Тема 2 ОПТИМІЗАЦІЯ ПРОГРАМНИХ РОЗРОБОК

2.1. ВИБІР ОПТИМАЛЬНОГО ВАРІАНТУ ПРОЕКТНОГО РІШЕННЯ

На різних етапах проектування (особливо часто на початкових етапах) перед розробником постає завдання вибору найкращого варіанта з безлічі допустимих проектних рішень, які задовольняють вимогам.

Неминучою платою за спробу отримати рішення за умов неповної інформації про об'єкт проектування є можливість помилкових рішень. Тому в такій ситуації особа, яка приймає рішення (ЛПР), повинна виробляти таку стратегію щодо прийняття рішень, яка хоч і не виключає можливості прийняття неправильних рішень, але зводить до мінімуму пов'язані з цим небажані наслідки. Для зменшення невизначеності ЛВР може провести експеримент, але це дорого і вимагає великих витрат часу. Тому ЛПР має прийняти рішення про форму, час, рівень експерименту.

Саме собою ухвалення рішення є компроміс. Приймаючи рішення, необхідно зважувати судження про цінність, що включає розгляд багатьох чинників, зокрема економічних, технічних, наукових, ергономічних, соціальних тощо.

Прийняти «правильне» рішення означає вибір такої альтернативи з-поміж можливих, в якій з урахуванням усіх різноманітних факторів буде оптимізовано загальну цінність. Процес прийняття рішення при оптимальному проектуванні характеризують такі основні риси: наявність цілей (показників) оптимальності, альтернативних варіантів об'єкта, що проектується, і облік істотних факторів при проектуванні.

Поняття «оптимальне рішення» при проектуванні має цілком певне тлумачення — найкраще у тому чи іншому сенсі проектне рішення, яке допускається обставинами. У переважній більшості випадків одна і та ж проектна задача може бути вирішена декількома способами, що призводять не тільки до різних вихідних характеристик, а й до класів програм. Найуніверсальніші програми - це текстові (процесори) редактори, що допускають використання графіки. Вони дозволяють оформляти вихідні дані, здійснювати ручний набір обґрунтування рішення з результатами розрахунку, робити висновок на друк. Для деяких цілей кращі електронні таблиці. Багатьох користувачів цілком влаштовують інтегровані системи, що включають текстовий процесор, процесор електронних таблиць, графічні процесори (малюнки та ділову графіку), систему управління базою даних (СУБД), системи модемного та мережевого зв'язку користувачів. При цьому одне з рішень може поступатися за одними показниками і перевершувати інші за іншими. Може бути так, що різні рішення взагалі характеризуються різним набором показників. У умовах важко зазначити, яка програмна система як оптимальна, і навіть краще. Найбільші збитки приносять помилки при виборі сукупності показників якості. Перепустка одного показника може виявитися трагічною. Щоб не робити таких помилок, треба накопичувати базу знань усіх сукупностей показників, які були використані для проектування конкретних систем. З іншого боку, використання традиційних сукупностей показників не дозволяє виходити нові вироби. Вихід із цього становища: певну частку у процесі проектування відводити під творчий пошук.

База знань сукупностей показників має складатися як із загальних (загально програмних), так і спеціальних (предметно-орієнтованих) показників. Нині використовують такі класифікації показників: 1) показники функціонування, що характеризують корисний ефект від використання програмної системи за призначенням, і область застосування (наприклад, бібліотечна інформаційно-пошукова система характеризується такими показниками функціонування: максимальним обсягом літературних

джерел, що зберігаються; максимальною кількістю одночасно працюючих користувачів; списком оброблюваних запитів; часом реакції на кожний запит при максимальній кількості користувачів;
2) показники надійності, що характеризують властивості програмної системи зберігати свою працездатність у часі;
3) показники технологічності, що характеризують ефективність конструкторсько-технологічних рішень для забезпечення високої продуктивності праці при виготовленні та супроводі;
4) ергономічні показники, що характеризують систему людина-виріб-середовище та враховують комплекс гігієнічних, антропологічних, фізіологічних та психічних властивостей людини, що виявляються у виробничих та побутових умовах;
5) естетичні показники, що характеризують зовнішні властивості системи: виразність, оригінальність, гармонійність, цілісність, відповідність середовищу та стилю;
6) показники стандартизації та уніфікації, що характеризують ступінь використання у програмній системі стандартизованих виробів та рівень уніфікації його частин;
7) патентно-правові показники, що визначають кількість патентів, ступінь патентного захисту, патентну чистоту;
8) економічні показники, що характеризують витрати на розробку, виготовлення, експлуатацію програмної системи та економічну ефективність експлуатації.
Середовище проекту характеризується можливостями капіталовкладення, можливостями колективу-виробника, науково-технічними досягненнями, соціальним та природним середовищем.

2.2. ПРИКЛАД ВИБОРУ ОПТИМАЛЬНОГО ВАРІАНТУ ПРОГРАМНОГО РІШЕННЯ

Щоб зробити вибір оптимального варіанта, треба мати кілька варіантів реалізації виробу. А щоб їх порівнювати, треба сформулювати ряд характеристик чи ненормованих критеріїв. Після нормування характеристик відповідно до шкал виходять нормовані критерії, які зручні для аналізу. Один із найпростіших способів отримання нормованих значень критеріїв - це проставлення експертами оцінок за п'яти-або десятибальною шкалою.
Фірма Borland Inc., створивши свій компілятор, вирішила розробити демонстраційну програму, яка могла б показати найбільшу кількість можливостей компілятора. У табл. 2.1 наводяться найменування критеріїв, варіанти реалізації програм та оцінки за п'ятибальною шкалою. Цю таблицю склали учні одному з практичних занять. Ними ж було виставлено оцінки.
Постачання компілятора у вихідних текстах може призвести до його іноді некваліфікованих масових модифікацій, що, своєю чергою, може викликати недовіру до виробу і завдати шкоди репутації фірми.

Таблиця 2.1
Бальна оцінка варіантів реалізації програми за критеріями

Критерії	Варіанти реалізації				
	СУБД	ЕТ	ОС	Редактор текстів	Гра
Обсяг програми	4	4	4	2,5	3
Зрозумілість	2	4	1	5	2
Нові знання	2	4	2	3	2
Інтерес	4	3	3	3	5
Використання у власних розробках	2	5	5	4	1

Система управління базами даних (СУБД) може бути великою або невеликою програмою. Головне в СУБД мало зрозумілі алгоритми обробки даних. Інтерес для користувача представляє бібліотека обробки даних, а чи не готова програма.

Електронна таблиця (ЕТ) надає можливість демонстрації користувачеві збирання програми із низки програмних файлів. Електронна таблиця містить функції редактора та інтерпретатора арифметичних виразів. Програма може бути прикладом реалізації обчислювальних алгоритмів. Сама програма та її окремі частини можуть вставлятися у програми користувачів. Програма таблиці має середній розмір. Операційна система може мати будь-який обсяг. Зрозумілість текстів ОС невисока.
Простий текстовий редактор є короткою програмою, що складається з декількох підпрограм. Складний текстовий процесор виходить з простого редактора екстенсивним доповненням великої кількості сервісних функцій з алгоритмами, що важко сприймаються.
Таким чином, саме для поставленої мети розробки перемагає варіант електронної таблиці (ЕТ), яка включає: клітинний редактор з ідеології функціонування, близький до текстового редактора; алгоритми роботи із файлами складної структури; інтерпретатор мови формул із виконавцем математичних розрахунків.
Фірма «Borland Inc.» з ранніми розробками компілятора (Turbo Pascal 4.0) постачала демонстраційну програму найпростішої електронної таблиці MicroCalc.
У пізнішому дистрибутиві Turbo Pascal 6.00 з'явилася нова демонстраційна версія електронної таблиці TurboCalc, реалізована з використанням об'єктно-орієнтованої технології. Оскільки й інші варіанти реалізації програм викликають інтерес у користувачів, фірма з пізніми розробками компілятора почала постачати їх. Так постачалися: гра в шахи з незрозумілими алгоритмами; текстовий редактор як бібліотечна програма; бібліотека підтримки роботи з базами даних Сам компілятор у вихідному коді фірмою Borland Inc. ніколи не постачався.

2.3. МЕТОДИ СИНТЕЗУ ВАРІАНТІВ РЕАЛІЗАЦІЙ ПРОГРАМ

Щоб відібрати оптимальне рішення необхідно синтезувати безліч можливих рішень (варіантів), що включають оптимальне рішення.
Жодне завдання не вирішується саме собою. Щоб отримати рішення, провадяться різні розумові дії. Ці дії не хаотичні, а мають методичну спрямованість, хоча зазвичай людина про це не підозрює.
Існує безліч методів синтезу варіантів проекту. Ось лише деякі з найбільш прийнятних для програмування: метод проб і помилок; евристичних прийомів; мозкового штурму; метод аналогій та морфологічних таблиць.
Раніше й до нашого часу більшість нестандартних завдань вирішувалася людиною на інтуїтивному рівні, т. е. шляхом спроб і помилок.
Метод проб та помилок- Це послідовне висування та розгляд ідей. Людина, стикаючись із проблемою, багаторазово подумки шукає відповідь, перебирає варіанти і, нарешті, знаходить рішення. Десятки, сотні, тисячі спроб упродовж днів, тижнів, років. Зрештою, в більшості випадків рішення знаходиться. У програмуванні цей метод зазвичай застосовується для оптимізації архітектури систем та структури програм.
Головний недолік методу спроб і помилок - це, по-перше, повільне генерування нових ідей, а по-друге, відсутність захисту від психологічної інерції, тобто висування ідей тривіальних, звичайних, неоригінальних.
Наступним кроком у вдосконаленні технології став перехід до спрямованих методів пошуку рішень, які базуються на розкритті та описі процесу рішення, представленні його у вигляді деякого евристичного алгоритму. Спрямованість евристичних методів - "розгойдувати" мислення, допомогти по-новому побачити завдання, подолати стереотипи.
Якщо, вирішуючи конкретне завдання, проектувальник не обмежиться досягненням лише миттєвої мети, а зможе «зазирнути у майбутнє» і виділити інваріантні частини системи, ці частини, будучи хіба що будівельним матеріалом цієї системи, можуть бути основою й у систем, які ще будуть проектуватись. Майбутні системи можуть вирішувати зовсім інші завдання. У цьому корисно було б створювати та накопичувати бібліотеки інваріантних частин системи або навіть паралельно проектувати кілька систем (об'єктів), що мають як подібні, так і різні цілі. «Зазирнути в майбутнє» можна лише добре знаючи минуле та сьогодення, а також нові досягнення програмування.
Цілеспрямовані методи творчості цілком застосовні не лише до технічних систем, а й до програмних. Розглянемо найбільш відомі з них, а також їхнє можливе застосування як при колективному, так і індивідуальному використанні.

Метод евристичних прийомів дозволяє як з'єднувати по-новому відомі частини, а й винаходити нові. Він базується на виділенні базових прийомів, знайдених під час аналізу кращих програмних виробів. При успішному розв'язанні будь-якої творчої задачі людина отримує два результати - саме розв'язання поставленої задачі та методичний досвід, тобто з'ясування процесу вирішення даної конкретної задачі. Але проблема полягає в тому, що вирішення одного завдання не можна просто перенести на рішення іншого. Тому лише після вирішення певної кількості завдань у людини з'являвся набір правил, вказівок чи прийомів розв'язання того чи іншого завдання. Такі методичні правила називають евристичними прийомами.

Евристичний прийом- Спосіб вирішення певного протиріччя. В евристичному прийомі міститься короткий розпорядження або вказівка, як перетворити вихідний прототип або в якому напрямку потрібно шукати, щоб отримати рішення. Евристичний прийом містить підказку, але гарантує перебування рішення.

Складність використання евристичних прийомів полягає в тому, що не будь-яка людина може бачити завдання в цілому, тобто не має системного підходу у вирішенні завдань. Причому необхідність у такому підході зростає із збільшенням складності завдання. Це призводить до того, що людина не може застосувати евристичний прийом до конкретного завдання і не розуміє, про що йдеться. Різним людям потрібно докласти різних зусиль, щоб здогадатися про те, як застосувати евристичний прийом та отримати розв'язання задачі. Але перед розв'язанням задачі повинні бути описані та з'ясовані критерії, за якими оцінюватиметься отримане рішення. Це допоможе відкинути непотрібні рішення при використанні евристичних прийомів і зрозуміти, в якому напрямку слід рухатись, щоб дійти до вирішення.

Отже, у кожної людини, яка займається створенням програм, згодом накопичується досвід і з'являються способи вирішення різноманітних завдань. Причому зі збільшенням досвіду у цих способах збільшується частка системного підходу, тобто згодом людина отримує такий спосіб, який стає застосовним для вирішення більшої кількості завдань, ніж було раніше.

Поступово у фахівця накопичується фонд таких практичних прийомів, але цей фонд є індивідуальним і не завжди доступним іншим користувачам. Тому необхідно систематизувати такі фонди та зробити їх більш систематичними. Актуальним завданням є створення фонду евристичних прийомів, застосовуваного на вирішення завдань оптимізації програмних розробок.

Найбільше евристичних прийомів спрямовано подолання протиріч. Суперечність у завданні — ситуація, яка потребує одночасного поліпшення двох суперечливих показників якості та поєднання, начебто, несумісних вимог. Ряд прийомів легко сприяє активізації мислення.

Перше, що спадає на думку в ситуації з протиріччями, — знайти найкраще співвідношення між показниками. Якщо потенційні можливості структури об'єкта великі, то цьому шляху іноді вдається отримати прийнятні значення всіх показників. Однак це вдається не завжди.

Наступний природний для людини крок у ситуації, коли потенційні можливості структури недостатні і компроміс виявляється неприйнятним, — перейти до нової структури з великими потенціями, які забезпечують досягнення прийнятних значень показників якості, що конкурують.

Однак кращим є інший варіант дій. Як показує досвід, у багатьох ситуаціях містяться приховані ресурси, використання яких дозволяє кардинально вирішити протиріччя. Такими прихованими ресурсами у багатьох випадках є три види ресурсів: тимчасові, просторові та «непрямі». До «непрямих» ми відносимо ресурси, наявні у системі (але, зазвичай, не вважаються ресурсом, по крайнього заходу, у межах розв'язуваного завдання), використання яких пов'язані з якимись додатковими затратами. Найбільш продуктивні способи вирішення протиріч - рознесення їх у часі, просторі або використання «непридатних» ресурсів. Ці загальні рекомендації можуть мати різне трактування у конкретних ситуаціях.

Доцільність використання методу евристичних прийомів для постановки завдань розробки програм перевірено педагогічною практикою авторів. Найчастіше досить короткої підказки учням як евристичного прийому, щоб вони самостійно правильно сформулювали завдання.

Використовуючи фонд евристичних прийомів, Б.С. Воїнів та В.В. Костерін успішно синтезували низку нових механізмів алгоритму пошуку глобального екстремуму функцій багатьох змінних на сітці коду Грея.

Приклад використання методу евристичних прийомів до створення алгоритмів описаний у книзі Д. Пойа. Укорочений фонд евристичних прийомів програмування описаний у додатку 3.

Метод мозкового штурму- Один із популярних методів колективної творчості. Його психологічна основа – взаємна стимуляція мислення у групі. Конкуренція для людей за кількість висунутих ідей робить цей метод більш ефективним порівняно з роботою кожної окремої людини поза групою. Метод мозкового штурму є по суті тим самим методом спроб і помилок, проте він створює умови для психологічної активізації творчого процесу, знижує інерцію мислення, чому особливо сприяє наявність групи людей з боку. Метод простий, доступний та ефективний. Реалізація методу виглядає так. Рішення проводиться у два етапи – генерації та аналізу. На етапі генерації створюється творча група з 5-15 осіб (фахівці-суміжники і люди «з боку», які не мають жодного досвіду в галузі, до якої належить вирішуване завдання). Групі пояснюється суть завдання, що потребує вирішення, та проводиться етап генерації ідей. На цьому етапі не допускається критика пропонованих ідей. Заохочується висування навіть навіжених ідей. Потім група експертів аналізує висловлені ідеї та відбирає ті, які заслуговують на більш ретельне опрацювання.

Методом мозкового штурму працюють команди знавців у популярній телепередачі: «Що? Де? Коли? П'ятдесят секунд іде генерація ідей і лише десять секунд витрачається на обговорення висунутих ідей. Стосовно програм методом мозкового штурму можна згенерувати ідеї щодо розподілу функцій обробки інформації для людей і машиною; набору основних прототипів та запозичених з них ідей; реалізації деяких евристичних алгоритмів обробки інформації; реклами та збуту програмної продукції; створення нових програм з урахуванням частин створюваних і раніше створених програм.

Методи аналогій є найпопулярнішими методами для програмістів. Подолати психологічну інерцію, знайти нове рішення допомагають несподівані порівняння, що дозволяють глянути ситуацію під незвичайним кутом.

Суть одного з методів полягає у наступному. Удосконалену систему тримають хіба що у фокусі уваги і переносять її у властивості інших програм із колекції, які мають до неї жодного відношення. У цьому виникають незвичайні поєднання, які намагаються розвинути далі.

Згідно з проведеним опитуванням, більшість професійних програмістів саме цим методом генерували зовнішній вигляд своїх програмних систем та визначили способи реалізації багатьох функцій програм.

Метод морфологічних таблиць є простим та ефективним особливо там, де необхідно знайти велику кількість варіантів досягнення мети. Останнім часом він використовується досить широко як розвиток творчої уяви. Метод по суті аналогічний складання різних варіантів будинку з кубиків дерев'яного конструктора і полягає в тому, що для об'єкта, що цікавить нас, формується набір відмітних ознак: найбільш характерних підсистем, властивостей або функцій. Потім кожного з них визначаються альтернативні варіанти реалізації (деталі конструктора). Комбінуючи альтернативні варіанти, можна отримати багато різних рішень. Аналізуючи їх, виділяють кращі варіанти.

Приклад морфологічної таблиці є прайс-лист комп'ютерної фірми. У прайс-листі міститься інформація про декілька типів корпусів ЕОМ; материнських плат; процесорів і т. д. Кожна частина забезпечена технічними характеристиками та ціною. Не всі варіанти елементів можуть бути зістиковані між собою. Головне, що характеризує прайс-лист, - це відсутність критерію якості комп'ютера для конкретного користувача. Дивлячись на прайс-лист, треба синтезувати цей критерій і вибрати оптимальний склад елементів. Як результат синтезу можуть бути виявлені варіанти побудови комп'ютерів, орієнтованих різні категорії користувачів.

> Проблема тут така сама, як і в ситуації підбору одягу для дівчини. Якщо ознайомитися з товаром у ряді магазинів, то неважко за рекомендацією «краще дороге» купити окремі елементи гардеробу. Швидше за все ці «найкращі» елементи не будуть загалом виглядати на дівчині через несумісність кольорів, фасону, та й самого вигляду та характеру дівчини, тобто відсутня оптимізація за критерієм цілого.
>
> Покупка сама може перетворитися на болісне ходіння по магазинах з тисячами примірок. При цьому, швидше за все, буде помилково придбаний не той товар.
>
> Для аналізу критерію цілого краще залучити досвідчених експертів (наприклад, піти магазинами з подругами), які можуть вказати на помилки вибору.
>
> Досвідчений кутюр'є допоможе зробити вибір дорогого модного одягу. Можливо, щоб краще підходити під запропонований одяг, вам доведеться позайматися з психологом, косметологом та інструктором фізкультури для зміни іміджу. На щастя, багато дівчат здатні самі одягатися «дешево та сердито» і вміють самостійно адаптувати свій імідж.

Морфологічна таблиця, складена в компактній формі, допоможе уникнути багаторазового ходіння по одним і тим же магазинам, що заощадить ваш час та час експертів. Морфологічна таблиця дозволить вам та експертам переглянути значно більше варіантів і зробити більш оптимальний вибір.

У 1983 р. В.В. Костериним успішно застосовано морфологічна таблиця для синтезу ідей побудови алгоритму нелінійного програмування пошуку глобального екстремуму функцій багатьох змінних на сітці коду Грея. Алгоритми нелінійного програмування призначені для пошуку екстремумів функцій багатьох змінних. У методах прямого пошуку екстремум виявляється шляхом розрахунку множини точок функції при аргументах, що визначаються самим алгоритмом пошуку. У табл. 2.2 наведено дану морфологічну таблицю, яка містить класифікаційні ознаки окремих механізмів алгоритмів нелінійного програмування на рівні основних принципів. Наведені класифікаційні ознаки виділялися за основними функціональними ознаками окремих механізмів. Кожній класифікаційній ознакі відповідає безліч реалізацій механізмів у вигляді значень класифікаційних ознак.

Цікаво відзначити, що кількість можливих реалізацій алгоритмів нелінійного програмування за цією таблицею становить N = 5 * 6 * 8 * 5 * 7 * 7 * 6 = 352800, що значно перевищує число опублікованих методів (близько 2000)!

Таблиця 2.2

Морфологічна таблиця принципів функціонування алгоритмів нелінійного програмування

Класифікаційні ознаки	Значення класифікаційних ознак
Початкова точка пошуку	1.1 1.2 1.3 1.4 1.5
Зондування гіперповерхні	2.1 2.2 2.3 2.4 2.5 2.6
Стратегія кроків пошуку	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8
Напрямок пошуку на кроці	4.1 4.2 4.3 4.4 4.5
Стратегія кроку пошуку	5.1 5.2 5.3 5.4 5.5 5.6
Механізм самонавчання	6.1 6.2 6.3 6.4 6.5 6.6 6.7
Механізм завершення пошуку	7.1 7.2 7.3 7.4 7.5 7.6

Значення класифікаційних ознак класифікаційної ознаки "Механізм початкової точки пошуку":
ознака 1.1 - з точки, вказаної користувачем;
ознака 1.2 - із середньої точки області визначення;
ознака 1.3 - з точки на межі області визначення;
ознака 1.4 - з випадкової початкової точки пошуку;
ознака 1.5 - початкова точка пошуку не задається.

Значення класифікаційних ознак класифікаційної ознаки «Первинне зондування гіперповерхні»:
ознака 2.1 - у вигляді великої кількості випадкових точок, зондуючих всю гіперповерхню;
ознака 2.2 - послідовні спуски з ряду випадкових початкових точок;
ознака 2.3 - конкуруючі спуски з випадкових точок, що додаються;
ознака 2.4 - зондування гіперповерхні випадковими точками з виявленням та більш ретельним дослідженням «підозрілих областей»;
ознака 2.5 - сканування всієї гіперповерхні з використанням різних розгорток, наприклад, Пеано;
ознака 2.6 - окремий механізм початку пошуку відсутня.

Значення класифікаційних ознак класифікаційної ознаки «Стратегія кроків пошуку»:
ознака 3.1 - один крок;
ознака 3.2 - Послідовні кроки до виявлення екстремуму;
ознака 3.3 - здійснювати всі кроки по тому самому механізму;
ознака 3.4 - перемикати механізми кроків від глобального методу до локального;
ознака 3.5 - перемикати механізми кроків від глобальних далі до усереднених і локальних;
ознака 3.6 - перемикати механізми кроків за евристичними правилами;
ознака 3.7 - мала кількість послідовних кроків з обмеженого ряду лідируючих конкуруючих початкових точок;
ознака 3.8 - кроки пошуку відсутні.

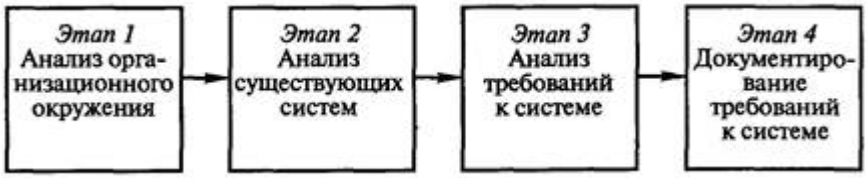
Значення класифікаційних ознак класифікаційної ознаки «Напрямок пошуку на кроці»:
ознака 4.1 - нова точка в напрямку апроксимації градієнта, побудованого на основі даних поточної та попередньої пробної точок;

ознака 4.2 - за результатами обробки невеликої кількості перспективних точок, отриманих на попередніх кроках;
ознака 4.3 - за результатами аналізу функції, що апроксимує випадкові точки в перспективному напрямку;
ознака 4.4 - зондування гіперповерхні великою кількістю випадкових точок та подальшою побудовою апроксимуючої функції;
ознака 4.5 - вздовж межі області визначення цільової функції;
ознака 4.6 – механізм відсутній.
Значення класифікаційних ознак класифікаційної ознаки «Механізм стратегії кроку пошуку»:
ознака 5.1 - пробні точки тільки на відстані передбачуваного екстремуму;
ознака 5.2 - пробні точки на більшій відстані, ніж передбачуваний екстремум;
ознака 5.3 - пробні точки на відстані, дещо меншій, ніж у передбачуваного екстремуму;
ознака 5.4 - об'єднання ознак 5.1 та 5.2;
ознака 5.5 - об'єднання ознак 5.1, 5.2 та 5.3;
ознака 5.6 - суміщення пошуку напряму та відстані до екстремуму;
ознака 5.7 - поділ пошуку напрямку та відстані до екстремуму.
Значення класифікаційних ознак класифікаційної ознаки «Механізм самонавчання»:
ознака 6.1 - звуження меж пошуку у міру просування до екстремуму;
ознака 6.2 - Поступове підвищення точності пошуку;
ознака 6.3 - виявлення форми гіперповерхні за результатами попередніх кроків та перехід на спеціальний механізм уточнення екстремуму;
ознака 6.4 - виявлення форми гіперповерхні за результатами попередніх кроків та перехід на спеціальний механізм просування вздовж ярів;
ознака 6.5 - виявлення форми гіперповерхні за результатами попередніх кроків та відмова від поточного знайденого екстремуму;
ознака 6.6 - Зміна щільності ймовірності випадкових точок для різних зон пошуку;
ознака 6.7 – механізм відсутній.
Значення класифікаційних ознак класифікаційної ознаки «Механізм завершення пошуку»:
ознака 7.1 - не виявляється напрямок поліпшення функції на наступному кроці;
ознака 7.2 - витрачений ресурс часу;
ознака 7.3 - досягнуто заздалегідь задане значення цільової функції;
ознака 7.4 - вичерпані можливості алгоритму пошуку екстремуму;
ознака 7.5 - виконано заздалегідь задану кількість кроків пошуку;
ознака 7.6 — немає поліпшень у «далекому» та «близькому» околицях.
Черговий принцип побудови методу нелінійного програмування виходить шляхом відбору за одним із значень класифікаційних ознак у кожному окремому рядку табл. 2.2.
Оболонки візуального програмування, наприклад Delphi, реалізують метод морфологічного синтезу під час побудови форм діалогу програм з урахуванням візуальних компонент.

2.4. АНАЛІЗ ВИМОГ ДО СИСТЕМИ (СИСТЕМНИЙ АНАЛІЗ) І ФОРМУЛЮВАННЯ ЦІЛІВ

Завдання оптимізації розробки програм полягає у досягненні цілей за мінімально можливої витрати ресурсів.
Системний аналіз на відміну попереднього системного дослідження — це поглиблене вивчення інформаційних потреб користувачів, яке буде покладено основою детального проектування нової автоматизованої системи (АС).
Кінцевий продукт цього етапу — набір функцій, що виконуються, або функціональні вимоги, тобто документована постановка системних вимог до нової АС. Коли йдеться про створення великої системи, цей документ є звітом про системний аналіз, який здійснюється за етапами, показаними на рис. 2.1.
Перший етап системного аналізу (аналіз організаційного оточення) пов'язаний з тим, що неможливо створити працездатну інформаційну систему, якщо дослідники нічого не знають про особливості функціонування організації, функції якої має обслуговувати автоматизована система (АС) та елементом якої вона є. Слід розуміти особливості та тип діяльності, управлінську структуру, методи управління,

зв'язки підрозділів, персонал, динаміку інформаційного обміну між окремими працівниками та робочими групами (форми документів та звітів, терміни, кількість екземплярів тощо).
Другий етап системного аналізу (аналіз існуючих систем) обумовлений тим, що в організаціях можуть існувати якісь АС з певними ресурсами (інформаційними, програмними, технічними, а також персоналом). Навіть тоді, коли проводиться повне оновлення технічної та програмної баз, існуюче інформаційне забезпечення відображає ядро головних потреб, які не можна не ігнорувати в новій системі, а навпаки, стартуючи від нього, слід розвинути і розширити.
Слід ретельно вивчати, які завдання вирішують «старі» системи, яке обладнання та програмне забезпечення мають, який персонал працює в інформаційному відділі; чи існують бази даних, якою є їх структура і якими методами формуються звіти про результати — усе це — найважливіші питання. Важливо з'ясувати: чи застосовується кодування інформації та які рівні кодів у своїй використовуються (місцеві, державні, міжнародні); існуючий регламент обробки даних, кого і чому він не влаштовує, чи бувають практичні затримки даних та звітів, причини затримки; чи є документація на стару систему. Такий перелік цілей слід пам'ятати в особистому щоденнику дослідника системи.



*Мал. 2.1.*Етапи проведення системного аналізу інформаційних систем

Другий етап системного аналізу — це найскладніший і найвідповідальніший крок у масу метаінформації, т. е. «даних даних». Головними орієнтирами в організації метаінформації є об'єкти (документи, діаграми, аналітичні тексти та записки, економічні показники, сукупності), а також процеси створення об'єктів, процеси їх передачі, обробки, зберігання.
На виконання третього етапу системного аналізу (аналіз вимог системи) дослідник повинен мати певні знання типових методів вирішення основних управлінських завдань (облікових, аналітичних, планових, оперативних). Все це є складовою спеціальних знань системного аналітика. Тому системний аналітик повинен простежити всі галузі комплексу вимог у розмовах з кінцевими користувачами, а потім зробити аналітичні системні висновки. Ці вимоги є та підтримуються існуючою інформаційною системою, а це має стати функціональною вимогою до нової покращеної системи.
Третій етап системного аналізу – визначення того, що має бути у новій АС. Це методологічний етап синтезу вимог до нової системи, які випливають із перших двох кроків аналізу, а також із спеціальних знань та засобів системних аналітиків. Фахівці з системного аналізу знають, що однією з підступних пасток у їхній роботі є можливість сплутати аналіз існуючої системи з тим, що має бути. Тому другий та третій етапи системного аналізу чітко розділені. Про це важливо пам'ятати, щоб не помилитися в оцінках основи, на якій будується нова система. У системотехнічної методології аналіз існуючого відокремлюється від аналізу характеристик майбутнього, ніж сприйняти бажане за дійсне.
Четвертий, підсумковий етап системного аналізу (документування вимог до нової системи) має узагальнити наявні аналітичні матеріали та створити документоване відображення функціональних вимог до нової АС. Документ «Вимоги до системи», або «Функціональні вимоги», є основою подальшої роботи спеціалістів інформаційного відділу для створення детального проекту нової системи, тобто створення специфікацій всіх її елементів, програм, інструкцій.
Отже, етап системного аналізу відповідає питанням, що повинна мати інформаційна система задоволення вимог користувачів, а етап системного проектування відповідає питанням, як конкретно здійснити таку АС.
У багатьох аспектах системний аналіз є найважчою частиною процесу створення системи. Проблеми, з якими стикається системний аналітик, пов'язані між собою, що є однією з головних причин складності їх вирішення:
1) аналітику складно отримати вичерпну інформацію з метою оцінки вимог до системи з погляду замовника;

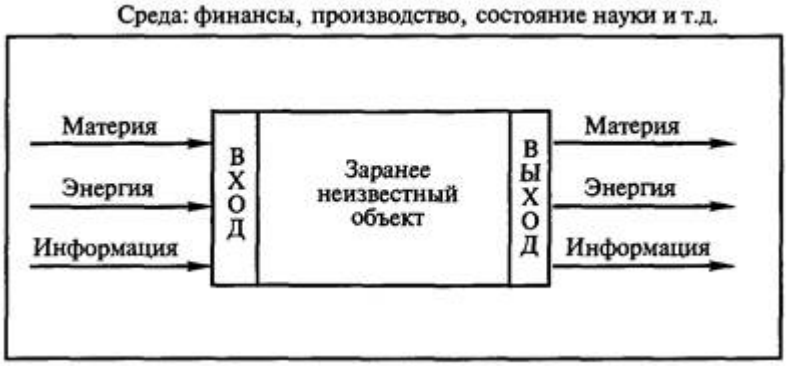
2) замовник, у свою чергу, не має достатньої інформації про проблему обробки даних, щоб судити, що є здійсненним, а що ні;
3) аналітик стикається з надмірною кількістю докладних відомостей про предметну область і про нову систему;
4) специфікація системи через обсяг та технічні терміни незрозуміла для замовника;
5) у разі зрозумілості специфікації для замовника, вона буде недостатньою для розробників, які створюють систему.
Отже, на даному етапі еволюційного розвитку ситуація в галузі проектування АС має такий вигляд:
• є суб'єкт — потенційний замовник, який зазнає дискомфорту, для подолання якого необхідно вирішити низку проблем, і тому цей суб'єкт є джерелом діяльності;
• є перелік потреб, які потрібно задовольнити;
• відомі прототипи програмних засобів з механізмами, які в сукупності могли б задовольнити наявні потреби, але ці механізми не пов'язані в єдине ціле так, щоб задовольнити всі потреби.
Тепер необхідно сформулювати цілі та визначити обмеження на реалізацію програмного продукту. Формулювання цілей – перший та найважливіший етап процесу проектування. Саме на цьому етапі закладаються основи успіху у вирішенні всього завдання. Помилки у виборі та формулюванні мети не можуть бути компенсовані на наступних етапах. Причина проста — все, що робиться на наступних етапах розробки, йде від поставленої мети. Отже, такі помилки жодними методами на наступних етапах неможливо компенсувати.
Зазвичай, всі помилки початкових етапів, виявлені наступних етапах, мають такі причини:
1) постановка недосяжної мети;
2) прагнення розробника та постановника завдання спростити завдання;
3) невміння розробника виділити із формулювання постановника окремо опис проблеми та постановку завдання;
4) не виявлені обмеження.
Ситуації, коли можлива конкуренція цілей, повинні виявлятися; необхідно узгодити цілі так, щоб найкращим чином у рамках можливостей, що визначаються обмеженнями, досягалися цілі кожного з можливих суб'єктів (замовників). Хороші цілі завжди є компромісом між бажанням якнайкраще задовольнити потреби та обмеженнями, що накладаються реалізацією та засобами.
Можна сформулювати послідовність рекомендацій (контрольних питань):
*Рекомендація 1.*Не довіряйте наявним формулюванням завдання; Рішення починайте з нуля, з виділення суб'єкта, виявлення причин його дискомфорту та потреб. Справа в тому, що найчастіше формулювання, пропоноване замовником, невдала або зовсім неприйнятна, оскільки описує насправді незадоволену потребу, видаючи її за завдання.
*Рекомендація 2.*Уточніть вимоги до кінцевого результату:
1) які функції та з якими показниками якості повинен виконувати функції об'єкт?
2) який ефект буде отримано у разі успішного розв'язання задачі?
3) які допустимі витрати, як вони пов'язані з показниками якості?
Може виявитися, що витрати істотно перевищать ефект, тому слід відмовитися від рішення, або шукати більш прийнятне.
*Рекомендація 3.*Уточніть умови, за яких передбачається реалізація знайденого рішення:
1) ретельно досліджуйте пов'язані з цим обмеження та переконайтеся, що всі вони дійсно мають місце;
2) виявите особливості реалізації, зокрема, допустимий ступінь складності, передбачувані масштаби застосування.
*Рекомендація 4.*Вивчіть завдання, використовуючи таку інформацію:
1) як вирішуються завдання, близькі до аналізованої?
2) як вирішуються завдання, обернені до розглядуваної? (Особливу увагу слід звернути при цьому на галузі застосування, для яких подібні завдання є найбільш актуальними.)
*Рекомендація 5.*Подумки змініть умови завдання та досліджуйте її розв'язання в нових умовах: змінюйте від нуля до нескінченності складність об'єкта, час процесу, витрати, умови середовища.
*Рекомендація 6.*Ретельно відпрацюйте формулювання завдання, бажано з використанням найзагальніших понять та термінів.
*Рекомендація 7.*Сформулюйте ідеальний кінцевий результат і у процесі рішення прагнете його отримати.

Аналіз вимог зосереджений на інтерфейсі системи людина-програма-машина та інформаційних потоках між ними. Тут вирішується, що робить людина, а що робить машина та як вона це робить. У результаті аналізу вирішується питання доцільності застосування ЕОМ.
У процесі аналізу розглядаються:
1) робота без ЕОМ та з ЕОМ з різним ступенем автоматизації;
2) варіанти використання існуючих програм як без модифікацій, так і з їхньою модифікаціями;
3) варіанти із спеціально створеними програмами;
4) час обробки інформації;
5) вартість обробки інформації;
6) ймовірність помилок, їх наслідки та якість обробки інформації.
Аналіз вимог сприяє кращому розумінню системи та досягненню найкращого задоволення потреби.
У ході проведення системного аналізу аналізуються надсистема, система та підсистема у вигляді складових проектованої системи.
Під час проектування необхідно враховувати такі ефекти.
Ефект заміни цілей. Процес діяльності з досягненню мети, зазвичай, пов'язані з подоланням деяких бар'єрів, часто непередбачених проблем. Намагаючись подолати їх, людина змушена змінювати початковий план дій, пристосовуючи його до конкретних умов. Природним для людини є і прагнення спростити своє завдання, діяти звичними, добре знайомими способами та засобами. Це може призвести найчастіше на завершальних етапах реалізації або в процесі експлуатації до заміни вихідної мети. У результаті досягнуто зовсім іншого результату, ніж передбачалося спочатку.
Тому чітке формулювання завдання, його відстеження на всіх етапах є необхідною умовою успішної діяльності.
Цілі та засоби. Наступний аспект, який не завжди оцінюється належним чином, — облік обмежень, що накладаються умовами реалізації, зокрема, властивостями реалізуючої системи та обмеженнями ресурсів. Здавалося б, забезпечення найбільшої ефективності об'єкта проектування з позицій над системи - головне завдання, але парадокс у тому, що не можна формулювати цілі, не маючи уявлення про те, як, за допомогою яких засобів, якої системи вони можуть бути досягнуті. Не будь-яка мета досяжна, і це або неможливістю чи незнанням, як реалізувати систему, що забезпечує досягнення мети, або є обмеження, які можуть зірвати її досягнення, тощо.
Повторимо раніше висловлену думку: «хороші цілі завжди є компромісом між бажанням якнайкраще задовольнити потреби та обмеженнями, що накладаються реалізацією та засобами». Практика показує, що цілі найкраще формулюються людьми, які знають можливості їх досягнення (або групою фахівців, які володіють різними аспектами проблеми).
Узгодження цілей. Насправді під час проектування цілей нерідко доводиться зіштовхуватися із ситуацією, коли функціонування одного об'єкта має задовольнити потреби низки суб'єктів (над систем). У цьому випадку стосовно кожної над системи можуть бути сформульовані свої цілі, які найчастіше виявляються суперечливими: більш повне задоволення потреб одного із суб'єктів може бути забезпечене лише за рахунок іншого. Якщо етапі проектування мети не узгоджені, то, зазвичай, погано задовольняються потреби як однієї, і інший над системи. Тому ситуації можливості конкуренції цілей повинні обов'язково виявлятися: необхідно узгодити цілі те щоб найкраще, у межах можливостей, визначених обмеженнями, досягалися мети кожної з над систем.
Формулювання та формалізація цілей. Цікавою видається інтерпретація цілей через потреби та обмеження ресурсів. При цьому можна виділити три варіанти формулювання цілей.
1. Обмеження відсутні. Досягнення кожної зі сформульованих цілей певною мірою задовольняє потребу, як правило, не знімаючи її повністю. Тому говорять про залишкову потребу: що менша залишкова потреба після досягнення мети, то вище оцінюється і сама мета. Отже, найкращою буде та мета, після досягнення якої залишкова потреба виявиться мінімальною.
2. Наведене вище формулювання привабливе, але мало реальне. Досягнення будь-якої мети потребує наявності ресурсів, причому величина їх (ресурсів) істотно залежить від формулювання мети. Тому найбільш прийнятною є формулювання, коли потрібно найкраще задовольнити потребу при заданих обмеженнях на ресурси.
3. Можлива ситуація, коли можуть бути визначені обмеження щодо ступеня задоволення потреб та потрібно при цьому мінімізувати витрати ресурсів. Тоді мета формулюється так: необхідно забезпечити заданий рівень задоволення потреби при мінімальному витраті ресурсів. Облік ресурсів та інших

обмежень є обов'язковим у більшості завдань проектування, отже, на етапі формулювання цілей обов'язковий аналіз, що дозволяє зіставити ступінь досягнення цілей і ресурси.
Незалежно від формулювання ціль як результат діяльності може бути задана формально через показники якості, що характеризують ступінь її досягнення. Вимоги до показників якості можуть бути задані у трьох видах: • прирівняти; обмежити; домогтися екстремуму.
У загальне формулювання цілей можуть входити складові, що задаються у різний спосіб.
Рівні опису цілей.У процесі проектування цілі можуть бути описані на рівні (виражені мовою) суб'єкта, зовнішнього та внутрішнього опису об'єкта. Суб'єкт при цьому характеризується своїми потребами, об'єкт при зовнішньому описі - показниками якості, а при внутрішньому, структурованому - через параметри та змінні стани. Тому нерідко говорять про завдання цілей у просторі потреб, показників якості та стану. Для опису цілей кожному рівні використовуються відповідні поняття та величини.
З урахуванням вищесказаного, загальна методика проектування цілей виглядає так. Будується опис над системи та визначаються показники, що характеризують ефективність її функціонування. Потім визначаються функції, які мають виконуватися проектованим об'єктом, і конкретні вимоги щодо них через показники якості над системи — цим визначаються цілі у просторі потреб. При подальшій конкретизації об'єкта рівня його показників якості досліджується вплив останніх на показники над системи. Це дозволяє виразити цілі через показники якості об'єкта - задати їх у просторі показників якості. Подальша конкретизація об'єкта дозволяє визначити зв'язки між показниками якості та значеннями параметрів та змінних стану, а також виразити цілі через вимоги до них – описати їх у просторі станів.
Незважаючи на прозорість методики, на практиці при проектуванні цілей доводиться стикатися з серйозними труднощами, пов'язаними в основному зі складністю опису над системи та взаємозв'язку її показників з показниками об'єкта, а також оцінки взаємозв'язку між значеннями показників та необхідними для досягнення цілей ресурсами.
Етап формулювання цілей може призвести до різних ситуацій.
*Ситуація 1.*Вихід на добре знайомі цілі, відомі розробнику. У цьому випадку знадобиться лише пошук та коригування відомих рішень, тобто в результаті аналізу потреб користувача проектувальник приходить до висновку, що задовольнити ці потреби може вже існуючий програмний продукт з невеликими змінами.
*Ситуація 2.*Діаметрально протилежний варіант – нові цілі. У цьому випадку ми маємо справу із завданням, яке має явно видиму мету і немає коштів для її безпосереднього досягнення.

2.5. ПРОЕКТНА ПРОЦЕДУРА ПОСТАНОВКИ ЗАВДАННЯ РОЗРОБКИ ПРОГРАМИ

Проектна процедура базується на володінні системним підходом стосовно аналізу програмних систем. Спочатку розглядається система - людина (люди), ЕОМ, програма, інші об'єкти, наприклад технічні, як ланка, що виконує весь комплекс обробки інформації. Далі ці елементи розглядаються окремо для уточнення вимог. Програма має лише інформаційний вхід та інформаційний вихід.
Полегшити первинну генерацію цілей часто дозволяє модель «чорної скриньки», зображена на рис. 2.2. У процесі проектних ітерацій дана модель «сіріє» і стає «білою», якщо на ній фіксувати всі знайдені факти.
Суть проектної процедури.
*Крок 1*Проаналізуйте вихід системи, визначте склад та форму вихідних даних.
*Крок 2.*Проаналізуйте вхід системи, визначте склад та форму вхідних даних.
*Крок 3*Визначте критерії якості перетворення вхідної інформації на вихідну інформацію.
*Крок 4.*Визначте обмеження.



*Мал. 2.2.*Модель «чорної скриньки» об'єкта, що проектується

*Крок 5.*Визначте основні алгоритми обробки інформації та розрахуйте час їх виконання.
*Крок 6*Визначте обмеження.
*Крок*7. До знаходження прийнятної постановки генеруйте варіанти постановок. Використовуйте класифікацію алгоритмів, критеріїв, методів прийняття рішень, евристичні прийоми, практичні прийоми.
При виконанні даної проектної процедури бажано розширити низку досліджуваних варіантів з використанням методу проб і помилок, методу аналогій, методу морфологічного синтезу рішень.
По-перше, з усього різноманіття можливих програмних рішень виділіть клас допустимих та перспективних.
По-друге, вивчіть варіанти, які є найкращими за окремими показниками. Ці варіанти можуть бути прийняті для подальшого аналізу.
Сукупність функцій системи, умов та обмежень їх існування називається безліччю споживчих властивостей системи. Під поняттям «споживач» розуміється система вищого ієрархічного рівня, що «експлуатує» споживчі властивості вихідної системи.
Системне опис об'єкта через комплексне опис споживчих властивостей дозволяє у межах єдиного методичного апарату вирішити питання вибору досконалішого варіанта рішення і підготувати об'єктивні умови включення людської фантазії. Поняття досконалості дуже складне, тому що до раціональних критеріїв досконалості дуже часто підмішуються такі суб'єктивні уподобання, як зиск, смак, мода тощо.
Загальна тенденція розвитку, т. е. еволюція об'єкта шляху до досконалості, має внутрішні закономірності і можна зрозуміти з урахуванням цих закономірностей. Які ж ці закономірності?
Закономірність 1.З точки зору споживчих функцій: «Чим ширший склад споживчих функцій, чим інтенсивніша кількісна сторона їхнього прояву, тим досконаліша система».

> *приклад.* З двох дівчат одного віку, однакової освіти, які проживали з дитинства до теперішнього моменту пліч-о-пліч в одному місті, за умови, що вони однаково улюблені, неможливо вибрати найкращу за дружину. Якщо ввести додатковий критерій, завдання спрощується. Наприклад, наявність посагу. Однак багатокритеріальне завдання вибору може стати важким і його легко вирішувати лише з використанням спеціальних методик.

Спробуйте самостійно розв'язати завдання: «Яка з відомих вам програм редактора текстів досконаліша?»
Закономірність 2.З погляду впливу чинників довкілля комп'ютера: «Чим ширше інтервал умов довкілля, всередині якого здатні реалізовуватися споживчі властивості конкретної системи, тим система досконаліше».

> *приклад.* З двох дівчат — кандидаток за дружину — набагато краще, ніж невибаглива — це вже зовсім банальна істина.

Чи згодні ви, що з двох програм редактора текстів краща та, яка може виконуватися на ЕОМ з меншою пам'яттю.

Закономірність 3.З погляду інтервалу обмежень штучної довкілля: «Чим вже інтервал обмежень реалізації споживчих функцій цієї системи, тим система досконаліше».

> *приклад.* При виборі майбутньої дружини, очевидно, що за інших рівних умов велику привабливість має наречена, здатна виконувати домашні (та інші) функції без допомоги асистентів у вигляді матінки, різних підозрілих друзів-приятелів і т.д.

Який редактор текстів краще: 1) із вбудованим орфографічним контролем; 2) із зовнішньою програмою орфографічного контролю, розробленою іншим виробником?
*Крок 1*Сформулювати завдання розвитку по кожному із споживчих властивостей.
*Крок 2.*Дослідити та спрогнозувати тенденції розвитку споживчого попиту по кожному із споживчих властивостей.
*Крок 3*На основі прогнозу про тенденції розвитку споживчого попиту скласти повний пріоритетний список усіх можливих завдань удосконалення об'єкта.

> *приклад.* Проведемо системний аналіз електронного архіву (ЕА), що забезпечує доступ до документів та їх зберігання в електронному вигляді. Мета створення ЕА полягає у забезпеченні оперативного та повноцінного доступу до всіх документів, що зберігаються і надходять. Для цього потрібно вирішити два основні завдання: запровадити масив наявних в архіві документів та забезпечити можливість оперативного повнотекстового доступу до електронних документів.

*Крок 1*Перелічимо основні функції ЕА:
• сканування;
• розпізнавання та коригування помилок;
• створення та міграція електронних документів та образів;
• індексування документів;
• оперативний пошук та відображення документів.
Для реалізації даних функцій в ЕА повинні бути підсистеми введення, зберігання, індексування, пошуку та відображення інформації, аналізу, управління потоками, адміністрування та науково-технічного супроводу.
У системі можна виявити наступний ряд обмежень на реалізованість споживчих функцій:
- неможливість зберігання образу документів з використанням магнітних дискових носіїв внаслідок їх високої вартості та невисокої надійності без багаторазового резервування;
— непридатність використовуваних нині офісних сканерів (не дозволяють вводити документи на паперових носіях низької якості: рукописні, зліплі, вицвілі, порвані, різних розмірів та щільності, погано про друковані, забруднені тощо);
— СУБД, особливо реляційного типу, спочатку не орієнтовані на інтенсивну обробку надвеликого обсягу інформації.
*Крок 2.*Завдання проектування:
1) розгортання високопродуктивної мережі, що включає графічні робочі станції та потужні сервери введення та обробки інформації;
2) використання сканерів та відповідні русифіковані програмні засоби для введення документів з паперових носіїв низької якості;
3) забезпечення ефективного індексування та повнотекстового пошуку неструктурованої інформації великого обсягу.
*Крок 3*Можливість технічної реалізації аналізованої системи:
- З'явилися дешеві носії - компактні диски; різко знизився показник вартість/продуктивність для високошвидкісних обчислювальних систем, мереж та пристроїв;
— набули розвитку апаратно-програмні системи, що реалізують паралельну обробку запитів; підвищився рівень інтерфейсу роботи із СУБД;
- З'явилися нові інформаційні технології індексування надвеликих масивів даних;
— розроблено та розвиваються вітчизняні технології та програмні продукти розпізнавання та аналізу російськомовних текстів;
— намітився напрямок впровадження засобів штучного інтелекту, що дозволяють моделювати та аналізувати великі масиви інформації.

*Крок 4.*Як пріоритетні завдання вдосконалення системи можна виділити такі:
1) використання комбінації різних технологій індексування та пошуку. Намітилося кілька напрямів побудови електронних архівів залежно від методів пошуку (використання атрибутного пошуку структурованих даних і повнотекстового індексування неструктурованих даних);
2) використання спеціалізованих промислових сканерів, орієнтованих потокове введення архівних документів. Відмінною рисою таких сканерів є ротаційний механізм переміщення документів, що дозволяє вводити дані з паперових носіїв поганої якості;
3) через високі вимоги до швидкості доступу до пошукового образу документа та його цілісності, здійснення його зберігання у високошвидкісних відмовостійких системах зберігання, наприклад RAID-масивах. Найбільш підходящими носіями можуть бути магнітооптичні, фазоінверсні (PD/CD), компакт-диски (CD-R) і WORM-диски. Для автоматизації пошуку інформації, розміщеної на цих дисках, її вилучення та роботи з дисками використовуються автоматичні бібліотеки, або оптичні дискові автомати (JukeBox);
4) використання тільки потужних RISC-платформ, що масштабуються, орієнтованих на паралельні обчислення.
Поданий спосіб опису та завдання споживчих властивостей систем дозволяє деталізувати результати тенденцій розвитку споживчого попиту, перекласти їх на мову розробників, поставити орієнтири превентивного вдосконалення систем.
Взагалі прагнення враховувати у будь-якій діяльності вимоги до кінцевого результату є проявом дії механізму зворотний зв'язок, воно підвищує керованість і спрямованість діяльності, отже, і якість результату. Образ ідеального рішення таки служить як порівняння між собою конкретних типів пошукової діяльності, а й утримання процесу пошуку у певних рамках, направляючи його до необхідному результату. Межі цих рамок можуть бути задані наступними ознаками ідеальності (вони ж критерії порівняння та вибору).
*Ознака 1.*Порівняння за рівнем розвитку споживчих властивостей, що максимально досягається.
*Ознака 2.*Порівняння систем та пошук рішення на основі максимального резерву розвитку.

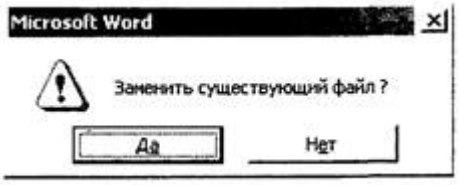
2.6. ПСИХОФІЗІОЛОГІЧНІ ОСОБЛИВОСТІ ВЗАЄМОДІЇ ЛЮДИНИ ТА ЕОМ

Психофізичні особливості взаємодії людини та ЕОМ - науково-дослідний напрямок, що вивчає процеси, що відбуваються в людино-машинній інформаційній системі.
ЕОМ доповнює людину, але не замінює її, тому розгляд основних особливостей їхньої співпраці необхідний.
Логічний метод міркування. У людини він ґрунтується на інтуїції, використанні накопиченого досвіду та уяві. Метод ЕОМ - суворий та систематичний. Найбільш вдалим є поєднання реалізованих на ЕОМ окремих розрахункових процедур із визначенням людиною їхньої логічної послідовності.
Здатність до навчання. Людина навчається поступово, ступінь «освіченості» ЕОМ визначається її програмним забезпеченням. Бажано, щоб кількість інформації, яка отримується на запит користувача, була змінною і могла змінюватися на вимогу користувача.
Поводження з інформацією. Місткість мозку людини для збереження деталізованої інформації невелика, але мозок має інтуїтивну, неформальну можливість організації інформації. Ефективність вторинного звернення до пам'яті залежить від часу.
У ЕОМ ємність пам'яті велика, організація формальна і деталізована, вторинне звернення залежить від часу. Тому доцільно накопичувати та організовувати інформацію автоматичним шляхом та здійснювати її швидкий виклик за зручними для людини ознаками.
Оцінка інформації. Людина вміє добре розділяти значну та несуттєву інформацію. ЕОМ такою властивістю не має. Тому повинна існувати можливість перегляду макроінформації великого обсягу, що дозволяє людині вибрати частину, що його цікавить, не вивчаючи всю накопичену інформацію.
Ставлення до помилок. Людина часто припускається суттєвих помилок, виправляючи їх інтуїтивно, при цьому метод виявлення помилок найчастіше також інтуїтивний. ЕОМ, навпаки, не виявляє жодної терпимості до помилок і метод виявлення помилок суворо систематичний. Проте у сфері формальних помилок можливості ЕОМ значно більше, ніж за виявленні неформальних. Тому потрібно забезпечити можливість користувачеві вводити в ЕОМ вихідну інформацію у вільній формі, написану за правилами, близькими до звичайних математичних виразів та розмовної мови. ЕОМ виконує контроль та наводить

інформацію до стандартного вигляду, зручного в процедурах обробки та формального усунення помилок. Потім бажано зворотне перетворення цієї інформації для показу користувачеві в наочній, наприклад, графічній формі, для виявлення смислових помилок.
Поводження зі складними описами. Людині важко сприйняти велику кількість інформації. Тому слід доручати ЕОМ автоматичне розбиття складних змін щодо незалежні частини, охоплювані одним поглядом. Природно, що зміни, зроблені в одній із цих частин, повинні автоматично проводитися у всіх інших.
Розподіл уваги кілька завдань. Виконати цю умову людині переважно не вдається. При вирішенні під завдання доводиться відволікатися від основного завдання. Тому в ЕОМ повинна бути організована система переривань, що відновлює стан основного завдання на момент, необхідний користувача. Аналогічним чином ЕОМ обслуговує процедуру аналізу кількох варіантів рішення.
Пам'ять щодо проведеної роботи. Людина може забути і те, що вже зроблено, і те, що йому заплановано зробити ще. Цей недолік компенсується ЕОМ, яка чітко фіксує та інформує користувача про виконані процедури та майбутню роботу.
Здатність зосереджуватись. Ця здатність у людини залежить від багатьох факторів, наприклад, від тривалості та напруги уваги, впливу середовища, загального стану. Втомою зумовлюються розсіяність, подовження реакцій, недоцільні дії. У зв'язку з цим інтерактивна система з розподілом часу має адаптуватися до часу реакції окремого користувача.
Терпіння. При багаторазовому повторенні тих самих дій людина може відчувати прикрість. Тому передбачається, наприклад, введення вихідних даних одним масивом при багаторазовому аналізі цих даних. До цього відноситься включення в систему макрокоманд або гнучких сценаріїв.
Самопочуття. ЕОМ має берегти самопочуття користувача, його почуття власної гідності та показувати йому, що саме машина його обслуговує, а не навпаки. Питання, відповіді та зауваження повинні відповідати розмові між підлеглим та його керівником, що визначає хід та напрямок роботи.
Емоційність. Це почуття властиве людині і далеке від ЕОМ. Програми повинні збуджувати у користувача позитивні емоції та не допускати негативних емоцій.

2.7. КЛАСИФІКАЦІЯ ТИПІВ ДІАЛОГУ ПРОГРАМ

В даний час поширені такі діалоги типу: питання-відповідь; вибір із меню; заповнення бланків; на основі команд; роботи у вікнах; за принципом електронної таблиці; гіпертексту; наближення до природної мови; віртуальної реальності.
Діалог типу питання-відповідь, поданий на рис. 2.3 є одним з універсальних. За запитом можливе введення значень різних типів. У найпростішому випадку діалог може бути реалізований із використанням трьох кнопок: так, ні, почати діалог спочатку. Проблема при реалізації діалогу полягає у скруті забезпечення перегляду користувачем усієї передісторії питань програми та відповідей на них.



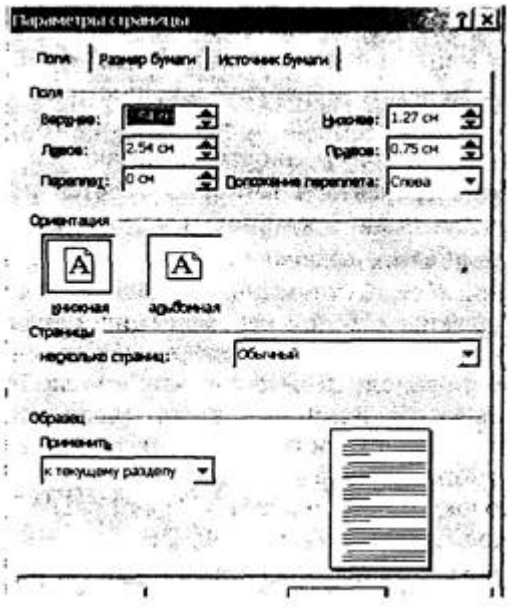
*Мал. 2.3.*Діалог типу питання-відповідь

Діалог типу меню не є універсальним. Досить складно реалізувати з використанням діалогу даного типу введення значень у широкому діапазоні. Вертикальні меню краще горизонтальних. Небажано в окремих меню пропонувати більше семи тем. Меню з великою кількістю тем бажано уявити системою ієрархічної системи підменю. Часто використовується в програмах меню в стилі фірми «Lotus» з одного головного горизонтального меню та ієрархічно побудованих інших вертикальних підменю. Рядок єдиного головного горизонтального меню надає впевненість користувачу-початківцю шляхом повідомлення йому факту про те, що є підменю в програмі. Діалог типу меню представлений на рис. 2.4.

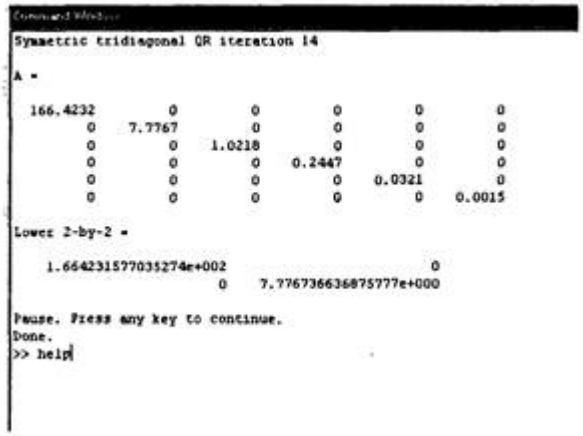
Діалог типу заповнення бланків відображає на окремому екрані якийсь бланк. За допомогою кліку миші або натискання клавіш <Tab>, а також клавіш-стрілок переміщення курсору забезпечується підведення курсору на будь-яке з виділених полів введення інформації. Зазвичай під час заповнення будь-якого поля бланка здійснюється контроль кодів натиснутих клавіш (введення по масці). Проблема реалізації діалогу полягає у прийнятті рішення за фактом введення неприпустимої за значенням інформації в полі або введення неприпустимої сукупності значень до ряду полів. Діалог типу заповнення бланків подано на рис. 2.5.

Діалог на основі команд, представлений на рис. 2.6, використовується в системах з апріорно незаданою послідовністю роботи. Складність використання діалогу цього типу полягає у необхідності попереднього вивчення користувачем мови команд та можливих послідовностей команд. Якщо команд мало у списку можливих команд, то користувачу на запит «?» може бути виданий перелік можливих команд. Після набору імені команди можливе видання списку полів команди та їх призначення.

*Мал. 2.4.*Діалог типу меню

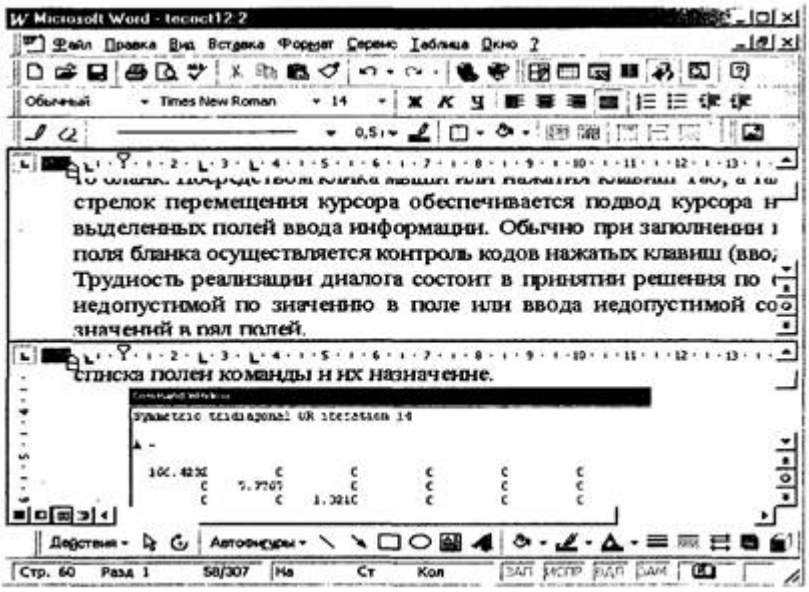


*Мал. 2.5.*Діалог типу заповнення бланків



*Мал. 2.6.*Діалог на основі команд

Діалог типу роботи у вікнах (Мал. 2.7) дозволяє на екрані монітора організувати сукупність вікон як окремих програм, так і окремих документів і здійснювати роботу в кожному з окремих вікон. Даний діалог необхідний фахівцям, «у яких стіл завалений паперами», тобто фахівцям, яким для отримання нового документа потрібна інформація одразу з кількох документів.
Діалог типу за принципом електронної таблиці відображає інформацію у двовимірній системі координат за принципом гри «у морський бій» і в режимі «що бачу, те і отримаю в роздруківці» дозволяє надати табличній інформації форму, необхідну користувачеві (рис. 2.8).
Діалог типу гіпертексту(Мал. 2.9) забезпечує користувачеві перегляд на екрані монітора текстової (графічної, відео, аудіо та ін.) інформації з можливістю отримання додаткової інформації при виборі користувачем виділених на екрані ключових слів (посилань).
Діалог наближення до природної мови зазвичай забезпечує видачу користувачем запитів обмеженою природною мовою. Найбільшу складність при реалізації діалогу даного типу є складання обмеженого тезаурусу слів запиту та вивчення даного тезаурусу користувачем.



*Мал. 2.7.*Діалог типу роботи у вікнах

	E	F	G	H
1	Цена	Тип товара	Ссылка	
2	40 руб.	книга	http://www.books.ru/books/6549	
3	40 руб.	книга	http://www.books.ru/books/8341	
4	167 руб.	книга	http://www.books.ru/books/12404	
5	65 руб	книга	http://www.books.ru/books/15714	
6	40 руб	книга	http://www.books.ru/books/2289	
7	48 руб.	книга	http://www.books.ru/books/19190	
8	56 руб.	книга	http://www.books.ru/books/15389	
9	40 руб.	книга	http://www.books.ru/books/7732	
10	48 руб	книга	http://www.books.ru/books/8560	
11	40 руб.	книга	http://www.books.ru/books/701	
12	40 руб	книга	http://www.books.ru/books/9280	
13	76 руб.	компакт-диск	http://www.books.ru/books/13062	
14	77 руб	компакт-диск	http://www.books.ru/books/18668	
15	800 руб.	компакт-диск	http://www.books.ru/books/16654	

Мал. 2.8. Діалог типу за принципом електронної таблиці

*Мал. 2.9.*Діалог типу гіпертексту

Діалог типу віртуальна реальність використовується в різних тренажерах і може ґрунтуватись на використанні особливого обладнання типу кібершолом, тактильні рукавички, система запахів тощо. Фонд різних діалогів полегшує вибір оптимального варіанта побудови зовнішніх специфікацій програм. Список дрібніших «будівельних елементів» діалогу можна отримати на панелі компонентів систем візуального програмування, наприклад Delphi.

ВИСНОВКИ

• На різних етапах проектування (особливо часто на початкових етапах) перед розробником постає завдання вибору найкращого варіанта з безлічі допустимих проектних рішень, які задовольняють вимогам. Прийняття «правильного» рішення означає вибір такої альтернативи з-поміж можливих, в якій з урахуванням усіх різноманітних факторів буде оптимізовано загальну цінність.

• Завдання оптимізації розробки програм полягає у досягненні цілей за мінімально можливої витрати ресурсів. Системний аналіз на відміну попереднього системного дослідження — це поглиблене вивчення інформаційних потреб користувачів, яке буде покладено основою детального проектування нової інформаційно-програмної системи (АС). Формулювання цілей розробки програмного продукту — перший і найважливіший етап процесу проектування. Помилки у виборі та формулюванні мети не можуть бути компенсовані на наступних етапах. Аналіз вимог сприяє кращому розумінню системи, і навіть досягненню найкращого задоволення потреби.

• Сукупність функцій системи, умов та обмежень їх існування називається безліччю споживчих властивостей системи. Функції системи — це властивості, що зумовлюють корисність (доцільність системи споживача).

• Будь-які класифікації програм підвищують можливість синтезу варіантів. Класифікації можна використовувати як під час роботи методом морфологічного синтезу, і методом аналогії.

• Перед тим, як шукати шляхи оптимізації розробки програм, необхідно виділити деякі ключові положення.
*Положення 1.*Проблема, яка має бути вирішена програмою, що розробляється, раніше якимось чином вирішувалася. Отже, необхідно вивчити методи, які раніше застосовувалися, по можливості, їх формалізувати і застосувати.
*Положення 2.*Для переважної кількості завдань у час існують програми, виконують схожі чи аналогічні функції. Тому необхідною умовою якісної розробки є ознайомлення з існуючими аналогами.
Контрольні питання
1. Перерахуйте основні види показників якості програмних систем.
2. Дайте визначення поняття "евристичний прийом".
3. Опишіть основні кроки, якими здійснюється системний аналіз.
4. Опишіть сутність проектної процедури розкриття проектної ситуації.
5. Наведіть приклади внутрішніх закономірностей під час опису споживчих властивостей системи.
6. Назвіть основні класи програм.
7. Назвіть основні особливості взаємодії людини та ЕОМ.
8. Дайте коротку характеристику основних типів діалогу програм.
9. У чому основне завдання оптимізації на етапі розробки програм?