнення типів. Наприклад, якщо функція приймає два аргументи типу char і повертає їхню суму, теж оголошену як тип char, то перевірка правильності функції на прикладах всередині діапазону 0..255 рідко дає змогу виявити проблему. Натомість передача функції значень 255 і 255 одразу виявить переповнення типу.

Для файлів, які можуть мати від 0 до N записів, необхідно переконатися, що програма працює коректно, коли файл порожній, коли є тільки один запис, коли кількість записів у ньому є максимально допустимою або ж на одиницю меншою за максимально допустиму.

Загалом, для впорядкованих даних (наприклад, для масивів чи динамічних списків) необхідно ретельно перевіряти перший і останній з елементів. Особливу увагу потрібно приділяти правильній обробці індексів масивів, особливо для кодів, написаних мовами, що не передбачають контролю виходу за межі масиву.

Типовими прикладами таких екстремальних значень є дуже великі числа, дуже малі числа і відсутність інформації. Для нечислових даних як екстремальні умови, окрім їх відсутності, потрібно використовувати спеціальні, нетипові символи, що охоплюють всі можливі ситуації. Процес використання екстремальних значень в якості тестових даних має назву ***граничні випробування***.

Необхідно обов’язково перевірити *усі* виявлені граничні умови.

Ще один тип екстремальних умов – це граничні обсяги даних, коли вони складаються з надто малої чи, навпаки, надто великої кількості записів. Необхідно встановити, що відбувається з програмою, якщо їй на оброблення не надходить жодного елемента даних або тільки один, і чи збережеться працездатність програми при дуже великих наборах даних.

Особливий інтерес представляють так названі нульові приклади. Для числового вводу – це звичайні нульові значення, для послідовності символів – це ланцюжок пропусків, для вказівників – нульове значення адреси. Якщо подібне тестування не виконується, то згодом часто доводиться зіштовхуватися з незрозумілою поведінкою програми.

Перевірка у виняткових ситуаціях передбачає використання даних, значення яких знаходяться **за межами допустимої області змін**. Відомо, що всі програми розробляються з розрахунку на оброблення обмеженого набору даних. Найгірший варіант виникає тоді, коли програма сприймає некоректні дані як правильні і видає неправильний, але правдоподібний результат. Програма має відкидати будь-які дані, які вона не в стані опрацювати правильно.

Щодо перевірки програми в екстремальних і виняткових умовах, то тут можна дати такі рекомендації:

1. Дані, які містять пропуски, цифри і букви, мають випробуватися в найрізноманітніших комбінаціях;
2. Інколи найбільші труднощі викликають дані, які містять відразу декілька помилок;
3. Особливими є помилки, викликані неправильним використанням першого та останнього елементів опрацьовуваної інформації.

Згідно з загальноприйнятою термінологією в галузі інженерії ПЗ, перевірку у виняткових ситуаціях для непередбачених вхідних даних називають негативним тестуванням. А позитивним тестуванням, відповідно, називають перевірку роботи програми для передбачених вхідних даних. Варто зазначити, що до таких даних належать і не зовсім типові вхідні дані, але ще дозволені, наприклад, файл з максимально можливою кількістю даних чи файл з одним записом, який буде видалено. Тобто тестування для нормальних та екстремальних вхідних даних називають позитивним тестуванням.

Як було розглянуто вище, правильний відбір даних для тестування тої чи іншої програми може значно полегшити завдання знаходження в ній помилок.

До способів перевірки правильності комп'ютерних результатів належать обчислення вручну, отримання результатів з документації та інших інформаційних джерел, отримання результату з допомогою деякої іншої аналогічної програми.

Структурне тестування

Структурне тестування (тестування "білої" або "скляної" скриньки) – це відлагодження алгоритму програми, яке, звісно, передбачає доступ до вихідного коду.

Зміст структурного тестування полягає у перевірці вихідного коду програми . Є такі різновиди структурного тестування:

- тестування на основі потоку управління програми;
- тестування на основі потоку даних програми;
- мутаційне тестування.

Тестування на основі потоку керування програми полягає у виконанні таких дій. Вихідний код програми подають у вигляді графа, в якому вершинами є оператори, а дугами – переходи між ними. Такий граф називають *графом управління*.

Для повного структурного тестування потрібно перевірити усі шляхи у графі управління ("пройтися" усіма можливими шляхами від початкової до кінцевої вершини принаймні один раз). Граф управління зручно будувати на основі блок-схеми програми.

Очевидно, що якщо кожна з двох чи більше вершин зв'язані тільки з наступною та попередньою (вершини "нанизані" на кшталт намиста, наприклад, як вершини на Рис. 3.4), то вони неодмінно належатимуть одному

шляху, відповідно, граф можна спростити, поєднавши такі вершини в одну (як показано на рисунку штриховою лінією).

Для проходження всіх шляхів необхідно підібрати відповідні значення коефіцієнтів квадратного рівняння, а саме:

1. $a = 0, b \neq 0$
2. $a = 0, b = 0$
3. $a \neq 0, D < 0$
4. $a \neq 0, D > 0$
5. $a \neq 0, D = 0$

Розглянутий приклад є гранично простим і має суто ілюстративний характер. У реальних програмних продуктах є величезна кількість галужень і, відповідно, велика кількість (десятки тисяч) можливих шляхів у графі управління, що робить повний перебір усіх шляхів графу надскладним або ж нерозв'язним завданням.

Тестування на основі потоків даних спрямоване на відстеження життєвого циклу змінних та виявлення неініціалізованих змінних і надлишкових присвоювань. Змінна може перебувати в одному з трьох станів: 1) визначення (змінна вже ініціалізована, але ще не використовувалася), 2) використання (в обчисленнях, в якості аргументів функцій, в логічних операторах тощо); 3) знищення. Нормальний життєвий цикл змінної передбачає, що вона створюється, ініціалізується, декілька разів використовується і після цього, можливо, знищується. Відхилення від цього сценарію має занепокоїти тестувальника.

Приклади переходів між станами, які вказують на помилку:

- "Визначення"–"Знищення" (немає коду, який використовує змінну, тож ініціалізація змінної – марна трата ресурсів),
- "Знищення"–"Використання" (небезпечна помилка, яка може не виявлятися тривалий час, якщо вказівник на звільнену пам'ять усе ще існує),

- “Знищення”–“Знищення” (особливо небезпечним є подвійне знищення вказівників).

Мутаційне тестування полягає у штучному внесенні у програму дрібних помилок – мутацій. Програма, одержана з оригінальної шляхом внесення мутації, називається мутантом. В одному мутанті повинна бути принаймні одна мутація, а типово їх є декілька.

Припускається, що професійні програмісти якщо і роблять помилки, то тільки “дрібні” – наприклад, неправильно вказують границі циклів, застосовують “не ті” константи, міняють місцями індекси при адресації елементів масивів, некоректно записують арифметичні операції і т.д. Відповідно, мутації – це змінені границі циклів, змінені індекси і т.д.

Оригінальну програму та її мутанти тестують на тому самому наборі тестів. Якщо на цьому наборі тестів виявляються усі помилки, штучно внесені у програми-мутанти, і підтверджується правильність оригінальної програми, то набір тестів відповідає мутаційному критерію, а тестована програма вважається коректною. Якщо виявлено не всі мутації, потрібно переглянути набір тестів і продовжити тестування.

Документування результатів тестування

Якщо відлагодження ПЗ виконується його розробниками, то тестування готових програм – командою тестувальників. Оскільки при функціональному тестуванні програма – це “чорна скринька” і тестувальник не знає, в якій ділянці коду міститься помилка, то важливим є не лише пошук помилок, але і їхній правильний опис. У розвинених компаніях (згадаймо модель CMM) для відстеження дефектів застосовуються спеціальні багтрекінгові (bugtracking, *англ.*) системи. У таких системах фіксують опис дефекту, достатній для його відтворення програмістом, особу, відповідальну за виправлення дефекту, рівень критичності дефекту, пріоритет і статус (відкритий, виправлений, закритий тощо; статус міняють як тестувальник, так і розробник).

Згідно з тим, чи відбувається повний опис (документування) тестування, розрізняють види тестування – формальне чи неформальне тестування.

У будь-якому випадку (використовується система відстеження дефектів чи ні), результати тестування варто документувати. Оскільки знаходить помилку одна людина, а виправляє її інша, то ймовірність виправлення помилки залежить від того, наскільки вона задокументована (взаємне розуміння учасників проекту є однією з основних проблем у розробці ПЗ).

При складанні звіту про знайдені помилки необхідно старатися максимально повно, просто і доступно її описати, причому одразу після виявлення, не покладаючись на пам’ять.

Зважаючи на сукупний практичний досвід компаній-розробників ПЗ, потрібно відображати у звіті про знайдені дефекти таку інформацію:

1. Назва програми.

2. Версія програми. Потреба в зазначенні версії зумовлюється тим, що після виправлення попередніх дефектів створюють нову версію; версія дає змогу уникнути плутанини.

3. Опис дефекту. Для складних програм з розгалуженим інтерфейсом і багатьма переходами між графічними екранами необхідно описати повну послідовність дій, що дала б змогу виявити дефект (можна долучити екранні знімки, якщо вони інформативні). Для консольної програми достатньо описати набір вхідних даних.

4. Тип звіту може бути зазначений як:

- Помилка кодування – (програма поводиться неправильно, на думку тестувальника);
- Помилка проектування;
- Пропозиція – тестувальник викладає свою ідею, як покращити програму (це не є дефект);

- Невідповідність документації – при цьому потрібно дати посилання на конкретні сторінки документації, помилка може бути у програмі, що спільно з'ясовується різними учасниками команди;
- Запитання – програма робить щось, чого тестувальник не розуміє і хоче уточнення;

5. Ступінь важливості (критичності) – суб'єктивний критерій, типово вказують рівень (низький, середній, високий);

6. Статус дефекту – широко вживане поняття, використовується для поліпшення взаємодії між розробниками та тестувальниками при моніторингу роботи над виявленням і виправленням дефектів, зокрема, у багтрекінгових системах. Типові приклади статусу дефекту: New (щойно описаний), In Process (опис дефекту прочитаний розробником, зрозумілий йому і над усуненням дефекту вже ведеться робота), Fixed (розробник встановлює цей статус, коли вважає, що йому вже вдалося усунути дефект), Verified (тестувальник перевірив роботу розробника і визнав, що дефект насправді усунуто), Cannot Verify (тестувальник не може перевірити результати виправлення дефекту, оскільки наявних у нього ресурсів недостатньо для відтворення умов, за яких виявлено дефект), Deferred (виправлення дефекту відкладено на деякий час).

Висновок. Тестування ПЗ – один з основних етапів ЖЦ ПЗ, мета якого контроль якості. Програма не вважається робочою, якщо непротестована. Програма є якісною, якщо правильно працює у різних ситуаціях і володіє рядом нефункціональних характеристик (зручність, ефективність, надійність й ін.) Тестування – це наука зі своїми моделями, методами та інструментами. Існує велика кількість критеріїв, згідно з якими можна проводити класифікацію видів тестування. Тестування виконують спеціальні фахівці з інженерії ПЗ – тестувальники, інженери з якості.

Контрольні питання для самоперевірки

1. Яка мета етапу тестування?
2. Дайте визначення, що таке тестування ПЗ.
3. Які є типи тестування в контексті умов функціонування програми?
4. Що таке граничні випробування?
5. Назвіть нульові приклади для різних типів даних?
6. Які основні принципи тестування?
7. Чим відрізняється функціональне та нефункціональне тестування?
8. Чим відрізняється тестування “чорної” та “білої” скриньки?
9. Що таке регресійне тестування?
10. Що таке приймальне тестування?
11. Що таке модульне тестування?
12. Чим відрізняється інтеграційне тестування від системного тестування?
13. Як класифікують тестування за методом проведення тестування?
14. Яке тестування називають позитивним?
15. Яке тестування називають негативним? Наведіть приклад.
16. Хто проводить бета-тестування?
17. Що таке альфа-тестування?
18. Яке тестування називають формальним?
19. Які є різновиди структурного тестування?
20. У чому суть мутаційного тестування? Наведіть приклади мутацій.
21. Що таке верифікація ПЗ?
22. Чим динамічний аналіз коду відрізняється від статичного?

Тема2.5. ЕКСПЛУАТАЦІЯ І СУПРОВІД – ЗАВЕРШАЛЬНІ ЕТАПИ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Означення експлуатації і супроводу ПЗ.

Коли програма протестована і в ній більше не залишилося серйозних дефектів, приходить час передачі її кінцевим користувачам.

Етап експлуатації ПЗ (Software operation, англ.) включає такі роботи:

- використання готової програмної системи;
- оцінка її ефективності використання;

Етап супроводу (Software maintenance, англ.) включає такі роботи:

- усунення знайдених в процесі експлуатації помилок;
- внесення необхідних змін для підтримки актуальності програмної системи;
- перевірка коректності внесених змін (зміни не повинні негативно впливати на функціонування системи).

Після випуску нової версії програми в роботу включається відділ технічної підтримки. Його співробітники забезпечують зворотний зв’язок з користувачами, їх консультування та підтримку. У разі виявлення користувачами тих чи інших помилок, інформація про них передається у вигляді звітів команді розробки, яка, в залежності від серйозності проблеми, або негайно випускає виправлення, або відкладає його до наступної версії програми.

Отже, етап супроводу – це процес створення і впровадження нових версій програмного продукту. Супровід ПЗ ще називають його еволюцією.

Причинами випуску нових версій можуть бути:

- ✓ необхідність виправлення помилок, виявлених в процесі експлуатації попередніх версій;

- ✓ необхідність вдосконалення попередніх версій, наприклад, покращення інтерфейсу, розширення складу виконуваних функцій або підвищення продуктивності розробленої системи;
- ✓ зміна середовища функціонування, наприклад, поява нових технічних засобів та/або програмних продуктів, з якими взаємодіє супроводжуване програмне забезпечення.

Проблема супроводу реальних програм є в тому, що вони настільки великі, що через певний час розробник вже не пам'ятає усіх деталей запису коду. Супровід може потребувати більше, ніж половини бюджету, закладеного на весь життєвий цикл програми.

Великі програми повинні розроблятись так, щоб їх супровід був максимально зручним. Це щонайменше потребує достатньої кількості коментарів у програмі. Для забезпечення етапу супроводу необхідна повна адекватна документація до розробленого ПЗ. Важливо також, щоб ПЗ оптимально/«правильно» спроектовано.

Модифікацією коду програми можуть займатись не розробники, а зовсім інші люди.

Внаслідок процесів, які виконуються під час супроводу, сама програма може зазнати значних змін, тому тут можливий

- ■ ***реінжиніринг*** програми
- ■ ***рефакторинг*** програми

Реінжиніринг – це процес створення нової функціональності або усунення помилок шляхом **суттєвої** зміни ПЗ, яке перебуває в експлуатації. Рефакторинг – це реорганізація коду для покращення характеристик і показників якості програм без зміни її поведінки .

Удосконалення застарілого ПЗ може включати рефакторинг та реінжиніринг, тобто реорганізацію або реструктуризацію, перепрограмування окремих елементів або налаштовування параметрів на іншу платформу або середовище виконання зі збереженням зручності його супроводу.

Особливістю етапу супроводу є те, що легше розробити новий програмний продукт, ніж внести зміни у готову програмну систему. Як правило, звичайному програмісту складно розібратися в чужому вихідному коді.

Реінжиніринг ПЗ з низьким індексом супроводжуваності, найчастіше, дорожче розроблення нового програмного забезпечення, оскільки потрібно прибрати обмеження попередніх версій, але при цьому залишити сумісність з попередніми версіями.

Реінжиніринг не може виконати програміст низької чи середньої кваліфікації. Потрібна робота програмістів з великим досвідом переробки програм і знанням різних технологій.

Якщо спочатку програма мала строгу та зрозумілу архітектуру, то провести реінжиніринг буде на порядок простіше. Тому при проектуванні, як правило, аналізується, що вигідніше – провести реінжиніринг або розробити програмний продукт «з нуля».

На етапі супроводу в програмний продукт вносять необхідні зміни, які так само, як в інших випадках, можуть вимагати перегляду проектних рішень, прийнятих на будь-якому попередньому етапі. Роль цього етапу істотно зросла, так як продукти тепер створюються ітераційно: спочатку випускається порівняно проста версія, потім наступна з великими можливостями, потім наступна і т.д. Саме це і є причиною виділення етапу супроводу та експлуатації в окремий процес життєвого циклу програмного забезпечення.

У зводі знань з інженерії програмного забезпечення SWEBOK визначаються основні розділи області знань, пов’язані зі супроводом ПЗ (Додаток К).

Зняття ПЗ з експлуатації здійснюється за рішенням замовника з участю експлуатуючої організації, служби супроводу і користувачів.

При цьому програмні продукти і відповідна документація підлягають архівуванню відповідно до договору.

Аналогічно, як при перенесенні ПЗ в інше середовище, з метою полегшити перехід до нової системи передбачається одночасна експлуатація старого і нового ПЗ протягом деякого періоду. У цей час виконується необхідне навчання користувачів для роботи з новим ПЗ.

Висновок. Призначення супроводу ПЗ – підвищити його якість. Однією зі складових якості ПЗ є зручність інтерфейсу користувача. Для забезпечення зручності інтерфейсу користувача розробникам необхідно дотримуватися загальноприйнятих правил побудови інтерфейсу. Інтерфейс користувача має бути професійно побудований, а не просто інтуїтивно зрозумілий. Важливу роль для користування програмною системою відіграє також необхідна користувачеві документація, довідкова система, допомога.

Контрольні питання для самоперевірки

1. Які етапи ЖЦ ПЗ є завершальними?
2. Які задачі інженерії ПЗ належать до етапу експлуатаційного використання ПЗ?
3. Яка мета етапу супроводу ПЗ?
4. Чому супровід ПЗ називають його еволюцією?
5. Чим експлуатаційне використання ПЗ відрізняється від супроводу ПЗ?
6. У чому полягає трудність виконання супроводу ПЗ?
7. У чому суть рефакторингу ПЗ?
8. У чому суть факторингу ПЗ?
9. Як рефакторинг пов'язаний з реінжинірингом?
10. Що є результатами реінжинірингу?
11. Чим рефакторинг відрізняється від реінжинірингу?
12. До якого типу робіт належить оптимізація програмного коду?
13. Яка типова вартість етапу супроводу для складних програмних систем по відношенню до всієї вартості ЖЦ ПЗ?

14. Що підвищує супроводжуваність ПЗ?
15. Які причини того, що етап супроводу може стати новим проектом з розроблення ПЗ?
16. Що таке якість програмного продукту?
17. Наведіть характеристики якості ПЗ.
18. Які характеристики якості можна забезпечити на етапі супроводу ПЗ?
19. Які задачі виконуються на етапі супроводу ПЗ?
20. Які задачі вирішуюся на етапі зняття з експлуатації ПЗ?
21. Які фахівці можуть бути задіяні на завершальному етапі життєвого циклу ПЗ?
22. Які є відомі методи програмування, метою яких є підвищення ефективності виконання етапу супроводу?

РОЗДІЛ 3. МОДЕЛІ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У розділі наведено означення та основні характеристики моделі життєвого циклу програмного забезпечення. Описано базові моделі життєвого циклу програмного забезпечення – каскадна, спіральна, формальна, компонентна моделі, а також змішані моделі – ітераційна, інкрементна, V-подібна, швидкого прототипування. Розглянуто схеми і принципи промислових методологій, утворених на основі базових і змішаних моделей життєвого циклу програмного забезпечення.

Тема 3.1. ОЗНАЧЕННЯ МОДЕЛІ ЖИТТЄВОГО ЦИКЛУ ПЗ

У загальному випадку модель – це узагальнений опис чого-небудь з певної точки зору.

Якщо життєвий цикл ПЗ описується набором фаз (етапів, стадій) проекту зі створення ПЗ, які виконуються через окремі процеси, розбиті на операції й завдання, то модель життєвого циклу ПЗ визначає вибір конкретного набору процесів (тобто склад ЖЦ ПЗ), послідовність їх виконання, кількість застосувань певних процесів згідно конкретних умов виконання програмного проекту.

Тобто модель життєвого циклу ПЗ – це структура, що визначає послідовність виконання та взаємозв'язок процесів, дій і завдань протягом життєвого циклу. Модель життєвого циклу залежить від специфіки, масштабу і складності проекту та умов, в яких система створюється і функціонує. У рамках специфічних моделей життєвого циклу, які приписують правила організації розроблення ПЗ у рамках даної галузі або організації, визначаються більш конкретні процеси розроблення. Відрізняються вони від стандартів, перш за все, більшою детальністю і чітким описом зв'язків між окремими видами діяльності, визначенням потоків даних (документів і артефактів) в ході життєвого циклу. Таких моделей досить багато, адже фактично кожен раз,

коли деяка організація визначає власний процес розроблення, в якості основи цього процесу розробляється деяка модель життєвого циклу ПЗ.

Правильний вибір моделі життєвого циклу ПЗ дуже важливий, тому що багато в чому визначає успіх проекту. Вибір затратних процесів може “втопити” проект, а , в свою чергу, занадто спрощене відношення до процесів приводить до втрати контролю над ходом виконання робіт.

Впродовж розвитку інженерії ПЗ на основі досвіду провідних компаній і фахівців були описані та запропоновані до використання типові моделі ЖЦ ПЗ, які проявили себе в певних умовах, мають певні переваги, недоліки й умови застосовності.

Завдання програмного інженера – в умовах реального програмного проекту підібрати правильну комбінацію типових моделей ЖЦ ПЗ, спрямовану на успішний кінцевий результат.

Не зважаючи на різноманіття, усі моделі життєвого циклу ПЗ мають спільний набір характерних властивостей. Серед них такі:

- Результатом виконання кожної фази (етапу) життєвого циклу є деяка модель ПЗ. Наприклад, опис вимог – це модель того, що повинен робити програмний продукт; результат проектування - модель ПЗ у вигляді архітектурних рішень стосовно структури та поведінки ПЗ, т.д.
- Результат виконання кожного етапу (фази) є входом для наступного етапу (фази). Наприклад, специфікація вимог – результат етапу аналізу і визначення вимог є вхідними даними для етапу проектування.
- Фази повинні виконуватися в певній послідовності, що визначається моделлю ЖЦ. Наприклад, у каскадній моделі – це строга послідовність усіх етапів, у ітераційній – це послідовність з поверненням з кожного етапу на один з попередніх при виявленні

помилок і відповідно повторення усіх етапів після проходження того, на який виконано повернення.

- Деякі процеси можуть виконуватися багаторазово, а деякі – лише раз. Наприклад, в ітераційній моделі можливе одноразове виконання процесу визначення вимог і багаторазове виконання процесів проектування.
- Моделі визначаються особливістю завдань, обмеженнями на ресурси, досвідом розробників, вимогами до технологій розроблення і т.д.

Першою моделлю ЖЦ ПЗ є каскадна або класична модель (автор У.Ройс), яка характеризується простотою і зручністю у застосуванні. У 70-80-х роках минулого століття ця модель була стандартом в інженерії ПЗ. Проте з середини 80-х років з ростом складністю самого ПЗ і відповідно процесів його розроблення недоліки каскадної моделі стали все більше проявлятися, а її застосування ставало все більш обмеженим. У відповідь на таку ситуацію була описана і запропонована як альтернатива спіральна або циклічна модель (автор Б.Боем). Ця модель характеризувалася великою гнучкістю на відміну від каскадної та враховувала можливі невизначеності етапів розроблення ПЗ. Проте занадто велика складність цієї моделі та намагання врахувати все можливі ситуації не сприяли її простому практичному застосуванню. Як вихід з такої ситуації фахівцями ІПЗ були запропоновані рішення у вигляді комбінації каскадної та спіральної моделей – так звані моделі-гібриди (ітераційна, інкрементна, V-подібна, швидкого прототипування).

З точки зору автоматизації процесів розроблення ПЗ перспективними моделями ЖЦ ПЗ є формальна і компонентна моделі. Основним недоліком цих моделей є обмеження у застосуванні, що пов’язано з недосконалими теоретичними методами, що лежать в основі цих моделей.

Успішний досвід провідних компанії та фахівців з розроблення ПЗ став основою для створення промислових методологій розроблення ПЗ. Серед них

альтернативами в інженерії ПЗ є формальні «важковагові» методології і гнучкі методології.

Аналіз моделей ЖЦ ПЗ показує, що саме програмування не є основним видом діяльності колективу, зайнятого промисловими розробленнями. Багато досліджень віддають на фазу програмування не більше 15-20% часу, витраченого на розроблення, а супровід взагалі може бути нескінченним.

Контрольні питання

1. Чим життєвий цикл ПЗ відрізняється від моделі життєвого циклу ПЗ?
2. Які характерні риси усіх моделей життєвого циклу ПЗ?
3. Які моделі ПЗ створюються на основних фазах ЖЦ ПЗ?
4. Що є результатами етапу тестування і для якого етапу ці результати є вхідними даними?
5. Що означає послідовність виконання етапів у ЖЦ ПЗ? Наведіть приклад.
6. Які критерії вибору конкретної моделі життєвого циклу ПЗ?
7. Які моделі життєвого циклу ПЗ належать до базових?
8. Скільки часу безпосередньо відводиться на програмування відносно усього ЖЦ ПЗ згідно типових моделей ЖЦ ПЗ?
9. Які моделі є перспективними з точки зору автоматизації процесів розроблення ПЗ?
10. Назвіть приклади змішаних моделей ЖЦ ПЗ?
11. Як умовно можна поділити промислові методології розроблення ПЗ?
12. Які фахівці є авторами базових моделей ЖЦ ПЗ?

Тема 3.2. КАСКАДНА МОДЕЛЬ.

Схема каскадної моделі

Каскадна модель (waterflow model) ЖЦ ПЗ виникла для потреби у систематизації робіт ще на ранніх етапах інженерії ПЗ.

Каскадна модель (інші назви – водоспадна, послідовна, класична, найпростіша) включає виконання таких етапів:

- **дослідження концепції** – початковий етап, коли відбувається дослідження предметної області і вимог, виробляється загальне бачення продукту й оцінюється можливість його реалізації.
- **визначення вимог** – етап роботи з вимогами, коли визначаються програмні вимоги до системи, її призначення, продуктивність, інтерфейси і т.д.
- **проектування** – конструкторський етап, коли розробляється й формулюється логічно послідовна технічна характеристика програмної системи, включаючи структури даних, архітектуру ПЗ, інтерфейсні подання й алгоритмічну деталізацію;
- **реалізація і тестування** – етап розроблення, коли проектний опис ПЗ перетворюється в повноцінний програмний продукт. Результат: вихідний код, база даних і документація. У реалізації звичайно виділяють два етапи: реалізацію компонент ПЗ й інтеграцію компонентів у готовий продукт. На обидвох етапах виконується кодування й тестування, які теж іноді розглядають як два підетапа. Важливо відзначити, що у сучасній версії каскадної моделі ці етапи розглядаються окремо порівняно з ранньою версію моделі, коли тестуванню не надавалося такої уваги.
- **експлуатація й підтримка** – етап використання готової програмної системи, а саме: запуск і поточне забезпечення, включаючи надання технічної допомоги, обговорення виниклих питань із користувачем,

реєстрацію запитів користувача на модернізацію й внесення змін, а також коректування або усунення помилок;

- **супровід** – етап усунення програмних помилок, збоїв, модернізації й внесення змін. Складається з ітерацій розроблення.

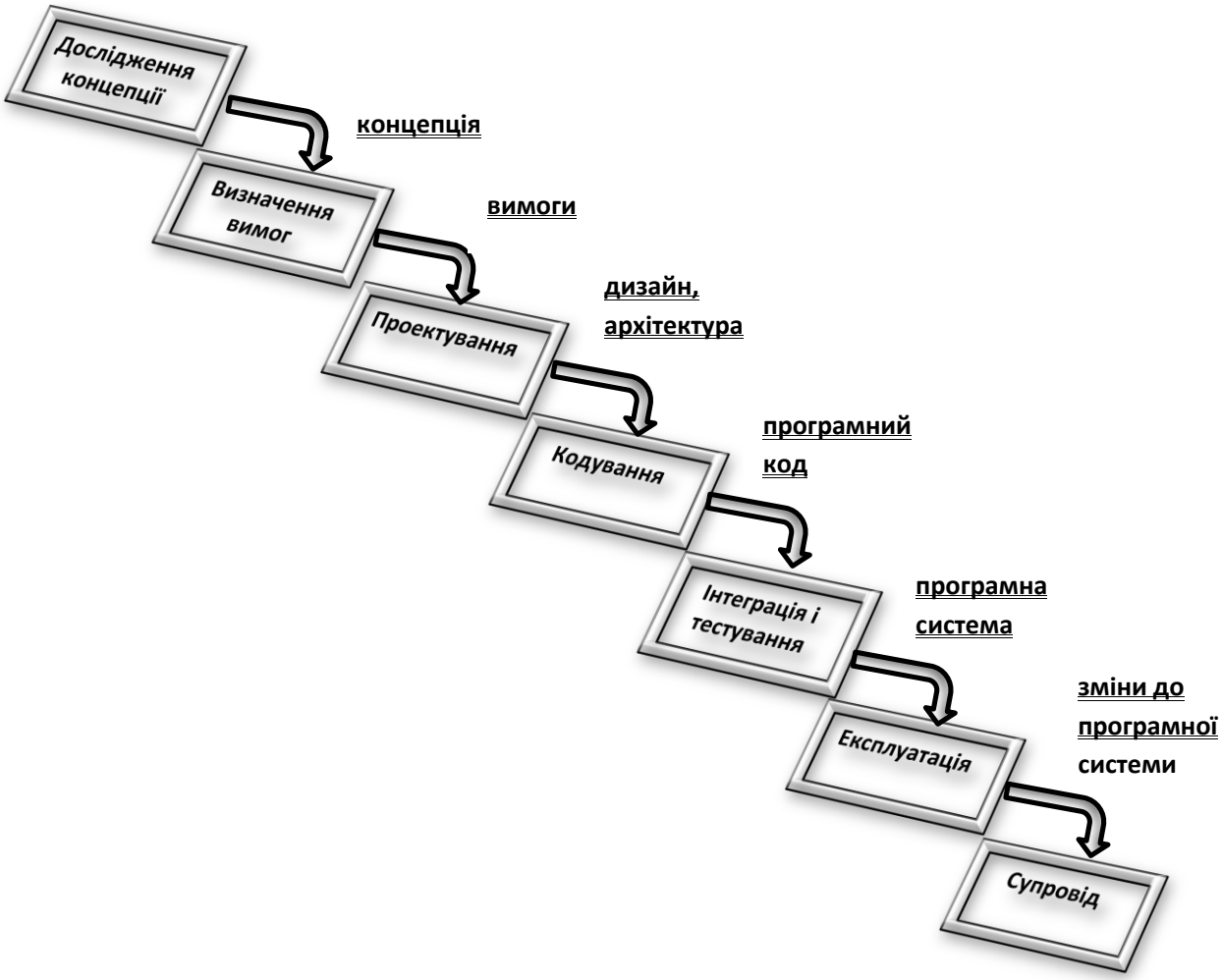


Рис. 3.1. Схема каскадної моделі.

Основними принципами каскадної моделі є:

- Строго послідовне виконання усіх фаз: кожна наступна фаза починається лише тоді, коли повністю завершене виконання попередньої фази.
- Кожна фаза має певні критерії входу й виходу: вхідні й вихідні дані.
- Кожна фаза повністю документується.

- Перехід від однієї фази до іншої здійснюється за допомогою формального огляду за участю замовника.
- Основа моделі – це те, що сформульовані вимоги (ТЗ) змінюватися не повинні.
- Критерій якості результату – це відповідність програмного продукту встановленим вимогам.

Переваги й недоліки каскадної модель.

Каскадна модель має такі переваги:

- Проста й зрозуміла замовникам, тому часто використається іншими організаціями для відстеження проектів, не пов'язаних з розробленням ПЗ.
- Проста й зручна в застосуванні:
  - процес розроблення виконується поетапно.
  - її структурою може керуватися навіть слабко підготовлений у технічному плані або - недосвідчений персонал;
  - вона сприяє здійсненню строгого контролю менеджменту проекту;
- Кожну стадію можуть виконувати незалежні команди, оскільки усе задокументовано.
- Дозволяє досить точно планувати строки й витрати.

При використанні каскадної моделі для «невідповідного» проекту можуть проявлятися наступні її основні недоліки:

- Спроба повернутися на одну або дві фази назад, щоб виправити яку-небудь проблему або недолік, приведе до значного збільшення витрат і збою в графіку;
- Інтеграція компонентів, на якій звичайно виявляється більша частина помилок, виконується наприкінці розроблення, що сильно збільшує вартість усунення помилок;

- Запізнювання з одержанням результатів, якщо в процесі виконання проекту вимоги змінилися, то вийде застарілий результат

Недоліки каскадної моделі особливо гостро проявляються у випадку, коли важко або неможливо сформулювати вимоги або вимоги можуть змінюватися в процесі виконання проекту. У цьому випадку розроблення ПЗ має принципово циклічний характер.

Не дивлячись на очевидні недоліки, каскадна модель є успішно застосованою в окремих чітко визначених умовах.

Уперше чітко сформульована в 1970 році Ройсом (W.W. Royce) каскадна модель на початковому періоді зіграла провідну роль як метод регулярного розроблення складного ПЗ. У 70-80-их роках XX століття модель була прийнята як стандарт міністерства оборони США. Згодом недоліки каскадної моделі стали проявлятися всі частіше й виникла думка, що вона безнадійно застаріла. Однак каскадна модель, багаторазово «розкритикована» і теорією, і практикою, продовжує зустрічатися в реальному житті.

Каскадна модель не втратила своєї актуальності при вирішенні таких типів завдань:

- Вимоги і їхня реалізація максимально чітко визначені й зрозумілі; використовується незмінне визначення продукту й цілком зрозумілі технічні методики. Це завдання типу:
    - науково-обчислювального характеру (пакети й бібліотеки наукових програм типу розрахунку несучих конструкцій мостів, розв’язування систем диференціальних рівнянь і т.п.);
    - операційні системи й компілятори;
    - системи реального часу керування конкретними об'єктами.
- Повторне розроблення типового продукту (бухгалтерський облік, навчальна система, редактор зображень і т.п.).

- Випуск нової версії вже існуючого продукту, якщо внесені зміни цілком визначені й керовані (перенесення уже існуючого продукту на нову платформу).
- Принципи каскадної моделі знаходять застосування як елементи моделей інших типів.

Каскадна модель життєвого циклу є ідеальною, оскільки лише дуже прості завдання проходять усі етапи без будь-яких ітерацій (повернень на попередні кроки процесу). Наприклад, при програмуванні може виявитися, що реалізація деякої функції неефективна і вступає у протиріччя з вимогами до продуктивності системи. У цьому випадку потрібні зміни проекту, а можливо, і переробка специфікацій. Для врахування повторюваності етапів процесу розроблення створювались альтернативи каскадної моделі.

Контрольні питання

1. Які основні фази каскадної моделі? Поясніть зміст кожної фази.
2. Яка послідовність виконання фаз в каскадній моделі?
3. Яка основна перевага каскадної моделі?
4. Назвіть умови застосування каскадної моделі.
5. Назвіть приклади розробок, коли каскадна модель може успішно застосовуватися.
6. Чому для більшості реальних проектів з розроблення ПЗ каскадна модель вважається застарілою?
7. У чому полягає ідеальність каскадної моделі?
8. Як по-іншому називають каскадну модель?
9. Хто автор каскадної моделі? Яке було її значення на ранніх етапах інженерії ПЗ?
10. Яке значення каскадної моделі в сучасних умовах інженерії ПЗ?
11. Які вимоги накладаються на специфікацію вимог у каскадній моделі?
12. Які вимоги є до процесу документування у каскадній моделі?

13. Поясніть, чому каскадна модель може застосовуватися для розроблення ОС.

14. Які обмеження накладаються на процес інтеграції в каскадній моделі?

15. Наведіть приклад використання каскадної моделі для проектів, що не стосуються розроблення ПЗ.

16. Чи може каскадна модель застосовуватися для розроблення ПЗ, коли вимоги нечітко визначені?

Тема 3.3. СПІРАЛЬНА МОДЕЛЬ.

Схема спіральної моделі

На практиці розроблення ПЗ має циклічний характер, тобто після виконання деяких стадій доводиться повертатися на попередні. Можна вказати дві основні причини таких повернень:

- Помилки розробників, допущені на ранніх стадіях і виявлені на пізніх стадіях – помилки аналізу, проектування, кодування, які виявляють, як правило, на стадії тестування.
- «Помилки» замовників, коли необхідна зміна вимог у процесі розроблення. Це пояснюється або неготовністю замовників сформулювати вимоги («Сказати, що повинна робити програма я зможу тільки після того, як побачу як вона працює»), або викликано змінами ситуації в процесі розроблення (зміни ринку, нові технології, ...).

Циклічний характер розроблення ПЗ розкрито у спіральній моделі ЖЦ, описаній Б. Боемом в 1988 році (Barry Boehm, "A Spiral Model of Software Development and Enhancement", IEEE Computer, Vol.21, No.5, pp. 61-72, 1988). Спіральна модель була запропонована як альтернатива каскадної моделі, що враховує повторюваний характер розроблення ПЗ.

Відмінною особливістю цієї моделі є спеціальна увага ризикам, які впливають на організацію життєвого циклу.

Схема роботи спіральної моделі виглядає як набір циклів спіралі, що розкручується. Кожен цикл представляється як розроблення варіантів програмного продукту.

При цьому, початкові стадії, пов'язані з аналізом і плануванням представлені більш докладно з додаванням нових елементів. У кожному циклі виділяються чотири базові фази:

- визначення цілей, альтернативних варіантів й обмежень.
- оцінка альтернативних варіантів, ідентифікація й дозвіл ризиків.

- розроблення продукту наступного рівня.
- планування наступної фази.

Рис. 3.2. Схема спіральної моделі ЖЦ ПЗ.

ПЕРШИЙ ЦИКЛ

«Розкручування» проекту починається з аналізу загальної постановки завдання на розроблення ПЗ. Тут на першій фазі визначаються загальні цілі, установлюються попередні обмеження, визначаються можливі альтернативи підходів до рішення завдання.

Далі проводиться оцінка підходів, установлюються їхні ризики.

На кроці розроблення створюється концепція (бачення) продукту й шляхів його створення.

НАСТУПНИЙ ЦИКЛ

Починається планування вимог і деталей ЖЦ продукту для оцінки витрат.

На фазі визначення цілей установлюються альтернативні варіанти вимог, пов'язані з аранжуванням вимог за важливістю й вартістю їхнього виконання.

На фазі оцінки встановлюються ризики варіантів вимог.

На фазі розроблення складається специфікація вимог (із вказівкою ризиків і вартості), готується демоверсія ПЗ для аналізу вимог замовником.

ПЕРЕДОСТАННІЙ ЦИКЛ

Розроблення проекту починається із планування розроблення.

На фазі визначення цілей установлюються обмеження проекту (за термінами, обсягом фінансування, ресурсами і т.п.), визначаються альтернативи проектування, пов'язані з альтернативами вимог, застосовуваними технологіями проектування, залученням субпідрядників і т.д.

На фазі оцінки альтернатив установлюються ризики варіантів і робиться вибір варіанта для подальшої реалізації.

На фазі розроблення виконується проектування й створюється демоверсія, що реалізує основні проектні рішення.

ЗАВЕРШАЛЬНИЙ ЦИКЛ

Наступний цикл пов'язаний з реалізацією ПЗ. Також починається із планування. Альтернативні варіанти реалізації пов’язані із застосовуванням різних технологій реалізації, використовуваними ресурсами.

Оцінка альтернатив і пов'язаних з ними ризиків на цьому циклі визначається ступенем «відпрацьованості» технологій й «якістю» наявних ресурсів. Фаза розроблення виконується за каскадною моделлю з виходом – діючим варіантом (прототипом) продукту.

Основні принципи спіральної моделі можна сформулювати в такий спосіб:

- Розроблення варіантів продукту відповідає різним варіантам вимог з можливістю повернутися до більше ранніх варіантів.

- Створення прототипів ПЗ як засобу спілкування із замовником для уточнення й виявлення вимог.
- Активне залучення замовника до роботи над проектом. Замовник бере участь в оцінці чергового прототипу ПЗ, уточненні вимог при переході до наступного, оцінці запропонованих альтернатив чергового варіанта й оцінці ризиків.
- Планування наступних варіантів з оцінкою альтернатив й аналізом ризиків, пов'язаних з переходом до наступного варіанта.
- Перехід до розроблення наступного варіанта перед завершення попереднього у випадку, коли ризик завершення чергового варіанта (прототипу) стає невиправдано високий.
- Використання каскадної моделі як схеми розроблення чергового варіанта.

Особливості спіральної моделі:

- До початку розроблення ПЗ є кілька повних циклів аналізу вимог і проектування.
- Кількість циклів моделі (як у частині аналізу й проектування, так й у частині реалізації) не обмежено й визначається складністю й обсягом завдання
- У моделі передбачаються повернення на попередні варіанти при зміні вартості ризиків.

Переваги і недоліки спіральної моделі.

Спіральна модель має наступні очевидні переваги:

- Більше ретельне проектування (кілька початкових ітерацій) з оцінкою результатів проектування, що дозволяє виявити помилки проектування на більше ранніх стадіях.

- Поетапне уточнення вимог у процесі виконання ітерацій, що дозволяє більш точно задовольнити вимогам замовника
- Участь замовника у виконанні проекту з використанням прототипів програми. Замовник бачить, що і як створюється, не висуває необґрунтованих вимог, оцінює реальні обсяги фінансування.
- Планування й керування ризиками при переході на наступні ітерації дозволяє розумно планувати використання ресурсів й обґрунтовувати фінансування робіт.
- Можливість розроблення складного проекту «вроздріб», виділяючи на перших етапах найбільш значимі вимоги.

Основні недоліки спіральної моделі пов'язані з її складністю:

- Складність аналізу й оцінки ризиків при виборі варіантів.
- Складність підтримки версій продукту (зберігання версій, повернення до ранніх версій, комбінування версій передбачає контроль над версіями).
- Складність оцінки точки переходу на наступний цикл.
- «Нескінченність» моделі виглядає так, що на кожному витку замовник може висувати нові вимоги, які приводять до необхідності наступного циклу розроблення.

Застосовність спіральної моделі визначається її перевагами та недоліками.

Спіральну модель доцільно застосовувати при наступних умовах:

- Коли користувачі не впевнені у своїх потребах або коли вимоги занадто складні й можуть змінюватися в процесі виконання проекту й необхідно виконувати прототипування для аналізу й оцінки вимог.
- Коли досягнення успіху не гарантовано й необхідна оцінка ризиків продовження проекту.
- Коли проект є складним, вартісним й обґрунтування його фінансування можливо тільки в процесі його виконання.

- Коли мова йде про застосування нових технологій, що пов’язане з ризиком їхнього освоєння й досягнення очікуваного результату.
- При виконанні дуже великих проектів, які в силу обмеженості ресурсів можна робити тільки врозділ.

Контрольні питання

1. Які основні фази спіральної моделі?

2. Яка послідовність виконання фаз в спіральній моделі?

3. Яка основна перевага спіральної моделі?

4. Назвіть умови застосування спіральної моделі.

5. Назвіть приклади розробок, коли спіральна модель може успішно застосовуватися.

6. Чому для більшості реальних проектів з розроблення ПЗ спіральна модель є більш застосовною, ніж каскадна?

7. У чому полягає складність спіральної моделі?

8. Яка особливість спіральної моделі стосовно аналізу ризиків?

9. Хто автор спіральної моделі? Яке було її значення на ранніх етапах інженерії ПЗ?

10. Яке значення каскадної моделі в спіральній розробці ПЗ ?

11. Які роботи виконуються з вимогами у спіральній моделі?

12. Які вимоги є до замовника у спіральній моделі?

13. Поясніть, як на практиці виглядає «нескінченість» спіральної моделі.

14. Як позначається вартість програмного продукту на схемі спіральної моделі?

15. Наведіть приклад використання спіральної моделі для проектів з розроблення ПЗ.

Тема 3.4. МОДЕЛІ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ РОЗРОБЛЕННЯ

Ключовим завданням інженерії ПЗ є створення методів і засобів автоматизованого розроблення ПЗ. Теоретики і практики вирішують проблему, як зробити з ІТ-компанії фабрику зі створення програмних систем.

Формальна та компонентна моделі ЖЦ ПЗ є спробами автоматизувати розроблення ПЗ.

Формальна модель

Модель формальних перетворень (формальна модель) полягає в автоматичній побудові програми на основі її формального опису, що гарантує отримання коду, який буде абсолютно точно відповідати початковій специфікації.

Формальні перетворення – це набір методів, що дозволяють математично коректно трансформувати вихідні специфікації в код цільової програмної системи. Оскільки перехід від вимог до коду відбувається математично коректно, то проблема тестування і доведення коректності кінцевої програми по відношенню до специфікації зникає.

Очевидно, що цей метод вимагає наявності формальних специфікацій, складання та доведення коректності яких є нетривіальними завданнями. Саме тому метод формальних перетворень рідко використовується для всього програмного комплексу. У більшості випадків він застосовується для тієї частини складної системи і реалізації тих алгоритмів, де вихідні вимоги добре формалізуються .Загалом формальна модель подібна до каскадної.

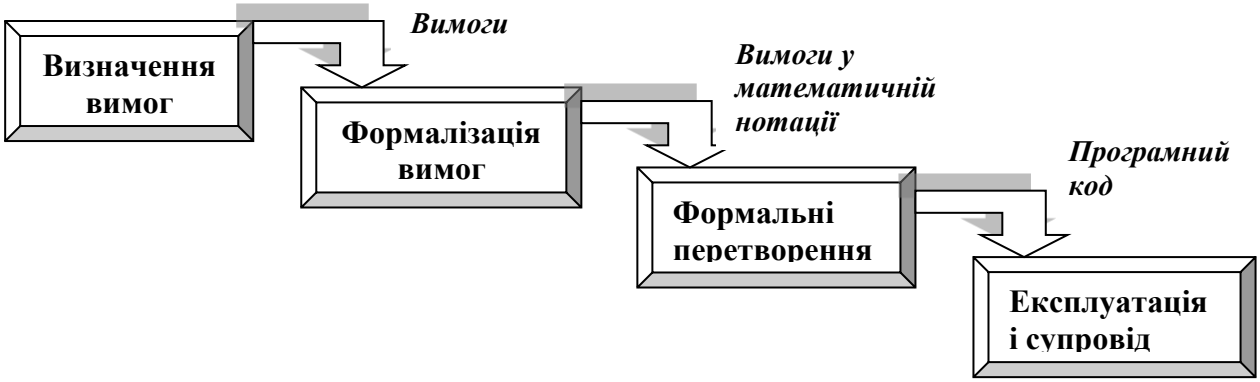


Рис. 3.3. Схема формальної моделі.

Принципи формальної моделі можна сформулювати в такий спосіб:

1) специфікація вимог має вигляд деталізованої формалізованої специфікації, записаної з допомогою математичної нотації;

2) процеси проектування, кодування, тестування замінюються автоматизованим процесом, в якому формалізована специфікація шляхом відповідних формальних перетворень трансформується в програму.

Перевага формальної моделі – це точна відповідність програми вимогам. Істотним недоліком є те, що вибір для застосування відповідних форм перетворень складний і неочевидний.

Приклад застосування формальної моделі є «модель чистої кімнати» (IBM), для неї характерна покрокове розроблення з формальними перетвореннями. Формальна модель застосовна до систем, які задовольняють строгим вимогам надійності, безпеки. Але вимагається наявність спеціальних знань і досвіду розробників. Загалом, очевидно, що описати програмну систему методом формальної специфікації досить складно.

Компонентна модель

Ще одною перспективною моделлю життєвого циклу ПЗ з точки зору автоматизації є компонентна модель. Компонентна модель передбачає

багаторазове використання окремих компонентів для конструювання програмної системи. Час створення цієї моделі – кінець 90-х років 20 століття.

Компонента – це як і проста бібліотечна програма, так і ціла програмна система. При компонентному програмуванні програмна система складається із окремо створених елементів (компонентів), які викликають один одного через інтерфейси. Зміни в існуючу систему вносяться додаванням нових компонентів або заміною існуючих. При цьому нові компоненти, які заміняють раніше створені, повинні наслідувати інтерфейси базового.

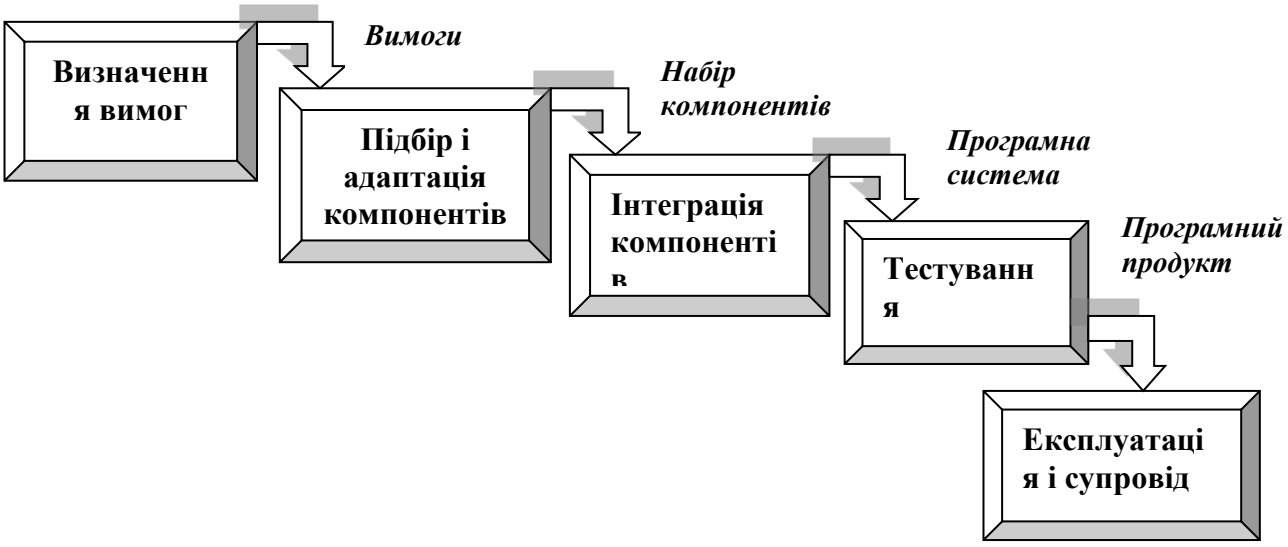


Рис. 3.4. Схема компонентної моделі.

Розглядають п’ять рівнів абстракції компонентів:

1) *Функціональна абстракція*. Компонента реалізує окрему функцію, інтерфейс з системою забезпечує сама функція.
2) *Безсистемне групування*. Компонента – це набір слабо пов’язаних між собою програмних об’єктів і підпрограм. Наприклад, оголошення даних, функції. Інтерфейс складається з усіх назв об’єктів в групуванні.
3) *Абстракція даних*. Компонента – це клас, описаний об’єктно-орієнтованою мовою програмування. Інтерфейс складається з

методів, які забезпечують створення, зміну і доступ до абстракції даних.

4) *Абстракція класів*. Компонента – це група зв'язаних класів, які працюють разом. Інтерфейс – це композиція усіх інтерфейсів об'єктів.

5) *Системна абстракція*. Компонента є автономною системою. Інтерфейсом є програмний інтерфейс додатків (API – Application Programming Interface).

Основна специфіка компонентної моделі – це комерційність компонентів і, відповідно, недоступність вихідного коду компонентів. Як наслідок, виникають такі проблеми:

1. Супровід і модернізація систем сконструйованих з компонентів є важко вирішуваними задачами.
2. Дуже часто в процесі розроблення виникає додаткова робота з адаптації компонентів, що значно збільшує вартість вихідного програмного продукту.

Для забезпечення повторного використання компонентів кожна компанія-розробник ПЗ накопичує свій досвід у вигляді бібліотек компонентів різноманітного призначення. Повторне використання власних компонентів у проектах з розроблення ПЗ значно підвищує прибутки ІТ-компаній.

Контрольні питання

1. Які основні фази формальної моделі ЖЦ ПЗ?
2. Яка послідовність виконання фаз у формальній моделі ЖЦ ПЗ?
3. Яка основна перевага формальної моделі ЖЦ ПЗ?
4. Назвіть умови застосування формальної моделі ЖЦ ПЗ.
5. Назвіть приклади розробок, коли формальна модель може успішно застосовуватися.

6. Чому для більшості реальних проектів з розроблення ПЗ формальна модель є менш застосовною, ніж інші?

7. У чому полягає складність формальної моделі ЖЦ ПЗ?

8. Яка особливість формальної моделі стосовно вимог?

9. Які основні фази компонентної моделі ЖЦ ПЗ?

10. Які є рівні абстракцій компонентів?

11. Які роботи виконуються з компонентами в однойменній моделі?

12. Які недоліки є в компонентній моделі ЖЦ ПЗ?

13. Яка особливість процесу тестування у формальній моделі ЖЦ ПЗ?

14. Що означає Application Programming Interface?

15. Що таке інтерфейс класу?

16. Чому формальна та компонентна моделі ЖЦ ПЗ вважаються найперспективнішими серед інших моделей з точки зору автоматизації процесів розроблення?

17. Що таке формальні перетворення в однойменній моделі ЖЦ ПЗ?

18. Що таке компонента програмної системи?

Тема 3.5. ЗМІШАНІ ТИПИ МОДЕЛЕЙ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Каскадна й спіральна моделі встановлюють деякі принципи організації життєвого циклу створення програмного продукту. Кожна з них має переваги, недоліки й області застосовності. Каскадна модель проста, але застосовна у випадку, коли вимоги відомі й змінюватися не будуть. Спіральна модель враховує такі важливі показники проекту як змінюваність вимог, неможливість оцінити заздалегідь обсяг фінансування, ризики виконання проекту. Але спіральна модель складна й вимагає більших витрат на підтримку.

Існують деякі інші типи моделей ЖЦ ПЗ, які можна розглядати як «проміжні» між каскадною й спіральною моделями, їх ще називають моделями-гібридами. Ці моделі використають окремі переваги каскадної й спіральної моделей і досягають успіху для певних типів завдань з розроблення ПЗ.

Ітераційна модель

Ітераційна модель життєвого циклу є розвитком класичної каскадної моделі й припускає можливість повернень з кожного етапу ЖЦ ПЗ на раніше виконані етапи. При цьому, причинами повернень у класичній ітераційній моделі є виявлені помилки, усунення яких і вимагає повернення на попередні етапи залежно від типу (природи) помилки – помилки кодування, проектування, специфікації або визначення вимог.

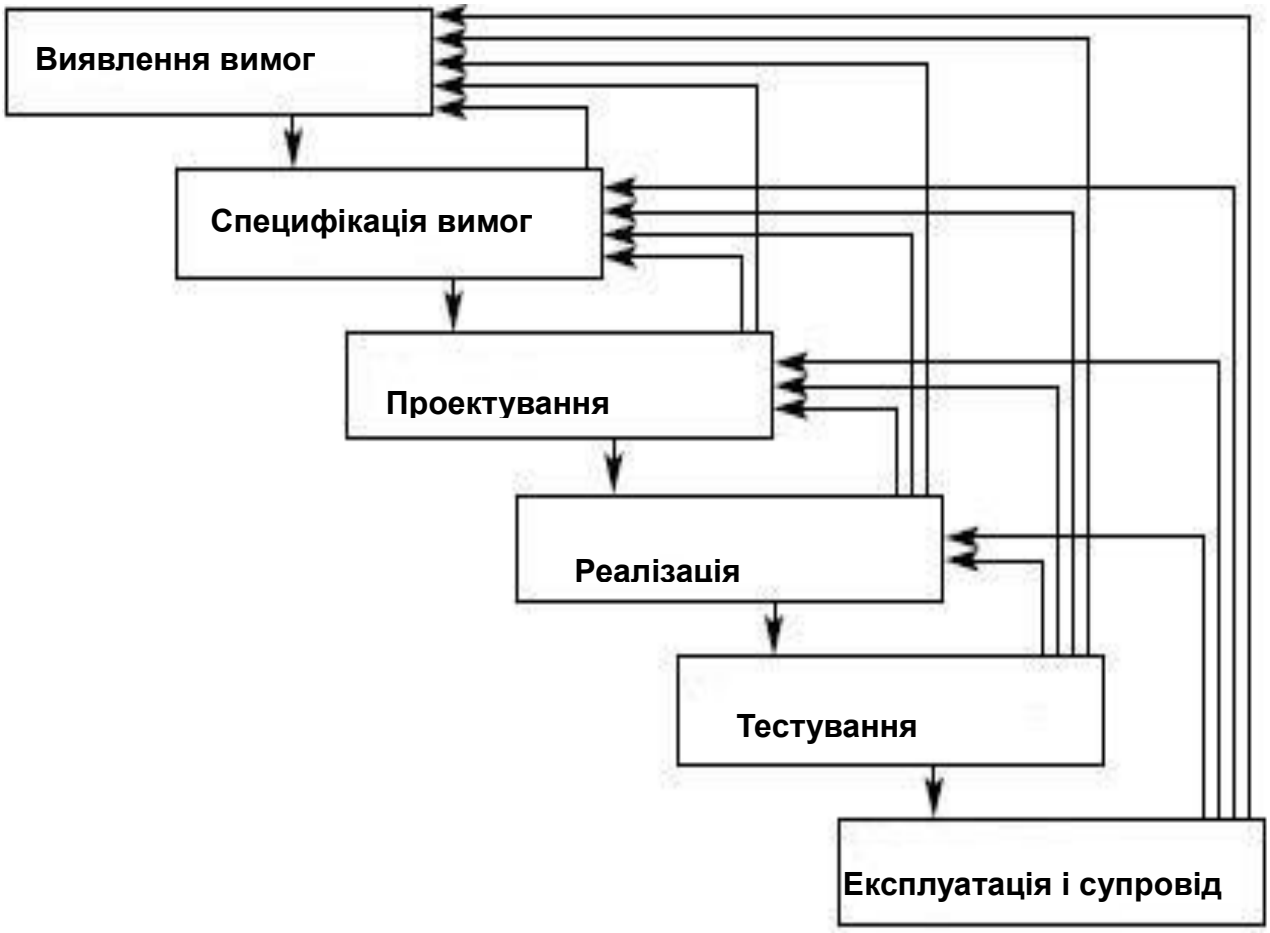


Рис. 3.5. Схема ітераційної моделі ЖЦ ПЗ

Реально ітераційна модель є більше життєвою, ніж класична каскадна модель, тому що створення ПЗ завжди пов'язане з усуненням помилок. Слід зазначити, що вже в першій статті, присвяченої каскадній моделі, Боем відзначав цю обставину й описав ітераційний варіант каскадної моделі. Практично всі застосовувані моделі життєвого циклу мають ітераційний характер, але цілі ітерацій можуть бути різними.

V-подібна модель

V-подібна модель була створена як ітераційний різновид каскадної моделі. Назва моделі відображає вигляд її схеми. Цілями ітерацій у цій моделі є забезпечення процесу тестування. Тестування продукту обговорюється, проектується й планується на ранніх етапах життєвого циклу розробки. План

випробування приймання програмного продукту замовником розробляється на етапі планування, а компонентного випробування системи – на фазах аналізу, розроблення проекту й т.д. Крім планів, на ранніх етапах розробляються також і тести, які будуть проводитись при завершенні паралельних етапів.

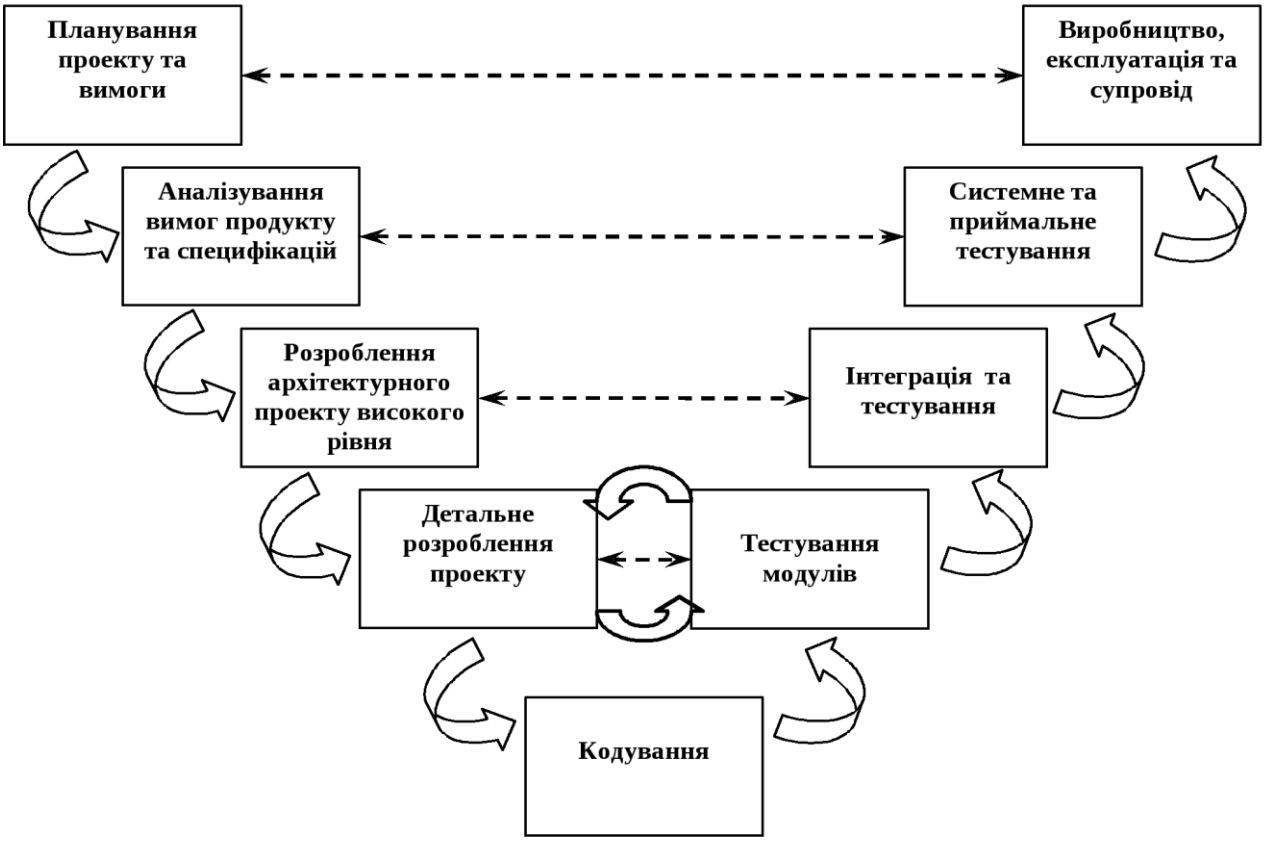


Рис. 3.6. Схема V-подібної моделі ЖЦ ПЗ.

Процес розробки планів випробування та відповідних тестів позначений на схемі пунктирною лінією з стрілкою праворуч між прямокутниками V-подібної моделі. Пунктирна лінія зі стрілкою ліворуч вказує на повернення у випадку непроходження відповідних тестів.

Інкрементна (покрокова) модель

Інкрементна розробка – це процес поетапної реалізації всієї системи й поетапного нарощування (збільшення) функціональних можливостей. На першому кроці необхідний повний заздалегідь сформульований набір вимог,

які діляться за деякою ознакою на частини (важливість, складність реалізації, вартість реалізації і т.д.). Далі вибирається перша група вимог і виконується повний прохід за каскадною моделлю. Після того, як перший варіант системи, що виконує першу групу вимог зданий замовникові, розробники переходять до наступного кроку (другого інкременту) з розроблення варіанта, що виконує другу групу вимог і т.д., поки не будуть реалізовані всі вимоги.

Особливістю інкрементної моделі є розроблення приймальних тестів на етапі аналізу вимог, що спрощує приймання варіанта системи замовником і встановлює чіткі цілі розроблення чергового варіанта системи.

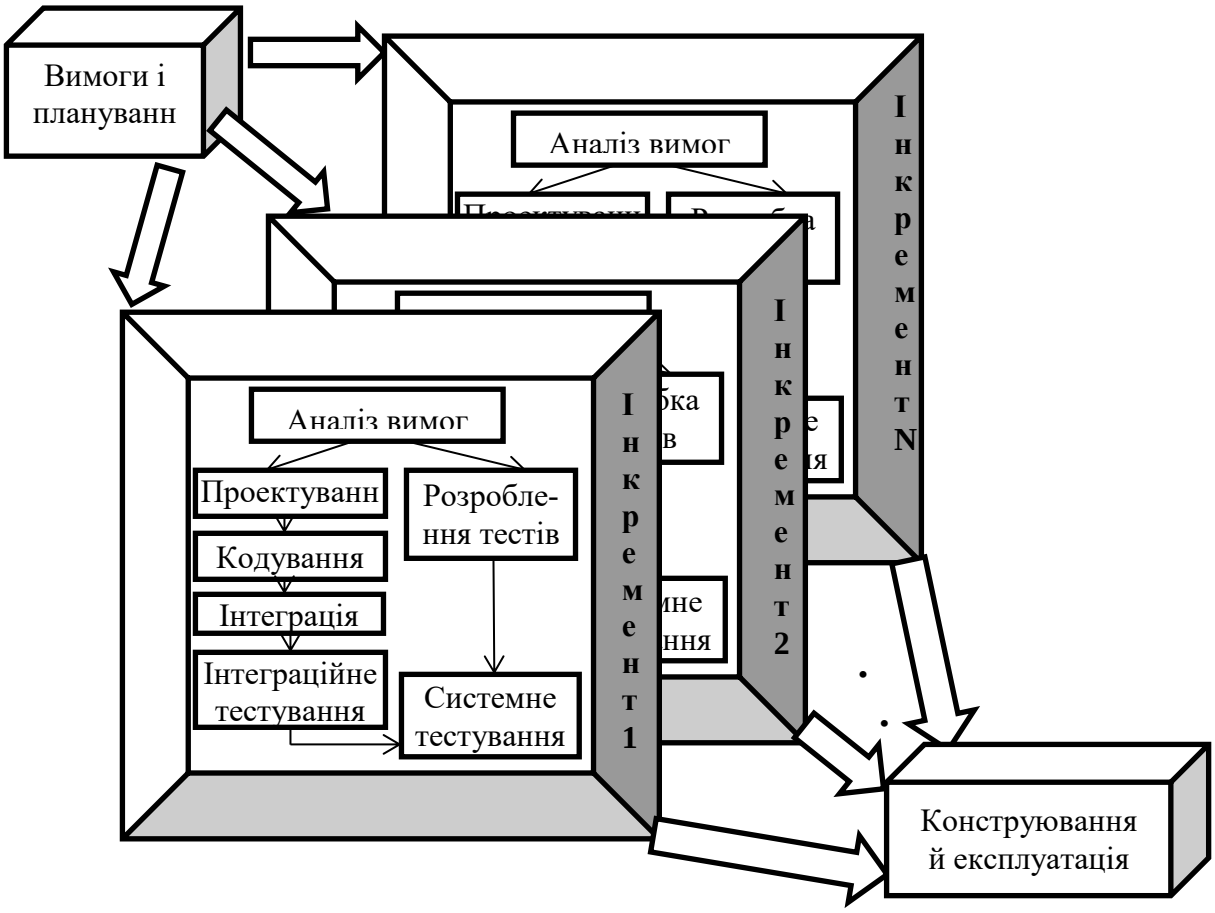


Рис.3.7. Схема інкрементної моделі ЖЦ ПЗ,

Інкрементна модель особливо ефективна у випадку, коли завдання розбивається на незалежні підзадачі (розроблення підсистем «Зарплата», «Бухгалтерія», «Склад», «Постачальники»).

Інкрементна модель може для внутрішньої ітерації використати не тільки каскадну, але й інші типи моделей.

Модель швидкого прототипування

Модель швидкого прототипування призначена для швидкого створення прототипів продукту з метою уточнення вимог і поетапного розвитку прототипів у кінцевий продукт. Таку модель можна вважати дуже спрощеним варіантом спіральної моделі.

Швидкість (висока продуктивність) виконання проекту забезпечується плануванням розроблення прототипів й участю замовника в процесі розроблення.

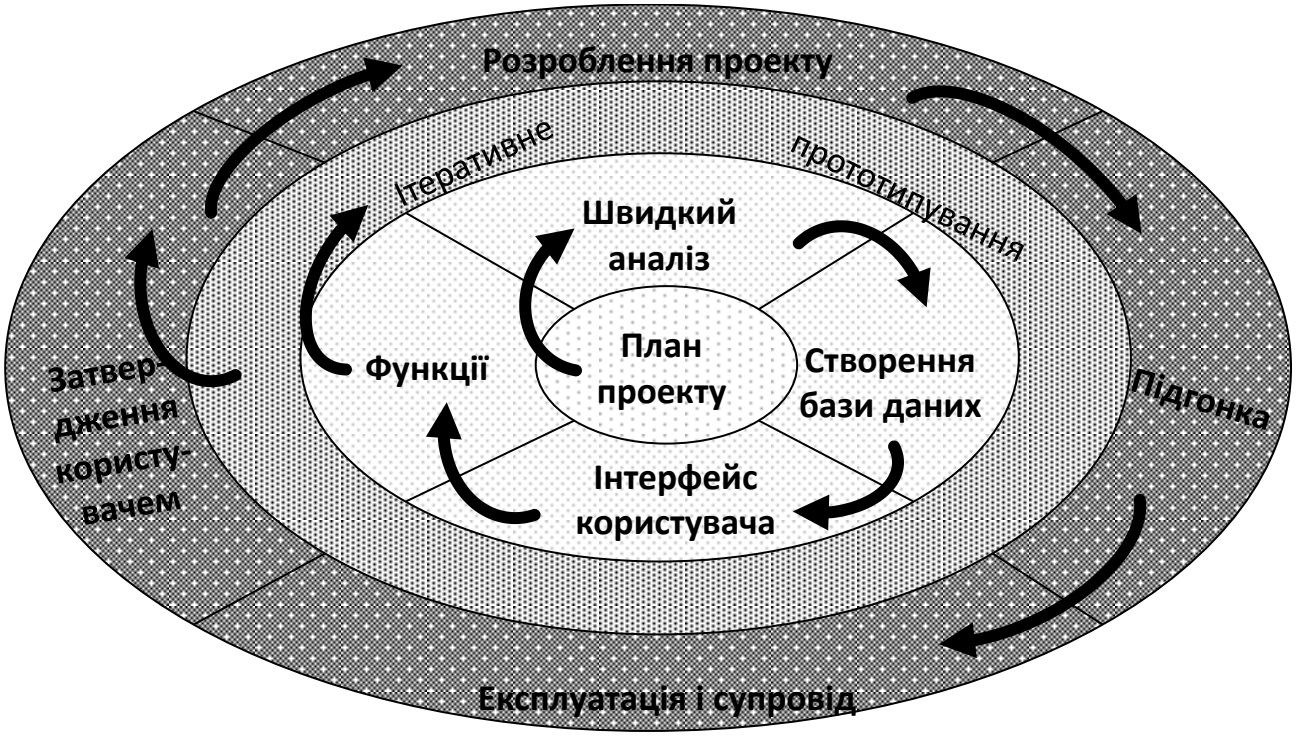


Рис. 3.8. Схема моделі швидкого прототипування .

Схема цієї моделі виглядає як багатошаровий еліпс. Початок життєвого циклу розроблення розміщено в центрі еліпса. Разом з користувачем розробляється попередній план проекту на основі попередніх вимог. Результат початкового планування – документ, що описує загалом графіки й очікувані результати.

Наступний рівень – створення вихідного прототипу на основі швидкого аналізу, проекту бази даних, користувальницького інтерфейсу й деяких функцій. Потім починається ітераційний цикл швидкого прототипування. Розробник проекту демонструє черговий прототип, користувач оцінює його функціонування, спільно визначаються проблеми й шляхи їхнього подолання для переходу до наступного прототипу. Цей процес триває доти, поки користувач не погодиться, що черговий прототип у точності відображає всі вимоги.

Одержавши схвалення замовника, швидкий прототип перетворюють у детальний проект, і систему набудовують на експлуатаційне використання. Саме на цьому етапі налаштування (підгонки) швидко створений прототип стає повністю діючою системою.

При розробці остаточної версії програми може реалізовуватися більш високий рівень функціональних можливостей, забезпечуватися використання різних системних ресурсів. Після цього проводиться тестування як в нормальних, так і в граничних режимах, а потім, як звичайно, передбачається супровід готової програмної системи.

Контрольні питання

1. Які основні фази ітераційної моделі?
2. Яка послідовність виконання фаз в ітераційній моделі ЖЦ ПЗ?
3. Як визначається ітерація в однойменній моделі ЖЦ ПЗ?
4. Порівняйте каскадну й ітераційну моделі ЖЦ ПЗ.
5. Які основні фази V-подібної моделі ЖЦ ПЗ?

6. Як місце тестування у V-подібній моделі ЖЦ ПЗ?

7. Як визначається ітерація у V-подібній моделі ЖЦ ПЗ?

8. Яке походження назви V-подібної моделі ЖЦ ПЗ?

9. Які основні фази інкрементної моделі ЖЦ ПЗ?

10. Яка інша назва інкрементної моделі?

11. Що таке інкремент в однойменній моделі ЖЦ ПЗ?

12. Які роботи проводяться з вимогами в інкрементній розробці ПЗ ?

13. Коли особливо ефективна інкрементне розроблення ПЗ?

14. Як інші моделі ЖЦ ПЗ можуть використовуватися в інкрементній моделі? Наведіть приклад.

15. Як інкременти пов’язані в одноіменній моделі ЖЦ ПЗ?

16. Які основні фази моделі швидкого прототипування?

17. Що таке прототип програмної системи?

18. З чого починається розроблення ПЗ за моделлю швидкого прототипування?

19. Чим забезпечується швидкість у розробленні за моделлю швидкого прототипування ?

20. Які необхідні вимоги для застосування моделі швидкого прототипування?

21. Які роботи виконуються після планування в моделі швидкого прототипування?

22. Який зміст ітеративного прототипування?

Тема 3.6. МЕТОДОЛОГІЇ РОЗРОБЛЕННЯ ПРОГРАМНИХ СИСТЕМ - MSF, RUP, XP

В індустрії ПЗ широке застосування отримали так звані промислові методології створення програмного продукту. Ці методології були розроблені відомими фірмами, що нагромадили великий досвід створення ПЗ. Методології представлені описами принципів, методів, процесів й операцій. Вони, як правило, підтримуються набором CASE-засобів, охоплюють всі етапи життєвого циклу продукту й успішно застосовуються для вирішення практичних завдань.

Слово CASE є скороченням від Computer Aided Software Engineering. При буквальному розгляді CASE-системою можна назвати будь-яку програмну систему, яка допомагає в розробці програм, включаючи будь-який транслятор. Але реально до CASE-систем можна віднести системи, що підтримують усі етапи розроблення програмного продукту. У міру розвитку CASE, цим терміном почали позначати різні програмні засоби, пов'язані з автоматизацією проектування і розроблення програм. Сучасні CASE-системи – це засоби не тільки розроблення програмних систем, але й організаційно-керуючі інструменти. З їх допомогою вирішуються завдання бізнес-моделювання, бізнес-аналізу, організації та реорганізації бізнес-процесів, планування, звітування і т. п.

Розглянемо особливості моделей життєвого циклу трьох найбільш відомих промислових методологій:

- Microsoft Solution Framework (MSF) – розробка фірми Microsoft, призначена для широкого кола завдань, які стосуються IT-рішень. Методологія масштабована, тобто налаштовується на вирішення завдань будь-якої складності колективом будь-якої чисельності.
- Rational Unified Process (RUP) – розробка фірми Rational, яка довгий час успішно займалася створенням CASE-засобів, що застосовуються на різних етапах життєвого циклу продукту від аналізу до тестування й

документування. Аналогічно MSF, RUP універсальна, масштабована на застосування в конкретних умовах.

- Extreme Programming (XP) – методологія, яка активно розвивається останнім часом, призначена для вирішення невеликих завдань відносно невеликими колективами професійних розробників в умовах жорстко обмеженого часу.

Кожна із цих методологій має свої особливості організації життєвого циклу створення програмного продукту. Перші дві методології називають «важкими», жорстко формалізованими, а остання – представник так званих гнучких методологій (Agile Development).

Модель Microsoft Solution Framework

MSF пропонує керівництво з організації людей і проекту для планування, побудови і розгортання успішних IT рішень.

Початок робіт датується 1991 роком. Випуск публічної версії був здійснено у 1993 р. (v1). Повна ревізія попередньої версії здійснена у 1998 р. (v2). Третя редакція - у 2003 р. (v3). У 2006 (v4) здійснена реалізація у вигляді Team Foundation Server Process Templates.

На відміну від методології framework – це набір концептуальних інструментів і кращих практик.

Одна з особливостей методології MSF полягає в тому, що вона орієнтована не просто на створення програмного продукту, який задовольняє перерахованим вимогам, а на пошук рішення проблем, що стоїть перед замовником. Відмінності полягають у тому, що перераховані замовником вимоги є проявами деяких більше глибоких проблем і неточність, неповнота, зміна вимог у процесі розроблення – наслідок недорозуміння проблем. Тому, у методології MSF велика увага приділяється аналізу проблем замовника й розробці варіантів системи для пошуку рішення цих проблем.

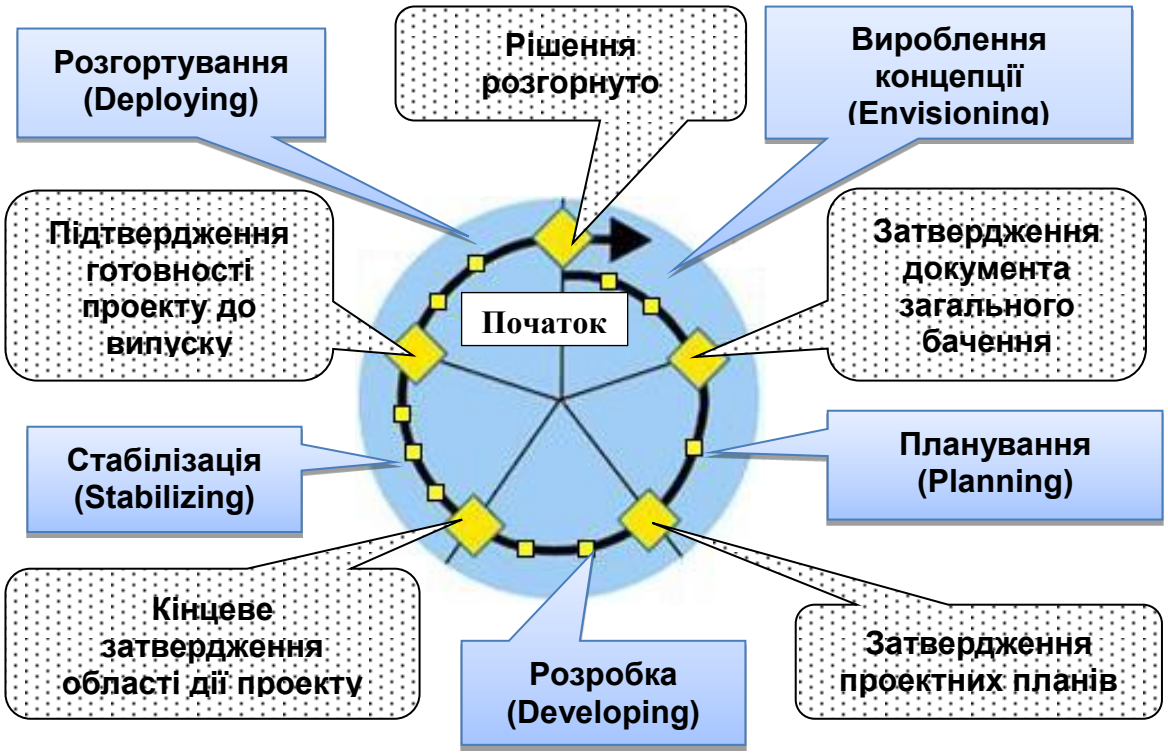


Рис. 3.9. Схема моделі MSF.

Модель життєвого циклу MSF є деяким гібридом каскадної й спіральної моделей, сполучаючи простоту керування каскадної моделі із гнучкістю спіральної.

Модель життєвого циклу MSF орієнтована на milestones – ключові точки проекту, що характеризують досягнення якого-небудь істотного результату. Цей результат може бути оцінений і проаналізований і дає відповідь на питання: “А чи досягли ми цілей, поставлених на цьому кроці?».

У моделі передбачається наявність основних фаз і проміжних, що відображають внутрішні етапи головних фаз.

Основними фазами моделі MSF є:

- Вироблення концепції/Створення загальної картини (Envisioning).

Концепція – це нічим необмежене уявлення про те, яким має бути рішення. На цьому етапі вирішуються такі основні завдання: оцінка існуючої ситуації; визначення складу команди, структури проекту, бізнесів-цілей, вимог

і профілів користувачів; розроблення концепції рішення й оцінка ризику. Установлюються дві проміжні фази: «Організований кістяк команди» й «Створена загальна картина рішення».

- Планування (Panning).

Цей етап включає планування й проектування продукту. На основі аналізу вимог розробляється проект й основні архітектурні рішення, функціональні специфікації системи, плани й календарні графіки, середовища розробки, тестування й пілотної експлуатації. Етап складається із трьох стадій: концептуальне, логічне й фізичне проектування.

На стадії **концептуального** проектування завдання розглядається з погляду користувацьких і бізнесів-вимог й закінчується визначенням набору сценаріїв використання системи. При **логічному** проектуванні завдання розглядається з погляду проектної команди, рішення представляється у вигляді набору сервісів. І вже на стадії **фізичного** проектування завдання розглядається з погляду програмістів, уточнюються використовувані технології й інтерфейси.

- Розроблення (Developing).

Створюється варіант вирішення проблеми у вигляді коду й документації чергового прототипу, включаючи специфікації й сценарії тестування. Основна віха етапу – «Остаточне затвердження області дії проекту». Продукт готовий до зовнішнього тестування й стабілізації. Крім того, замовники, користувачі, співробітники служби підтримки й супроводи, а також ключові учасники проекту можуть попередньо оцінити продукт і вказати всі недоліки, які потрібно усунути до його поставки.

- Стабілізація (Stabilizing).

Підготовка до випуску остаточної версії продукту, доведення його до заданого рівня якості. Тут виконується комплекс робіт з тестування (виявлення й усунення дефектів), перевіряється сценарій розгортання продукту. Коли рішення стає досить стійким, проводиться його пілотна експлуатація в

тестовому середовищі із залученням користувачів і застосуванням реальних сценаріїв роботи.

- Розгортання (Deploying).

Виконується установка рішення й необхідних компонентів оточення, проводиться його стабілізація в промислових умовах і передача проекту в руки групи супроводу. Крім того, аналізується проект у цілому на предмет рівня задоволеності замовника.

Модель Rational Unified Process

Модель життєвого циклу RUP є досить складною, детально пророблено ітеративно-інкрементною моделлю з елементами каскадної моделі. У моделі RUP виділяються 4 основні фази, 9 видів діяльності (процесів). Крім того, у моделі описується ряд практик, які варто застосовувати або керуватися для успішного виконання проекту. RUP орієнтований на поетапне моделювання створюваного продукту за допомогою мови UML (Unified Modelling Language).

Основними фазами RUP є:

- **Фаза початку проекту (Inception).** Визначаються основні цілі проекту, бюджет проекту, основні засоби його виконання – технології, інструменти, ключовий персонал, складаються попередні плани проекту. Основна мета цієї фази – досягти компромісу між всіма зацікавленими особами щодо завдань проекту.
- **Фаза пророблення (Elaboration).** Основна мета цієї фази – на базі основних, найбільш істотних вимог розробити стабільну базову архітектуру продукту, що дозволяє вирішувати поставлені перед системою завдання й надалі використається як основа розроблення системи.

- **Фаза побудови (Construction).** Основна мета цієї фази – детальне прояснення вимог і розроблення системи, що задовольняє їм, на основі спроектованої раніше архітектури.
- **Фаза передачі (Transition).** Ціль фази – зробити систему повністю доступною кінцевим користувачам. Тут відбувається остаточне розгортання системи в її робочому середовищі, підлаштування дрібних деталей під потреби користувачів.

У рамках кожної фази можливе проведення декількох ітерацій, кількість яких визначається складністю виконуваного проекту.

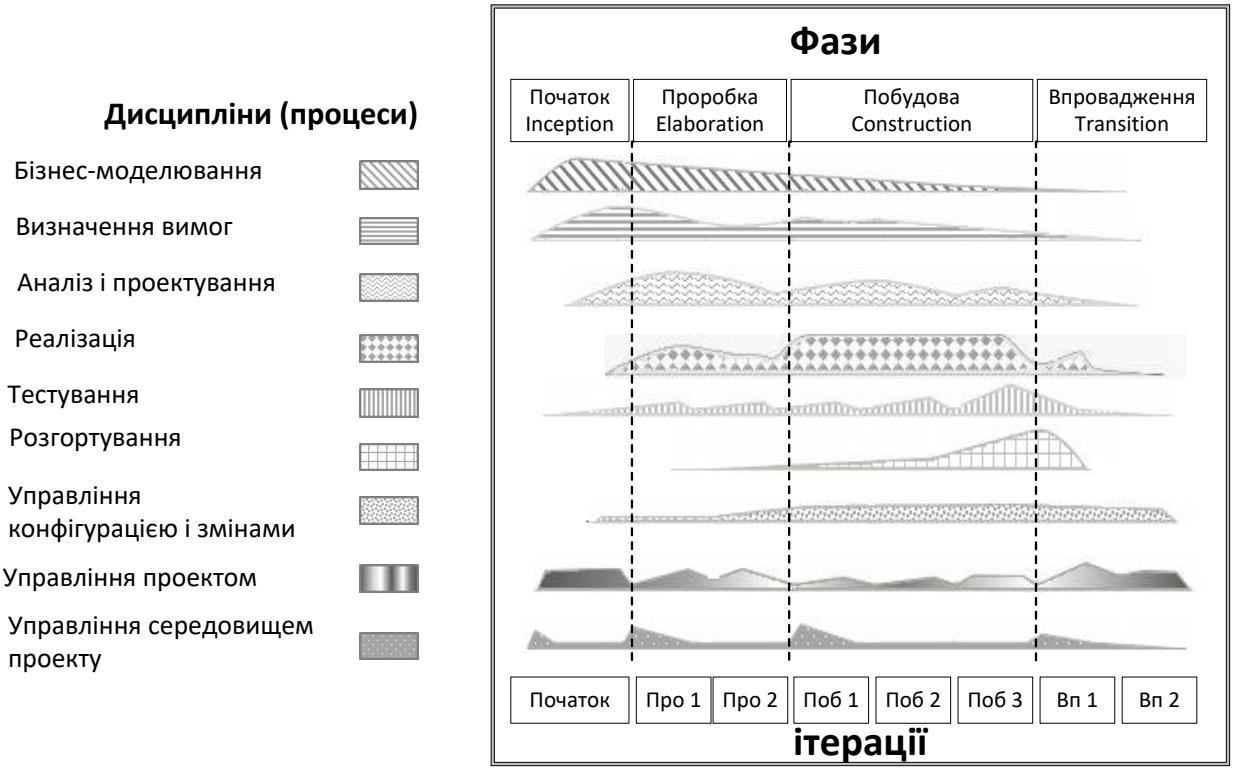


Рис.3.10. Схема методології RUP

Діяльності (основні процеси) RUP діляться на п’ять основних і чотири підтримуючі. До основних відносяться:

- Моделювання предметної області (бізнес-моделювання, Business Modeling). Мети цієї діяльності – зрозуміти бізнес-контекст, у якому

повинна буде працювати система (і переконатися, що всі зацікавлені особи розуміють його однаково), зрозуміти можливі проблеми, оцінити можливі їхні рішення і їхні наслідки для бізнесу організації, у якій буде працювати система.

- Визначення вимог (Requirements). Мета – зрозуміти, що повинна робити система, визначити межі системи й основу для планування проекту й оцінювання витрат ресурсів у ньому.
- Аналіз і проектування (Analysis and Design). Вироблення архітектури системи на основі ключових вимог, створення проектної моделі, представленої у вигляді діаграм UML, що описують продукт із різних точок зору.
- Реалізація (Implementation). Розроблення вихідного коду, компонент системи, тестування й інтеграція компонентів.
- Тестування (Test). Загальна оцінка дефектів продукту, його якість у цілому; оцінка ступеня відповідності вихідним вимогам.

Підтримуючими процесами є:

- Розгортання (Deployment). Мета – розгорнути систему в її робочому оточенні й оцінити її працездатність.
- Керування конфігураціями й змінами (Configuration and Change Management). Визначення елементів, що підлягають зберіганню й правил побудови з них погоджених конфігурацій, підтримка цілісності поточного стану системи, перевірка погодженості внесених змін.
- Керування проектом (Project Management). Включає планування, керування персоналом, забезпечення зв’язків з іншими зацікавленими особами, керування ризиками, відстеження поточного стану проекту.
- Керування середовищем проекту (Environment). Налаштування процесу під конкретний проект, вибір і зміна технологій й інструментів, використовуваних у проекті.

Модель Extreme Programming

Екстремальне програмування (XP)– це приклад так званого методу «живої» розробки (Agile Development Method). У групу «живих» методів входять, крім екстремального програмування, ще й інші методи – SCRUM, DSDM (Dynamic System Development Method), FDD (Feature Driven Development).

Модель життєвого циклу XP є ітераційно-інкрементною моделлю швидкого створення (і модифікації) протопопів продукту, що задовольняють черговій вимозі (user story).

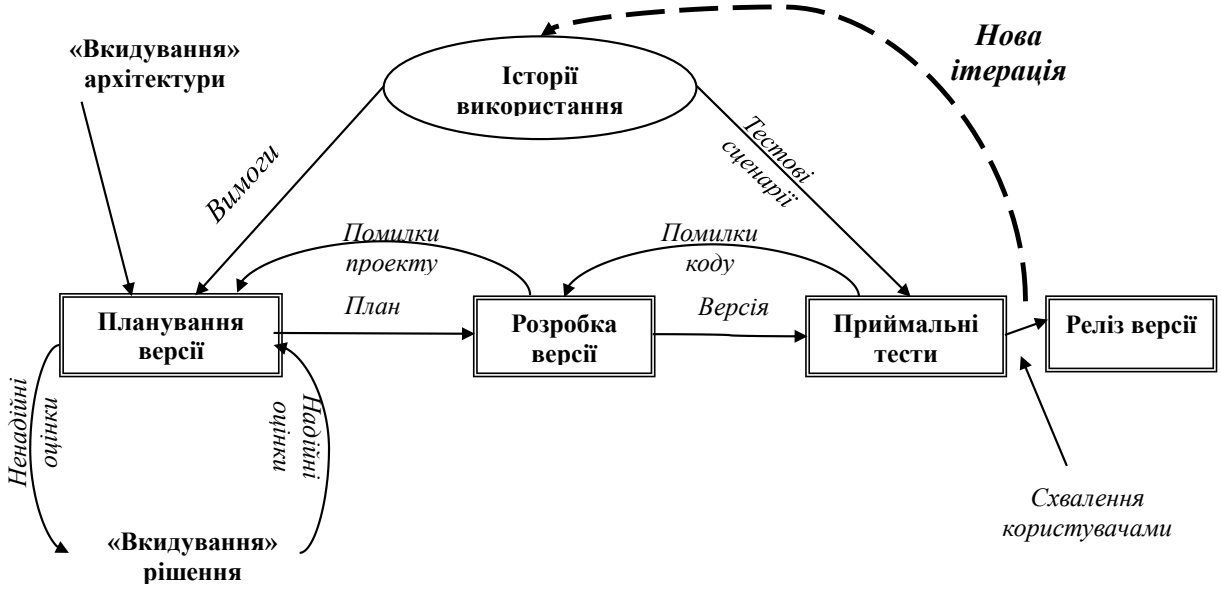


Рис. 3.11. Схема моделі XP.

Основними фазами моделі можна вважати:

- «Вкидування» архітектури – початковий етап проекту, на якому створюється бачення продукту, приймаються основні рішення по архітектурі й застосовуваних технологіях. Результатом початкового етапу є метафора (metaphor) системи, що у досить простому й зрозумілому команді виді повинна описувати основний механізм роботи системи.

- Історії використання (User Story) – етап збору вимог, записуваних на спеціальних картках у вигляді сценаріїв виконання окремих функцій. Історії використання є вимогами для планування чергової версії й одночасного розроблення приймальних тестів (Acceptance tests) для її перевірки.
- Планування версії (релізу). Проводиться на зборах за участю замовника шляхом вибору User Stories, які ввійдуть у наступну версію. Одночасно приймаються рішення, пов’язані з реалізацією версії. Ціль планування – одержання оцінок того, що і як можна зробити за 1-3 тижні створення наступної версії продукту.
- Розроблення проводиться відповідно до плану й включає тільки ті функції, які були відібрані на етапі планування.
- Тестування проводиться за участю замовника, що бере участь у складанні тестів.
- Випуск релізу – розроблена версія передається замовникові для використання або бета-тестування.

Після завершення циклу робиться перехід на наступну ітерацію розробки .

Особливості моделі життєвого циклу XP визначаються принципами цього методу. Насамперед, це принципи «живої» розробки ПЗ, зафіксовані в маніфесті «живої» розробки:

- Люди їхнє спілкування більше важливі, чим процеси й інструменти
- Працююча програма більше важлива, чим вичерпна документація.
- Співробітництво із замовником більш важливо, ніж обговорення деталей контракту
- Відпрацьовування змін більше важливе, чим проходження планів.

Крім того, в XP є кілька правил (технік), що характеризують особливості моделі:

- Живе планування (planning game) – якнайшвидше визначити обсяг робіт, які потрібно виконати до наступної версії ПЗ. Рішення приймається на основі, у першу чергу, бізнесів-пріоритетів замовника й, в-другу, технічних оцінок. Плани змінюються, як тільки вони починають розходиться з дійсністю або побажаннями замовника.
- Часта зміна версій (small releases) – перша працююча версія повинна з'явитися якнайшвидше, і відразу наступні версії підготовлюються через досить короткі проміжки часу.
- Прості проектні рішення (simple design) – у кожен момент часу система повинна бути сконструйована так просто, наскільки це можливо. Нові функції додаються тільки після чіткої вимоги. Вся зайва складність видаляється, як тільки виявляється.
- Розробка на основі тестування (test-driven development) – спочатку пишуться тести, потім реалізуються модулі так, щоб тести спрацьовували. Замовники заздалегідь пишуть тести, що демонструють основні можливості системи, щоб можна було побачити, що система дійсно зробила.
- Постійна переробка (refactoring) – система переписується, змінюється для усунення зайвої складності, збільшення зрозумілості коду, підвищення його гнучкості. При цьому перевага віддається більше елегантним і гнучким рішенням, у порівнянні із такими, що просто дають потрібний результат.
- Програмування парами (pair programming) – весь код пишеться двома програмістами на одному комп'ютері, що підвищує його якість (відсутність помилок, зрозумілість, читабельність,…), також відбувається обмін досвідом і навчання фахівців.

- Постійна інтеграція (continuous integration) – система збирається й проходить інтеграційне тестування якнайчастіше, по кілька разів у день, щораз, коли пари програмістів закінчують реалізацію чергової функції.
- 40-годинний робочий тиждень – понаднормова робота розглядається як ознака великих проблем у проекті. Не допускається понаднормова робота 2 тижні підряд. Це виснажує програмістів і робить їхню роботу значно менш продуктивної.

Контрольні питання

1. Що таке CASE-засіб? Що є типовими складовими CASE-системи?
2. Як умовно поділяють промислові методології створення програмних систем?
3. Які основні фази MSF-моделі?
4. Яка послідовність виконання фаз в MSF-моделі?
5. Що таке ключові точки MSF-моделі?
6. Порівняйте каскадну, спіральну й MSF-моделі ЖЦ ПЗ.
7. Які основні фази RUP-моделі ЖЦ ПЗ?
8. Які процеси використовуються в RUP-моделі?
9. Як визначається ітерація й інкремент у RUP-моделі?
10. З допомогою чого виконується моделювання програмної системи в RUP-моделі?
11. Що таке концепція системи?
12. Які методології входять в групу гнучких («живих») методологій розроблення ПЗ?
13. Які основні фази екстремального програмування?
14. Що таке метафора системи ?
15. Як визначається ітерація й інкремент у XP-моделі?
16. У чому зміст практики planning game?
17. Навіщо необхідний рефакторинг у XP-моделі?

18. Обґрунтуйте ефективність практики pair programming?

19. Що таке user story в XP-методології?

20. Який зміст маніфесту «живої» розробки ПЗ?

21. Проаналізуйте переваги й недоліки «важких» і «гнучких» методологій розроблення ПЗ.

22. Як термін «методологія» пов’язаний з поняттям моделі життєвого циклу ПЗ?

23. Поясніть походження назви Agile-методології.

24. До яких методологій належать методології компанії Microsoft і Rational? Що є спільного у цих методологіях?

25. Які CASE-системи підтримують які методології розроблення ПЗ?

РОЗДІЛ 4. ГРУПОВА ДИНАМІКА ТА КОМУНІКАЦІЇ

У розділі наведено характеристику груп в системі суспільної взаємодії при підготовці програмного забезпечення. Розглянуто комунікативні зв'язки, механізми управління групою, стилі лідерства і керівництва, види конфліктів, моделі поведінки та механізми їх реалізації, засоби підвищення ефективності групової роботи. Описано види і форми ділового спілкування, засоби комунікацій та групової роботи в процесі створення і реалізації програмного забезпечення.

ТЕМА 4.1. ОСНОВНІ ПОНЯТТЯ ГРУПОВОЇ ДИНАМІКИ

Колективна робота в галузі ІТ

Великі програмні проекти, як правило, виконуються групою розробників. На кожному етапі розроблення програмного забезпечення (аналіз вимог, проектування, кодування, збірка, тестування, впровадження/супровід) залучаються групи розробників різних спеціальностей, які об'єднані спільними завданнями. Відносини між людьми в групі виявляються через ***комунікації,*** тобто в процесі обміну інформацією. ***Групова динаміка*** - це процес взаємодії членів групи на основі взаємовпливу з метою задоволення особистих і суспільних інтересів.

Майбутній інженер-програміст повинен мати навики колективної роботи над створенням програмного продукту, а також вміти налагоджувати особисті стосунки з членами колективу для більш продуктивної роботи.

Працівники відділів кадрів чи керівники підприємств (фірм), які проводять співбесіди з потенційними працівниками, звертають велику увагу не лише на необхідний рівень кваліфікації, але і на особисті здібності людини. Велике значення має вміння і готовність навчатися, наявність навиків командної роботи, вміння співпрацювати з іншими членами команди, налагоджувати позитивні стосунки з колегами. Менеджери по роботі з

кадрами чи проект-менеджери використовують розроблені і перевірені набори психологічних тестів. Розуміння психології допомагає задіяти працівників з максимальним ефектом на користь проекту. Людей тестують за багатьма критеріями: наприклад, за рівнем прояву емоцій – інтроверт чи екстраверт, за типом темпераменту – сангвінік, флегматик, меланхолік чи холерик, за притаманними командними ролями, за навичками роботи в команді та ін. Для ефективної роботи команди важливим є мікроклімат в колективі, який підвищує продуктивність праці та рівень задоволення кожного від участі в спільній роботі.

Поняття групи

Психологія і поведінка кожної окремої людини як особистості істотно залежать від соціального середовища. ***Соціальне середовище*** - це суспільство, в якому люди об'єднані у численні, різноманітні більш-менш стійкі формування і об'єднання, які прийнято вважати групами. Саме в групах відбувається спілкування і взаємодія людей.

Група - це обмежена у просторі сукупність людей, що виділяється із суспільного цілого на основі певних ознак (характеру виконуваної діяльності, соціальної чи класової належності, структури, композиції, рівня розвитку тощо).

Людина – істота суспільна. Все життя, від народження до смерті, кожен з нас належить до безлічі груп. До їх переліку належать сім'я, студентська група, групи наших друзів і знайомих, спортивна команда, вболівальники певної команди, гуртки, суспільні організації, групи співробітників, жителі міста, області, країни тощо.

Усі перераховані групи - це ***реальні групи,*** тобто утворення, що реально існують, в яких люди об'єднані якоюсь спільною ознакою, різновидом спільної

діяльності, які перебувають в ідентичних обставинах і певним чином усвідомлюють свою належність до цього утворення.

В психології, статистиці, соціології вживається поняття ***умовної групи.*** Це довільні об'єднання людей за якоюсь спільною ознакою, необхідною в даній системі аналізу, які мають близькі параметри за певними критеріями. До умовних груп належить, наприклад, група людей, які мають однаковий рівень освіти, або які потребують житла, або які хворіють на однакову хворобу.

Поняття групової динаміки

Незважаючи на величезну різноманітність груп, в них діють однакові базові закони, що називаються ***груповою динамікою***. Не обов’язково зосереджуватися на вивченні якогось одного типу групи, оскільки, навчившись розпізнавати групову динаміку в дії, можна застосувати свої знання до будь-яких груп, до яких ми належимо. Поняття групової динаміки об'єднує дуже широке коло питань: поділ влади всередині групи, способи комунікації, ролі, виконувані членами групи, ступінь лояльності її учасників та ін.

Будь-яка мала група націлена на вирішення цілком конкретних завдань. Це може бути розроблення складного програмного забезпечення, перемога в спортивному змаганні, спільна підготовка до іспиту, виховання дітей чи інша мета діяльності. Спільний зміст діяльності породжує спільні ***психологічні характеристики*** групи, до яких належать такі групові утворення як групові інтереси, групові потреби, групові норми, групові цінності, групова думка та групові цілі.

Для індивіда, який входить в групу, усвідомлення належності до неї відбувається перш за все через прийняття цих характеристик, тобто через усвідомлення факту деякої психічної єдності з іншими членами даної соціальної групи, що й дає йому змогу ідентифікуватися з групою.

Універсальним принципом психічного формування спільноти є відмінність для індивідів, які входять в групу, деякого утворення «ми» від іншого утворення «вони».

Діловий та соціальний аспекти групи

Люди об'єднуються в групу заради спільної справи. ***Діловий аспект*** описує відношення членів групи до завдання, яке їм необхідно виконати. Але група складається з окремих індивідуумів, тому необхідно враховувати і ***соціальний аспект*** – емоційні стосунки між членами групи, їх ставлення до групи і свого перебування в ній. Перший аспект впливає на продуктивність групи, а другий впливає на її згуртованість.

Обидва аспекти тісно переплетені між собою. Недостатньо згуртовану групу важко націлити на розв'язання завдання чи виробництво якісного продукту. І навпаки, якщо група успішно працює і досягає високих результатів спільної праці, то між її членами переважно налагоджуються гарні стосунки і вони горді з того, що належать до даної групи. Прикладом може бути спортивна команда, яка отримала перемогу на відповідальних змаганнях. Постає питання: команда перемогла, бо вона добре згуртована, чи перемога згуртувала команду?

Аналізуючи будь-яку групу, необхідно враховувати обидва вказані аспекти. Однак для конкретної групи один аспект може виявитись більш суттєвим, ніж інший.

Діловий аспект є визначальним для футбольної команди, колективу кафедри університету чи відділу тестувальників в ІТ-фірмі. Наприклад, головною метою футболістів є перемога в матчі. Якщо в команді немає згуртованості, їй важко буде досягти мети, але рівень гри визначається не тим, наскільки добре члени команди ладнають один з одним, а їх професіоналізмом.

Соціальний аспект є основним для гуртка авіамоделювання, групи людей, які займаються йогою, пенсіонерів, які в парку грають в доміно та ін. Перед ними не стоїть завдання закінчити роботу до певного терміну, для них набагато важливіша можливість побути в колі людей, які поділяють їх інтереси.

Новостворені групи часто недооцінюють важливість соціальних моментів, прагнуть якнайшвидше розпочати виконання завдання. Необхідно дати можливість членам групи ближче познайомитися один з одним і зав'язати особисті стосунки, перш ніж вони візьмуться за роботу. У протилежному випадку група працюватиме з меншою ефективністю і продуктивністю. На щастя, існує безліч прийомів і завдань, використовуючи які можна послабити напругу перших днів. Неформальна взаємодія членів групи в рамках якогось завдання, не пов'язаного з формальними цілями групи, допомагає налагодити соціальні контакти, що створюють фундамент групової згуртованості.

При розподілі завдань лідеру групи нерідко доводиться вирішувати, доручити виконання завдання всій групі чи одній людині. І тут важливими чинниками є продуктивність і згуртованість групи. Група може виявитися продуктивнішою за рахунок більших ресурсів; але через складності, пов’язані з соціальним аспектом, може бути доцільнішим доручити завдання одній людині.

Завдання, які постають перед групою поділяються на три основні типи:

– виробничі завдання – це завдання, результатом виконання яких є конкретний продукт: програмна система, автомобіль, тощо;

– дискусійні завдання – результати яких не обов’язково включають паперовий звіт;

– завдання, що вимагають вирішення проблеми – результатом таких завдань є не матеріальний продукт, а пропозиції щодо вирішення існуючої проблеми у формі звіту.

Формальні і неформальні групи

Формальна (офіційна) група - реальна або умовна соціальна спільнота, яка має юридично фіксований статус, члени якої в умовах суспільного розподілу праці об'єднані єдиною соціально-значущою діяльністю. Формальна група завжди має певну нормативно закріплену структуру, призначене або вибране управління, нормативно закріплені права і обв'язки її членів. Прикладами формальних груп можуть бути студентська група, різноманітні комісії, групи референтів, консультантів і т.п.

Неформальна (неофіційна) група - реальна соціальна сукупність людей, яка не має юридично фіксованого статусу, добровільно об'єднана на основі як професійних, так і непрофесійну інтересів, дружби, симпатій або на основі прагматичної вигоди. Неформальні групи можуть виступати як ізольовані спільноти або утворюватись в середині офіційних груп. Прикладами є неформальні молодіжні об'єднання, дружні компанії.

Мету офіційних груп формують відповідні установи, а мета неофіційних груп звичайно виникає на основі особистих інтересів їх учасників. Вони можуть співпадати або відрізнятися від мети офіційних організацій.

Кожна організація, установа, підприємство складається з підрозділів. Підрозділ (трудовий колектив) - мала група, що має перелік функціональних обов'язків (завдань) і є елементом організаційної структури підприємства. Ефективність діяльності колективів залежить від розміру і складу робочих груп, від групових норм, згуртованості людей, ступеня конфліктності, статусу і функціональних ролей членів груп. Існують два види колективів: формальні і неформальні.

Формальні колективи створюються керівництвом на певний термін, тимчасово чи постійно, з метою виконання певного офіційного завдання. Це підрозділи, які знаходяться в рамках ієрархічної структури підприємства чи організації, можуть бути і міжфункціональні, необхідні для корекції діяльності підрозділів, спільного пошуку важливих рішень.

До формальних колективів відносять:

- керівні групи - топ-менеджери та їхні команди;

- робочі групи - відділи розробників ПЗ, тестувальників і т.п.;

- цільові групи або комітети – групи, які мають право на групове прийняття рішень і створені для реалізації певних завдань чи проектів. Це ради, комісії, проектні команди тощо. Їх виникнення пов'язане з появою в організації певної проблеми, яка не може бути ефективно вирішена у межах існуючої організаційної структури. Цільові групи усувають формальні відомчі перешкоди, що існують між різними підрозділами організації і перешкоджають координуванню їхньої діяльності. До складу таких груп вводять осіб, які мають великий досвід роботи у відповідній галузі. Це сприяє концентрації інтелектуальних зусиль і швидкому вирішенню проблеми. На період роботи в цільовій групі цих осіб, як правило, звільняють від основної роботи, а після вирішення проблеми вони повертаються до своїх постійних обов'язків.

Неформальні колективи - спонтанно виникають із співробітників формальних колективів, які знаходять в процесі службового спілкування точки зіткнення і об’єднуються для досягнення своєї власної мети, далекої від офіційної.

У будь-якій групі можна виділити як формальний, так і неформальний рівні функціонування.

Контрольні питання

1. Що таке соціальне середовище?
2. Що таке група?
3. Які групи називаються реальними?
4. Які групи називаються умовними?
5. Назвіть групи, до яких ви належите? Які з них реальні, а які умовні?

6. Які ви знаєте психологічні характеристики групи?
7. У чому полягає соціальний і діловий аспекти групи? Назвіть приклади.
8. На які типи поділяються завдання, які стоять перед групою?
9. У чому пролягає різниця між формальними і неформальними групами? До яких формальних і неформальних груп ви належите?
10. Як, на вашу думку, мікроклімат в колективі впливає на результати командної роботи?
11. Які типи формальних колективів ви знаєте?
12. Що таке цільові групи?

ТЕМА 4.2. ОСНОВНІ ХАРАКТЕРИСТИКИ ГРУПИ

До елементарних параметрів будь-якої групи належать розмір, композиція групи (або її склад), структура групи, групові процеси, групові норми і цінності, система санкцій. Розглянемо детальніше кожну з цих характеристик.

Розмір групи

Часто перед дослідниками виникає запитання: «Скільки осіб становить групу?» Питання в тому, зі скількох осіб «починається» група - з двох чи трьох? Цікаво, що в сімейному консультуванні теж іноді виникає питання, скільки людей утворює сім'ю. Чи можна вважати сім'єю подружню пару, або ж в сім'ї обов'язково повинні бути діти? Деякі психологи вважають, що дві людини ще не група.

Що роблять люди в групі? Вони спілкуються один з одним, координують зусилля, притираються один до одного, дотримуються групових цілей чи їх ігнорують, розподіляють ресурси і владу, разом працюють, змінюючи свою поведінку під впливом один одного. У такому випадку подружня пара – теж група.

Особлива мова, якою спілкується група, називається сленгом або жаргоном. У різних професійних групах прийняті свої спеціальні слова і вирази, які зрозумілі тільки членам цих груп і значно полегшують комунікацію всередині групи. Іноді професійні терміни переходять у повсякденну мову, набуваючи при цьому іншого сенсу. Жаргон, полегшуючи внутрішньогрупову комунікацію, може ускладнювати спілкування між групами.

У груповій динаміці існує ряд термінів, що стосуються розміру групи. Дві людини утворюють діаду. Тріада - це група з трьох людей. Групу, в якій не більше двадцяти учасників, прийнято називати малою групою, хоча очевидно, що група з двадцяти чоловік буде мати іншу динаміку, ніж група з трьох-чотирьох.

Часто виникає питання про «ідеальний розмір» групи. Однозначної відповіді на це питання немає. Все залежить від того, яке завдання виконує група, від індивідуальних особливостей її членів, від характеру ситуації і т. п.

Якщо перед групою стоїть завдання обговорити або вирішити проблему, то краще за все з цим завданням впорається група з чотирьох-шести чоловік. У разі виробничих завдань розмір групи залежить від масштабності завдання: чим більш масштабним є завдання, тим більше людей необхідно для його виконання. До речі, коли людей просять розподілитися на групи для вирішення якогось завдання, вони стихійно об'єднуються в групи саме по четверо-шестеро чоловік.

Групові процеси - це процеси групової динаміки, що відображають весь цикл життєдіяльності групи (утворення, функціонування, розвиток, стагнацію/розвиток і розпад групи) та її етапи (керівництво та лідерство, прийняття групових рішень, вироблення групових норм, формування структури групи). Це всі ті процеси, які фіксують і забезпечують психологічні зміни, що відбуваються в групі в процесі її існування.

Місце індивіда в групі

Іншою вагомою характеристикою групи є дослідженні місця індивіда в групі, тобто визначення його статусу і ролі.

Статус (з лат. *status* - положення, стан) - місце суб'єкта в системі міжособистісних відносин, що визначають його права, обов'язки і привілеї. В різних групах індивід може мати різний статус.

Роль (з франц. *role* - роль) - соціальна функція особистості; спосіб поведінки людей відповідно до прийнятих норм, залежно від їх статусу чи позиції в суспільстві, в системі міжособистісних відносин. Роль - динамічний аспект статусу, що розкривається через перелік тих реальних функцій, які задані особистості групою, змістом групової діяльності.

Якщо взяти до прикладу таку групу, як сім'я, то можна показати взаємозв'язок між статусом (позицією) і роллю. В сім'ї різні статусні позиції існують для кожного з її членів: є статус матері, батька, старшої доньки, молодшого сина і т.д. Якщо описати набір функцій, які визначені групою для кожної позиції, то отримаємо характеристику кожної з перерахованих ролей. Не можна визначати роль, як щось незмінне: її динамізм полягає у тому, що при збереженні статусу, набір функцій, які йому відповідають, може сильно коливатись в різних однотипних групах, а головне під час розвитку як самої групи, так і більш широкої соціальної структури, до якої вона належить.

Сприйняття ролі - це ваше розуміння того, як ви повинні поводитися. ***Рольове очікування*** - це те, якої поведінки чекають від вас інші.

Тут підходить аналогія з роллю, яку виконує актор в п'єсі чи фільмі. В даному випадку сприйняття ролі - це ваша інтерпретація ролі. Рольові очікування режисера -- це його розуміння того, як ви повинні грати вашу роль. Добре, якщо перше і друге збігаються.

Входячи в нову роль або вступаючи в нову групу, ви спочатку не знаєте, як себе поводити, як співвідноситься ваша роль з ролями інших учасників. Через деякий час, після декількох репетицій, члени групи (або акторський

склад) починають розуміти, як пов'язані між собою виконувані ними ролі, і їх поведінка стає більш передбачуваною і стабільною. Таким чином, відбувається узгодження ролей, або, інакше кажучи, ролі стають ***комплементарними*** (взаємнодоповнювальними). Прикладом можуть бути очікування між студентами і викладачем або між членами сім'ї. Якщо з якихось причин один з членів групи веде себе не так, як наказано роллю, це може створювати проблеми в групі.

Уявімо собі наступну ситуацію: актори на сцені грають якусь п'єсу, а виконавець однієї з ролей раптом починає виголошувати монолог з іншої п'єси. Очевидно, що його поведінка викличе загальну розгубленість і спектакль, швидше за все, буде зірвано. Розгубленість виникне і тоді, коли посередині спектаклю хтось із акторів забуде свої слова. Це може вибити з колії всіх акторів. Тому актори часто завчають не лише свої, але і чужі репліки, щоб у разі потреби зуміти «підказати» партнеру правильну поведінку (забуті слова).

Іншою характеристикою є ***ступінь ясності ролі***. Грати роль у п'єсі в якомусь сенсі просто - адже ваші слова і рухи визначені заздалегідь іншими людьми. Інша справа - перші дні на новій роботі. Зрозуміло, ви знаєте, з якої метою вас прийняли, і, можливо, вам навіть вдалося пройти попереднє навчання і вам пояснили, що і як треба робити. І все одно ви зіштовхнетеся з деякою невизначеністю. У перший день роботи обов'язково трапиться щось непередбачуване, і ви не будете знати, як повестися в даній ситуації. Поступово, у міру набуття досвіду, ви будете почувати себе у своїй ролі все більш впевнено. Нерідко людина знаходить свою нішу в групі завдяки поведінці інших. ***Рольова невизначеність*** служить однією з причин того, чому багато людей уникають вступати в нові групи. Ніхто не знає, що сказати і що зробити, і всі зазнають незручностей. Багато груп розпадаються, ледве сформувавшись, саме тому, що люди не розуміють, як їм себе поводити.

Розглянемо ще одну дихотомію: ролі, що досягаються і приписані або призначені ролі. В перших є елемент вибору: ми здобуваємо освіту, щоб стати дипломованими спеціалістами, ми працюємо, сподіваючись зробити кар’єру. Приписані ролі - це рольові очікування до нас, очікування, що ґрунтуються на тому, ким ми є, а не на тому, що ми робимо. Стать, вік - ось лише деякі характеристики, на основі яких можна нав'язати людині ту чи іншу роль. Ці характеристики непідвладні нам, однак навколишні люди часто очікують від нас певної поведінки, виходячи з існуючих стереотипів.

Рольове напруга і рольовий конфлікт

Кожен з нас виконує одночасно декілька ролей. Ви граєте певну роль в кожній групі, в яку ви входите. Якщо поведінкові вимоги, пропоновані різними ролями, ясно окреслені, не вступають в суперечність один з одним, то людині неважко суміщати різні ролі. На жаль, в житті не завжди складається так, як би хотілося.

Існують стереотипи. Наприклад, очікуваний рівень інтелекту блондинки та худорлявого хлопця в окулярах. Або бабуся-спортсменка…

Рольова напруженість виникає тоді, коли до нас пред’являється занадто багато вимог і ми не в змозі виконати все те, що повинно бути виконано в рамках виконуваних нами ролей. Це не обов'язково означає, що прийняті людиною ролі несумісні одна з одною, просто їх надто багато. Рольова напруга може виникати і в тому випадку, коли людина змушена грати ролі, занадто для неї складні.

Важливим компонентом характеристики становища індивіда в групі є система «групових очікувань». Цей термін означає, що кожен член групи не просто виконує в ній свої функції, але й обов'язково сприймається, оцінюється іншими. Зокрема, це стосується того, що від кожної позиції, від кожного статусу, а також від кожної ролі очікується виконання цих функцій. Група

через систему очікуваних зразків поведінки, що відповідають кожній ролі, певним чином контролює діяльність своїх членів. В ряді випадків можуть виникати розбіжності між очікуваннями, які має група до певного свого члена, і його реальною поведінкою, реальним способом виконання ним своєї ролі. Для того, щоб ця система очікувань була якось визначена, в групі існує ще два надзвичайно важливих утворення: групові норми і групові санкції.

Групові норми

Норми регламентують, «як треба і як не треба себе вести». Норми є у будь-якої групи. Якщо вони прописані на папері, то це вже не норми, а правила. З правилами простіше: про них можна дізнатися. Припустимо, вас запросили на офіційний обід. Ви знаєте, що, сівши за стіл, ви побачите перед собою чотири виделки, три ложки і п'ять келихів. Щоб не схибити за обідом, ви можете заглянути в довідник з етикету і дізнатися коли, де і як користуватися цим начинням.

Довідатися про норми складніше. Припустимо, ви влаштувалися на нову роботу. Там не потрібно носити уніформу. Що ви одягнете в перший день - строгий костюм чи джинси? Як вам вести себе з новими колегами? Як ви дізнаєтеся, чи прийнято тут обговорювати особисті справи чи всі обмежуються суто діловим спілкуванням? Вам залишається тільки підлаштовуватись під оточуючих і сподіватися, що ви не помилитеся. Вступаючи в нову групу, намагайтеся триматися якомога більш нейтрально і спостерігайте за поведінкою інших - у ньому міститься немало підказок. Не хвилюйтеся. Якщо ви порушите якусь норму, група дасть вам знати про це. Правду кажучи, саме таким чином ми найчастіше дізнаємося про існування і зміст тих чи інших норм.

Діючі в більшості груп норми можна розділити на чотири класи:

– ділові норми підказують членам групи, як їм працювати, скільки зусиль докладати, до яких результатів прагнути і як спілкуватися один з одним в робочий час;

– норми, що стосуються зовнішнього вигляду, підказують, як одягатися, щоб у вас побачили члена групи;

– неформальні соціальні норми управляють поведінкою членів групи в неробочих ситуаціях, наприклад: диктують з ким пити каву під час перерв;

– норми, що регламентують розподіл ресурсів.

Норми - це механізм контролю, без якого неможливі виживання і процвітання групи. Норми впорядковують соціальне життя, роблять його більш передбачуваним. Вони знижують ступінь невизначеності в незвичайній, непередбачуваній ситуації. Якщо ви бажаєте бути членом групи, ви повинні дотримуватися її норм. Помиляється той, хто думає, що його поведінка не контролюється жодними нормами. Спробуйте спуститися з пішохідного мосту по лівій стороні сходів, спробуйте подати ліву руку при вітанні, спробуйте поговорити з малознайомою людиною, стоячи впритул до неї. Слово «норма» лежить в основі слова «ненормальний». Ми називаємо ненормальними тих, хто не хоче або не може дотримуватись прийнятих у суспільстві норм.

Норми групи тісно зв'язані із цінностями, так як будь-які правила можуть бути сформульовані тільки на основі прийняття або відторгнення певних соціально значущих явищ. Цінності кожної групи утворюються на основі вироблення певного ставлення до соціальних явищ, продиктованого місцем даної групи в системі суспільних відносин, її досвідом в організації певної діяльності. Групові норми покращують стабільність групи, вони не тільки впливають на членів групи, а й становлять основу соціального контролю.

Соціальний контроль (з франц. controle - перевірка) - система способів впливу суспільства і соціальних груп на особистість з метою регуляції її поведінки і приведення її до відповідності із загально прийнятими в даній спільноті нормами. Вплив соціального контролю найбільше відчувають на собі

індивіди, чия поведінка може бути охарактеризована як така, що відхиляється від групових норм. Залежно від виду та інтенсивності цих відхилень, група застосовує до індивіда ті чи інші санкції. Характер цих санкцій, адекватність їх використання в тій чи іншій ситуації, їх диференційованість визначаються рівнем соціально-психологічного розвитку даної конкретної групи. Будь-яке порушення групових норм сприймається суспільством як загроза його існуванню і призводить до негайного покарання.

Характер цих санкцій залежить від рівня розвитку групи. Санкції можуть бути двох типів - позитивні і негативні. Поведінка членів групи залежить від системи очікувань щодо виконання групових норм. Ці очікування - різновид санкцій, що також впорядковують систему взаємин і взаємодії в групі. На відміну від офіційних регуляторів поведінки ці санкції не завжди мають усвідомлюваний характер. Соціальний контроль, який здійснює група високого рівня розвитку, характеризується гнучкістю і диференційованістю, що сприяє формуванню самоконтролю у членів колективу і дозволяє їм успішно інтегруватися не лише в групі, але й у цілому соціумі.

Контрольні питання

1. Назвіть основні характеристики групи.
2. Що таке діада і тріада?
3. Який на вашу думку «ідеальний розмір» групи?
4. Для чого використовується внутрішньогруповий сленг?
5. У чому різниця між сприйняттям ролі та рольовим очікуванням?
6. Які ролі називаються комплементарними? Назвіть кілька прикладів.
7. Як призначені ролі пов’язані з стереотипами?
8. Від чого залежить поведінка людини в групі?
9. Коли виникає рольова напруга?
10. Що таке групові норми і групові санкції?

ТЕМА 4.3. ФУНКЦІЇ, ЯКІ ВИКОНУЄ ГРУПА В ЖИТТІ ЛЮДИНИ

Вчені виділили низку людських потреб, що можуть бути задоволені в групі. Якщо класифікувати їх, то отримаємо три категорії потреб: фізичне виживання, психологічне виживання та соціальні потреби.

Фізичне виживання

З найдавніших часів об'єднання індивідуумів в групи підвищувало ймовірність їх виживання, а саме: добування їжі, самооборона, продовження роду та турботи про потомство.

Здійснимо подорож у глибину століть, коли людина виживала за рахунок полювання. Можливості мисливця-одинака були дуже обмежені, він міг впоратися лише з дрібною дичиною, тоді як група могла дозволити собі полювати на крупного звіра. З іншого боку, здобиччю треба було ділитися, але з точки зору соціальної біології краще, коли всі члени групи забезпечені нехай невеликою, але достатньою для виживання кількістю їжі, ніж коли у когось одного достатньо їжі і він виживає, а інші помирають з голоду.

Що стосується самооборони, то зрозуміло, що наскільки б не була людина вправна в цьому, вона не може весь час бути напоготові. Людині потрібні відпочинок та сон, під час якого вона стає дуже вразливою. Об'єднавшись у групу, яка складається навіть з двох людей, можна забезпечити собі цілодобовий захист, якщо спати по черзі. Крім того, що могла зробити первісна людина, яка отримала поранення під час полювання. Без сторонньої допомоги їй загрожувала смерть. Якщо жінка помирала під час пологів, у немовляти не залишалося шансів вижити, тоді як у групі турботу про дитину брали на себе інші жінки. Ще одна очевидна причина для об'єднання полягає в тому, що завдання продовження роду вимагає участі двох людей.

Більшість перерахованих вище функціональних причин існування груп актуальне і зараз. Наше життя не можливе без допомоги інших людей. Ми

залежимо від тих, хто вирощує, виробляє і продає їжу, від армії та поліції, які захищають нас, ми піклуємося про хворих та немічних, ми одружуємось і народжуємо дітей. Отже, група дійсно забезпечує людині фізичне виживання.

Психологічне виживання

Розглянемо окремі елементи теорій особистості різних психологів. На думку творця «теорії прив'язаності» британського психоаналітика Боулбі (1907-1990) «у немовлят є вроджена потреба у фізичному контакті з людською істотою». Кожна людина потребує ласки, розради та спілкування. Для дітей величезне значення має зв'язок з дорослими людьми. В якості доказу зв'язку між матір'ю і дитиною він приводив той факт, що вже в чотиримісячному віці дитина починає впізнавати матір і реагує на неї інакше, ніж на інших людей. Маленькій дитині необхідна фізична присутність матері: вона плаче, коли мати виходить з кімнати, а пізніше, почавши ходити, всюди слідує за матір'ю.

Процес розвитку особистості має кілька стадій. На відміну від австрійського психолога Зи́гмунда Фрейда (1856-1939), який вважав, що формування особистості закінчується до шести-семи років, німецький психолог Ерік Еріксон (1902-1994) у своїй теорії особистості виділив стадії розвитку індивідуума і у дорослому житті. На його думку на першому році життя головним завданням є формування довіри. Якщо на цій стадії довіра до світу не сформується, то людина ніколи не зможе довіряти людям. Практика показує, що у дітей, які виросли в чуйному середовищі, виявляється сформоване почуття безпеки, стійкий тип прихильності, або, за термінологією Еріксона, здатність до довіри.

Американський психолог Генрі Мюррей (1893-1988) стверджував, що кожна людина має потребу в афіліації, тобто бажанні бути з людьми, та владі, тобто бажанні ними управляти.

Теорія міжособистісних відносин австрійського та американського філософа і соціолога Альфреда Шюца (1899-1959) «Базова орієнтація» створена для вимірювання трьох потреб: потреби у включенні (кожна людина з одного боку хоче бути серед людей, а з іншого - бажає бути на самоті), потреби у контролі (бажання впливати на людей, бажання керувати людьми, а також бажання підкорятися і бути веденим) та в любові (бажання мати близькі і теплі стосунки з людьми і, як протилежність - прагнення дистанціюватися від людей).

Соціальні потреби

Перша категорія соціальних потреб, що задовольняються групою і членством у групі, включає потреби в інформації. Якщо ми хочемо дізнатися температуру повітря на вулиці, можемо виміряти її термометром, якщо ми хочемо знати розмір кімнати, можемо виміряти довжину стін рулеткою. Якщо ж треба з'ясувати, наскільки складний курс групової динаміки або як пройшла церемонія посвяти в студенти університету, нам доведеться звернутися за інформацією до інших людей. У даному випадку відгуки у соціальних мережах – це інформація, надана іншими людьми.

Якщо задуматися, то за допомогою об'єктивних інструментів на зразок термометра або лінійки ми отримуємо лише невелику частину знань. У переважній більшості випадків джерелом інформації для нас служать інші люди. (Цікавимось думкою інших при виборі одягу, про те, чи варто переглянути фільм, де відпочити і т. п.)

Американський психолог Леон Фестінґер (1919-1959) назвав цей процес соціальним звіренням. Використовуючи процес соціального порівняння, люди оцінюють себе: вони відчувають потребу оцінити власну думку і власні здібності, порівнюють себе з іншими людьми, зазвичай з тими, хто чимось схожий на них.

Деякі психологи вважають, що навіть при визначенні пережитих нами емоцій ми використовуємо інформацію, отриману від інших людей. Згідно теорії емоцій американського соціального психолога Стенлі Шехтера (1922-1997) емоція складається з двох компонент: фізіологічного збудження, що являє собою викид адреналіну, який запускає реакцію на кшталт «бий або біжи», та оцінки цього збудження.

Страх і тривога змушують людей тягнутися один до одного. Розглянемо ще одне дослідження, яке поверне нас до інформаційних потреб. Пацієнтам, які очікують на складну операцію, пропонувалося вибрати собі сусіда по лікарняній палаті. Вони могли поселитися з людиною, яка теж очікує такої самої операції, або з вже прооперованим пацієнтом. Більшість пацієнтів віддали перевагу другому варіанту. Очевидно, пацієнтам хотілося спілкуватися з людиною, що пройшла через операцію, щоб довідатися, чого їм чекати.

Є й інші не менш вагомі соціальні основи для об'єднання людей в групи. Ми об’єднуємося в групи, щоб отримати від людей емоційну підтримку або розраховуючи на їх сприяння. Ми вступаємо в політичні партії, тому що поділяємо проголошені ними переконання і цілі. Ми приєднуємось до групи, тому що ідентифікуємо себе з нею. Членство в групі і ототожнення себе з групою відіграють важливу роль у формуванні самоповаги. Прагнення людей об'єднуватися в групи визначено людською природою, соціальними потребами та є необхідною умовою виживання.

Соціальний обмін

Приймаючи рішення про входження в групу, людина розуміє, що їй доведеться чимось поступитися, відмовитися від звичного способу життя. Натомість вона отримає певні можливості для задоволення власних потреб. Тобто крім офіційної угоди з організацією людина укладає своєрідну

психологічну угоду, що являє собою сукупність очікувань особи щодо її внеску в організацію і того, що організація надаватиме їй взамін. Ця угода не зафіксована на папері та її умови нечіткі. На рис. 4.1 зображено сутність психологічної угоди.

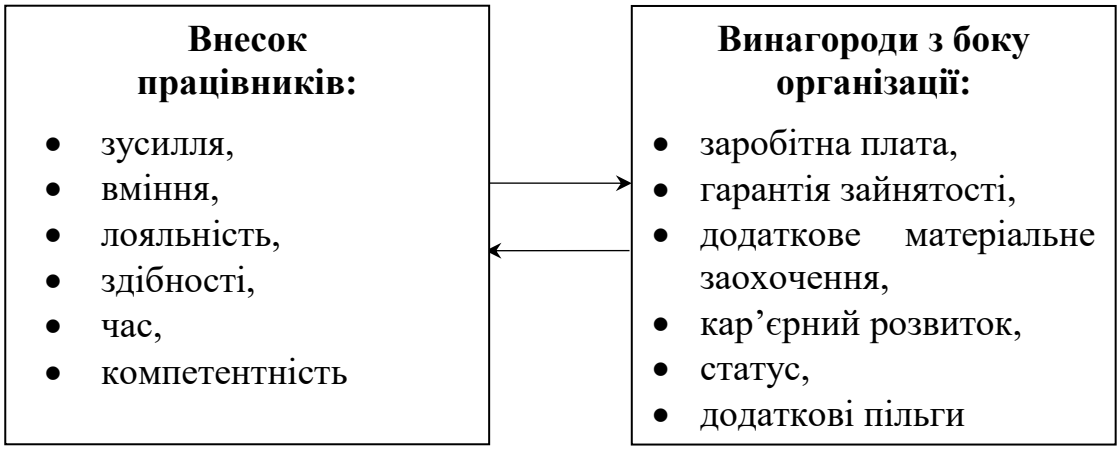


Рис. 4.1. Сутність психологічної угоди

Соціальний обмін має місце тоді, коли людина прораховує, що вона придбає і які збитки понесе, якщо приєднається до групи. В суспільстві прийнято відповідати люб'язністю на люб'язність.

Ізраїльська психолог Една Фоа (народилася у 1937році) виділяє шість типів винагород, які можна отримати в групі: любов, гроші, статус, інформацію, речі та послуги. Ці винагороди можна класифікувати в залежності від ступеня партикулярності і конкретності. Партикулярність означає, що для того, хто отримує винагороду, не байдуже, від кого вона виходить.

Цінність любові, а якщо говорити конкретніше - цінність обіймів і ніжних слів, - визначається тим, від кого ми їх отримуємо. Таким чином, любов - це партикулярна винагорода. На відміну від любові, гроші мають цінність незалежно від того, хто їх платить. Вони є універсальним винагородою.

Конкретність - це відчутність винагороди. Гроші являють собою найвищою мірою конкретну винагороду, тоді як любов і статус - винагороди символічні.

Отримуючи винагороди, людина оцінює їх кількість і якість, порівнюючи зі своїми витратами. До витрат належать час, енергія, матеріальні та особистісні ресурси.

Співвідношення між витратами і винагородою визначає ступінь привабливості групи для індивіда, але не обов'язково визначає, чи залишиться індивід у групі чи покине її. Якщо людина задоволена групою, це ще не означає, що вона залишиться її членом. Так само невдоволення індивіда групою не обов'язково означає, що вона покине її. Напевно, в житті кожної людини бувають періоди, коли вона незадоволена своїм місцем праці. Але людина працює, бо не може знайти іншу роботу.

Згідно теорії Фоа ступінь привабливості групи або відносин для індивіда і ступінь його задоволеності визначаються рівнем порівняння - кількістю поваги і винагород, яке людина вважає для себе достатньою. Ми задоволені, коли отримуємо достатньо, і щасливі, коли отримуємо більше. Рівень вимог і достатній обсяг винагород для кожного індивіда різний.

Питання про продовження членства в групі вирішується, виходячи з рівня порівняння альтернатив. Будучи членом якоїсь групи, індивід знає, що крім цієї групи існує безліч інших, альтернативних, груп. У людини завжди є можливість порвати стосунки, що склалися, звільнитися з роботи або вийти з клубу. Вона так і вчинить, якщо альтернативні варіанти будуть здаватися їй більш привабливими. Суб'єктивний вибір альтернативного варіанту змушує людину покинути групу. Очевидно, що людина, оцінюючи альтернативи, може помилятися.

Наприклад часто виникає ситуація, коли з групи йде людина, яка почувалася цілком щасливою, а той, хто почував себе нещасним, залишається. Припустимо, ви працюєте в якійсь фірмі. Вам подобається ваша робота, у вас

хороші стосунки з колегами, ви отримуєте високу платню, вас все влаштовує. Але вам раптом пропонують роботу, заробітна плата за яку вдвічі вища. Цілком можливо, що ви приймете цю пропозицію. Часто люди залишаються в групі, в якій почувають себе некомфортно, через відсутність альтернатив.

Привабливість групи

Характеристики, які підвищують привабливість інших людей і сприяють тому, щоб ми захотіли зав'язати з ними відносини і сформувати групи – це схожість, географічна близькість та фізична привабливість.

Нас приваблюють ті люди і ті групи, які нам здаються схожими на нас. Ця ***схожість*** може спочатку ґрунтуватися виключно на поверхневих характеристиках. Наприклад, коли люди потрапляють у велику аудиторію, їх комфортніше зайняти місце біля людини, з якою вони знайомі або яку вони вже десь бачили. Два туристи з одного міста, які випадково зустрілися закордоном, можуть стати близькими приятелями на час відпустки, а, повернувшись додому, обмежать або й зовсім припинять спілкування.

Нам подобається бути з тими людьми, чиї схильності і цінності співпадають з нашими, і не подобається бути з тими, хто з нами не згоден. Дуже зручно мати поруч того, хто підтримує нашу точку зору. Це підкріплює наше переконання в тому, що ми праві. До того ж нас привертають ті люди, які «настільки розумні», що розділяють нашу думку.

Існують, однак, і ситуації, в яких ми уникаємо людей, яких вважаємо схожими на нас. Якщо нам здається, що деякі наші характеристики знижують наш статус або плямують нас, ми не хочемо приєднуватися до групи.

Другим фактором, що впливає на привабливість групи для індивіда, є ***географічна близькість***. Люди не випадково селяться в певному районі. Вони вибирають той будинок, який їх влаштовує за ціною (фінансовий схожість), і часто звертають особливу увагу на абсолютно конкретні характеристики

району. Якщо у них є маленькі діти, вони шукають район, де поблизу багато дітей. Якщо це літні люди, вони можуть вибрати район, де тихо і спокійно.

Але навіть, коли близькість носить випадковий характер, наприклад, коли мова йде про місце в аудиторії, про столи в офісі або про квартири у гуртожитку, більш імовірно, що у людини зав'яжуться близькі стосунки саме з сусідами, а не зі студентом, що сидить в іншому кінці аудиторії, з працівниками з іншого поверху або з тими, хто живе в іншому гуртожитку. Можливо, головною причиною сили близькості є те, що ці люди більш доступні для спілкування.

Велика кількість контактів дає нам відчуття близькості з людиною, а це, в свою чергу, може призвести до відчуття комфорту. Якщо ви йдете по безлюдній вулиці близько півночі і почуєте кроки за спиною, то ви з більшою ймовірністю відчуєте себе спокійніше, якщо, повернувшись, побачите однокласника або сусіда, ніж якщо виявите, що до вас прямує незнайомець.

Незважаючи на те, що стандарти ***фізичної привабливості*** в різні часи і в різних культурах різні, існують певні загальноприйняті культурні стандарти того, як люди повинні виглядати. Як показують дослідження, практично від самого народження до людей, що відповідають стандартам привабливості, ставляться краще, ніж до інших. Отже, ми хочемо виглядати привабливо. За інших рівних умов нас будуть притягати привабливі люди.

Ми прагнемо увійти в одну групу з людьми, відносини з якими будуть нам корисні. Розподіл витрат і доходів у групі впливає на нашу задоволеність від членства в цій групі, а також на рішення, залишатися нам у цій групі чи ні.

Контрольні питання

1. Які соціальні потреби людини задовольняються при її вступі в групу?
2. Що робить групу більш привабливою для індивіда?
3. Які види винагород людина може отримати в групі?

4. Що таке партикулярність? Назвіть винагороди, які є для вас партикулярними.

5. Які винагороди є конкретними?

6. Що таке психологічна угода? Чи оформлюється вона юридично?

7. З якими людьми комфортніше об’єднуватися у групи?

8. Чому людині комфортніше спілкуватися з людиною, схожою на себе?

9. Як географічна близькість впливає на привабливість групи для індивіда?

10. Які переваги для членства у групі надає людині її фізична привабливість?

ТЕМА 4.4. КЛАСИФІКАЦІЯ ГРУП

Велика кількість малих груп в суспільстві передбачає їх величезну різноманітність, і тому для дослідницьких цілей необхідна їх класифікація (рис. 4.2). Розглянемо найбільш розповсюджені ***критерії,*** за якими класифікують групи: безпосередність взаємозв'язків, розмір, суспільний статус, значущість для особистості та рівень розвитку.

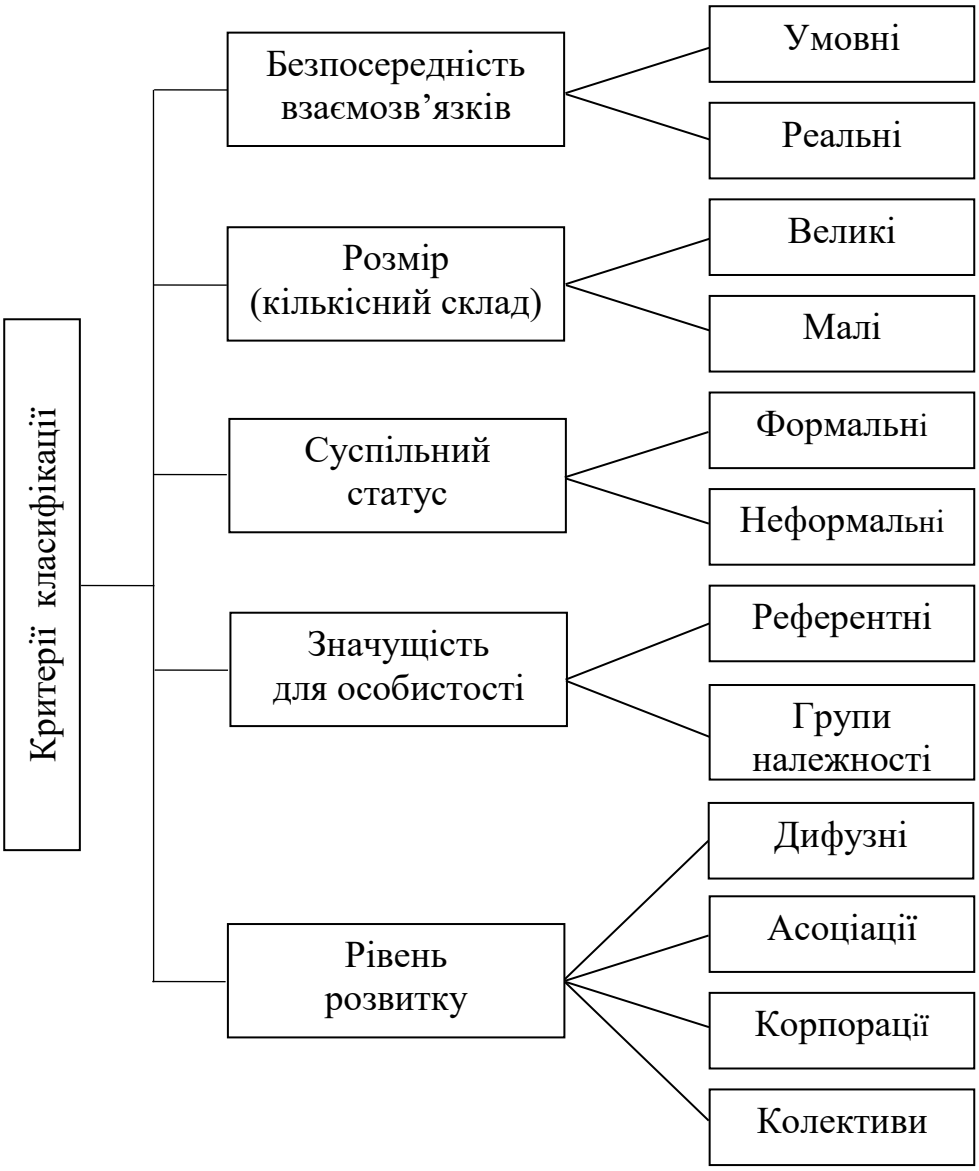


Рис.4.2. Класифікація соціальних груп

За безпосередністю взаємозв'язків групи поділяються на реальні (контактні) і умовні. ***Реальна група*** - обмежена у розмірі спільнота людей, яка існує в спільному просторі і часі і члени якої об'єднані реальними стосунками (наприклад, студентська група, військовий підрозділ, сім'я тощо). Найбільша за розміром реальна група - людство як історична і соціальна спільнота людей, яка об'єднана економічними, політичними та іншими зв'язками. Найменшою реальною групою є діада, тобто група, яка складається з двох індивідів. ***Умовна група*** - об'єднана за певною ознакою (характером діяльності, статтю, віком, рівнем освіти, національністю і. т.п.) спільнота людей, яка є об'єктом дослідження соціальної психології і яка містить суб'єктів, які не мають прямих або непрямих об'єктивних взаємовідносин однин з одним. Люди, з яких складається ця спільнота, можуть не тільки ніколи не зустрічатись, але і не знати нічого один про одного, хоча при цьому вони перебувають в певних, більш-менш однакових, відносинах з іншими членами своїх реальних груп.

Серед реальних груп існують природні групи та лабораторні групи, які фігурують в загальних психологічних дослідженнях. Найбільшої уваги в дослідження приділяють вивченню реальних природних груп.

За розміром виділяють великі та малі. В деяких класифікаціях ще виокремлюють мікрогрупи, до яких відносять діади та тріади. ***Великі групи*** – це держави, нації, народності, партії, класи, інші соціальні спільноти, які виділяються за професійними, економічними, релігійними, культурними, освітніми, віковими, статевими та іншими ознаками. Через ці групи опосередковано здійснюється вплив ідеології суспільства на психологію людей, які до них належать. У великих групах формуються норми поведінки, суспільні і культурні цінності і традиції, суспільна думка і масові рухи, які через малі групи доводяться до свідомості кожного індивіда.

Малі групи – це невеликі об'єднання людей (від двох-трьох до двадцяти-тридцяти осіб), які об’єднані спільною справою і перебувають у прямих стосунках один з одним. У таких групах людина проводить більшу частину

свого життя. Малу групу характеризує психологічна і поведінкова спільність її членів, яка виділяє і відокремлює групу, робить її відносно автономним соціально-психологічним утворенням. Ця спільність може виявлятися за різними характеристиками - від суто зовнішніх (наприклад, територіальна спільність людей як сусідів) до досить глибоких внутрішніх (наприклад, члени однієї сім'ї). Міра психологічної спільності визначає згуртованість групи - одну з основних характеристик рівня її соціально-психологічного розвитку. Структура малої групи зображена на рис. 4.3.

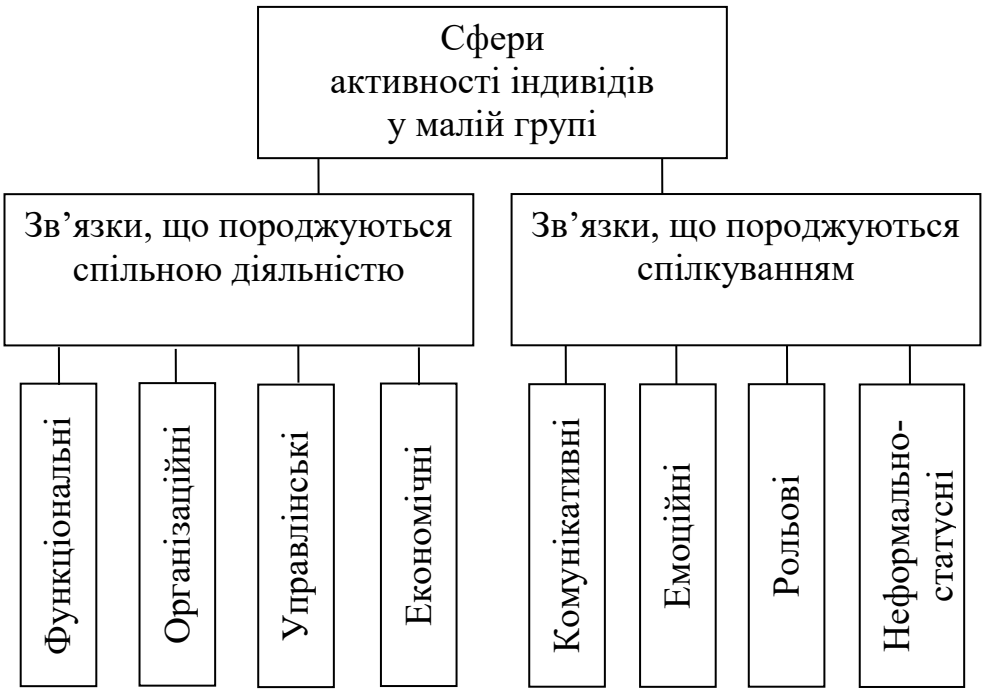


Рис. 4.3. Структура малої групи

Малі групи можуть бути різними за величиною, за характером і структурою відносин між їх членами груп, за індивідуальним складом, цінностями, особливостями норм і правил взаємовідносин, міжособистісними стосунками, цілями і змістом діяльності.

Кількісний склад групи на мові науки називається її розміром, індивідуальний - композицією. За суспільним статусом групи поділяються на формальні і неформальні (див. Тема 4.1).

За значенням для індивіда групи поділяються на референтні і групи належності. Спільнота, до якої людина добровільно себе зараховує, з якою себе порівнює і на норми та цінності якої орієнтується у своїй поведінці та самооцінці називається ***референтною*** групою. У ***нереферентній*** групі психологія і поведінка групи чужі або байдужі для індивіда, а в ***антиреферентній*** - людина зовсім не сприймає, засуджує і відкидає поведінку і психологію інших членів групи. Референтна група виконує дві функції — нормативну й порівняльну. Нормативна функція виявляється в мотивації: референтна група є джерелом норм поведінки, соціальних установок та ціннісних орієнтацій людини. Порівняльна функція полягає в тому, що референтна група стає еталоном, за яким індивід оцінює себе та інших.

Група належності — це така група, до якої людина реально належить.

Крім того, можна класифікувати групи за рівнем розвитку: низького рівня розвитку (дифузні групи, асоціації, корпорації) і високого рівня розвитку (колективи).

Дифузна група - нестійке короткочасне об'єднання людей, що виникає лише на основі особистісно значущої діяльності. У ній міжособистісні стосунки не опосередковує зміст спільної діяльності, оскільки вона відсутня. Склад дифузної групи випадковий, відносини встановлюються на рівні симпатії - антипатії.

Асоціація - добровільне об'єднання людей для досягнення спільної мети. Ролі і статуси в такій групі суворо детерміновані, але від цього не залежить ефективність спільної діяльності.

Корпорація - організована група, яка характеризується замкнутістю, максимальною централізацією і авторитарністю керівництва, що протиставляє себе іншим соціальним спільнотам на основі своїх вузькоіндивідуальних інтересів. Міжособистісні відносини в корпорації опосередковані асоціальними, а часто антисоціальними ціннісними орієнтаціями.

Персоналізація індивіда в корпорації здійснюється шляхом деперсоналізації інших індивідів.

Колектив - група людей, об'єднаних спільними цілями і задачами, яка досягла високого рівня розвитку в процесі соціально цінної спільної діяльності. Для того щоб назвати малу групу колективом, вона повинна відповідати ряду досить високих вимог: бути ефективною, тобто успішно виконувати свої функції, мати високу мораль, гарні людські відносини, створювати для кожного свого члена можливість розвитку як особистості, бути здатною до творчості, тобто як група давати людям більше, ніж може дати сума тієї ж кількості індивідів, які працюють окремо. Останнім часом слово «колектив» часто заміняють словом «команда».

Контрольні питання

1. Назвіть критерії, за якими прийнято класифікувати групи?
2. Що таке реальна група?
3. Що таке умовна група?
4. Чи можуть члени умовної групи ніколи не спілкуватись між собою?
5. На які види групи поділяються за рівнем розвитку?
6. На які види групи поділяються за безпосередністю зв’язків?
7. На які види групи поділяються за суспільним статусом?
8. На які види групи поділяються за значенням для індивіда?
9. До якої групи людина добровільно себе зараховує і знаходить для себе зразки для наслідування?
10. Яка різниця між референтною і антиреферентною групами?
11. До кого типу належить студентська група, в якій ви навчаєтесь?
12. До яких груп ви належите?
13. Назвіть групи низького і високого рівня розвитку.

ТЕМА 4.5. РОЗВИТОК ГРУПИ

Групова динаміка - це розвиток групи, її зміна в часі. Можна виділити наступні стадії групового розвитку: створення групи, поділ на підгрупи («притирання»), конфронтація, співпраця та розпад.

Стадії групової динаміки

Створення групи. Коли ми вперше зустрічаємося в групі, всі перебувають в напрузі, не знають, чого очікувати, хвилюються і намагаються це приховати. Складається враження, що ніхто не хвилюється, тому кожен думає, що нервує лише він. Це абсолютно нормальне відчуття називається первинною напруженістю. Частково ця напруженість пояснюється тим, що на даній стадії рівень довіри ще низький. Люди знають, що вони будуть залежати від відповідальності та компетентності інших членів групи, але не знають, наскільки інші члени групи відповідальні, компетентні і надійні. Багато хто нервує через те, що його у групі завантажать роботою, а інші зможуть ухилятися від завдань. Люди часто ставляться з підозрою до тих, хто намагається захопити владу в групі і веде себе як лідер. Конфлікту на стадії членства ще немає, тому що кожен з усіх сил намагається викликати симпатію у інших членів групи й водночас намагається оцінити інших як потенційних колег. Кожен поводиться м'яко, люб'язно і поверхнево, прагнучи справити враження «хорошого» члена команди. Отже всі видаються не такими, якими вони є насправді. Враження членів групи один про одного базуються на стереотипах. Для того щоб група стала успішним колективом, необхідно змінити перші враження. Якщо групі на ранній стадії розвитку доводиться виконати важливе завдання, якість його виконання буде дуже невисокою. Ще однією важливою причиною неефективності групи на даній стадії розвитку є

те, що кожен член групи боїться вступати в конфлікт і проявити себе з «поганого боку», тому з великим ентузіазмом схвалює будь-які рішення.

Існують способи пройти цю стадію більш гладко. Треба поставити перед групою завдання, яке не має відношення до загальних цілей групи, але при цьому вимагає взаємодії між її членами. Таким чином, члени групи побачать одне одного в справі, їх діяльність буде певним чином структурована, а тим часом всі пізнають один одного краще. Така діяльність відома як «розбивання льоду». Завдання може бути будь-яким, наприклад, вишикуватись за зростом або розбитися на підгрупи з однаковим кольором очей. Такі заняття допомагають учасникам групи зняти напругу, покращити настрій і почати поводитися більш природно.

Якщо ролі не прояснюються досить швидко, люди можуть прийняти рішення вийти з групи. Винятком з проблеми неоднозначності є заздалегідь призначений лідер групи. Ця роль досить зручна, оскільки члени групи не лише знають, як повинен поводитись лідер, але і розуміють, як самі повинні слідувати за ним, щоб бути «хорошим» членом групи.

Взаємодія на ранніх стадіях відбувається скуто, напружено і обережно. На жаль, ця взаємодія формує норми, які будуть керувати поведінкою групи доти, поки хтось їх не опротестує. Будь-яка дія, вчинена в групі вперше, створює прецедент для подальшої поведінки. Якщо на першу зустріч групи всі прийдуть причепурені, логічно припустити, що вони будуть так виглядати і на кожній наступній зустрічі. Якщо більшість учасників запізнилися, можна бути впевненим, що і надалі зустрічі не будуть розпочинатися вчасно - принаймні, доти, доки пунктуальні члени групи не зможуть поставити питання про запізнення.

Поділ на підгрупи. Цю стадію ще називають «притиранням». Люди швидше зближуються з тими, з ким у них є щось спільне. Вони впізнають тих, з ким разом навчалися або зустрічалися в гостях, або ж ці дві людини - єдині дві жінки в кімнаті, повній чоловіків. На цій стадії рівень напруженості в групі

починає спадати, а рівень довіри – навпаки зростати. Створюються коаліції, які можуть почати маневрувати з метою захоплення влади в групі, але відкритого конфлікту ще немає. Продуктивність групи на даній стадії досить низька. Незважаючи на поверхневий спокій, зростає незгода, яка виявиться на наступній стадії розвитку.

Конфронтація - вкрай важлива фаза групового розвитку. Досить ймовірно, що групи, які намагаються перескочити цю стадію, ніколи не зможуть працювати продуктивно. Цю стадію можна уникнути лише в групах, які об'єднуються на дуже короткий термін.

На стадії членства кожен намагався справити на інших добре враження. Вступ в стадію конфронтації в групі є позитивною ознакою. Це ознака налагодження довірливих відносин між членами групи. Ми не вступаємо в конфронтацію, якщо не можемо передбачити реакцію і не вважаємо, що реакція буде адекватною. Тому виникнення конфлікту свідчить про те, що група розвивається нормально. Кожен, кому доводилось проходити у відносинах період першої суперечки мабуть відчув, що якщо залагодити стосунки, зберігши відносини і не порушивши цілісність групи, це додасть групі впевненості в тому, що конфлікт можна вирішити, нікого не образивши. Це також стимулює гігантський стрибок продуктивності групи.

На початковому етапі функціонування групи її члени не бояться виносити на обговорення необдуману пропозицію, адже впевнені, що група її прийме. Усвідомивши, що ідеї можуть зазнати критики, учасники групи починають ретельніше обдумувати свої ідеї. Якість матеріалу зростає, в групі вперше з'являється контроль якості, тому що група буде вибирати краще з кращого. Тепер група впевнена, що може ефективно вирішувати конфлікти, і акцент зміститься з соціальних проблем на первинну причину створення групи.

Розробка норм. Наступна стадія групового розвитку - це стадія розробки норм або індивідуальної диференціації. Це щасливий і продуктивний час для

групи. Напруженість досить низька, навички вирішення конфліктів група вже виробила, довіра перебуває на стабільно високому рівні.

Характеристикою цієї стадії групового розвитку є феномен довіри. В групі встановлюється рівень довіри, достатній для прийняття розумних рішень про поділ праці в групі. Найбільш ефективний розподіл праці передбачає розподіл завдань у відповідності з сильними сторонами і навичками кожного члена групи. Якщо в групове завдання входить фінансовий аналіз, має сенс доручити його людині, що володіє бухгалтерськими навичками і необхідною підготовкою. Люди краще виконують роботу, якщо задіяти їх сильні сторони, і з більшою готовністю беруться за ту роботу, яку, як вони впевнені, їм вдасться виконати якісно. Люди починають ідентифікувати себе з групою і піклуються про загальний успіх групи. На цьому етапі люди покидають групи, як правило, лише за необхідності, наприклад, якщо вони закінчують навчання в університеті, переходять на інше місце роботи, їх сім'ї переїжджають і т.п.

Співпраця. Четверта стадія групового розвитку - стадія співробітництва. Група стала досить зрілим колективом, здатним ефективно виконувати роботи високої якості. Рівень напруженості дуже низький, рівень довіри високий, а всі конфлікти виявлені і ефективно вирішуються. Поділ праці ґрунтується на навичках кожного, навіть роль лідера часом переходить до того, хто при виконанні певного завдання виконує лідерські функції краще, ніж інші.

Розпад групи. Але всі групи, навіть чудово спрацьовані, розпадаються. Важливо знати, яким чином можна зробити розпад групи настільки ж успішним і результативним, як і всі інші стадії. Групи можуть припинити своє існування двома способами: заплановано і не заплановано. Коли людина вступає до університету, вчиться в школі або проходить службу в армії, вона знає, скільки часу триватиме її членство в даній групі. Вона може підготуватися до моменту виходу з групи. Лідерам тимчасових груп важливо обговорювати їх розпад , що дозволить людям подумки підготуватися до цього

моменту. Якщо це можливо, необхідно організувати останню зустріч групи, присвячену обговоренню набутого досвіду та припиненню існування групи.

Найстрашніше, що може статися з групою - це раптовий кінець. Фірма може збанкротувати, футбольну команду можуть залишити кілька ключових гравців, що ті, які залишилися, не зможуть продовжувати виступати. Пара, яка мріяла все життя бути разом, розлучається. Люди, що раптово випали з групи, дезорієнтовані і приголомшені, вони переживають негативні відчуття і шок від раптового розпаду.

Основні процеси соціалізації

Процес індивідуального вибору, вступу, групової належності і відходу з групи ми називаємо соціалізацією. На кожній стадії групового розвитку перед нами постає ряд питань: оцінка, прийняття зобов'язань та рольові переходи. Кожне з цих питань треба розглядати з точки зору індивіда і точки зору групи.

Оцінка - це процес визначення всіх «за» і «проти» вступу в групу. Як тільки людина вирішує, в який виш вступати, навчальний заклад починає оцінювати цю людину. Ідеальна ситуація передбачає, що вибір влаштовує обидві сторони. Але це відбувається не завжди. Найкращим кандидатом на вступ в будь-яку групу є людина, яка в неї вписується.

Під ***прийняттям зобов'язань*** ми розуміємо особисте бажання людини вкладати в групу свої ресурси - час, гроші чи енергію. Ми можемо піти з групи, яка нам подобається, якщо нам пропонують більш привабливий варіант, або залишитися в принизливому становищі, якщо немає іншої альтернативи.

Останнім важливим питанням залишається питання про ***рольовий перехід***. Коли ми в процесі соціалізації переходимо з однієї стадії на іншу, змінюються очікування групи відносно нашої поведінки, а, оскільки набір очікуваних дій ми називаємо роллю, наші ролі теж змінюються. Подібним чином ми очікуємо, що немовля, школяр, студент і батько сімейства будуть

поводити себе по-різному: очікування змінюються в процесі соціалізації. Часто переходи супроводжуються певним ритуалом або церемонією: закінчення школи чи університету супроводжується церемонією випуску, формальні зобов'язання нареченого і нареченої підкріпляються весіллям.

Розвиток особистості в групі

Розвиток особистості в групі містить наступну послідовність стадій: дослідження, вступ до групи, соціалізація, прийняття, відхилення, повторна соціалізація і спогади.

Дослідження (вибір групи). На цій стадії людина ще не належить до жодної групи і вивчає різні можливості. Перед тим як вступити в університет або на роботу, ми повинні з'ясувати якомога більше переваг і недоліків кожної альтернативи. При цьому виборі кожен має свої пріоритети: репутація, фінансовий стан, географічне розміщення, наявність гнучкого розкладу і т.п.

Поки ми оцінюємо університет, роботу або потенційного партнера, вони оцінюють нас. У момент вивчення альтернатив вступу в групу наші зобов'язання перед групою досить низькі. Але групи не хапаються щосили за нашу кандидатуру. Якщо нам пощастить і нам запропонують вступити до кількох навчальних закладів, нам доведеться здійснити вибір з представлених альтернатив. Людина, яка хоче досягнути високого рівня в організації, повинна одягатися і поводитися так, ніби вона вже перебуває на цьому рівні.

Вступ до групи і соціалізація. Якщо оцінка з обох боків виявилась позитивною, ми стаємо новим членом групи і починаємо поводитися відповідно до нової ролі. Розпочинається знаходження свої ніші в групі. Якщо мова йде про нову посаду, ця стадія є випробувальним терміном. На даному етапі дуже важлива двостороння оцінка. Якщо хтось хоче, щоб ви приєдналися до групи, він буде поводитись дуже чемно і всіляко применшувати проблеми групи. Ви не побачите свого майбутнього чоловіка або дружину неохайним

або сердитим. Начальник не буде показувати свій характер, поки ви не освоїтеся на новому місці. Поки не пізно, ви можете вирішити, що ваша початкова оцінка була занадто позитивною, і піти, або вас можуть звільнити.

Прийняття і підтримка. Випробувальний термін закінчився, і ви стаєте повноцінним членом групи. Ви повинні працювати з партнером або групою над визначенням своєї ролі. Це називається процесом рольових переговорів. Коли ви вже достатньо освоїлися зі своєю роллю, рівень ваших зобов'язань досягає максимуму. Ви можете вирішити, що вам подобається ця робота, і залишитися тут назавжди. Ви - повноцінний член групи.

Розчарування. Не всі групи і відносини «живуть довго і щасливо». Можливо, вам не вдається знайти своє місце в групі або налагодити відносини, про які ви мріяли, або вас не дуже влаштовує нова робота. Коли позиції наших зобов'язань похитнулися, ми знову повертаємося до оцінювання. Ви можете почати думати про те, щоб піти, відокремитися від групи. Але група також починає оцінювати вас. Якщо група хоче вас утримати, вас знову накриє хвиля уваги. Це етап повторної соціалізації. Якщо ви приймете рішення залишитися, ви повертаєтеся на стадію прийняття та підтримки, яку називають «нарешті вдома». У вас знову з'являється можливість залишитися на даній стадії слабкого оцінювання, високих зобов'язань і повноцінного членства назавжди.

Вихід з групи. В іншому випадку, спираючись на повторне оцінювання, ви вирішуєте, що не хочете повертатися до колишніх відносин. Ви виходите з групи і знову опиняєтеся в ролі «одинака». Можливий також варіант, коли ви вирішуєте повернутися, але група вже вас не приймає. Або, що найкраще, ваше рішення взаємне.

Спогади. Однак, якщо мова йде про групу, роботу, школу, романтичні стосунки і т.п., залишилося зробити ще дещо. Ви повинні проаналізувати свій досвід і винести уроки з того, що сталося. Ви вступаєте на стадію спогадів. Ця стадія часто асоціюється з негативними почуттями. Багато хто вважає, що спогади записані в нашому мозку, як слова на сторінці, і все, що потрібно

зробити, - це повернутися і знайти їх, коли ми захочемо щось згадати. Але подальші події можуть змінити наші спогади. Дослідження пам'яті показують, що до зміни або до спотворення результату може призвести навіть те, яким чином вас просять щось пригадати. Те, яким чином припинилися ваші стосунки з групою, може істотно вплинути на ваші спогади про групу.

Ви, ймовірніше за все, збережете хороші спогади про групу, якщо розрив з групою був частиною угоди, або якщо ваш відхід був спровокований зовнішніми силами, скажімо, ви залишаєте місце праці через переїзд вашої сім'ї в інше місто. Обидві сторони знали про тимчасовість відносин, або група хотіла утримати вас, ви хотіли залишитися, але не змогли. Згодом ви будете згадувати про це з любов'ю. Практично в будь-який інший ситуації спогади обох сторін будуть негативними. Ви покидаєте групу, коли інші хочуть, щоб ви залишилися. Природно, що ваш крок обґрунтований. Однак з точки зору групи ви - дезертир. Або, навпаки, вас звільняють. Можливо, ви не сподобалися членам групи, і ви ображені, що вас недооцінили.

Контрольні питання

1. Назвіть стадії розвитку групи.
2. Яку стадію групової динаміки називають «притиранням»?
3. На якій стадії групової динаміки група досягає найвищої ефективності?
4. Яка стадія розвитку групи характеризується найвищим рівнем напруженості і хвилювання?
5. На якій стадії групової динаміки виникає довіра між членами групи?
6. Чи можна обійти стадію конфронтації в процесі розвитку групи?
7. Чи можна розпад групи зробити успішним? Яким чином?
8. Що таке соціалізація? Які її основні процеси?
9. Назвіть етапи розвитку особистості в групі.

10. Як зробити так, щоб після розпаду групи люди залишились з добрими спогадами?
11. У якому випадку не вдасться зберегти добрі спогади про групу, навіть якщо перебування в групі було тривалим і комфортним?

ТЕМА 4.6. КОМАНДНІ РОЛІ

Команда, на відміну від звичайної робочої групи, не лише успішно здійснює свою місію, досягає високої продуктивності і зниження витрат, привертає і утримує нових клієнтів і т.п., але і задовольняє особистісні та міжособистісні потреби своїх членів (рис. 4.4). Ефективна команда - це група однодумців, об'єднаних спільною метою, які злагоджено і ефективно працюють для її досягнення.

а *б*

Рис. 4.4. Робоча група (а), команда (б)

Суть взаємодоповняльної збалансованої команди в тому, що кожен її член грає в ній свою унікальну роль. Якщо командна робота правильно побудована, члени команди разом можуть домогтися кращих результатів, ніж сума їх індивідуальних досягнень. У цьому виявляється синергетичний ефект - головна цінність командної роботи. ***Командна роль*** - це спосіб виконання членами команди своєї роботи, який визначається їхніми вродженими і набутими психологічними особливостями.

Однак не кожна людина здатна до групової роботи. Є цілковиті індивідуалісти, котрі в команді працюватимуть нижче середнього рівня або навіть можуть стати причиною її руйнування.

Розрізняють три види командних ролей: цільові, підтримуючі і негативні ролі (див. рис. 4.5).

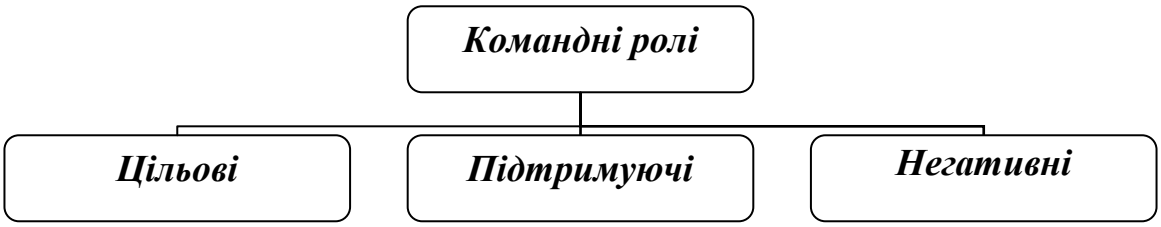


Рис. 4.5. Види групових ролей

Цільові ролі

Цільові ролі розподіляють таким чином, щоб була можливість відбирати і виконувати групові завдання. До цільових належать наступні ролі: «голова», «генератор ідей», «комунікатор-добувач інформації», «скептик-аналітик», «контролер-фінішер».

«Голова» - авторитетна людина з високою самодисципліною, яка вміє поставити перед командою стратегічні цілі та завдання. Він досягає цього не за рахунок своїх прав і влади, а завдяки такій організації роботи, за якої члени команди реалізують поставлені перед ними завдання як свої власні. «Голова» ініціює діяльність групи, відшукує нові сфери використання її можливостей, координує та інтегрує зусилля різних підгруп чи членів групи, керує виконавчою роботою, спрямовує творчий процес у потрібне русло.

«Генератор ідей» - індивід з найвищим рівнем інтелекту і найрозвинутішою фантазією. Його основне завдання — ідеї. Це автор оригінальних ідей, думок, пропозицій, нових підходів, нетрадиційних рішень. Радикальність його мислення, зорієнтованість на функціональні проблеми можуть «заносити» його в теоретичні хащі, з яких важко вибратися. Він може

припускатися помилок у деталях через власну неуважність. Інколи доводиться потурати певним його вчинкам, оскільки важливим для організації є кінцевий результат його діяльності.

«Скептик-аналітик» - людина з високим рівнем інтелекту і флегматичним характером. Він рідко пропонує оригінальні ідеї, однак визначає критерії, за якими слід оцінювати подані ідеї, і обов'язково виявить допущені при їх обґрунтуванні помилки, що зробить обговорюваний проект реальнішим.

«Комунікатор-добувач інформації» - своєрідний розвідник з неабиякими здібностями і можливостями у міжособистісному спілкуванні. Він інформує групу про всі події, підтримує стосунки з зовнішнім середовищем і в такий спосіб запобігає ситуаціям, коли рішення приймаються без урахування реалій навколишнього для даної команди світу.

«Контролер-фінішер» - це людина порядку, яка не любить неорганізаваності, змушує дотримуватись планів і відповідної якості в роботі. «Контролер» ніколи не заспокоїться, якщо сам не перевірить кожну деталь і не переконається, що все зроблено так, як слід. Має сильний характер і загострене відчуття ритму й часу, орієнтований на кінцевий результат.

Підтримуючі ролі

Підтримуючі ролі передбачають поведінку, яка сприяє підтримці та активізації діяльності групи. До цих ролей належать «секретар-оформлювач рішень», «організатор» та «душа команди».

«Секретар-оформлювач рішень». Його основна функція — чітке оформлення результатів. Об'єднує всі ідеї і думки в один завершальний проект, тому заохочує усіх до участі у підведенні підсумків, обговоренні проекту, домагається чіткості формулювань, дбає про те, щоб висновки підтверджували реальність та вигідність проекту.

«Організатор» раціоналізує виконання роботи і доводить справу до кінця. Його діяльність спрямована на практичне виконання завдань.

«Душа команди» - індивід, який працює краще від інших і водночас добре знає і відгукується на проблеми своїх колег. Яскравий колективіст, дружелюбний, чуйний; «цементує» команду, емоційно підтримує ініціативу, позитивно оцінюючи внесок усіх в загальну справу. За його відсутності справи йтимуть значно гірше, особливо в кризових, стресових ситуаціях.

Негативні ролі

Негативні ролі виникають за обставин, коли позиція певних людей або властиві їм норми поведінки не збігаються з позицією більшості членів групи. Негативними ролями можуть бути «король», «кавалер», «красуня», «критикан» та «блазень».

«Король» - це людина, яка звикла до головної ролі в групі і вважає, що має беззаперечне право на прийняття остаточного рішення, не зважаючи на сумніви інших. Своїм тиском цей член групи придушує ініціативу і робить групу неспроможною до прийняття деяких рішень.

«Кавалер» - це той, хто у всьому підтримує лідера групи, перший підхоплює висунуті ним пропозиції. Внаслідок цього створюється враження, що рішення вже знайдене, і робота групи завершується не оптимальним чином.

«Красуня» - це той, хто претендує на роль «найкращого працівника». Охоче розповідає про досягнуті успіхи, про те, якими вдалими були його ідеї, наголошує, що готовий із запалом взятись за вирішення нової проблеми. Але проект, яким займається група, йому нецікавий, тому він не хоче витрачати на нього свій час. Така поведінка дезорієнтує групу, оскільки її члени втрачають впевненість у перспективності справи, якою займаються.

«Критикан-скептик» - це той, хто готовий заперечити будь-яку ідею, довести її ефемерність і неспроможність. Він отримує задоволення від самого

процесу критики, яка викликає у нього почуття самоствердження. Перетворення «скептика-аналітика» на «критикана» є небезпечним для групи, оскільки налаштовує її песимістично щодо можливості вирішення проблеми.

«Блазень» - особа, що звикла завжди бути в центрі уваги. Якщо в групі вона не може виділитися своїми професійними здібностями, то бодай тим, що доречно і недоречно жартуватиме. Це вносить роздратування серед членів групи, сповільнює їх роботу і знижує ефективність групи. «Блазнем» може стати людина, яка претендувала на роль «душі команди».

Існує вірогідність перетворень позитивних ролей на негативні, тому керівнику групи слід бути готовим вживати застережних заходів.

Контрольні питання

1. Які види групових ролей ви знаєте?
2. Назвіть командні ролі, які відносяться до цільових ролей.
3. Які командні ролі є негативними?
4. Які командні ролі належать до підтримуючих?
5. Які функції в групі виконує «генератор ідей»?
6. Які функції в групі виконує «душа команди»?
7. Які функції в групі виконує «контролер-фінішер»?
8. Які функції в групі виконує «король»?
9. Які функції в групі виконує «красуня»?
10. Які функції в групі виконує «організатор»?
11. Які функції в групі виконує «кавалер»?
12. Як називається командна роль людини, яка претендує на роль «найкращого працівника»?
13. Як називається командна роль людини, яка у всьому підтримує лідера?
14. Як називається командна роль людини, яка звикла завжди бути в центрі уваги?

15. Хто з членів групи пропонує нові підходи, нетрадиційні рішення проблем?
16. Хто з членів групи керує діяльністю групи, скеровує її в необхідне русло?
17. Хто з членів групи підтримує стосунки з зовнішнім середовищем?
18. Хто з членів групи змушує дотримуватись планів і відповідної якості в роботі?
19. Порівняйте групові ролі «скептик-аналітик» і «критикан-скептик».

ТЕМА 4.7. ТЕОРІЯ РОЛЬОВОЇ ПОВЕДІНКИ БЕЛБІНА

Передбачити поведінку людини в команді можна з допомогою всесвітньо відомого тесту Белбіна.

Реймонд Мередіт Белбін (народ. В 1926 р.) - американський психолог, доктор психологічних наук, творець теорії і моделі «Ролі в команді менеджерів». Свої дослідження він розпочав у 1967 році, а у 1981 році світ побачила книга Белбіна «Команди менеджерів. Як пояснити їх успіх чи невдачу», яка визнана однією з п’ятдесяти кращих книг з менеджменту двадцятого століття. Сьогодні майже 70 % промислових підприємств і державних організацій Великобританії використовують методику Белбіна для створення ефективних команд.

Одна з головних тез Белбіна: «Команда - не збіговисько людей, що займають різні посади, а спільнота індивідуумів, кожен з яких відіграє роль, зрозумілу іншим». Жоден індивід не може володіти всіма якості, необхідними для ефективної діяльності, тоді як команда індивідів, безумовно, може їх поєднувати. Для того, щоб група професіоналів утворила ефективну команду, необхідне виконання певних умов.

У 1961 році стартував «пробний академічний експеримент»: в Школі менеджменту Хенлі, найстарішому управлінському коледжі Європи, почали проводити бізнес-гру для слухачів. Результатом багаторічного проведення гри

стала знаменита теорія рольової поведінки людей та їх взаємодії з командою. Як працюють команди, що зумовлює їх успіх, через що вони руйнуються і як змусити їх працювати краще? Провівши ґрунтовне дослідження, Белбін проаналізував командні ролі, порівняв склад команд-переможниць і команд-невдах, пояснив, які люди корисні в команді, описав сильні сторони членів команди, перерахував ключові командні ролі. Доктор Белбін та його колеги розробили процедуру формування успішних команд за результатами тестування без необхідності особистої зустрічі з людьми, які відповідали на тестові запитання.

Згідно з теорію Белбіна для ефективної роботи в команді потрібно вісім ролей, поділених на чотири категорії: лідери, трудяги, інтелектуали та парламентарі. На думку вченого «недосконалі люди можуть створити досконалу команду». Розглянемо кожну роль командну роль зокрема.

Лідери

До лідерів у групі належать Координатор та Мотиватор.

Координатор або ***Голова (Coordinator)*** – це лідер, який заохочує і підтримує членів своєї команди. Він необхідний для організації процесу, керівництва, розподілу повноважень, прийняття рішень. Координатор спокійний, терплячий, впевнений у собі, самодисциплінований, не метушиться. Він вміє чітко формулювати мету, делегувати повноваження, самостійно приймати рішення, беручи на себе відповідальність за них. Такий тип лідера схильний довіряти людям, радо приймає внесок кожного працівника в спільну діяльність і оцінює їх відповідно до загальної мети. Він здатний оптимально використовувати ресурси, добре знає сильні і слабкі сторони команди і використовує потенціал кожного члена команди якнайкраще.

Координатор може не володіти блискучим інтелектом, але вміє чудово організовувати діяльність команди. Ідеальний Координатор є хорошим

менеджером, що легко знаходить спільну мову з колегами, але ніколи не втрачає контролю над ситуацією і здатності приймати рішення.

Мотиватор (Shaper) - підприємницький тип лідера, для якого головне - перемога. Він орієнтований на завдання, повний енергії, зацікавлений в досягненні мети і стимулює інших до цього. Увага Мотиватора скерована на формування цілей і пріоритетів. Він прагне надати певну форму і структуру дискусіям в команді, «мотивує» команду на досягнення результату, володіє собою в напруженій обстановці, наполегливий у подоланні перешкод, вимогливий, товариський та динамічний.

Мотиватор не дозволяє команді нудьгувати або спокійно спочивати на лаврах попередніх перемог. Він постійно підштовхує групу до активних дій, захоплює колег своєю енергією, кидає виклик до подолання перешкод. Представники цього типу лідерів більші індивідуалісти, ніж Координатори, вони володіють високою самооцінкою, не терплять поразок, схильні до провокацій. Їхній запал і тиск на команду можуть призвести як до успіху, так і до невдачі. Мотиватори стають в пригоді не на етапі становлення команди, а для сформованої команди, яка переживає складні ситуації.

Трудяги

Ця група об'єднує собі ролі Реалізатора та Контролера.

Реалізатор або ***Виконавець (Implementer)*** виконує рутинну роботу. Саме Реалізатори перетворюють плани на практичні дії, хоча повільно реагують на нові можливості. Це самодостатні особистості, які володіють високими професійними знаннями і навичками.

Ці члени команди дисципліновані, надійні, мають розвинене почуття обов'язку, передбачувані та консервативні. Вони ефективно працюють, завжди виконують доручені завдання. Виконавці володіють внутрішньою стабільністю і низьким рівнем занепокоєння, працюють переважно на

команду, а не заради задоволення власних інтересів. При виконанні робіт вони дотримуються поставленої перед ними мети, систематично складають плани і виконують їх. Виконавці є ефективними організаторами та адміністраторами, але можуть бути недостатньо гнучкими і не люблять неперевірені ідеї.

У великих, добре структурованих організаціях кар'єра таких людей зазвичай складається дуже успішно. Завдяки своїм хорошим організаторським здібностям і компетентності Реалізатори часто просуваються до високих посадових позицій в управлінні.

Контролер (***Completer Finisher***) – член команди, який кожну справу доводить до кінця, вчасно виконує всі завдання.

Як правило, про успіх команди судять за остаточними результатами її роботи. При цьому багато людей майже патологічно не можуть довести розпочате ними до кінця, і вміння завершувати справу є досить рідкісною якістю. Контролери-фінішери - це люди, які володіють цим даром в повній мірі. Вони педантичні, старанні та сумлінні. Ці члени колективу методично забезпечують максимальний захист команди від помилок, пов'язаних з роботою. Саме вони контролюють терміни виконання доручень. Ці працівники занурюються у дрібниці та із задоволенням виконують діяльність, що вимагає підвищеної уваги. Контролери знаходить помилки, недоопрацювання, вміють концентруватися на цілях і допомагають це зробити іншим. Їх відрізняє увага до деталей і вміння тримати в голові заплановане, контролюючи доведення до завершення всіх деталей плану. Вони орієнтовані на виконання завдань, а не цікавляться ефектним і гучним успіхом. Схильність до досягнення досконалості у всьому, за що вони беруться, і непохитність у досягненні наміченого - їх неодмінні якості.

Недоліком Контролерів-фінішерів є схильність до надмірних хвилювань та недостатня гнучкість, в результаті чого вони можуть занадто багато зусиль витрачати на досягнення мети, що змінилася в процесі виконання робіт.

Інтелектуали

До цієї групи Белбін зараховує Генератора ідей та Аналітика.

Генератор ідей або ***Мыслитель (Plant)*** креативний, оригінальний, нестандартний. У нього високі розумові здібності і чудово розвинута уява. Він здатний вирішувати найскладніші завдання, створювати величезну кількість нових ідей, пропонувати концептуально нові стратегії, не вдаючись до подробиць.

Генератори ідей пропонують ідеї, з яких розвивається більшість розробок і проектів. Вони люблять працювати самостійно, відокремившись від інших членів команди, використовуючи свою уяву і вибираючи нетрадиційні шляхи. Зазвичай Генератори ідей - інтроверти, які гостро реагують як на критику, так і на похвалу. Вони незалежні, розумні і оригінальні, але можуть бути слабкими в спілкуванні з людьми іншого рівня.

Генератори ідей необхідні на початкових стадіях проектів або коли проект знаходиться під загрозою зриву. Дуже часто саме Генератори ідей є засновниками підприємств або організаторами нових виробництв. Але присутність кількох Генераторів ідей в одній компанії може мати негативні наслідки, оскільки вони мають тенденцію конфліктувати один з одним.

Аналітик - людина, що залишається в тіні до тих пір, поки не потрібно приймати рішення, володіє критичним мисленням і відмінною здатністю до аналітики. Він обдумує стратегії, тверезо мислить і вміє прослідковувати динаміку просування до мети. Він аналізує всі можливі варіанти і дає свої пропозиції, але дуже часто Аналітику не вистачає наполегливості, щоб переконувати інших членів команди.

Парламентері

Четверта категорія включає ролі Колективіста і Дослідника.

Колективіст або ***Гармонізатор (Team Worker)*** - м'який і дипломатичний. Він привносить в групу співробітництво, попереджує

виникнення суперечок, робить можливим застосування навичок членів команди з важкими характерами, покращує взаєморозуміння між ними, вміє слухати, розряджає напружену обстановку в команді. Його життєвими орієнтирами є люди, цінності, процес.

Якщо в команді є складні в спілкуванні люди, то Колективісти здатні м'яко впливати на ситуацію і попереджувати можливі конфлікти, допомагаючи формальному лідерові команди у виконанні поставленого завдання. Недоліком Колективіста є нерішучість у складних непередбачуваних ситуаціях.

Представники цього типу нерідко зустрічаються серед вищого керівництва організацій. Проте Колективісти можуть стати відмінними наставниками молодих працівників.

Дослідник або ***Розвідник (Resource Investigator)*** - екстраверт, сповнений ентузіазму, комунікабельний.

Це ще один член команди, орієнтований на пропозицію нових ідей. Розвідники схильні не стільки самі пропонувати оригінальні ідеї, скільки «виловлювати» фрагменти пропозицій інших членів команди і розвивати їх. Вони особливо вправні у вивченні ресурсів за межами команди. Стиль побудови команди Розвідника - створити мережу і збирати корисні ресурси для команди. При середніх показниках інтелектуального рівня і креативності, вони товариські, допитливі і соціально орієнтовані. Завдяки цим якостям і вмінню використовувати ресурси Розвідники легше, ніж Генератори ідей інтегруються в команду. При вмілому керівництві лідера команди Мислитель і Розвідник можуть успішно співіснувати разом, не зазіхаючи на територію один одного і вносячи кожен свій внесок в пропозицію нових ідей.

Дослідники повідомляють про ідеї, розробки і ресурси за межами групи, встановлюють корисні зовнішні контакти, добре проводять переговори, підтримують та розвивають ідеї інших. Недоліком Дослідників є те, що вони можуть швидко втрачати ентузіазм.

Теорія Белбіна припускає наявність у кожної людини схильності до виконання двох-трьох яскраво виражених ролей. Якщо «ваша» стандартна роль в команді вже «зайнята», то знаючи, яких ролей у команді не вистачає, ви можете зайняти одну з відсутніх ролей і успішно її виконувати.

Підбір команди

За традицією, члени команди відбираються за розумовими показниками. Підставою для прийняття на роботу цілком може стати диплом поважного вишу і підтверджена здатність до вирішення будь-яких завдань. Але не завжди можна розраховувати на таких «розумників» в командній грі.

Дослідження Белбіна показали, що люди з гострим розумом володіють критичним мисленням, здатністю бачити і критикувати недоліки чужих пропозицій, але не націлені на пошук ефективного рішення. На додаток до цього, такі люди не вчаться на своїх помилках, перекладаючи відповідальність на інших. Багато керівників, розглядаючи нових кандидатів, враховують їх характер. А більша частина компаній бере в свій штат людей одного типу.

Формула успішної команди, на думку Белбіна, починається з того, що повинен бути правильно обраний керівник, виходячи з його особистісних характеристик. Бажано, щоб майже всі члени команди володіли приблизно однаковими інтелектуальними здібностями. Необхідна наявність одного топ-менеджера, який інтелектуального перевершує інших. Найкраще, якщо в команді будуть представники всіх ролей, а ролі кожного учасника повинні відповідати їх особистісним якостям. Найуспішнішими є команди, учасники яких прислухаються один до одного і можуть заміняти один одного.

Реальні команди формують виходячи з того, які завдання планується вирішувати. Якщо планується довготривала співпраця, то члени команди підбираються, виходячи з результатів особистісних тестів, командних ігор та

навчання.

Тести рівня інтелекту разом з психологічними тестами є дуже важливими інструментами для підбору кадрів. Однак не можна розраховувати лише на них, адже в реальному житті все відбувається набагато складніше. Тести здатні підказати, яку роль слід віддати новому гравцеві. Після цього він повинен почати виконувати відповідні функції в командній грі під час експерименту, щоб переконатися, що новий гравець зможе стати частиною даної команди.

Підбираючи людей в свою команду, необхідно мати на увазі, що максимально ефективною є змішана команда, тому що в ній мають можливість розкритися люди будь-якого психологічного типу, включаючи навіть інтровертів.

У будь-якій команді головну роль буде грати керівник, що володіє позитивним мисленням, здатний завжди контролювати себе і нести відповідальність за прийняті рішення. Інтелектуальний рівень такої людини повинен перевищувати інтелектуальний рівень всіх інших. У команді «розумників» керівник повинен бути найрозумнішим учасником або бути здатним домінувати над іншими. А в команді емоційно стійких екстравертів керівник повинен бути першим серед рівних або якомога яскравіше виділятися, будучи «суперзіркою».

Контрольні питання

1. Назвіть основні командні ролі за класифікацією Белбіна.
2. Які ролі належать до інтелектуальних?
3. Які ролі належать до соціальних?
4. Які ролі належать до діяльних?
5. Як можна групувати командні ролі за напрямками діяльності?
6. Як ви вважаєте, якими ролями володієте ви. Перевірте свої здібності до групової роботи з допомогою теста Белбіна.

7. На скільки категорій розподіляються ролі в команді? Назвіть їх.
8. Охарактеризуйте кожну з ролей, які згідно теорії Белбіна необхідні для ефективної роботи в команді.
9. Які поради дає Реймонд Белбін для створення нової команди?

ТЕМА 4.8. СПІЛКУВАННЯ ЯК ОБМІН ІНФОРМАЦІЄЮ

Процес комунікації

Спілкування між членами групи формує психологічний клімат команди, її організаційну і виробничу мобільність, конкурентні позиції на ринку. Характер спілкування обумовлюється етнічними, професійними, віковими, гендерними та багатьма іншими параметрами, які необхідно знати і враховувати. Адже правильно організоване спілкування не лише забезпечує ефективний обмін інформацією, а й дає змогу глибше пізнати партнера, спрогнозувати особливості подальшої ділової взаємодії з ним.

Комунікація - процес обміну інформацією. Від успішного управління комунікаціями в організації залежить ефективність її життєдіяльності, рівень інформованості працівників. В будь-якій організації ***комунікація*** займає центральне місце, тому що структура, розміри і масштаби діяльності організації майже повністю визначаються засобами комунікації.

Комунікація (від лат. *зв'язую, спілкуюся*) - це шлях, повідомлення, зв'язок одного місця з іншим; спілкування, передача інформації від людини до людини - специфічна форма взаємодії людей у процесах їх пізнавально-трудової діяльності, що здійснюється головним чином за допомогою мови (рідше за допомогою інших знакових систем).

Комунікація - це процес, за допомогою якого якась ідея передається від ***джерела*** до ***одержувача*** з метою змінити поведінку одержувача. Цією поведінкою можуть бути знання, соціальні установки і т.п. Коли керівник

віддає розпорядження, то він чекає, що розпорядження буде виконано (хоча його можуть і не виконати).

Комунікація між керівником і підлеглими суттєво відрізняється від комунікації між людьми з однаковим статусом в організації.

Моделі комунікаційного процесу

Модель ДКПО

Модель комунікаційного процесу (модель ДПКО) містить наступні компоненти: джерело, повідомлення, канал та одержувач.

Рис. 4.5. Модель комунікаційного процесу ДПКО

Джерело (***комунікатор)*** - особа, що передає інформацію. Це творець повідомлення, відправник.

Повідомлення - власне інформація, стимул, який джерело передає одержувачу за допомогою символів (кодування і декодування).

Канал - це засіб (масової інформації, безпосереднього спілкування), за допомогою якого повідомлення передається від джерела до одержувача, шлях фізичної передачі повідомлення.

Одержувач (***реципієнт)*** - особа, яка отримує інформацію. Це найбільш важливий елемент комунікаційного процесу.

Модель мовного комунікативного процесу Г.Лассуелла

Запропонована американським дослідником Г. Лассуеллом модель впливу засобів масової інформації на свідомість людей включає п'ять елементів:

1. **Хто?** (передає повідомлення) - Комунікатор.
2. **Що?** (передається) - Повідомлення (текст).
3. **Як?** (здійснюється передача) - Канал.
4. **Кому?** (спрямоване повідомлення) - Аудиторія.
5. **З яким ефектом?** - Ефективність.

Комунікаційний процес складається з наступних ***етапів***: зародження ідеї, кодування, вибору каналу і передачі інформації та декодування.

Специфіка міжособистісної комунікації розкривається насамперед у наступних процесах і феноменах: процесі зворотного зв'язку, наявності комунікативних бар'єрів, феномені комунікативного впливу, існування різних рівнів передачі інформації (вербального й невербального).

Ефективність комунікаційного процесу визначається ***зворотнім зв'язком*** – реакцією одержувача на повідомлення. Зворотний зв'язок - це інформація, що містить реакцію реципієнта на поведінку комунікатора. Мета зворотного зв'язку - допомога партнерові по спілкуванню в розумінні того, як сприймаються його вчинки, які почуття вони викликають в інших людей.

Існує три основних ***типи ефекту комунікації***, що функціонують послідовно: зміни в знаннях одержувача; зміни установок одержувача (уявлень про об'єкт його дій); зміни явного поведінки одержувача повідомлення.

Ступінь подібності пари «джерело-одержувач» називається гомофілією, а ступінь відмінності - гетерофілією.

Перешкоди і спотворення в комунікації позначені як її шум (надлишкова інформація, занадто голосна розмова, тощо).

Комунікаційні ролі

Комунікаційні ролі - це функції, які виконують члени організації в процесі передачі інформації.

Виділяють наступні комунікаційні ролі:

- **Сторож** (контролює вхідні потоки повідомлень). Цю роль може виконувати секретар, який не дозволяє спілкуватися з керівником або відсіває інформацію, передаючи лише її частину;

- ***Зв'язковий*** – людина, яка пов’язує групи, не належачи ні до однієї з них; без нього організація розвалюється на групи, руйнується її єдність;

- ***Лідер думки*** - неформальний лідер, який здійснює неформальний вплив на поведінку інших;

- ***Космополіт*** – людина, яка частіше за інших взаємодіє із зовнішнім середовищем, відкриваючи «вікна у світ».

Рівень обміну інформацією в колективі визначається стосунками між його членами. Відносини між членами колективу поділяються на такі типи:

- ***лінійні*** (відносини між керівником і підлеглими);

- ***функціональні*** (відносини фахівців та інших працівників, часто носять консультативний характер);

- ***відносини управлінського апарату*** (подання чиїхось повноважень або прав);

- ***латеральні відносини*** (бічні, віддалені від центру, їх також називають колегіальними - між службовцями, підпорядкованими одному начальнику і паралельними - між службовцями, які займають однакове положення в організації).

У процесі спілкування обмін інформацією між його учасниками здійснюється як на ***вербальному***, так і ***невербальному (немовному)*** рівні.

Вербальна і невербальна комунікація

На ***вербальному рівні*** засобом передачі інформації є людське мовлення, природна звукова мова, тобто система фонетичних знаків, що включає два принципи: лексичний і синтаксичний. Мовлення є найуніверсальнішим засобом комунікації. За допомогою мовлення здійснюється кодування й

декодування інформації: комунікатор у процесі говоріння кодує, а реципієнт у процесі слухання декодує цю інформацію.

Існують характеристики комунікатора, що сприяють підвищенню ефективності його мовлення, зокрема його ***позиції*** під час комунікативного процесу:

- ***відкрита***, коли комунікатор демонструє свою прихильність та відкрито викладає свою думку і факти на її підтвердження;
- ***відсторонена***, коли комунікатор тримається підкреслено нейтрально, надаючи суперечливі точки зору і не орієнтуючись на жодну з них;
- ***закрита***, коли комунікатор приховує свою думку.

Невербальна комунікація - вся сукупність засобів, покликана виконувати наступні функції: доповнення мовлення, заміщення мовлення та репрезентація емоційних станів партнерів при комунікативному процесі. До невербальної комунікації відносяться зовнішній вигляд, виразні рухи людини: жести, міміка, пози, хода, а також контакт очей. Ці засоби доповнюють мову, проектуючи емоційні реакції людини, які ми «зчитуємо» у процесі спілкування. Тоді жести, міміка й контакт очей допомагають визначити його щирість.

Види невербальної комунікації

Розрізняють такі п’ять видів невербальної комунікації: кінесика, паралінгвістика, екстралінгвістика, проксеміка та візуальне спілкування.

Кінесика (др.-грец. κίνησις - рух) - сукупність рухів (жестів, міміки), що застосовуються в процесі людського спілкування (за винятком рухів мовного апарату).

Кінесика вивчає відображення поведінки людини в її ***невербальних проявах***:

- міміка (рух м'язів обличчя),

- жестикуляція (рухи руками);
- пантоміміка (рухи всього тіла, пози),
- просторовий малюнок (зона, територія, власність і переміщення),
- експресія (виразність, сила прояву почуттів, переживань).

Необхідно враховувати той факт, що в різних культурах один і той самий жест може трактуватися по різному. Наприклад, у дослідженнях М. Аргайла вивчалися частота й сила жестикуляції в різних культурах (протягом однієї години фіни жестикулювали 1 раз, італійці – 80 разів, французи – 120 разів, а мексиканці – 180 разів).

Паралінгвістична система – це «вокальна міміка» (інтонація, тембр, ритм, діапазон голосу, тональність, фразові й логічні наголоси, яким надає перевагу конкретна людина).

Екстралінгвістична система – це включення в мовлення пауз, інших вкраплень, наприклад покахикування, плачу, сміху, нарешті, сам темп мовлення. Всі ці доповнення збільшують семантичне значення інформації, але не за допомогою додаткових мовних включень, а навколомовними прийомами.

Проксеміка - спеціальна галузь, що займається нормами просторової та часової організації спілкування. Засновник проксеміки Е. Хол назвав її «просторовою психологією». Простір і час організації комунікативного процесу виступають також особливою знаковою системою, несуть смислове навантаження як компоненти комунікативних ситуацій. Так, розміщення партнерів один навпроти одного сприяє виникненню контакту, символізує увагу до промовця, у той час як окликування у спину може мати певне негативне значення.

Американський психолог Греенвілл Стенлі Гол (1844-1924) зафіксував норми наближення до партнера по спілкуванню, властиві американській культурі:

- інтимна відстань (0-45 см);
- персональна відстань (45-120 см);

- соціальна відстань (120-400 см);
- публічна відстань (400-750 см).

Кожна відстань характерна для окремої ситуації спілкування.

Візуальне спілкування. Як і інші невербальні засоби, «контакт очима» значно доповнює вербальну комунікацію, тобто повідомляє про готовність підтримати комунікацію або припинити її, заохочує партнера до продовження діалогу, нарешті, сприяє тому, щоб виявити повніше своє "Я", або, навпаки, приховати його.

Нараховують більше двадцяти тисяч описів виразів обличчя. Розглянемо одну з методик їх класифікації, запропоновану американським психологом Полом Екманом (нар. 1934 р.). Спеціаліст у сфері психології емоцій, міжособистісного спілкування та розпізнавання брехні запропонував принцип FAST, відповідно до якого обличчя ділиться на три зони горизонтальними лініями: очі й чоло, ніс і ділянка носа, рот і підборіддя.

Фіксація емоції кожної зони дозволяє реєструвати більш-менш виразно мімічні рухи. Виділяють шість ***основних емоцій***, що виражаються за допомогою міміки: радість, гнів, подив, відраза, страх, смуток.

Дослідження, які пов'язані з виявленням механізмів утворення різних емоційних відносин до людини, що сприймається отримали назву атракції й емпатії.

Атракція - поняття, що означає виникнення при сприйнятті людини людиною привабливості один до одного. Говорячи іншими словами: атракція - це мистецтво подобатися іншим людям, справляти на них приємне враження.

Емпатія - співпереживання іншій людині, здатність відчувати те ж, що й співрозмовник, розуміти його не «розумом», а «серцем» (тобто проникнення в переживання іншої людини).

Комунікативні бар'єри

Адекватність сприйняття інформації багато в чому залежить від наявності або відсутності в процесі спілкування ***комунікативних бар'єрів***. У випадку виникнення бар'єру інформація або спотворюється, або втрачає свій зміст, або взагалі не доходить до реципієнта.

Комунікативними перешкодами може бути механічний обрив інформації, що призводить до її перекручування; нечіткість переданої думки, через що можливе спотворення переданої інформації. Ці варіанти можна назвати ***інформаційно-дефіцитним бар'єром.***

Трапляється, що реципієнт чітко чує передані слова, але надає їм іншого значення. Проблема полягає в тому, що передавач може навіть не здогадуватись, що його сигнал викликав неправильну реакцію. ***Перекручування*** інформації, що проходить через одну людину, може бути незначним. Але коли вона проходить через кількох людей-ретрансляторів, перекручування може бути дуже істотним.

Значно більша можливість перекручування пов'язана з емоціями. ***Емоційні бар'єри*** мають місце, коли отримувачі інформації більше зайняті своїми почуттями, припущеннями, ніж реальними фактами. Слова мають сильний емоційний заряд, до того ж не стільки самі слова (символи), скільки асоціації, які вони породжують у людині. Слова мають первинне (буквальне) значення й вторинне (емоційне).

Можна говорити про існування ***бар'єрів нерозуміння, соціально-культурного розходження й бар'єрів відносин***.

Бар’єр нерозуміння може виникати через похибки в самому каналі передачі інформації; це так називане ***фонетичне нерозуміння.*** Насамперед, воно виникає, коли учасники спілкування розмовляють різними мовами або діалектами, мають істотні дефекти мовлення чи дикції. Бар’єр фонетичного

нерозуміння породжує також невиразне швидке мовлення, мовлення-скоромовка й мовлення з використанням слів-паразитів.

Існує також ***семантичне нерозуміння***. Цей бар'єр пов'язаний, у першу чергу, з розбіжностями у системах значень (тезаурусах) учасників спілкування. Це, насамперед, проблема жаргонів і сленгів. Не менш важливу роль у руйнуванні нормальної міжособистісної комунікації може зіграти ***стилістичний бар'єр***, що виникає при невідповідності стилю мовлення комунікатора й ситуації спілкування або стилю мовлення й актуального психологічного стану реципієнта та ін. Нарешті, можна говорити про існування ***логічного бар'єра*** нерозуміння. Він виникає в тих випадках, коли логіка міркування, пропонована комунікатором, або занадто складна для сприйняття реципієнта, або здається йому не вірною, суперечить властивій йому манері доказу. Для одних людей логічно й доказово те, що не суперечить розуму, для інших те, що відповідає обов'язку й моралі. Можна говорити про існування «жіночої» і «чоловічої» логіки, про "дитячу" логіку і т.п.

Засоби передачі інформації

На жаль, досить часто люди в процесі спілкування «не чують» один одного. У науковому плані говорять про ***ефективне й неефективне слухання***. Слухання неефективне в тих випадках, коли воно не забезпечує правильного розуміння слів і почуттів співрозмовника, створює в мовця відчуття, що його не чують, замінюють його проблему іншою, більш зручною для співрозмовника, вважають його переживання незначними. Слухання неефективне й у тих випадках, коли не забезпечує просування партнерів по спілкуванню в розумінні обговорюваної проблеми, не приводить до її рішення або правильної постановки, не сприяє встановленню довірливих відносин між партнерами по спілкуванню.

Розрізняють два види ***ефективного слухання***, що відрізняються за ситуаціями їхнього використання.

Нерефлексивне слухання – уміння уважно мовчати, не втручаючись у мовлення співрозмовника своїми зауваженнями. Такий прийом застосовується на етапах постановки проблеми, коли вона тільки формулюється, а також у ситуації, коли ціль розмови з боку промовця – «вилив душі», емоційна розрядка. Уважне мовчання - це слухання з активним використанням невербальних засобів: кивків, мімічних реакцій, контакту очей і поз, уважного інтересу. Використовуються також мовні прийоми на кшталт повторення останніх слів промовця або підтакування.

Рефлексивне слухання – це об'єктивний зворотний зв'язок з промовцем за допомогою таких прийомів, як задавання відкритих і закритих запитань по темі розмови (з'ясовування), перефразовування слів співрозмовника, що дозволяє викласти ту ж думку іншими словами (парафраз), відбиття почуттів і резюмування - виклад проміжних і остаточних висновків до бесіди (звичайно використається в тривалих розмовах).

У спілкуванні фахівця будь-якого профілю із клієнтом чи колегою адекватність зворотного зв'язку потребам обох партнерів - важлива й необхідна умова встановлення довірливих стосунків.

Під засобами спілкування розуміється те, яким чином людина реалізує певний зміст і мету спілкування. Залежать вони від культури людини, рівня її розвитку, виховання та освіти. Коли мова йде про розвиток у людини здібностей, умінь і навичок спілкування, насамперед, мають на увазі техніку й засоби спілкування. ***Техніка спілкування*** – це способи налаштування людини на спілкування, її поведінка в процесі спілкування, а ***прийоми*** - засоби спілкування, які включають вербальні й невербальні.

На початковому етапі спілкування його техніка включає такі елементи, як прийняття певного виразу обличчя, пози, вибір початкових слів, тону

висловлення, рухів і жестів, що залучають партнера до дій, спрямованих на сприйняття переданої інформації.

Існує безліч ***прийомів підвищення ефективності спілкування***, подолання комунікативних бар'єрів. Назвемо деякі з них.

1. Прийом ***«власне ім'я»*** полягає в проголошенні вголос імені та по-батькові партнера, з яким спілкується працівник. Це підкреслює увагу до даної особистості, викликає в неї почуття задоволення й супроводжується позитивними емоціями, тим самим формується атракція, довіра працівника до клієнта або партнера.

2. Прийом ***«дзеркало відносин***» полягає в добрій посмішці й приємному виразі обличчя, який ніби промовляє: «Я - ваш друг». А друг - це прихильник, захисник. У співрозмовника виникає почуття захищеності, що супроводжується позитивними емоціями і мимоволі формує атракцію.

3. Прийом ***«золоті слова»*** полягає у висловлюванні компліментів на адресу співрозмовника, що також породжує позитивні емоції.

4. Прийом ***«терплячий слухач***» випливає з терплячого й уважного вислуховування проблем клієнта. Це призводить до задоволення однієї з найважливіших потреб будь-якої людини - потреби в самоствердженні.

5. Прийом ***«особисте життя***» проявляється у виявленні уваги до захоплень клієнта чи партнера, що також підвищує його вербальну активність і супроводжується позитивними емоціями.

Контрольні питання

1. Які компоненти містить модель комунікаційного процесу?
2. Як називається особа, яка передає інформацію?
3. Як називається особа, яка отримує інформацію?
4. Що таке атракція?
5. Що таке емпатія?

6. Які прояви поведінки людини вивчає кінетика?
7. Що таке паралінгвістика?
8. Що таке екстралінгвістика?
9. Що таке проксеміка?
10. У чому полягає візуальне спілкування?
11. Що таке комунікативні бар’єри?
12. Які бувають види слухання? Чим вони відрізняються?
13. Які ви знаєте прийоми подолання комунікативних бар'єрів?

ТЕМА 4.9. ДІЛОВЕ СПІЛКУВАННЯ ДЛЯ ПРАЦЕВЛАШТУВАННЯ

Пошук роботи

У добу доступного інтернету існує багато способів знайти роботу, окрім звернення до рекрутингового агентства чи на біржу праці. Це спеціалізовані інтернет-ресурси, електронні дошки, газети безкоштовних оголошень. До того ж, практично усі великі організації мають власні сайти, де постійно оновлюється інформація про наявні вакансії.

Пошук роботи розпочинається з написання ***резюме*** або ***CV*** – «Curriculum Vitae», що перекладається з латини як «хід життя» – короткого опису життя і професійних навичок. Терміни CV і резюме часто застосовуються як синоніми і означають документ, який претендент надає потенційному роботодавцю, виставляючи свою кандидатуру на відкриту вакансію. Їх головна відмінність полягає в тому, що ***CV*** містить більш детальну інформацію про кандидата і може займати з десяток сторінок. У цьому документі повинні бути перераховані всі спеціальності, які ви отримали, додаткові курси, місця роботи, посади, які ви займали, стажування і обов'язки в хронологічному порядку. Такий докладний опис корисний для наукових працівників або журналістів, наприклад, якщо треба перерахувати всі публікації, конференції, в яких вони брали участь і т. п. Коли вас просять надіслати CV, це означає, що

вашою кандидатурою цікавляться в цілому: де ви вчилися, працювали, чим займалися в різні періоди життя, як складалася ваша біографія. У CV не потрібно фокусувати увагу на чомусь конкретному (на відміну від резюме). На дві різні посади можна відправляти одне CV, і це не буде помилкою.

Але при працевлаштуванні в більшості компаній докладний опис кандидата не потрібний. Роботодавці або агентства часто обмежені в часі, їм доводиться розглядати велику кількість відгуків на вакансії. У такому випадку застосовується ***резюме*** - короткий опис, що займає не більше двох аркушів формату А4. До резюме можна докласти супровідний лист. Напишіть в листі, чому ви хочете працювати саме в даній компанії, і чому вважаєте свою кандидатуру гідною уваги. Продемонструйте, що ви знаєте компанію і сферу її діяльності, що ви не випадкова людина.

На відміну від CV, резюме - більш стислий документ, в якому вказані ключові, найбільш важливі моменти вашої професійної діяльності. Також в резюме місця роботи вказуються в зворотному хронологічному порядку. Акценти щодо досвіду, навичок і досягнень можуть бути індивідуально розставлені під конкретну вакансію. Таким чином, резюме одного кандидата можуть мати різний зміст в залежності від вакансії, на яку його відправляють. Це необхідно для того, щоб привернути увагу роботодавця до вашої кандидатури і збільшити шанси на працевлаштування.

Завдання резюме – зацікавити потенційного працедавця. Серед сотень переглянутих резюме він повинен помітити саме ваше. Як цього досягти?

Влаштовуючись на роботу, ви продаєте свою кваліфікацію. Отже, про неї слід заявити грамотно і лаконічно. Вдало складене резюме стане щасливим квитком на співбесіду. Адекватна оцінка своєї кваліфікації справляє позитивне враження. Відомості про знання і навички мають відповідати дійсності, дати прийому-звільнення з попередніх місць роботи повинні співпадати з датами, вказаними у трудовій книжці.

Подайте коротке оголошення до газет та на дошки оголошень, позначте у ньому свою кваліфікацію та бажану вакансію, рекрутингові агенції та працівники відділів кадрів великих компаній передивляються такі оголошення регулярно. Передивіться пропозиції на сайтах, перегляньте газету та розішліть резюме по всіх вакансіях, що вас зацікавили. Якщо отримаєте запрошення на співбесіду від багатьох фірм, будете мати з чого вибирати – не варто відмовлятися від жодної із запропонованих зустрічей.

Структура резюме

Перше враження про претендента роботодавець складає, аналізуючи його резюме. Від того, наскільки професійно ви оформите резюме, буде залежати не лише те, чи запросять вас на співбесіду, але і саме працевлаштування.

Для того, щоб роботодавець міг швидко знайти необхідну для себе інформацію, резюме повинно мати наступну структуру:

1. **Персональні дані**. Подайте повне ім’я, по батькові та прізвище, а не ініціали.

2. **Контактні дані**. Включіть якомога більше контактних даних, щоб роботодавець міг легко зв’язатися з вами. Вкажіть один або два номери телефонів та електронну адресу. Якщо у вас немає особистої адреси, яка в назві містить ваше ім’я, краще спеціально створити її для пошуку роботи. Також бажано додати посилання на вашу сторінку в соціальній мережі, наприклад, професійної мережі LinkedIn.

3. **Узагальнення**. Відразу після вашого імені пропишіть дві-три коротких речення, в яких покажіть всі свої переваги як кандидата. Не пишіть загальних фраз. Визначте свої найсильніші професійні якості, які безпосередньо пов’язані з майбутньою роботою, наприклад, великий досвід в певній сфері або знання кількох мов. Цей пункт дуже важливий, оскільки роботодавець

витрачає на кожне резюме не більше 20 секунд, відповідно, захопити його увагу потрібно в перший момент.

4. **Компетенції.** Приведіть невеликий перелік ваших професійних знань і навичок. Простіше кажучи, напишіть, що саме ви вмієте робити. При цьому відштовхуйтесь від списку вимог і функціональних обов'язків, вказаних у вакансії. Наприклад, якщо, згідно вакансії, ви повинні створювати рекламні кампанії, вкажіть конкретні вміння: «Медіапланування», «Уміння працювати з замовниками» і т.д. Якщо ви сумніваєтеся, які вміння краще вказати в резюме, подивіться готові резюме кандидатів з вашої сфери на різних інтернет-порталах - не виключено, що ви знайдете цікаві підказки.

5. **Освіта.** Сюди можна включити як навчальний заклад, де ви отримали вищу освіту, так і всі великі тренінги, семінари, програми, які ви додатково проходили. Якщо ви відвідали велику кількість таких заходів, експерти радять відбирати лише ті, які близькі до вашої професії. Роботодавець не витрачатиме час та вчитуватися в довгий список ваших курсів, і може не помітити важливий семінар, який додасть вам ваги як фахівцю.

6. **Досвід роботи**. Ваш кар'єрний шлях краще описати в зворотному хронологічному порядку - від останнього місця роботи до першого. Оформити цю інформацію краще наступним чином: після назви місця роботи відразу прописати свої ключові досягнення в компанії. Це повинні бути конкретні приклади, чого ви домоглися на попередньому місці роботи: скоротили витрати компанії; налагодили систему співпраці між відділами; успішно управляли командою з п'ятдесяти працівників і т.п.

Тільки після цього прописуйте свої обов'язки. Уникайте загальних фраз і зосередьтеся на конкретних завданнях. Прописуючи обов'язки, потрібно описувати їх від найбільших обов'язків до дрібніших. Пам'ятайте, що ваша мета - вразити роботодавця з перших рядків, показавши професійне резюме.

7. **Додаткова інформація**. Тут роботодавець хоче побачити інформацію про ваше знання мов, опис навичок володіння комп'ютером, додатковий

досвід роботи, не пов’язаний безпосередньо з вашою професійною діяльністю (наприклад, волонтерство). Тут також розміщується інформація про наявність водійських прав.

Вимоги до професійного резюме

1. **Компактність.** Обсяг ідеального резюме викликає суперечки. При цьому експерти точно знають, яким воно бути не повинно: три сторінки - це недозволений розмір, який не подужає наймач. Разом з тим, деякі експерти вважають, що одна сторінка - це резюме для працівників-початківців. Оптимальний розмір - дві сторінки. Однак враховуйте, що з великою увагою буде переглядатися перша сторінка, тому зосередьтеся на тому, щоб вмістити туди всю найважливішу інформацію про себе.

2. **Виділяйте свої досягнення і унікальні якості**. Це один з найважливіших моментів при заповненні резюме. Резюме повинно бути переліком ваших досягнень, а не описом кар’єрного шляху. Складаючи резюме, пам’ятайте, що ваше завдання - показати роботодавцю, яку цінність ви представляєте для компанії.

Для цього виділіть свої досягнення за кожним місцем роботи в окремі блоки. Саме ваші досягнення, а не просто назви компаній є переконливим аргументом для прийому на роботу. Все, що ви впишете в резюме повинно показувати, що ви - найкращий кандидат.

3. **Говоріть з роботодавцем на одній мові**. Використовуйте в резюме так звані ключові слова. Для цього уважно прочитайте вимоги до вакансії і виділіть для себе основні терміни, якими користується роботодавець. «Наповніть» своє резюме тією ж лексикою, навіть якщо раніше ви користувалися іншою термінологією. У такому разі ваше резюме переконливо продемонструє, що ваші навички і вміння підходять компанії і ви відповідаєте її вимогам.

4. **Грамотність**. Перевіряйте, перевіряйте і ще раз перевіряйте резюме. Увімкніть функцію перевірки правопису і обов'язково попросіть когось переглянути резюме. Помилки відразу кинуться в очі наймачеві і, ще не читаючи резюме, він сформує про вас не найкращу думку.

5. **Зовнішній вигляд**. Форма резюме при влаштуванні на роботу теж має велике значення. Важливо, щоб ваше резюме легко читалося. Використовуйте типові в діловому стилі шрифти. Calibri, Arial, Times New Roman з 12 кеглем - найкращий варіант. Не перенасичуйте текст резюме виділеннями та підкресленнями. Якщо ж ви все-таки хочете виділити якісь елементи або назви, то переконаєтеся, що виділення однакові у всьому тексті.

6. **Відмовтеся від універсального резюме**. Багато роботодавців звертають увагу на те, наскільки ваше резюме адаптовано до конкретної вакансії. Не потрібно повністю переписувати резюме для тієї чи іншої компанії, однак додати специфічних деталей або змістити акценти треба обов'язково.

По-перше, винесіть найбільш підходящі для цієї вакансії навички та досягнення на перший план. По-друге, пам'ятайте про вже згадані ключові слова. Не забувайте, що роботодавець очікує побачити в вашому резюме відповідь на питання, чому ви найкращий кандидат на цю посаду.

7. **Фото тільки за необхідності**. Додавати фото до резюме бажано лише у тому випадку, якщо ваша зовнішність буде безпосередньо пов'язана з функціональними обов'язками. Наприклад, якщо ви претендуєте на посаду, яка вважається «обличчям компанії» або ж робота передбачає певну публічність (наприклад, PR-менеджери, продавці-консультанти, офіс-менеджери, промоутери). Крім того, експерти радять додавати фото, щоб зіграти собі в плюс, наприклад, якщо ви виглядаєте набагато молодшими від свого віку, але побоюєтеся, що за віком вам можуть відмовити в роботі.

8. **Не залишайте відкритих питань**. Це вкрай важливо, якщо ви хочете зробити ваше резюме справжнім «провідником» на роботу. Будь-яка

двозначність або недомовка в резюме може викликати у роботодавця додаткові запитання і сумніви. Постарайтеся прояснити всі моменти, які можуть викликати запитання. Не допускайте, щоб в вашому резюме з'явилися «білі» плями. Наприклад, якщо якийсь проміжок часу ви не працювали на постійній роботі, а займалися фрілансом - не втрачайте цей період, а вкажіть, що ви робили. Також вказуйте час, коли ви займалися волонтерською роботою або навчалися.

Роботодавців насторожує, коли вони бачать, що здобувач занадто мало пропрацював в тій чи іншій компанії (наприклад, менше ніж півроку). Це викликає підозри, що людина не пройшла випробувальний термін або її звільнили. Але іноді людина залишається без роботи з незалежних від неї причин. Наприклад, якщо вас скоротили, то допишіть про це в дужках. Якщо вас звільнили, допишіть «звільнення за згодою сторін».

Чого не повинно містити резюме?

1. **Обману**. Приписуючи собі додаткові знання і досвід, ви навряд чи отримаєте роботу. Максимум - вас запросять на співбесіду, однак досвідчений HR (від англ. *Human Resource* – людські ресурси, менеджер з підбору персоналу) в особистій розмові, задаючи додаткові запитання, відразу розкриє обман.

2. **Химерності стилю**. Якщо ви хочете створити резюме, яке запам'ятовується, не намагайтеся виділити своє резюме з допомогою незвичайного шрифту, формату або яскравих кольорів. Чітко структуроване резюме, в якому виразно продемонстровано ваші досягнення і переваги, набагато привабливіше для роботодавця.

3. **Кліше**. Викресліть з резюме все фрази в стилі «високомотивований фахівець з хорошими комунікативними здібностями і націленістю на результат» або «шукаю роботу, в якій зміг би проявити всі свої найкращі

якості і вміння». Пам’ятайте, що роботодавцю щодня надходять десятки, а то і сотні резюме. Ваше завдання - виділитися серед них.

4. **Інформації про загальні навички**. Користування Word і Excel вже давно перестало бути спеціальним навиком. Ним володіє будь-яка людина, що працює в офісі. Такі дані тільки обтяжать ваше резюме і відвернуть від дійсно істотних переваг.

5. **Особистих даних**. Сімейний статус, стать, наявність дітей не цікавлять роботодавця. Ваше завдання в лаконічному вигляді показати свої переваги як фахівця, а не розповісти свою життєву історію.

Підготовка до співбесіди

Для того, щоб роботодавець серед інших претендентів вибрав саме вас, потрібно ретельно підготуватись до співбесіди. Для цього необхідно дотримуватись наступних кроків:

1. Перевірте свій зовнішній вигляд. Переконайтеся, що ваш зовнішній вигляд відповідає корпоративній культурі компанії. Подумайте, який стиль буде найбільш доречним – діловий чи повсякденний. Ваше завдання на співбесіді - сподобатися роботодавцю, а зовнішній вигляд одна з важливих складових першого враження.

2. Перегляньте резюме. Перечитайте перед співбесідою своє резюме і супроводжуючі листи, щоб під час розмови з роботодавцем було простіше формулювати логічні відповіді з конкретними прикладами. Зверніть увагу на цифри і дати, які ви вказали в резюме, щоб не помилитися. Крім того, потрібно бути готовим уточнити і деталізувати кожен з пунктів резюме, на якому може зупинитися роботодавець. На співбесіді у вас точно не буде часу шукати пункт, який зацікавив майбутнього роботодавця, і згадувати, в якому контексті ви про це згадували.

3. Зберіть інформацію про компанію. Чим більше ви будете знати про компанію, тим краще і переконливіше звучатимуть ваші відповіді, ви

продемонструєте справжню зацікавленість в роботі і професійний підхід. Варто уважно переглянути останні новини про компанію (конкуренти, рейтинги, нагороди, проблеми) і подумати, чому вакансія, на яку ви претендуєте, взагалі відкрилася. Можливо компанія хоче активно розвивати новий бізнес-напрямок або у роботодавця існують певні проблеми або недоробки, які він хоче виправити, розширивши штат. Все це допоможе вам правильно побудувати свою розмову з роботодавцем і продемонструвати, що ви знайомі з тенденціями у даній галузі і орієнтуєтеся в специфіці роботи компанії.

4. Продумайте відповіді на типові питання. На співбесіді у вас, швидше за все, запитають про ваші сильні та слабкі сторони, чому ви хочете працювати в конкретній компанії і чому ви вважаєте себе найсильнішим кандидатом. Щоб не розгубитися, продумайте відповіді на всі ці питання. Для того, щоб краще зрозуміти спосіб вашого мислення і визначити, чи легко ви зорієнтуєтеся у складній, незнайомій ситуації, HR може використовувати підступні запитання та задачки на логіку і кмітливість.

5. Продумайте питання, які ви задасте роботодавцю. Наприкінці співбесіди вас також запитають, чи є у вас якісь питання. Продумайте заздалегідь, що б ви хотіли запитати і як це найбільш коректно сформулювати. У стресовій ситуації ви можете просто забути щось важливе.

6. Подумайте що ви будете говорити про зарплату. Перед співбесідою проаналізуйте ситуацію на ринку праці у вашій сфері та визначитеся, на яку суму ви розраховуєте. Деякі експерти вважають, що найгіршим варіантом відповіді на це питання може бути: «Скільки ви можете запропонувати?». Щоб такого не сталося, продумайте всі фінансові моменти заздалегідь.

7. Роздрукуйте кілька копій резюме. Про всяк випадок роздрукуйте кілька копій резюме, щоб бути готовим до будь-якого формату співбесіди. У певний момент вони можуть виявитися корисними.

8. Подолайте хвилювання. Справитися з хвилюванням безпосередньо перед дверима офісу компанії, до якої ви йдете на співбесіду, може виявитися непростим завданням. Тому краще постаратися зробити це заздалегідь, переконавши себе, що приводів для хвилювання немає, оскільки ви добре підготувалися і зможете презентувати себе на вищому рівні. І навіть, якщо вас не приймуть на бажану вакансію, ви отримаєте корисний досвід, необхідний для подальшого розвитку.

Основні питання, які задають на співбесідах

1. Розкажіть про себе. З відповіді здобувача працівник відділу кадрів оцінює, наскільки той вміє презентувати себе, свої сильні сторони, виділяти найголовніше зі своєї професійної біографії і встановлювати контакт. У жодному випадку не потрібно питати у HR-а, що б йому хотілося дізнатися. Пам’ятайте, що на співбесіді ви повинні бути в певній мірі «першою скрипкою». Відповідаючи на це питання, треба пояснити роботодавцю, чим ви можете бути корисні компанії на тій посаді, на яку претендуєте. Не потрібно описувати весь свій кар’єрний шлях. Резюмуйте весь свій досвід, виділивши свої сильні якості і ключові професійні вміння, які сформувалися у вас протягом усіх років роботи. Безумовно, перш за все, варто виділяти ті, які роботодавець вказав в описі вакансії.

2. Ким ви бачите себе через п’ять років? Ваша відповідь на це запитання допоможе зрозуміти, до чого прагне людина і наскільки вона цілеспрямована, наскільки серйозні ваші наміри і чи надовго ви збираєтеся залишитися в компанії. Розкажіть, якою ви уявляєте собі кар’єру в довгостроковій перспективі. Не обмежуйтеся порожніми фразами типу «планую професійно розвиватися». Вкажіть конкретні навички та вміння, які ви хочете розвивати в майбутньому, і як ви збираєтеся це зробити. Підкріплюйте свою відповідь конкретними прикладами з вашого досвіду. Наприклад, розкажіть, які навички ви мали змогу розвинути на ваших

попередніх місцях роботи, чим ви там займалися. HR повинен зрозуміти, чому у вас сформувалися ті чи інші кар'єрні плани і чи ви розумієте, які кроки потрібно зробити для реалізації задуманого.

Чим чіткіше і детальніше ви будете відповідати, і чим переконливіше звучатиме ваша відповідь, тим більше виникне довіри до вас, і HR-фахівець зрозуміє, що ви дійсно плануєте свою кар'єру.

3. Чому ви покидаєте попереднє місце праці? Роботодавець хоче зрозуміти ваші критерії пошуку нової роботи і те, наскільки успішно ви справляєтеся зі складними обов'язками, навіть якщо в компанії вам щось не подобається. Відповідаючи на це запитання, ні в якому разі не потрібно розповідати про всі деталі, звинувачуючи або критикуючи свого попереднього роботодавця. Це тільки відштовхне вашого потенційного наймача. Якщо ви мали конфлікт з керівництвом або вам не подобалися певні умови роботи, то сформулюйте це максимально коректно. Наприклад: «Мені було важко працювати без конкретних термінів і чітко поставлених завдань». Як правило, HR-и великих компаній знають один одного, і орієнтуються, що відбувається в тій чи іншій компанії. Неправда може легко розкритися, а вашу коректність гідно оцінять.

Якщо ж ви покидаєте попереднє місце роботи через відсутність можливостей росту, то розкажіть, яким чином ви хочете розвиватися в професійному плані, зробивши акцент на тому, як нова робота може допомогти вам в цьому. Для цього уважно проаналізуйте текст вакансії і знайдіть інформацію про кар'єрні можливості компанії.

Відповідаючи на запитання, будьте обережні у висловлюваннях - не вигадуйте те, чого не було. Досвідчений HR обов'язково запитає, яких заходів ви вжили для усунення причини свого відходу з попередньої компанії. Наприклад, чи прагнули ви налагодити відносини з керівником (колегою), чи обговорювали з ним ваші перспективи в компанії. У вас повинна бути конкретна відповідь. Нечіткі формулювання можуть викликати побоювання,

що ви так само вчините і наступного разу - просто підете мовчки, нічого не пояснивши.

4. Які помилки (невдачі, критичні ситуації) у вас були на минулій роботі? Це питання - перевірка на чесність і адекватність. Роботодавець не прагне дізнатися, скільки помилок ви припустили. Набагато важливіше - чи готові ви в цьому зізнаватися. Якщо ви можете вільно говорити про свої невдачі - значить, вам можна довіряти і, швидше за все, у відносинах з колегами і керівництвом ви теж будете відверті і «прозорі». Чого робити точно не варто - це говорити, що у вас не було помилок або невдач. Для HR-а це тривожний знак: або людина обманює, або вона поганий працівник, якому не довіряли ніяких відповідальних завдань.

Не варто розповідати про всі провали в деталях. Але поділитися однією з неприємних історій доведеться. Тому заздалегідь продумайте, про який «складний» досвід ви можете розповісти. Також обов'язково зробіть акцент на тому, як ви вийшли з проблемної ситуації і які зробили висновки на майбутнє.

5. Чим найбільше пишаєтеся у вашій роботі? Роботодавець хоче зрозуміти, чи дійсно людина захоплена своєю справою. Ви повинні назвати всі свої професійні досягнення, якими пишаєтеся. Якщо ви скажете на співбесіді, що не можете пригадати нічого підходящого - у HR-а тут же виникне кілька підозр. По-перше - вам дійсно нічого показати, і ви просто пасивно виконували поточні завдання. По-друге - вам, швидше за все, не подобається те, чим ви займаєтеся, а значить і повної віддачі від вас очікувати не варто.

6. Чому хочете працювати саме у нас? Працівник відділу кадрів хоче дізнатися, якою інформацією про його компанію ви володієте. Так він зрозуміє, чи готувались ви до співбесіди і наскільки серйозний і свідомий ваш вибір. Роботодавець хоче переконатися, що людина дійсно зацікавлена у вакансії і не покине роботу, як тільки отримає вигіднішу пропозицію.

Відповідаючи, продемонструйте свої знання про компанію, покажіть, що ви добре підготувалися. Для цього ретельно вивчіть сайт компанії, дізнайтеся

все про її проекти, корпоративну культуру, соціальні ініціативи. Якщо на співбесіді ви розповісте, чим саме вам сподобався той чи інший проект або ініціатива, відповідь буде звучати переконливо.

Ще здобувачеві не завадить почитати інтерв'ю з топ-менеджерами і останні новини про роботодавця в ділових виданнях. Так ви будете знати, як компанія себе «почуває» на ринку в даний момент. А якщо ще вам вдасться використовувати якісь нетривіальні факти про компанію (наприклад, вони викладені лише в певній статті, яка опублікована на англійській мові), то це буде великий плюс для вас.

На основі зібраної інформації дайте собі відповідь на такі запитання: які для вас ключові фактори вибору роботи і чому; які важливі можливості може запропонувати вам саме цей роботодавець; що цінує і шукає компанія в кандидатах, і що з цього маєте ви?

7. На яку зарплату ви розраховуєте? Це питання задають для того, щоб дізнатися, наскільки оцінює себе здобувач на ринку праці, і чи адекватна його оцінка.

Щоб дати коректну відповідь, заздалегідь перегляньте сайти з працевлаштування, прочитайте огляди зарплат, щоб ваша «цінова пропозиція» була адекватною. Якщо ви ніяковієте, говорячи про гроші, або питаєте, скільки вам готові запропонувати, це означатиме, що ви просто торгуєтесь, самі не знаючи, скільки вартуєте на ринку праці.

Обов'язково будьте готові пояснити, чому називаєте ту чи іншу суму, підкріплюючи свою відповідь описом ваших сильних якостей і аналітикою ринку праці. Також будьте готові до прямого питання про те, скільки ви заробляєте зараз, не забуваючи, що цю інформацію можна легко перевірити.

Відповідайте на всі питання максимально конкретно. На співбесіді ви повинні демонструвати впевненість в собі. Тому сидіть прямо, поклавши руки на стіл, дивіться на свого співрозмовника під час бесіди. Так роботодавець буде бачити, що ви вмієте вести діалог, володіти емоціями. Впевненість

повинна бути і в ваших відповідях. Не використовуйте займенник «ми» («ми зробили ...», «ми розробили ...»). Дуже добре, якщо ви - командний гравець, але на співбесіді роботодавець хоче зрозуміти, що можете саме ви, а не команда, в якій ви працювали. Тому якомога частіше у співбесіді вживайте займенник «я», розповідайте про свій досвід.

Проявляйте активний позицію. Напередодні співбесіди подумайте, що ви могли б запропонувати компанії, якщо б вас взяли на роботу. Обов'язково озвучте свої ідеї на співбесіді. Їх навіть можна систематизувати в невелику презентацію. Навіть якщо це будуть не найкращі пропозиції - роботодавець оцінить ваші старання.

Співбесіда - це діалог, а не монолог роботодавця або претендента. І ви повинні вести діалог на рівних. Мова не тільки про те, щоб добре відповідати на питання, а й самому їх задавати. Вас адже цікавить, в чому полягатиме ваша робота, яка організаційна структура компанії, цілі? Швидше за все, так, якщо ви хочете не просто працевлаштуватися, а працювати там, де вам буде комфортно. Логіка HR-а така: якщо людина ставить запитання, то вона зацікавлена в роботі, якщо немає - то навряд чи вона серйозно розглядає працевлаштування в дану компанію. Тому заздалегідь продумайте, про що ви можете запитати.

Говоріть правду. Може, це і звучить банально, але будь-яка брехня може розкритися дуже несподівано для вас. Наприклад, може бути так, що ваші колеги вже працюють в тій компанії, куди ви хочете потрапити, або HR-и компаній добре знайомі. Ви, звичайно, можете щось прикрашати в процесі співбесіди, але обманювати категорично заборонено.

Часто HR-и спеціально роблять невірні висновки зі слів претендента або ставлять провокаційні запитання, щоб зачепити людину і подивитися на її реакцію. Головне - розуміти, що вас перевіряють і реагувати спокійно.

Будьте готові, що на співбесіді вас можуть попросити виконати тестове завдання або вирішити якесь завдання (загадку, головоломку). Головне - не

панікуйте і не відмовляйтеся виконувати завдання. В даному випадку для роботодавця важливо не тільки те, як ви впораєтеся, але і вашу поведінка в стресовій і критичної ситуації. Ваша реакція розповість йому більше, ніж будь-яка відповідь на питання.

Контрольні питання

1. У чому пролягає важливість резюме?
2. Яка різниця між резюме та CV (Curriculum Vitae)?
3. Який, на вашу думку, оптимальний розмір резюме?
4. Які основні вимоги до резюме?
5. Чи варто вказувати у резюме такі особисті дані, як сімейний статус, стать та наявність дітей?
6. Коли доречно додавати до резюме своє фото?
7. Чому у резюме не варто детально описувати свій кар’єрний шлях?
8. Які запитання найчастіше задають на співбесідах?
9. Як правильно підготуватись до співбесіди?

ТЕСТОВІ ЗАВДАННЯ ДЛЯ САМОПЕРЕВІРКИ ЗНАНЬ

Подано типові тестові завдання для самоперевірки знань. До кожного тесту задається набір з п’яти варіантів відповіді, з яких лише один правильний.

Тести можуть бути використані як для підготовки до поточного, так і до семестрового контролю.

1. Що не входить до основних етапів життєвого циклу ПЗ?

А) документування ПЗ;

Б) супровід ПЗ;

В) проектування ПЗ;

Г) розроблення ПЗ;

Д) усі відповіді правильні.

2. Які етапи належать до неосновних в життєвому циклі ПЗ?

А) оцінка якості ПЗ;

Б) проведення тренінгів для розробників ПЗ;

В) створення робочих місць;

Г) оцінка ризиків виконання проекту;

Д) усі відповіді правильні.

3. Функціональною вимогою до програмного забезпечення може бути:

А) ПЗ повинне забезпечити безперебійну роботу 1000 користувачів;

Б) над проектом повинні працювати лише 10 програмістів та 3 тестувальники;

В) проект повинен бути виконаний до 1 січня 2020 року;

Г) щодня надсилати замовнику звіт про виконання роботи розробниками проекту;

Д) немає правильної відповіді.

4. Технічне завдання є результатом етапу ЖЦ:

А) документування ПЗ;

Б) проектування ПЗ;

В) визначення та аналізу вимог ПЗ ;

Г) роботи з замовником;

Д) немає правильної відповіді.

5. Для опису вимог не можуть використовуватися такі види документів:

А) технічне завдання;

Б) прототип ПЗ;

В) блок-схема ПЗ;

Г) діаграма прецедентів;

Д) усі відповіді правильні.

6. Чи можна вносити зміни до технічного завдання?

А) можна завжди;

Б) не можна;

В) можна лише на початку кожного етапу ЖЦ ПЗ;

Г) можна лише за згоди замовника;

Д) немає правильної відповіді.

7. У якому вигляді не можуть бути описані вимоги до ПЗ?

А) графічному;

Б) табличному;

В) текстовому;

Г) словесно-усному;

Д) немає правильної відповіді.

8. Які види діяльності розрізняють під час роботи з вимогами:

А) документування вимог;

Б) аналіз узгоджених із замовником вимог ;

В) оцінка зрозумілості вимог для майбутньої реалізації;

Г) виявлення джерел вимог;

Д) усі відповіді правильні.

9. Вимоги необхідно документувати для:

А) програмістів, які на їхній основі проектують ПЗ;

Б) оцінки коштів розроблення ПЗ ;

В) оцінки термінів розроблення ПЗ;

Г) уникання непорозумінь із замовником;

Д) усі відповіді правильні

10. Організаційні процеси ЖЦ ПЗ

А) діють протягом усього ЖЦ;

Б) пов’язані із додатковими етапами ЖЦ ;

В) належать до основних процесів ЖЦ;

Г) завжди присутні під час розроблення ПЗ;

Д) немає правильної відповіді.

11. Вимога стосовно супроводжуваності програми означає:

А) скільки розробників бере участь в проекті;

Б) як швидко буде розроблена програма;

В) як швидко буде працювати програма;

Г) як довго буде використовуватися програма;

Д) немає правильної відповіді.

12. Модель життєвого циклу ПЗ:

А) зображає порядок виконання етапів ЖЦ ПЗ;

Б) є вхідними даними для написання вимог до ПЗ;

В) будується на етапі проектування ЖЦ ПЗ;

Г) обов’язково містить усі етапи ЖЦ ПЗ;

Д) немає правильної відповіді.

13. Процес життєвого циклу ПЗ:

А) певна множина дій і задач;

Б) призводить до значимого результату;

В) є складовою частиною ЖЦ ПЗ;

Г) іноді є синонім до етапу ЖЦ ПЗ;

Д) усі відповіді правильні.

14. Термін «життєвий цикл » у галузі ІПЗ

А) є оригінальним;

Б) запозичено з технічних наук;

В) запозичено з біології;

Г) є невідомого походження;

Д) немає правильної відповіді.

15. Визначення вимог відбувається

А) інтегрально;

Б) інтерактивно;

В) лише за вимогою замовника;

Г) лише при потребі зі сторони розробників;

Д) немає правильної відповіді.

16. Вимога стосовно ефективності програми означає:

А) чи правильно працює програма;

Б) чи програмою зручно користуватися;

В) як співвідносяться її вартість й якість;

Г) як довго буде використовуватися програма;

Д) немає правильної відповіді.

17. Які види діяльності можуть бути віднесені до етапу проектування:

А) розроблення алгоритму;

Б) розроблення прототипу графічного інтерфейсу;

В) створення блок-схем;

Г) проектування структур даних;

Д) усі відповіді правильні.

18. Основна мета етапу проектування:

А) створити блок-схеми;

Б) розробити архітектуру ПЗ;

В) розробити окремі компоненти ПЗ;

Г) проаналізувати технічне завдання;

Д) немає правильної відповіді.

19. Які види діяльності відбуваються на етапі кодування ЖЦ ПЗ?

А) написання коментарів;

Б) написання модулів;

В) реалізація алгоритмів;

Г) налагодження тексту програми;

Д) усі відповіді правильні.

20. Архітектура ПЗ передбачає формування

А) структурного вигляду програми;

Б) поведінкового вигляду програми;

В) фізичного вигляду програми;

Г) логічного вигляду програми;

Д) усі відповіді правильні.

21. Патерн проектування – це

А) фрагмент програмного коду програми;

Б) план робіт проекту;

В) схема розподілу компонентів системи за вузлами;

Г) строгий опис поведінки програми;

Д) немає правильної відповіді.

22. Результатами етапу проектування є:

А) зображення компонент ПЗ і взаємозв’язків між ними;

Б) вибір алгоритмів реалізації деякої функціональності;

В) вибір шаблонів ПЗ для розв’язання певних підзадач;

Г) вибір і обґрунтування структур даних для реалізації ПЗ;

Д) усі відповіді правильні.

23. Проектування « top-level design» має завданням –

А) визначити поведінку системи;

Б) визначити структуру системи;

В) визначити алгоритми розв’язку задачі;

Г) розробити графік робіт проекту;

Д) немає правильної відповіді.

24. Проектування належить до категорії/групи процесів:

А) допоміжні;

Б) інженерні основні;

В) інженерні неосновні;

Г) замовник-постачальник;

Д) немає правильної відповіді.

25. Проектування «software detailed design» має завданням –

А) визначити поведінку системи;

Б) визначити структуру системи;

В) визначити алгоритми розв’язку задачі;

Г) розробити графік робіт проекту;

Д) немає правильної відповіді.

26. Архітектура ПЗ передбачає формування

А) плану робіт;

Б) команди розробників;

В) програмного коду;

Г) специфікації вимог;

Д) немає правильної відповіді.

27. Рефакторинг коду – це

А) покращення програмного коду;

Б) створення програмного коду;

В) перевірка програмного коду на оптимальність;

Г) інтеграція модулів системи;

Д) немає правильної відповіді.

28. Як можна описати структуру програмної системи?

А) блок-схемою алгоритму роботи програми;

Б) інструкцією користувача;

В) прототипом інтерфейсу користувача;

Г) складовими модулями;

Д) усі відповіді правильні.

29. Яке тестування полягає в перевірці коду без виконання програми?

А) статичне;

Б) динамічне;

В) системне;

Г) бета-тестування;

Д) немає правильної відповіді.

30. Яке тестування передбачає внесення дрібних помилок у програмний код?

А) системне;

Б) навантажувальне;

В) динамічне;

Г) мутаційне;

Д) усі відповіді правильні.

31. Яке тестування спрямоване на перевірку безпеки та зручності використання?

А) димове;

Б) нефункціональне;

В) бета-тестування;

Г) санітарне;

Д) немає правильної відповіді.

32. Яке тестування проводиться «зовнішніми» користувачами?

А) димове;

Б) регресійне;

В) бета-тестування;

Г) функціональне;

Д) немає правильної відповіді.

33. За рівнем тестування розрізняють:

А) модульне;

Б) системне;

В) інтеграційне;

Г) приймальне;

Д) усі відповіді правильні.

34. Тестування окремих найменших частин програми називають:

А) модульним;

Б) «чорної» скриньки;

В) санітарним;

Г) альфа-тестуванням;

Д) усі відповіді правильні.

35. Яке тестування передбачає побудову графа управління?

А) мутаційне;

Б) димове;

В) бета-тестування;

Г) «чорної» скриньки;

Д) немає правильної відповіді.

36. Доступ до вихідного коду програми передбачається у:

А) структурному тестуванні;

Б) мутаційному тестуванні;

В) тестуванні на основі потоку даних програми;

Г) тестуванні «скляної» скриньки;

Д) усі відповіді правильні.

37. Яке з цих видів тестування проводиться першим?

А) димове;

Б) регресійне;

В) санітарне;

Г) альфа-тестування;

Д) усі відповіді правильні.

38. Яке тестування проводиться перед здачею в експлуатацію?

А) модульне;

Б) системне;

В) інтеграційне;

Г) приймальне;

Д) усі відповіді правильні.

39. Оберіть правильні твердження:

А) Якісне тестування однозначно доводить відсутність помилок.

Б) Про тестування слід думати лише , коли завершений етап кодування.

В) Тестування – це те саме, що відлагодження.

Г) Важливою є не стільки кількість, скільки якість підібраних тестових наборів.

Д) Усі відповіді правильні.

40. Яке з видів тестування відноситься до структурного тестування?

А) функціональне

Б) мутаційне

В) димове

Г) бета-тестування

Д) немає правильної відповіді.

41. Використання даних, значення яких знаходяться за межами допустимої області змін, передбачене у:

А) перевірці нормальних умов

Б) перевірці у виняткових ситуаціях

В) перевірці екстремальних умов

Г) перевірці граничних умов

Д) немає правильної відповіді.

42. Інтеграційне тестування має за мету:

А) перевірку правильності роботи окремих компонент системи;

Б) перевірку правильності роботи системи в цілому;

В) перевірку правильності інтерфейсу програми;

Г) перевірку правильності обчислень інтегралів;

Д) немає правильної відповіді.

43.Тестування належить до категорії/групи процесів:

А) допоміжні;

Б) інженерні основні;

В) інженерні неосновні;

Г) замовник-постачальник;

Д) немає правильної відповіді.

44. Якого тестування не існує?

А) мутаційне;

Б) регресійне;

В) димове;

Г) автоматичне;

Д) немає правильної відповіді.

45. Етап супроводу ПЗ передбачає:

А) виправлення знайдених дефектів;

Б) додавання нового функціоналу;

В) реорганізацію програмного коду;

Г) використання програми для виявлення недоліків роботи програми;

Д) усі відповіді правильні.

46. Супровід належить до категорії/групи процесів

А) допоміжні;

Б) інженерні основні;

В) інженерні неосновні;

Г) замовник-постачальник;

Д) немає правильної відповіді.

47. Зручність користування програмою – це

А) основна функціональна вимога до будь-якого ПЗ;

Б) складова супроводжуваності ПЗ;

В) складова ефективності роботи програми;

Г) складова якості ПЗ;

Д) усі відповіді правильні.

48. Якісна характеристика ПЗ супроводжуваність означає, що:

А) програму можна зручно використовувати;

Б) програма немає дефектів;

В) програму можна легко модифікувати;

Г) програму можна використовувати на різних платформах;

Д) усі відповіді правильні.

49. До принципів модульного програмування не належить:

А) виділення фрагментів в модулі;

Б) повторне використання модулів

В) створення бібліотеки модулів;

Г) написання коментарів у тексті програми;

Д) організація інтерфейсу модуля

50. До принципів структурного програмування належить

А) створення бібліотеки модулів;

Б) висхідне проектування

В) дисципліна проектування і розроблення ;

Г) модульне кодування

Д) розроблення інтерфейсів користувача

51. Виберіть чого насамперед стосуються ознаки кризи інженерії програмного забезпечення:

А) вартість створення ПЗ, термін створення ПЗ, професійний рівень розробників ПЗ;

Б) вартість створення ПЗ, вартість апаратного забезпечення, вартість використання ПЗ;

В) вартість використання ПЗ, невідповідність ПЗ рівню апаратного забезпечення;

Г) вартість створення ПЗ, термін створення ПЗ, функціональність ПЗ;

Д) немає правильної відповіді .

52. CASE-засіб використовується для

А) автоматичного навчання користувачів програмного забезпечення;

Б) пошуку замовників програмних продуктів;

В) автоматизованого розроблення програмного забезпечення;

Г) генерації веб-сайтів;

Д) немає правильної відповіді

53. Ієрархічна декомпозиція життєвого циклу ПЗ означає

А) обов’язковість виконання основних етапів ;

Б) застосування допоміжних процесів;

В) поділ на основні етапи;

Г) існування складових, які деталізуються іншими елементами;

Д) усі відповіді правильні.

54. Вибрати справедливе твердження для каскадної моделі:

А) розроблення тестів з обов’язковим залученням замовника;

Б) точне планування термінів виконання та витрат;

В) сформульовані вимоги у формі технічного завдання можуть змінюватися, уточнюватися замовником;

Г) ітераційний характер;

Д) легке застосування для складних проектів

55. Вибрати справедливе твердження для спіральної моделі:

А) замовник визначає необхідність наступного циклу розроблення ;

Б) зміна вимог не передбачається;

В) точне планування термінів виконання та витрат;

Г) строго послідовний характер розроблення;

Д) історично перша модель описана в інженерії програмного забезпечення

56. Як називається спосіб поведінки людей відповідно до прийнятих норм і позиції в суспільстві?

А) статус;

Б) роль;

В) рольове очікування;

Г) рольова невизначеність;

Д) немає правильної відповіді.

57. Як називається місце суб’єкта в системі міжособистісних стосунків, що визначають його права, обов’язки і привілеї?

А) статус;

Б) роль;

В) рольове очікування;

Г) рольова невизначеність;

Д) немає правильної відповіді.

58. За яким критерієм групи поділяються на умовні і реальні:

А) за безпосередністю зв’язків;

Б) за розміром;

В) за суспільним статусом;

Г) за рівнем розвитку;

Д) немає правильної відповіді.

59. До якої групи людина добровільно себе зараховує і знаходить для себе зразки для наслідування?

А) референтної;

Б) нереферентної;

В) антиреферентної;

Г) групи належності;

Д) немає правильної відповіді.

60. За яким критерієм групи поділяються на формальні і неформальні?

А) за безпосередністю зв’язків;

Б) за значенням для індивіда;

В) за суспільним статусом;

Г) за рівнем розвитку;

Д) немає правильної відповіді.

61. За яким критерієм групи поділяються на референтні групи і групи належності?

А) за безпосередністю зв’язків;

Б) за значенням для індивіда;

В) за суспільним статусом;

Г) за рівнем розвитку;

Д) немає правильної відповіді.

62. До кого типу груп належить студентська група, в якій ви навчаєтесь?

А) реальна;

Б) умовна;

В) дифузна;

Г) неофіційна;

Д) усі відповіді правильні.

63. До скількох малих груп належить кожна людина?

А) до однієї;

Б) до кількох;

В) до багатьох;

Г) може не належати до жодної групи;

Д) усі відповіді правильні.

64. Яка з перерахованих груп є високорозвиненою групою ?

А) асоціація;

Б) корпорація;

В) колектив;

Г) дифузна група;

Д) усі відповіді правильні.

65. На яких стадіях розвитку групи немає конфлікту?

А) створення групи;

Б) поділ на підгрупи;

В) конфронтація;

Г) розробка норм;

Д) співпраця;

Е) розпад групи.

66. Які з перерахованих командних ролей відносяться до цільових ролей?

А) «генератор ідей»;

Б) «душа команди»;

В) «контролер-фінішер»;

Г) «король»;

Д) немає правильної відповіді.

67. Які з перерахованих командних ролей відносяться до негативних ролей?

А) «генератор ідей»;

Б) «душа команди»;

В) «красуня»;

Г) «король»;

Д) немає правильної відповіді.

68. Які з перерахованих командних ролей відносяться до підтримуючих?

А) «генератор ідей»;

Б) «душа команди»;

В) «красуня»;

Г) «організатор»;

Д) немає правильної відповіді.

69. Як називається командна роль людини, яка претендує на роль «найкращого працівника»?

А) «король»;

Б) «кавалер»;

В) «красуня»;

Г) «організатор»;

Д) немає правильної відповіді.

70. Як називається командна роль людини, яка у всьому підтримує лідера?

А) «король»;

Б) «кавалер»;

В) «красуня»;

Г) «блазень»;

Д) немає правильної відповіді.

71. Як називається командна роль людини, яка звикла завжди бути в центрі уваги?

А) «король»;

Б) «кавалер»;

В) «красуня»;

Г) «блазень»;

Д) немає правильної відповіді.

72. Хто з членів групи пропонує нові підходи, нетрадиційні рішення проблем?

А) «контролер-фінішер»;

Б) «генератор ідей»;

В) «король»;

Г) «стратег»;

Д) немає правильної відповіді.

73. Хто з членів групи керує діяльністю групи, скеровує її в необхідне русло?

А) «король»;

Б) «генератор ідей»;

В) «голова»;

Г) «стратег»;

Д) немає правильної відповіді.

74. Хто з членів групи підтримує стосунки з зовнішнім середовищем?

А) «комунікатор-добувач інформації» ;

Б) «генератор ідей»;

В) «голова» ;

Г) «контролер-фінішер»;

Д) немає правильної відповіді.

75. Хто з членів групи змушує дотримуватись планів і відповідної якості в роботі?

А) «контролер-фінішер»;

Б) «лідер»;

В) «голова»;

Г) «стратег»;

Д) немає правильної відповіді.

76. Які компоненти містить модель комунікаційного процесу?

А) канал;

Б) інформація;

В) реципієнт;

Г) джерело;

Д) усі відповіді правильні.

77. Особа, яка передає інформацію - це

А) фасилітатор;

Б) комунікатор;

В) реципієнт;

Г) космополіт;

Д) немає правильної відповіді.

78. Особа, яка отримує інформацію - це

А) фасилітатор;

Б) комунікатор;

В) реципієнт;

Г) космополіт;

Д) немає правильної відповіді.

79. Як називається вміння співпереживати іншій людині, здатність відчувати те ж, що й співрозмовник?

А) атракція;

Б) емпатія;

В) симпатія;

Г) антипатія;

Д) немає правильної відповіді.

80. Як називається вміння співпереживати іншій людині, здатність відчувати те ж, що й співрозмовник?

А) атракція;

Б) емпатія;

В) симпатія;

Г) антипатія;

Д) немає правильної відповіді.

81. Оптимальний розмір резюме становить:

А) одну сторінку;

Б) дві сторінки;

В) три сторінки;

Г) 1-10 сторінок (залежно від попереднього стажу);

Д) усі відповіді правильні.

Покажчик основних термінів

належності	
нереферентна	
неформальна	
реальна	
референтна	
умовна	
формальна	
групова динаміка	
детальне проектування	
димовий тест	
динамічний аналіз коду	
емпатія	
ефективність ПЗ	
життєвий цикл ПЗ	
зв'язний список	
зручність інтерфейсу користувача	
інженерія програмного забезпечення	
інкапсуляція	
інкремент	
інтеграційне тестування	
інтегрованість	
інтерактивність	
інтерфейс користувача	
інтерфейс модуля	
інформаційна система	
інформаційні технології	
клас	
клас еквівалентності	

колектив	
комп'ютерна інженерія	
комп'ютерні науки	
комунікація	
вербальна	
невербальна	
коректна програма	
корпорація	
масив	
метафора системи	
модель ЖЦ ПЗ	
модуль	
модульне програмування	
модульне тестування	
мутаційне тестування	
надійна програма	
напівавтоматизоване тестування	
негативне тестування	
нефункціональна вимога	
нефункціональне тестування	
об'єкно-орієнтоване програмування	
об'єкт	
патерн проектування	
позитивне тестування	
поліморфізм	
приймальне тестування	
програмне забезпечення	
програмний продукт	

проект	
проектування ПЗ	
процес	
психологічні характеристики групи	
регресійне тестування	
резюме	
реінженерія	
рефакторинг коду	
рольова невизначеність	
рольове очікування	
рольовий перехід	
роль	
командна	
комплементарна	
комунікаційна	
негативна	
підтримуюча	
цільова	
ручне тестування	
санітарне тестування	
системна вимога	
системне тестування	
слухання	
нерефлексивне	
рефлексивне	
соціальне середовище	
соціальний контроль	
специфікація вимог	

СПИСОК ЛІТЕРАТУРИ

1. Alan Mark Davis Just Enough Requirements Management: Where Software Development Meets Marketing. - Dorset House, 2005.
2. Budiu R. Memory Recognition and Recall in User Interfaces. – [Електронний ресурс]. – Режим доступу: https://www.nngroup.com/articles/recognition-and-recall/.
3. Design applications for the Windows desktop. - [Електронний ресурс]. – Режим доступу: https://developer.microsoft.com/en-us/windows/desktop/design.
4. Google C++ Style Guide. [Електронний ресурс]. – Режим доступу: https://google.github.io/styleguide/cppguide.html
5. Graham D. Foundations of software testing. ISTQB certification / E. Veenendaal, I. Evans, R. Black. – Intl. Thomson Business Pr, 2007. – 243 p.
6. Guide to the Software Engineering Body of Knowledge: Edition – SWEBOKv3.0. IEEE, 2005. - [Електронний ресурс]. – Режим доступу: http://www4.ncsu.edu/~tjmenzie/cs510/pdf/SWEBOKv3.pdf
7. ISO/IEC 12207: 1995. Information technology - Software life cycle processes. Информационные технологии - Процессы жизненного цикла программного обеспечения.
8. ISO/IEC 9126, Information Technology - Software quality characteristics and metrics (Part 1-4).
9. ISO/IEC TR 15504, Information Technology - Software Process Assessment (Part 1-9).
10. IT професії [Електронний ресурс]. – Режим доступу- http://brainbasket.org/who-are-it-specialists-2/
11. Laubheimer P. Preventing User Errors: Avoiding Unconscious Slips. – [Електронний ресурс]. – Режим доступу: https://www.nngroup.com/articles/slips/.

12. Lauesen S. User Interface Design: A Software Engineering Perspective. Addison Wesley, 2004. – 624 pp.
13. Molich R., Nielsen J. Improving a human-computer dialogue // Communications of the ACM 33. – 1990. – N 3. – P. 338-348.
14. Nielsen J., Molich R. Heuristic evaluation of user interfaces // Proceedings of ACM CHI'90 Conf. (Seattle, WA, 1-5 April). – 1990. – P. 249-256.
15. Nilsen J. 10 Usability Heuristics for User Interface Design. – [Електронний ресурс]. – Режим доступу: https://www.nngroup.com/articles/ten-usability-heuristics/.
16. Resharper C++. Visual Studio Extension for C++ Developers. [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/resharper-cpp/#.
17. SWEBOK. Основные области знаний/ Островский А. Физико-технический учебно-научный центр НАН Украины [Електронний ресурс]. – Режим доступу: http://softandware.org.ua/wp-content/uploads/2014/11/base-areas.pdf
18. The Standish Group [Електронний ресурс]. – Режим доступу: https://www.standishgroup.com
19. Van Veendabl E. Standard glossary of term used in Software testing / E. Van Veendabl. - ISTQB. - 2007. Vol. 1,2.
20. Англо-український тлумачний словник з обчислювальної техніки, Інтернету, програмування. - К.: СофтПрес, 2006. – 823 с.
21. Андон П.И. Основы качества программных систем/ Андон П.И., Коваль Г.И., Коротун Т.М., Лаврищева Е.М., Суслов В.Ю. – К.: Академперіодика, 2007.– 860с.
22. Андрушків Б.М., Кузьмін О.Є. Основи менеджменту. - Львів: Світ, 1995.
23. Андрушків Б.М., Кузьмін О.Є. Основи менеджменту. - Львів: Світ, 1995. – 296 с.
24. Ахо Альфред. Структуры данных и алгоритмы/ Ахо Альфред, Хопкрофт Джон, Ульман Джеффри. - М.: Издательский дом «Вильямс», 2003.-384 с.

25. Бабенко Л. П. Основи програмної інженерії: Навчальний посібник/ Бабенко Л. П., Лавріщева К.М. – К.: Знання, 2001. – 269 с.
26. Бахтизин, В. В. Технология разработки программного обеспечения : учеб. пособие / В. В. Бахтизин, Л. А. Глухова. – Минск : БГУИР, 2010. – 267 с.
27. Бевз О. М. Проектування програмних засобів систем управління : навч. посіб. Ч. 1. Основи об'єктно-орієнтованого проектування / О. М. Бевз, В. М. Папінов, Ю. А. Скидан; Він. нац. техн. ун-т. - Вінниця, 2010. - 124 с.
28. Бейзер Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем / Б. Бейзер – СПб.: Питер, 2004. – 320 с.
29. Белбин М. Типы ролей в командах менеджеров: пер. с англ. - М. : HIPPO, 2003. - 232с.
30. Белбин М. Типы ролей в командах менеджеров: пер. с англ. - М.: HIPPO, 2003. - 232с.
31. Бенеш Г. Психологія: dtv-Atlas: Довідник: Пер. з нім. - К.: Знання-Прес, 2007. - 510 с.
32. Бенеш Г. Психологія: dtv-Atlas: Довідник: пер. з нім. – К.: Знання-Прес, 2007. – 510 с.
33. Білас О. Якість програмного забезпечення та тестування / О.Білас – Нац. Ун-т «Львів. Політехніка». – Л.: Вид-во Нац. Ун-ту «Львів. Політехніка», 2011. – 214 с.
34. Браунси К. Основные концепции структур данных и реализация в C++/ Браунси К. – М.: Изд. Дом «Вильямс», 2002. – 320с.
35. Вакалюк Т.А. Технології тестування програм. Навч. Посібник / Т.А. Вакалюк – Житомир, Вид-во ЖДУ, 2013. – 96 с.
36. Введение в программную инженерию и управление жизненным циклом программного обеспечения = Guide to Software Engineering Base of Knowledge (SWEBOK): Пер. с англ. С.Орлик [Електронний ресурс]. - Режим доступу: sorlik.blogspot.com/

37. Вигерс К. И. Разработка требований к программному обеспечению/ Вигерс К. И. – Пер. с англ. –М.: Издательско-торговый дом «Русская редакция», 2004. – 576 с.
38. Вітенко І.С., Вітенко Т.І. Основи психології. - Вінниця: НОВА КНИГА, 2008. - 256 с.
39. Вітенко І.С., Вітенко Т.І. Основи психології. – Вінниця: НОВА КНИГА, 2008. – 256 с.
40. Гудлиф П. Ремесло программиста. Практика написания хорошего кода/ Гудлиф П. - Пер. с англ. - СПб.: Символ Плюс, 2009. - 704 с.
41. Джонсон Девід В. Соціальна психологія: тренінг спілкування: Пер. з англ. - К., 2003.
42. Джонсон Девід В. Соціальна психологія. Тренінг міжособистісного спілкування: пер. з англ. - К.:Академія, 2003. – 288 с.
43. Довідка Word [Електронний ресурс]. – Режим доступу: https://support.office.com/uk-ua/word
44. Довідка з програми Visio [Електронний ресурс]. – Режим доступу: https://support.office.com/uk-ua/article/Довідка-з-програми-Visio.
45. ДСТУ 4163-2003 Уніфікована система організаційно-розпорядчої документації. Вимоги до оформлювання документів.
46. Казмиренко В. П. Социальная психология организаций. - К.: МЗУУП, 1993. - 384 с.
47. Канер С. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений / Сэм Канер, Джек Фолк, Енг Нгуен – К.: Диасофт, 2001. – 544 с.
48. Коберн А. Сучасні методи опису функціональних вимог до систем/ Коберн А. - М .: Лорі, 2002.
49. Кознов Д.В. Введение в программную инженерию/ Д.В. Кознов - Изд-во СанктПетербургского ун-та, 2005 . - 154 с.

50. Коротєєва Т.О. Алгоритми і структури даних. Навчальний посібник / Т. О. Коротєєва. Львів : Видавництво Львівської політехніки, 2014. – 280 с.
51. Кулямин В. В. Формализация требований на практике/ Кулямин В. В., Пакулин Н. В., Петренко О. Л., Сортов А. А., Хорошилов А. В. - Препринт ИСП РАН 2005.- 50 с.
52. Кулямин В.В. Технология программирования. Компонентный подход/ Кулямин В.В. - М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2007. - 463 с.
53. Лаврищева К.М. Методы программирования. Теория, инженерия, практика/ К.М. Лавріщева. – К.; Наукова Думка, 2006.–451с.
54. Лавріщева К.М. Програмна інженерія / К.М. Лавріщева – К.: Видавничий дім «Академперіодика» НАН України, 2008. – 319 с.
55. Левус Є. Життєвий цикл програмного забезпечення: навч. посібник / Євгенія Левус, Тетяна Марусенкова, Оксана Нитребич – Львів: Видавництво Львівської політехніки, 2017. – 240 с.
56. Левус Є.В., Марусенкова Т.А. Життєвий цикл ПЗ: тестування. Методичні вказівки до виконання лабораторної роботи із дисципліни «Основи програмної інженерії» для студентів базового напряму «Програмна інженерія»/ Укл.: Є.В. Левус, Т.А. Марусенкова – «Львівська політехніка», 2016. – 26 с.
57. Леффінгуелл Д. Принципи роботи з вимогами до програмного забезпечення/ Леффінгуелл Д., Уідріг Д. - М .: Вільямс, 2002.
58. Липаев В.В. Программная инженерия. Методологические основы / В.В. Липаев — Высшая школа экономики, – М. : ТЕИС, 2006. – 608 с.
59. Макконнелл С. Совершенный код. Мастер-класс / Пер. с англ. - М.: Издательско-торговый дом "Русская Редакция" ; СПб.: Питер, 2007. - 896 с.
60. Мартин Р. Быстрая разработка программ: принципы, примеры, практика. Пер. с англ.—М.: Издательский дом “Вильямс”, 2004.—752 с.

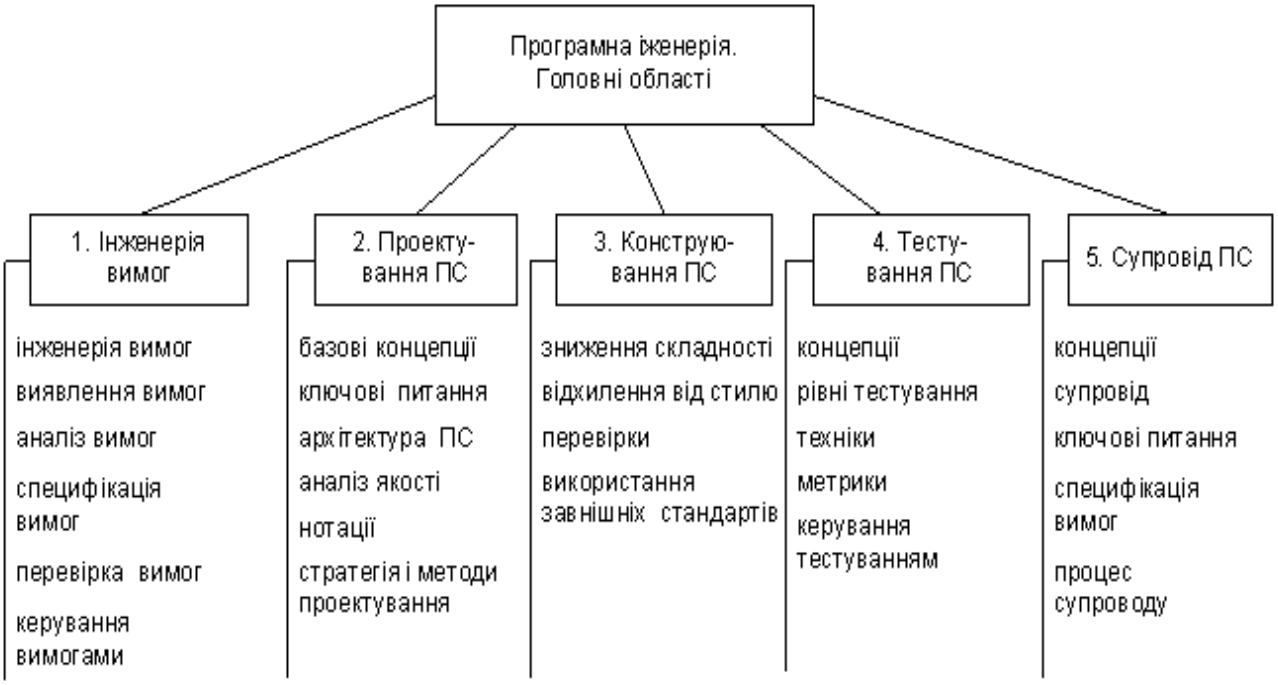
61. О`Коннор К. Основи ефективного лідерства: Пер. з англ. - К.: Британська Рада в Україні, 2000.
62. О'Коннор К. Основи ефективного лідерства: пер. з англ. - К.: Британська Рада в Україні, 2000. – 96 с.
63. Ойстер Кэрол. Социальная психология групп. - С.-Пб.: Прайм-ЕВРОЗНАК, 2004.- 224 с.
64. Ойстер Кэрол. Социальная психология групп. - С.-Пб.: Прайм-ЕВРОЗНАК, 2004. - 224 с.
65. Орбан-Лембрик Л.Е. Соціальна психологія : Підручник: У 2 кн. Кн. 2: Соціальна психологія груп. Прикладна соціальна психологія. - Київ: Либідь, 2006. - 558 с.
66. Орбан-Лембрик Л.Е. Соціальна психологія : Підручник: У 2 кн. Кн. 2: Соціальна психологія груп. Прикладна соціальна психологія. – Київ: Либідь, 2006. – 558 с.
67. Орлов С.А. Технологии разработки программного обеспечения/ Орлов С.А., Цилькер Б.Я.- 4-е изд. — СПб.: Питер, 2012. — 608 с.
68. Офісні пакети [Електронний ресурс]. - Режим доступу: http://programy.com.ua/ua/office_suite.
69. Павич Н.Я. Людино-машинна взаємодія: Конспект лекцій для студентів Інституту комп'ютерних наук та інформаційних технологій базового напряму 6.050103 Програмна інженерія/ Павич Н.Я. — Львів: Видавництво Львівської політехніки, 2013. – 100 с.
70. Поморова О.В. Проектування інтерфейсів користувача: навч. посібник/ О.В. Поморова, Т.О. Говорущенко. – Хмельницький: ХНУ, 2011. – 206 с.
71. Почепцов Г. Коммуникативные технологи двадцатого века. - М.: Рефл-бук; К.: Ваклер, 2000. - 352 с.
72. Почепцов Г. Коммуникативные технологи двадцатого века. - М.; К., 2000.
73. Роббинс Х., Финли М. Почему не работают команды? Что идет не так, и как это исправить. - М.: Добрая книга, 2005. - 304с.

74. Роббинс Х., Финли М. Почему не работают команды? Что идет не так, и как это справить: пер. с англ. - М.: Добрая книга, 2005. – 304с.
75. Савин Р. Тестирование DOT COM или пособие по жесткому обращению с багами в интернет-стартапах/ Савин Р. — М.: Дело, 2007. — 312 с.
76. Сидоров М.О. Вступ до програмної інженерії: конспект лекцій/ Сидоров М.О. - К.: НАУ, 2009.- 130с.
77. Скотт Б. Проектирование веб-интерфейсов/ Скотт Б., Нейл Т. – М.: Символ-плюс, 2010. – 352 с.
78. Соммервилл И. Инженерия программного обеспечения / Иан Соммервилл. – М.: Вильямс, 2002. – 624 с.
79. Тетерук И. Тестирование программного обеспечения - основные понятия и определения [Електронний ресурс] / Тетерук И., Булат А. - Портал знань - Режим доступу: http://www.znannya.org/?view=software-testing
80. Тидвелл Д. Разработка пользовательских интерфейсов / Д.Тидвелл. – СПб.: Питер, 2008. – 416 с.
81. У. Ройс. Управление проектами по созданию программного обеспечения. – М.: из-во «Лори», 2002, – 160 с.
82. Фетискин Н. П., Козлов В. В., Мануйлов Г. М. Социально-психологическая диагностика развития личности и малых групп. - М.,2002.
83. Фетискин Н. П., Козлов В. В., Мануйлов Г. М. Социально-психологическая диагностика развития личности и малых групп. - М.: Изд-во Института Психотерапии, 2002. – 490 с.
84. Цимбалюк І.М. Психологія: Навч. Посіб. - К.: ВД "Професіонал", 2006. - 576 с.
85. Цимбалюк І.М. Психологія: Навч. Посіб. – К.: ВД «Професіонал», 2006. – 576 с.

86. Шатовська Т. Б. Аналіз вимог до інформаційних систем. Лабораторний практикум [Електронний ресурс]. – Режим доступу: http://hire2.hzmk.com.ua/courses/AVPZ/003/content/content1.html.

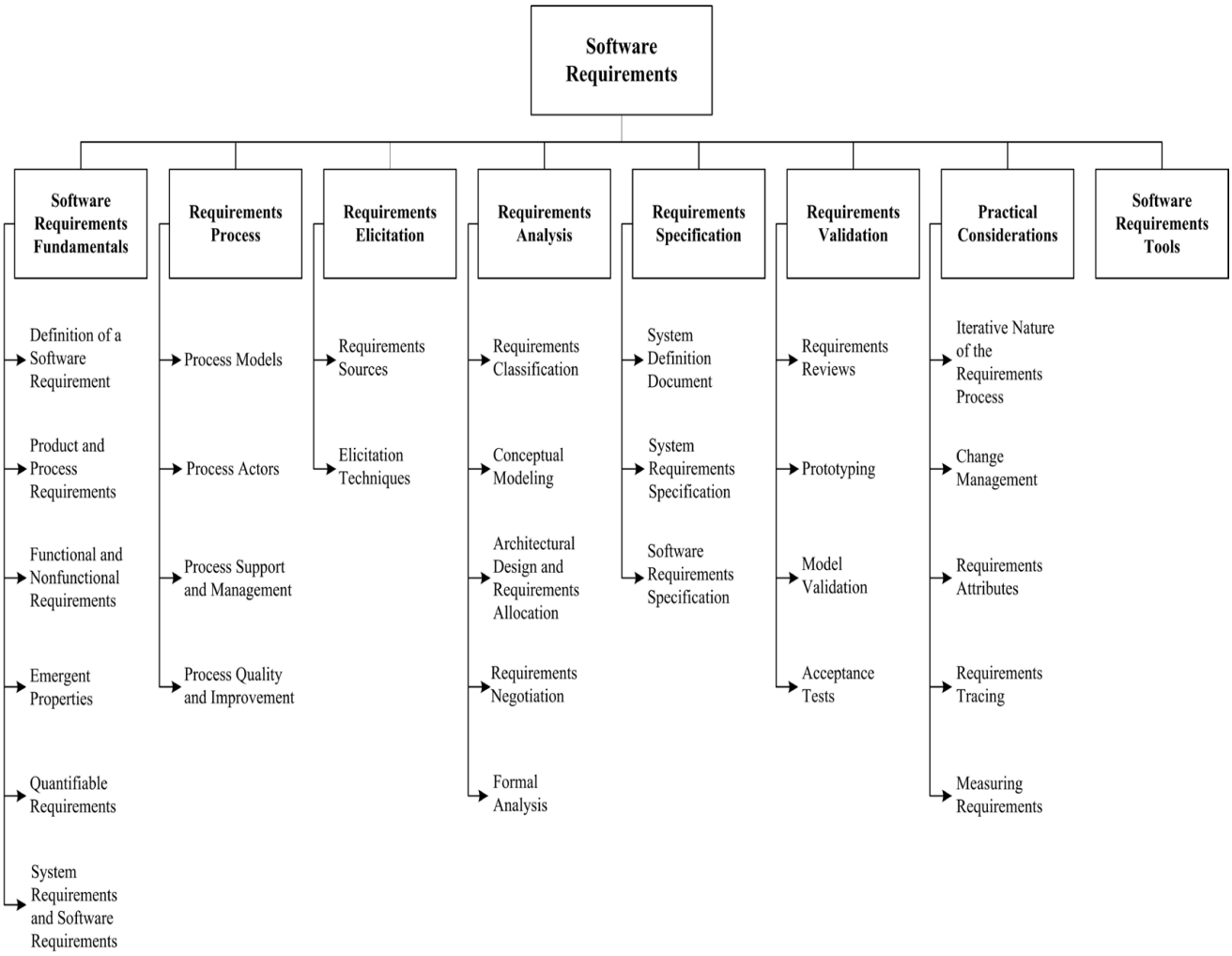
Додаток А.

Десять областей галузі знань інженерії програмного забезпечення згідно SWEBOK



Додаток Б.

Опис області знань Software Requirements (Програмні Вимоги) згідно SWEBOK



Додаток В.

Приклад сценарію використання

Замовлення літератури в читацькому залі бібліотеки

Головний сценарій (успішний)*:*

Користувач підходить до бібліотекара і надає йому замовлення у вигляді переліку видань. Бібліотекар заносить номер читацького квитка до системи і перевіряє облікову картку користувача. Бібліотекар шукає кожне видання в базі і визначає кількість екземплярів вільних у даний момент часу. Система видає шифри (коди), за якими бібліотекар знаходить видання у сховищі, заносить номери книг до картки користувача і видає йому літературу. При цьому читацький квиток лишається у бібліотекара.

Альтернативні сценарії:

1. Користувач бібліотеки є боржником, тому бібліотекар не може видати йому замовлену літературу.

2. Бібліотекар не може видати користувачу літературу, оскільки в даний момент часу немає вільних екземплярів у сховищі.

3. Читацький квиток користувача є недійсним. Бібліотекар вилучає його.

4. Технічний збій роботи системи. Видається повідомлення «немає зв’язку з сервером баз даних». Бібліотекар викликає адміністратора системи

Додаток Г.

Приклад діаграми варіантів використання

Інформаційна система «Бібліотека»

Коментар.

Інформаційна система бібліотеки передбачає два типи користувачів з розділеною функціональністю.

Додаток Д.

Складові специфікації вимог до програмного забезпечення (стандарт IEEE/ANSI 830-1993)

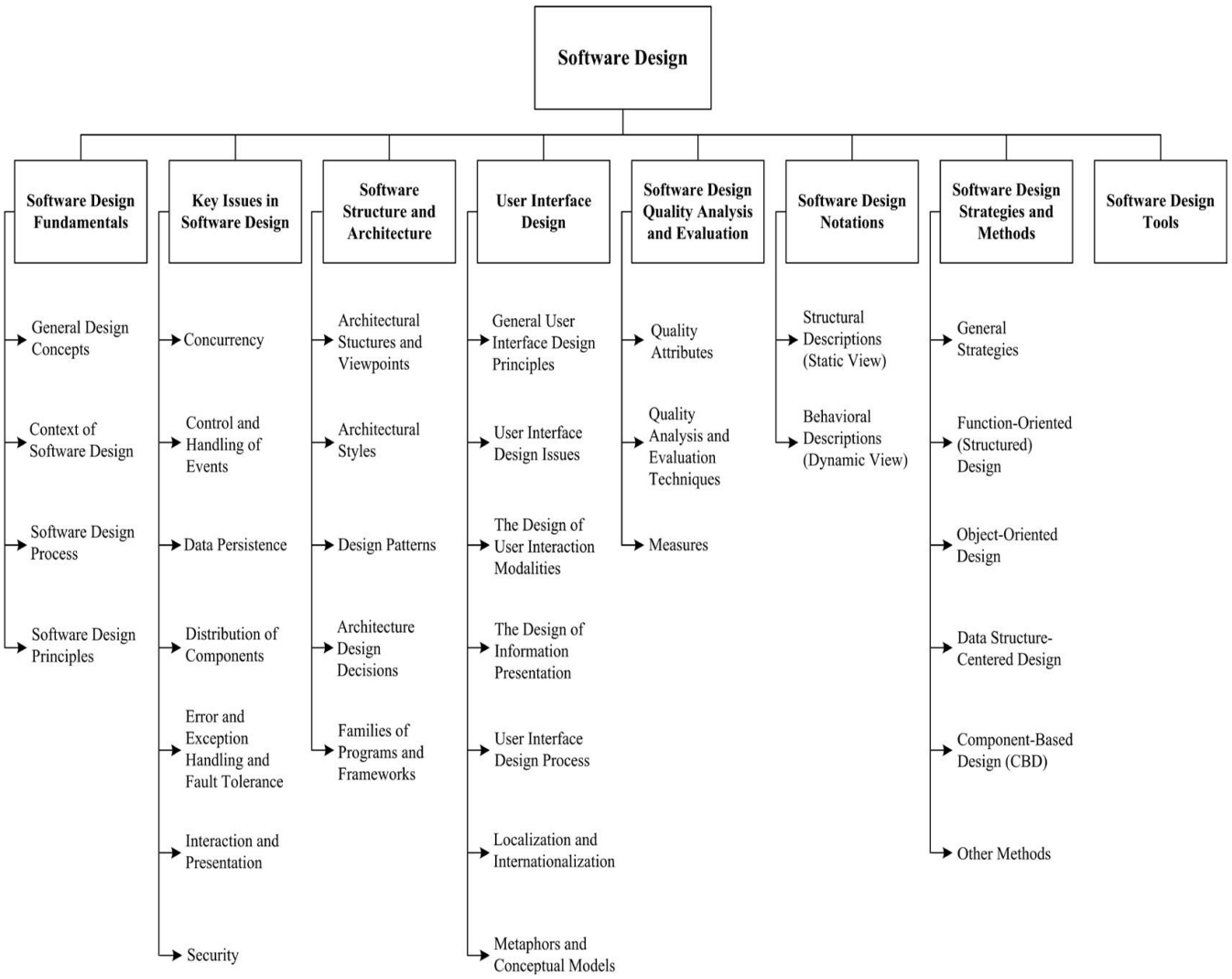
- **Передмова**
  - Особи, для яких розробляють документ;
  - Попередні версії та зміни внесені в ПЗ
  - Обґрунтування нової версії
- **Вступ**
  - Потреби в створенні системи
  - Системні функції та пояснення роботи із іншими системами
  - Пояснення, який ефект для організації після впровадження системи
- **Глосарій** – опис технічних термінів документу
- **Вимоги користувачів**
  - Опис сервісів (функцій), які надаються користувачу;
  - Не функціональні системні вимоги;
  - Стандарти на ПЗ та на процес створення;
- **Системна архітектура**
  - Високорівневе представлення архітектури із розподілом функцій по компонентах системи із виділенням існуючих компонент
- **Системні вимоги:** Функціональні та не функціональні
- **Системні моделі**
  - Показують взаємодію між компонентами та зовнішнім середовищем
  - Типи: моделі потоків даних; об'єктні; моделі даних
- **Еволюція системи**

- Припущення на яких базується система
- Зміна апаратних засобів, необхідна для реалізації системи
- Зміна системи відповідно до потреб користувачів в майбутньому

- **Додатки**
  - Опис апаратних засобів (мінімальна та максимальна конфігурація) та баз даних(логічна структура із якою працюватиме ПЗ) із якими працюватиме система

Додаток Е.

Опис області знань Software Design (Проектування Програмного Забезпечення) згідно SWEBOK



Додаток Ж.

Приклад патерну Factory Method (фабричний метод).

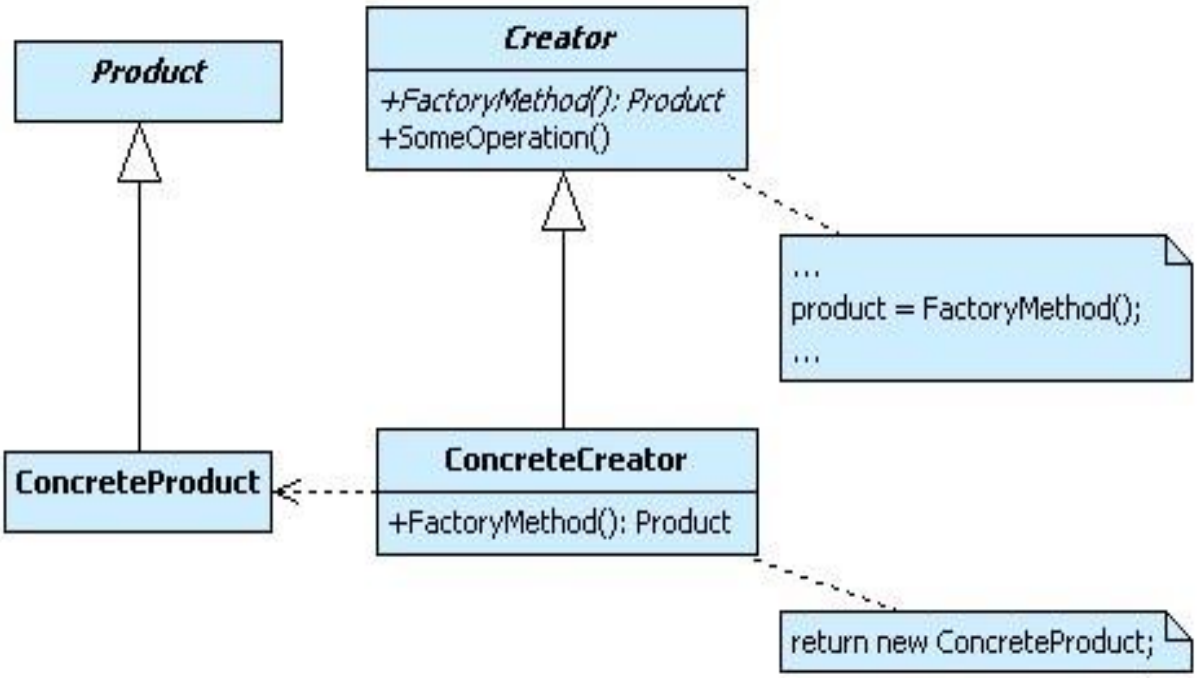
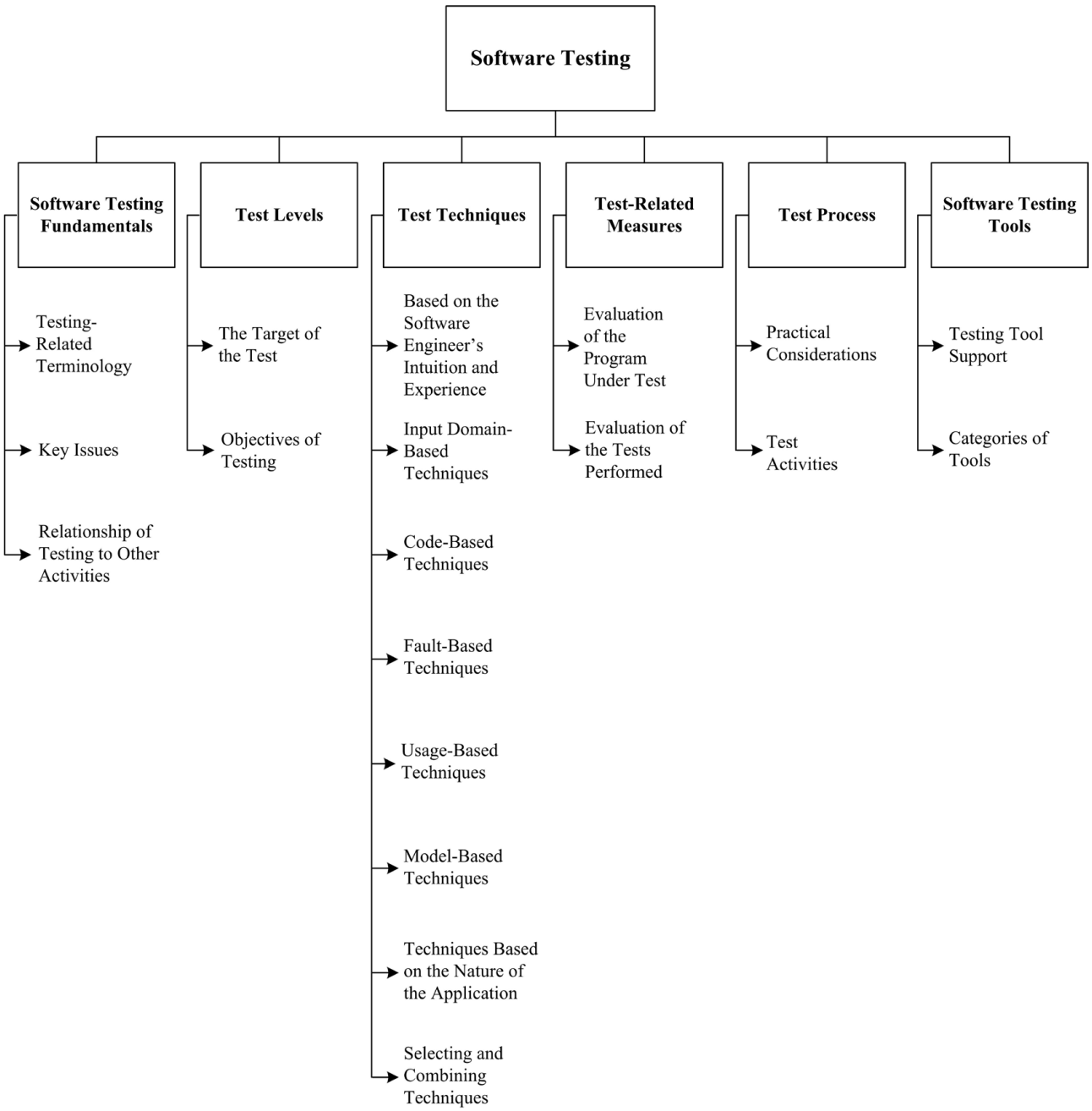


Рис.З.1. UML-діаграма класів патерна Factory Method (класична реалізація).

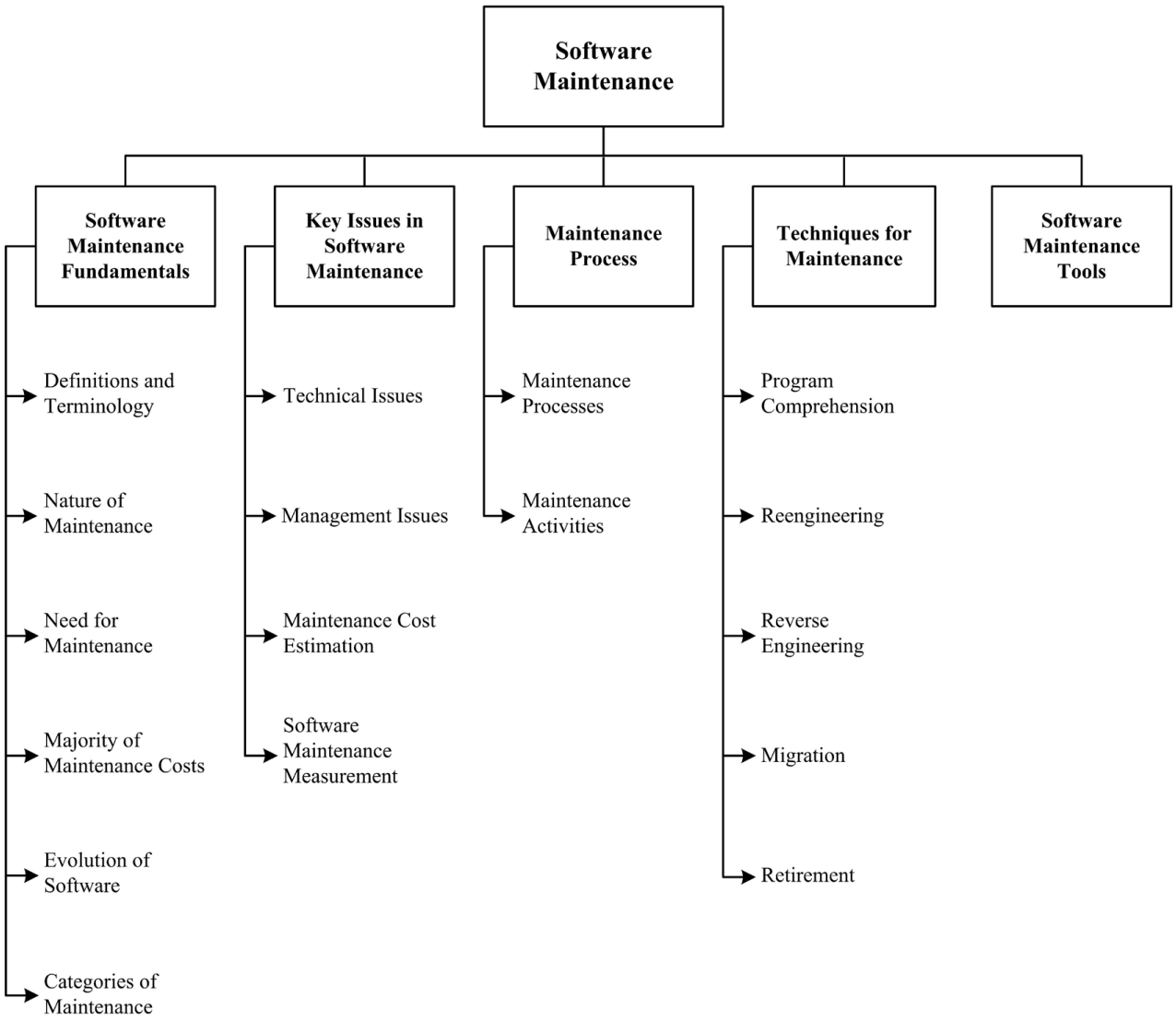
Додаток З.

Опис області знань Software Testing (Тестування Програмного Забезпечення) згідно SWEBOK



Додаток К.

Опис області знань Software Maintenance (Супровід Програмного Забезпечення) згідно SWEBOK



Додаток Л

Тест для оцінювання лідерських здібностей»

За 2-3 хвилини необхідно дати відповідь на кожне запитання у вигляді «Так», «Ні» або «Не знаю». Не слід витрачати багато часу на обмірковування запитань. Давайте ту відповідь, яка першою спала вам на думку.

Запитання тесту

1. Чи любите ви бути ватажком у компанії, очолювати групу, керувати і скеровувати інших?
2. Чи любите ви описувати людину, яка вам подобається?
3. Чи прагнете ви навчати людей правилам і порядку?
4. Чи любите ви бути ватажком там, де вас люблять?
5. Чи йдуть за вами люди, які визнають ваш авторитет?
6. Чи можете ви наполегливо і без вагань приймати рішення, не зупиняючись перед труднощами?
7. Командувати і керувати вам комфортніше, ніж підкорюватися?
8. Чи вважаєте ви себе досить здібним і кмітливим порівняно з більшістю людей?
9. Коли вам доручають якусь справу, чи завжди ви наполягаєте на тому, щоб зробити її по-своєму?
10. Чи прагнете ви бути першим завжди і скрізь?
11. Якби ви серйозно зайнялися наукою, чи змогли б рано чи пізно стати професором?
12. Чи важко вам сказати «Ні», навіть якщо впевнені, що ваше бажання не здійсниться?
13. Чи вважаєте, що ваші досягнення в житті будуть значно більшими, ніж у ваших однолітків?
14. Чи встигнете ви зробити в житті більше ніж інші?

15. Якби вам довелося починати життя спочатку, чи змогли б ви досягти значно більше?
16. Чи охоче ви йдете за авторитетними людьми?
17. Чи любите ви керувати, чи не лякає вас відповідальність?
18. Чи вмієте бути ватажком і керувати людьми?
19. Чи любите ви бути попереду інших в думках, проте свідомо відмовляєтеся від керівництва і першості?
20. Чи все заплановане ви встигаєте роботи вчасно?
21. Чи ви вважаєте, що в будь-якій ситуації краще бути непомітним, щоб менше було вимог до вас?
22. Чи вважаєте ви, що при рівній матеріальній винагороді роль підлеглого краща, ніж роль керівника?

Підсумовування балів

За кожну відповідь «Так» додайте 2 бали, «Не знаю» - 1 бал, «Ні» - 0 балів. Із суми відповідей на 1-15 запитання необхідно відняти суму відповідей на 16-22 запитання.

Результати тестування

Сумарна кількість балів менша від 0 - найнижчий рівень;

від 0 до 6 балів – нижчий від середнього;

від 7 до 12 балів - середній рівень;

від 13 до 19 балів - високий рівень;

від 20 балів і більше - дуже високий рівень лідерських здібностей.

Додаток М

Тест для оцінювання комунікативних та організаторських здібностей

Необхідно відповісти на сорок запитань. Якщо ваша відповідь на питання позитивна (Ви згодні), то у відповідній графі таблиці відповідей (табл. 2) поставте «плюс», якщо ж ваша відповідь негативна – «мінус». Відповідайте швидко, не замислюючись. Якщо на деякі запитання Вам важко відповісти, дайте відповідь, яку вважаєте переважаючою. Відповідаючи на питання, не прагніть справити краще враження на оточуючих.

Таблиця М.1.

1		2		3		4	
5		6		7		8	
9		10		11		12	
13		14		15		16	
17		18		19		20	
21		22		23		24	
25		26		27		28	
29		30		31		32	
33		34		35		36	
37		38		39		40	

Запитання тесту

1. Чи багато у вас друзів, з якими ви постійно спілкуєтеся?
2. Чи часто вам вдається переконати більшість своїх друзів у правильності вашої думки?
3. Чи довго вас турбує почуття образи, причиненої вам кимось з ваших знайомих?
4. Чи завжди вам важко орієнтуватися у складній ситуації?

5. Чи прагнете ви встановлювати нові знайомства з різними людьми?
6. Чи подобається вам займатися суспільною роботою?
7. Чи правда, що вам комфортніше проводити час вдома за книжкою або за якимось іншим заняттям, ніж у товаристві інших людей?
8. Якщо виникли будь-які перешкоди для здійснення ваших намірів, чи легко ви відступаєте від них?
9. Чи легко ви встановлюєте контакти з людьми, які значно старші від вас за віком?
10. Чи любите ви організовувати різні ігри та розваги з вашими знайомими?
11. Чи важко ви включаєтеся в нову для вас компанію?
12. Чи часто ви відкладаєте на пізніше ті справи, які потрібно було б виконати зараз?
13. Чи легко вам вдається встановлювати контакти з незнайомими людьми?
14. Чи прагнете ви добиватися, що б ваші друзі діяли відповідно до вашої думки?
15. Чи важко ви освоюєтеся в новому колективі?
16. Чи правда, що у вас не буває конфліктів з друзями через невиконання ними своїх зобов'язань?
17. Чи прагнете ви при будь-якій слушній нагоді познайомитися і поговорити з новою людиною?
18. Чи часто у вирішенні важливих справ ви приймаєте ініціативу на себе?
19. Чи дратують вас навколишні люди і чи хочеться вам побути на самоті?
20. Чи правда, що ви зазвичай погано орієнтуєтеся в незнайомій для вас обстановці?
21. Чи подобається вам постійно перебувати серед людей?
22. Чи виникає у вас роздратування, якщо вам не вдається завершити розпочату справу?
23. Чи переживаєте ви незручності, якщо доводиться проявити ініціативу для знайомства з новою людиною?

24. Чи правда, що ви втомлюєтеся від частого спілкування з іншими?
25. Чи подобається вам брати участь у колективних іграх?
26. Чи часто ви проявляєте ініціативу при вирішенні питань, що зачіпають інтереси ваших друзів?
27. Чи правда, що ви відчуваєте себе невпевнено серед малознайомих вам людей?
28. Чи правда, що ви рідко прагнете доводити свою правоту?
29. Чи вважаєте ви, що вам нескладно внести пожвавлення у малознайомій компанії?
30. Чи берете ви участь у суспільній роботі?
31. Чи прагнете ви обмежити коло своїх знайомих невеликою кількістю людей?
32. Чи вірно, що ви не будете відстоювати свою думку або рішення, якщо воно не було відразу сприйняте вашими друзями?
33. Чи відчуваєте ви себе невимушено, потрапивши в незнайому компанію?
34. Чи охоче ви організовуєте різних заходів для своїх знайомих?
35. Чи правда, що ви не почуваєте себе досить впевнено і спокійно, коли доводиться виступати перед великою групою людей?
36. Чи часто ви запізнюєтеся?
37. Чи маєте Ви багато друзів?
38. Чи часто опиняєтеся в центрі уваги своїх товаришів?
39. Чи часто ви бентежитеся, відчуваєте незручність при спілкуванні з малознайомими людьми?
40. Чи правда, що ви не дуже впевнено почуваєтеся в оточенні великої групи своїх знайомих?

Підсумовування балів

Для опрацювання результатів слід порівняти відповіді з дешифратором (табл. 3) і порахувати окремо кількість збігів за комунікативними та організаторськими здібностями.

Таблиця М.2.

Здібності	**Відповіді**	
	Позитивні	**Позитивні**
комунікативні	запитання 1-го стовпчика	запитання 3-го стовпчика
організаторські	запитання 2-го стовпчика	запитання 4-го стовпчика

Далі необхідно обчислити коефіцієнти комунікативних ($К_к$) та організаторських ($К_о$) здібностей за формулами:

$$К_к = З_к /20, \quad К_о = З_о /20,$$

де $З_к$, $З_о$ - кількість співпадаючих відповідей відповідно за комунікативними та організаторськими здібностями, 20 - максимально можлива кількість збігів.

Для якісної оцінки результатів необхідно порівняти отримані коефіцієнти з наступною шкалою (табл. 4).

Таблиця М.3.

$К_К$	$К_О$	**Оцінка**
0,10 – 0,45	0,20 – 0,55	1
0,46 – 0,55	0,56 – 0,65	2
0,56 – 0,65	0,66 – 0,70	3
0,66 – 0,75	0,71 – 0,80	4
0,76 – 1,00	0,81 – 1,00	5

Опис комунікативних та організаторських здібностей

Оцінка 1. Рівень ваших організаторських та комунікативних здібностей дуже низький.

Оцінка 2. Ваші комунікативні та організаторські здібності перебувають на рівні, нижчому від середнього. Ви не прагнете спілкування, скуто почуваєтеся в новому товаристві, вважаєте за краще проводити час на самоті, обмежуєте коло знайомств, відчуваєте труднощі під час встановлення контактів з новими людьми та при виступі перед аудиторією, погано орієнтуєтеся у незнайомій ситуації, не відстоюєте свою думку, важко переживаєте образи. Прояв вашої ініціативи в суспільній діяльності вкрай занижений, у багатьох справах ви вважаєте за краще уникати прийняття самостійних рішень.

Оцінка 3. У вас середній рівень організаторських та комунікативних здібностей. Ви прагнете до спілкування з людьми, не обмежуєте коло своїх знайомих, відстоюєте власну думку, плануєте свою роботу. Проте потенціал цих здібностей не має високої стійкості.

Оцінка 4. Рівень ваших комунікативних та організаторських здібностей високий. Ви не губитеся в новій обстановці, швидко знаходите друзів, постійно прагнете розширити коло своїх знайомств, займаєтеся громадською діяльністю, допомагаєте близьким, друзям, виявляєте ініціативу у спілкуванні, із задоволенням берете участь в організації суспільних заходів, здатні у складній ситуації приймати самостійні рішення.

Оцінка 5. У вас дуже високий рівень комунікативних та організаторських здібностей. Ви відчуваєте в ній потребу в організаторській та комунікативній діяльності. Швидко орієнтуєтеся у складних ситуаціях, невимушено поводитеся у новому колективі. У важливій справі або скрутній ситуації вважаєте за краще ухвалювати самостійні рішення, відстоюєте свою думку і добиваєтеся, щоб вона була прийнята колегами. Можете внести пожвавлення у незнайомому товаристві, любите організовувати різні заходи. Ви наполегливі у цікавій для вас діяльності, самі знаходите шляхи для комунікативної та організаторської діяльності.

ЗМІСТ

НАВЧАЛЬНЕ ВИДАННЯ

ВСТУП ДО ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

НАВЧАЛЬНИЙ ПОСІБНИК

з дисципліни “Вступ до інженерії програмного забезпечення ”

для студентів спеціальності

“Інженерія програмного забезпечення ”

Укладач Левус Євгенія Василівна

Мельник Наталія Богданівна

Редактор

Комп’ютерне верстання