ЛЕКЦІЯ №7

СУЧАСНІ ПРОГРАМНІ ЗАСОБИ ПОБУДОВИ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ

ПЛАН

7.1 Об'єктно-орієнтована мова Visual Basic.
7.2 Можливості мови Visual Basic для створення ЕС.
7.3 Мова логічного програмування Visual Prolog.
7.4 Інтегроване інструментальне середовище GURU.
7.5 Інтегроване інструментальне середовище G2 для створення інтелектуальних систем реального часу.

Час: 4 години.
Література: [1-23]

7.1 Об'єктно-орієнтована мова Visual Basic

Visual Basic – мова, яка підтримує подвійно-кероване програмування (event-driven programming): візуальне проектування й елементи об'єктно-орієнтованого програмування. Випустивши в 1991 р. першу версію VB, Microsoft досить скромно оцінювала можливості цієї системи, орієнтуючи її, насамперед, на категорію початківців і непрофесійних програмістів. Основним завданням тоді було випустити на ринок простий і зручний інструмент розробки в тоді ще досить новім середовищі Windows, програмування в якій представляло проблему й для досвідчених фахівців. В 1992 р. була випущена друга версія, а в 1993-94 рр. - третя версія. Ця версія дозволила продукту увійти до числа серйозних інструментальних засобів програмування й значно розширити свій ринок.

1996-97 рр. була випущена п'ята версія; В VB5 було багато вдосконалень, він забезпечував помітно більш високу продуктивність і пропонував довгоочікуваний компілятор, що перетворить програму у внутрішній машинний код.

В 1998 р. з'явився Visual Basic 6.

Протягом декількох років ідуть постійні дебати про те, може чи Visual Basic уважатися мовою об'єктно-орієнтованого програмування (ООП). З одного боку, елементи ООП у ньому були завжди, і їхнє число росло від версії до версії. З іншого боку — багатьох потрібних можливостей ООП в Visual Basic не було. Поява Visual Basic.NET повинне покласти кінець усім цим дискусіям, тому що в ньому будуть реалізовані всі необхідні атрибути ООП. Нагадаємо, що модель ООП має на увазі наявність трьох обов'язкових механізмів інкапсуляції, поліморфізму й спадкування. Перші два минулі реалізовані в попередніх версіях і

одержали розвиток у новій, а останній з'явиться в ній уперше.

Visual Basic нарешті став повноцінною об'єктно-орієнтованою мовою. Безумовно, Visual Basic.NET серйозно додав у потужності засобів, але працювати з ним буде складніше.

Адже об'єктно-орієнтовані методи програмування пред'являють більш серйозні вимоги до кваліфікації розроблювача, на которою перекладаються багато проблем забезпечення працездатності програми.

Дамо також опис деяких нових елементів мови на концептуальному рівні.

Web Services - це якась принципово нова платформно-незалежна технологія, пов'язана з використанням стандарту XML і протоколу SOAP (Simple Object Access Protocol - протокол доступу до простих об'єктів), яка буде широко інтегрована в засоби розробки. Ключова ідея полягає в створенні компонентів рівня бізнес-логіки**,** які взаємодіють із зовнішніми об'єктами за допомогою стандартних Web-Протоколів.

Object-oriented: Для того, щоб називатися об'єктно-орієнтованим, мова повинен задовольняти трьом критеріям:

1. Він повинен підтримувати інкапсуляцію(encapsulation). VB робить це з 4 версії

2. Зобов'язано підтримувати спадкування (inheritance), VB 7 буде мати повноцінне спадкування.

3 Ну й підтримка поліморфізму(ро1утогрш8т). В VB це працює з 4 версії. Отже, в 7 версія буде цілком задовольняти цим критеріям.

Encapsulation **(**Інкапсуляція**).** Ідея полягає в тому, що ви можете створювати схований набір процедур (методів і властивостей), які формують якийсь програмний інтерфейс. Інший код може звертатися до цих методів і властивостям, не вдаючись у подробиці їх внутрішньої реалізації.

Free-threaded (Многопоточность). Це комплексна концепція підтримки виконання більш ніж одного потоку завдань у те саме час. Наприклад, користувач може продовжувати працювати з додатком, після того як він задав операцію фонової печатки документа. Подібний режим украй необхідний для створення масштабованих серверних компонентів і може бути корисний для реалізації користувацького інтерфейсу. Створення таких обчислювальних потоків виконується приблизно в такий спосіб:

Inheritance (Спадкування). Це одне із ключових понять об'єктно-орієнтованого програмування можливість використання (у тому числі розширення) поведінки чужого об'єкта. Спрощено говорячи, можна створити об'єкт Продукт, а потім на його основі об'єкти Програмний Продукт і Технічний Продукт. Обоє нових об'єкта будуть успадковувати властивості й методи об'єкта Продукт, і при цьому ви зможете змінити поведінку об'єкта, що успадковує. Visual Basic-Розроблювачі тепер можуть використовувати ключове слово Inherits для підключення процедур уже існуючого класу:

Overloading. У російській мові немає відповідного терміна в даному контексті: використання того самого ідентифікатора для позначення різних процедур. Вибір потрібної процедури виконується залежно від числа й типу параметрів. Це особливо корисно для створення одного властивості, що підтримує різні типи аргументів.

Polymorphism (Поліморфізм). Можливість мати кілька об'єктів різного типу, але з однаковими методами. Це дозволяє писати код, що викликає той метод, який потрібний залежно від використовуваного в цей момент об'єкта.

Structured Exception Handling (Структурна обробка особливих ситуацій). Це нова структура для обробки помилок, уже реалізована в багатьох мовах. Вона повинна замінити стару й досить негнучку (точніше, ненаглядну) конструкцію On Error Goto|resume|next. Новий блок містить ключові слова Try, Catch, Finally:

Type Safety (Контроль типів даних). Заборона неявного перетворення типів за допомогою нового оператора Option Strict. Комусь із програмістів це не сподобається, тому що даний режим змушує задуматися про типи змінних і використовувати спеціальні функції при присвоєнні змінної значення інший змінної іншого типу. Але це зовсім необхідно, якщо ви прагнете писати надійні програми й знизити витрати на налагодження додатків.

User Interface Inheritance (Спадкування користувацького інтерфейсу). VB7 буде включати спадкування форм, тобто створення нових форм на основі деяких шаблонів. На відміну від існуючого сьогодні механізму підключення нових форм на основі шаблонів, у цьому випадку буде автоматично підтримуватися механізм спадкування: зміни в батьківському шаблону (наприклад, корпоративному логотипі) будуть відбиватися 1 у дочірніх формах.

7.2 Можливості мови Visual Basic для створення ЕС.

В основу програмного пакета був покладений передовий архітектурний розв'язок, що дозволяє не писати, а проектувати програми, подібно інженерові-дизайнерові. Інакше кажучи, у ньому був одним з перших реалізований популярний нині стиль візуального програмування. І ключовим словом у назві мови є Visual -екранні форми й безліч вбудованих компонентів (текстові, графічні вікна, кнопки, діалоги й т.п.) рятують від складностей, зв’язані з виведенням, обробкою, відновленням усіх цих елементів, що особливо важливо при розробці експертної системи, для якої легка модифицируемость є ледве чи не найбільш важливою відмітною рисою.

В Visual Basic реалізоване "керування від подій", як ми вже відзначали раніше, що дозволяє розроблювачеві спроектувати інтерфейс користувача максимально зручно для користувача. Після етапу візуального проектування програміст просто пише програмний код для обробки пов'язаних з об'єктом подій, VB надає зручне середовище для розробки додатків, тестування їх роботи й знаходження й виправлення помилок.

Розглянемо основні принципи, покладені в основу Visual Basic, що й дозволяють максимально полегшити праця розроблювача й зробити роботу з написаним додатком максимально комфортним.

Об'єкт одне з основних понять Visual Basic і об'єктно-орієнтованого програмування. Об'єктом називається якась сутність, яка, по-перше, чітко проявляє свою поведінку, а по-друге, є представником деякого класу подібних об'єктів.

Класом об'єктів в об'єктно-орієнтованих мовах називається загальний опис таких об'єктів, для яких характерна наявність безлічі загальних властивостей і дій.

ПОДІЯ - дія або ситуація, пов'язана з об'єктом, наприклад: клацання миші або натискання клавіші. Події можуть ініціюватися в програмному коді додатка або безпосередньо в системнім середовищі.

ВЛАСТИВОСТІ - визначають вистава, поведінка й інші риси об'єкта. Колір тла форми, рядок з'єднання (у сеансі БД), розмір елементів і т.п. - усе це властивості тих або інших об'єктів.

Властивості: Caption - ім'я, Enabled - доступ, Height - висота.

МЕТОДИ - програмні процедури, які виконують деяку обробку, пов'язану з об'єктом, певні дії над об'єктом.

Об'єктно-орієнтована модель аналогічна багато в чому фреймовій моделі, реалізуючи обмін повідомленнями між об'єктами, більшою мірою орієнтована на розв'язок динамічних завдань і відбиття поведінкової моделі. Відмінності від фреймової моделі полягають у чіткім понятті клас об'єктів і екземпляр об'єкта, а також у способі активації процедур об'єктами.

Перелічимо й дамо коротку характеристику тем засобам VB які допомагають у розробці професійних комерційних додатків взагалі й експертних систем зокрема.

Пакет Microsoft Visual Basic являє собою ідеальну платформу для створення інтерфейсів з локальними базами даних і базами даних типу клієнт/сервер у середовищі Windows.

Visual Basic дозволяє створювати різні додатки для роботи з базами даних - від найпростіших локальних баз даних до багаторівневої архітектури клієнт/сервер, а також додатків для роботи в intranet і Internet з використанням таких передових технологій, як DHTML, XML і ASP.

Також дуже важливі питання проектування реляційних баз даних і використання мови SQL. Мова структурованих запитів є стандартним засобом для роботи з базами даних і може використовуватися як для інтерактивної роботи з базами даних, так і включатися в мови програмування. Стосовно до Visual Basic SQL дозволяє:

- створювати, модифікувати або видаляти таблиці в базі даних Access;
- вставляти, видаляти або модифікувати записи таблиць;
- одержувати зведену інформацію про дані в таблиці;

•пошук даних в одній або більш таблицях по запиту.

Експертна система, як ніяка інша, повинна надавати користувачеві максимально "дружелюбний" інтерфейс, оскільки вона в більшості випадків є діалоговою й вимагає від користувача максимальної взаємодії.

Це справедливо у великому ступені для так званих оболонок експертних систем, правильність настроювання й заповнення яких є ключовим чинником у побудові адекватно реагуючої експертної системи. У даній ситуації багато чого залежить від логіки вистави інформації й керуючих елементів у програмі, наявності користувацького меню й розвиненою довідковою системою.

Visual Basic надає розроблювачеві всі засоби для створення «графічного інтерфейсу користувача»:

•можливість побудови багато документального інтерфейсу;

•створення користувацького меню;

•травлення ACTIVE-X і багато чого ін.

У висновку приведемо деякі подробиці реалізації механізму висновку за допомогою дерев.

В ЕС прийнято представляти процес логічного висновку у вигляді схеми, яка показується деревом висновку. У системах, база знань яких нараховує сотні правил, досить бажаним є використання якої-небудь стратегії керування висновком, що дозволяє мінімізувати час пошуку розв'язку й тим самим підвищити ефективність висновку. До таких стратегій належать пошук у глибину, пошук завширшки, розбивку на підзадачи.

Методи сліпого перебору, повного перебору або пошуку в глибину є вичерпними процедурами пошуку шляхів до цільової вершини. У принципі ці методи забезпечують розв'язок завдання пошуку шляхи, але часто ці методи неможливо використовувати, оскільки при переборі прийде розкрити занадто багато вершин. Перш ніж потрібний шлях буде знайдений. Для багатьох завдань можна сформулювати правила, що дозволяють зменшити обсяг перебору. Усі такі правила, використовувані для прискорення пошуку, залежать or специфічної інформації про завдання, що представляється у вигляді графа. Будемо називати таку інформацію евристичною інформацією (, що допомагає знайти розв'язок) і називати ее процедури, що використовують, пошуку евристичними методами пошуку. Нам необхідний захід, який дозволяв би оцінювати "перспективність" вершин. Такі заходи називають оцінними функціями. Оцінна функція повинна забезпечувати можливість ранжирування вершин- кандидатів на розкриття- для того, щоб виділити ту вершить', яка з найбільшою ймовірністю перебуває на кращому шляху до мети.

Visual Basic дозволяє розроблювачеві створювати дерева й здійснювати пошук по них по кожному з перерахованих вище алгоритмів.

Розглянемо деякі загальні способи реалізації дерев мовою Visual Basic, Один зі способів - створити окремий клас для кожного типу вузлів дерева. Для побудови

дерева порядку 3 можна визначати структури даних для вузлів, які мають нуль, один, два або три дочірні вузли. Цей підхід був би досить незручним.

Крім того, що потрібно було б управляти чотирма різними класами, у класах потрібні були б якісь прапори, які б указували тип дочірніх вузлів. Алгоритми, які оперували б цими деревами, повинні були б уміти працювати з усіма різними типами дерев.

У якості простого розв'язку можна визначити один тип вузлів, який містить залишкове число покажчиків на нащадків для вистави всіх потрібних вузлів. Назвемо це методом *повних вузлів,* тому що деякі вузли можуть бути більшого розміру, чому необхідно па самій справі.

Для побудови дерева порядку 3 з використанням методу повних вузлів, потрібно визначити єдиний клас, який містить покажчики на три дочірні вузли.

Якщо порядки вузлів у дереві сильно різняться, метод повних вузлів приводить до даремної витрати великої кількості пам'яті. Деякі програми додають і видаляють вузли, змінюючи порядок вузлів у процесі виконання. У цьому випадку метод повних вузлів не буде працювати. Такі динамічно мінливі дерева можна представити, помістивши дочірні вузли в списки. Є кілька підходів, які можна використовувати для створення списків дочірніх вузлів. Найбільш очевидний підхід полягає в створенні в класі вузла відкритого масиву дочірніх вузлів, як показано в наступному коді. Тоді для оперування дочірніми вузлами можна використовувати методи роботи зі списками на основі масивів.

Другий підхід полягає в тому, щоб зберігати посилання на дочірні вузли у зв'язних списках. Кожний вузол містить посилання на першого нащадка. Він також містить посилання на наступного нащадка на тому ж рівні дерева. Ці зв'язки утворюють зв'язний список вузлів одного рівня, тому я називаю цей метод виставою у вигляді зв'язного списку вузлів одного рівня.

Третій підхід полягає в тому, щоб визначати в класі вузла відкриту колекцію, яка буде містити дочірні вузли.

Вистава *нумерацією зв'язків* дозволяє компактно представити дерева, графи й мережі за допомогою масиву. Для вистави дерева нумерацією зв'язків, у масиві Firstlink записується індекс для перших галузей, що виходять із кожного вузла. В інший масив, Tonode, заносяться вузли, до яких веде галузі.

Сигнальна мітка наприкінці масиву Firstlink указує на крапку відразу після останнього елемента масиву Tonode. Це дозволяє визначити, які галузей виходять із кожного вузла.

Використовуючи виставу нумерацією зв'язків, можна швидко знайти зв'язки, що виходять із певного вузла. З іншого боку, дуже складно змінювати структуру даних, представлених у такому виді.

Послідовний обіг до всіх вузлів називається обходом дерева. Обхід у глибину починається із проходу вглиб дерева доти, поки алгоритм не досягнеться листів. При поверненні з рекурсивного виклику підпрограми, алгоритм

переміщається по дереву у зворотному напрямку, переглядаючи шляху, які він пропустив при русі вниз.

Інший метод перебору вузлів дерева - це обхід завширшки. Цей метод звертається до всіх вузлів на заданому рівні дерева, перед тем, як перейти до більш глибоких рівнів. Алгоритми, які проводять повний пошук по дереву, часто використовують обхід завширшки.

Деталі реалізації обходу залежать від того, як записано дерево. Для обходу дерева на основі колекцій дочірніх вузлів, програма повинна використовувати трохи інший алгоритм, чому для обходу дерева, записаного за допомогою нумерації зв'язків.

Особливо просто обходити повні дерева, записані в масиві. Алгоритм обходу завширшки, який вимагає додаткових зусиль в інших виставах дерев, для вистав на основі масиву тривіальний, тому що вузли записані в такому ж порядку.

7.3 Мова логічного програмування Visual Prolog.

У жовтні 1981 р. була широко поширена інформація про японський проект створення ЕОМ п'ятого покоління. Багатьох фахівців здивувало, що в основу методології розробки програмних засобів було покладено логічне програмування. Метою проекту декларувалося створення систем обробки інформації, що базуються на знаннях. Тоді ж з'являється безліч комерційних реалізацій Прологу практично для всіх типів комп'ютерів. До найбільш відомих можна віднести Cprolog, Quintus Prolog. Silogic Knowledge Workbench. Prolog-2, Arity Prolog, Prolog-86, Turbo Prolog і ін.

Найбільшу популярність у нашій країні одержала система програмування Turbo Prolog — комерційна реалізація мови для Ibm-Сумісних ПК. Його перша версія була розроблена датською компанією Prolog Development Center (PDC) у співдружності з фірмою Borland International в 1986 р. Система створювалася із серйозними відступами від неофіційного стандарту мови, самим істотним з яких було введення строгої типізації даних, але це дозволило значно прискорити трансляцію й виконання програм. Новий компілятор відразу ж був по гідності оцінений праграммистами-практиками, хоча й викликав критикові в академічних колах.

В 1988 р. вийшла значно могутніша версія Turbo Prolog 2.0, що включає вдосконалене інтегроване середовище розробки програм. швидкий компілятор і засобу низкоуровневого програмування. Крім того, вона надавала можливість роботи із власними зовнішніми БД, dbase 111 і Reflex, інтегрованим пакетом Lotus 1-2-3. графічним пакетом Paint Brush і іншими додатками. Фірма Borland поширювала цю версію до 1990 р., а потім компанія PDC придбала монопольне право на використання вихідних текстів компілятора й подальше просування системи програмування на ринок за назвою PDC Prolog. У червні 1992 р. з'явилася версія 3.31 -ефективний універсальний інструмент професійних програмістів, який

незабаром став одним з найбільше широко використовуваних. PDC Prolog 3.31 працював у середовищі MS DOS, OS/2, UNIX, XENIX, Pharlap DOS Extender, MS Windows. Ця версія була добре сумісна із традиційними мовами програмування, у першу чергу із Си. У ній були розширені можливості створення додатків з інтерфейсом GUI (Graphical User Interface), прийнятим в MS Windows і OS/2.

Хоча версія PDC Prolog 3.31 уже включала засобу для написання програм, що працюють під керуванням графічних операційних систем, процес розробки подібних додатків усе ще носив рутинний характер. Для того щоб зробити більш простими, зручними й швидкими процеси написання, тестування й модифікації додатків мовою PDC Prolog, фахівці Prolog Development Center створили систему програмування за назвою Visual Prolog 4.0, випущену 7 січня 1996 р.

При розробці додатків у середовищі Visual Prolog використовується підхід, що одержав назва «візуальне програмування», при якім зовнішній вигляд і поведінка програм визначаються за допомогою спеціальних графічних засобів проектування без традиційного програмування алгоритмічною мовою. У результаті одержали систему програмування, що відрізняється винятковою логічністю, простотою й ефективністю.

В Visual Prolog входять різні елементи: насамперед, інтерактивне середовище візуальної розробки (VDE - Visual Develop Environment), яка включає текстовий і різні графічні редактори, інструментальні засоби генерації коду, що конструюють керуючу логіку (Experts), а, що також є розширенням мови інтерфейс візуального програмування (VPI - Visual Programming Interface), Пролог-Компілятор, набір різних файлів, що підключаються, і бібліотек, редактор зв'язків, файли, що містять приклади й допомога.

Visual Prolog підтримується різними ОС, у тому числі MS-DOS Pharlap-extended DOS. усіма версіями Windows. 16- і 32-бітовими цільовими платформами OS/2, а також деякими іншими системами, що вимагають графічного користувацького інтерфейсу.

Залежно від обраного інтерфейсу розроблювачеві забезпечується доступ до безлічі генераторів коду (Code Expert), усіляким ресурсним редакторам і особливим додатковим Vpi-Предикатам. визначенням і бібліотекам. Ресурсні редактори застосовуються для створення, компонування й редагування вікон, діалогів, меню, панелей інструментів, рядків допомоги, строкових таблиць, ярликів, курсорів, бітових карт і оперативної допомоги. Генератори коду на основі подібних структур створюють необхідний первинний Prolog-Код. У результаті з'являється первинний код («кістяк»), готовий для компіляції, редагування зв'язків і виконання.

За бажанням програміста генератори коду можуть відобразити будь-яку частину первинного тексту програми у вікні редактора для перегляду й доповнення відповідно до логіки додатка, а також для перетворення «кістяка» у повноцінний додаток. Цей процес реалізується за допомогою різних функцій: редагування, вибору, пошуку, переміщення й вставки.

Прикладний програмний інтерфейс високого рівня полегшує проектування Пролог-Додатків з витонченим видовим користувацьким розв'язком, що використовують графічні можливості сучасних ОС і апаратних засобів відображення інформації.

Ресурси й інструментальні засоби, що вимагаються для таких додатків (вікна, меню, діалоги, органі керування, пір'я, кисті, курсори миші, графічні курсори, малюнки й т.п.), представляються у вигляді нескладних Пролог-Структур.

За допомогою VPI можна створити мобільний вихідний текст, а потім перекомпілювати його для роботи як в 16-бітовому режимі під керуванням MS DOS або Windows, так і в 32-бітовому режимі під керуванням Windows NT, OS/2 PM і інших ОС.

У грудні 1997 р. фірма PDC випустила Visual Prolog 5.0, а із січня 1999 р. приступилася до поширення версії 5.1.

Пролог - це мова, яка дотепер перебуває в розвитку. Область його застосування постійно розширюється, у нього вносяться нові додаткові функціональні можливості, покликані задовольнити зростаючі потреби користувачів,

Visual Prolog є Prolog-Системою з 100% оболонкою, виконаної в ідеології Visual, що спрощує розробку програм для систем Windows 3.x. Windows 95/98/2000, Windows NT. Середовище розробки додатків системи Visual Prolog включає зручний текстовий редактор, різні редактори ресурсів, засобу розробки гіпертекстових Help-Систем, оптимизирующий компілятор. Visual Prolog автоматизує праця програміста по побудові складних процедур. З його допомогою проектування користувацького інтерфейсу й пов'язаних з ним вікон, діалогів, меню, кнопок, рядків стану й т.п. проводиться в графічнім середовищі. Зі створеними об'єктами відразу ж можуть працювати різні Кодові Експерти, що генерують програмні коди мовою Пролог.

Потужність мови Пролог і візуального середовища розробки програм робить простий і інтуїтивно зрозумілої розробку експертних систем, заснованих на знаннях, систем підтримки прийняття розв'язків, що планують програм, розвинених систем керування базами даних і ін., а також забезпечує підвищення швидкості розробки додатків.

7.4 Інтегроване інструментальне середовище GURU.

В інструментальнім середовищі побудови ЕС GURU, розробленою фірмою Micro Data Base Systems, Inc., методи експертних систем сполучаються з такими засобами обробки даних, як складання електронних відомостей, керування базою даних і діловою графікою, і в такий спосіб формується унікальне середовище для підтримки прийняття розв'язків і розробки прикладних інтелектуальних систем.

Система GURU легка у вживанні для новачків і в той же час є досить ефективною й гнучкою системою для професіоналів - розроблювачів.

У звичайних «інтегрованих» програмних продуктах або кілька окремих програм поміщені в операційне середовище, або трохи, другорядних компонентів вкладаються в головний компонент (як, наприклад, програма обробки електронних відомостей або текстовий редактор).

Труднощі, з якими зустрічаються при таких стилях «інтеграції», добре відомі. Перші труднощі полягає в тому, що користувач змушено переходити назад і вперед по окремих програмах і передавати дані між ними.

Метод вкладень змушує користувача виконати всю обробку в межах головного компонента, і в результаті виходять відносно слабкі вторинні компоненти.

Метод Інтеграції, використовуваний у системі GURU, зовсім відрізняється від вищезгаданих. Він ґрунтується на принципі синергизма. При цьому під «синергизмом» тут розуміється наступне. У системі GURU усі засоби завжди доступні. Численні компоненти можна з'єднувати за бажанням у межах однієї операції, а це характеризує систему як гнучку й зручну у використанні. Наприклад:

- у посилці будь-якого правила ЕС можна робити прямі посилання на поля бази даних, на гнізда електронних відомостей, на статичні змінні, на програмні змінні й масиви;
- виведення будь-якого правила ЕС може містити в собі операції керування базою даних, запити мовою SQL (мові структурованих запитів), операції обробки електронних відомостей, генерацію статистичних даних, дистанційну передачу даних, виконання процедур, генерацію ділової графіки
- оскільки ЕС обґрунтовує завдання, вона може брати консультації в інших ЕС, виконувати процедурні моделі, переглядати бази даних, становити електронні відомості або проводити статистичний аналіз;
- будь-яке гніздо електронної відомості можна визначити в термінах пошуку в базі даних або в термінах усієї програми, або в термінах консультації з ЕС.

Взаємодіяти із системою можна кожним із чотирьох різних способів: за допомогою меню: на обмеженій природній мові, у режимі команд або через спеціально розроблені інтерфейси. Кожний тип інтерфейсу системи GURU призначений для задоволення потреб і смаків різних класів користувачів. Усіма чотирма інтерфейсами можна користуватися під час того самого сеансу взаємодії із системою GURU.

Як і в більшості оболонок, в GURU використовується продукційна модель вистави знань у вигляді сукупності «If-then» правил зі зворотною стратегією висновку в якості основної є можливість моделювання нечітких і неточних міркувань. Крім посилок і висновку в правила можна включати команди, які

будуть виконуватися перед перевіркою умови, а також пояснювальний текст для генерації пояснень. Правила також включають необов'язкові параметри ціни й пріоритету, що дозволяють управляти процесом вибору із сукупності, готових, до виконання правил чергового. З кожним правилом можна також зв'язати число, що визначає, скільки раз це правило може виконуватися в процесі консультації.

Правила, що ставляться до розв'язку деякого загального завдання, утворюють базу знань, або набір правил.

У цей набір крім: властиво правил включаються дві спеціальні процедури: ініціалізація й завершення, які повинні виконуватися до й після виконання правил.

У набір правил також включаються описи змінних, що брав участь у правилах специфікації, що містять, типу, точності й т.п.

За замовчуванням в GURU прийнята стратегія зворотного висновку, однак, можна використовувати чисто пряме виведення, а також комбінувати його зі зворотним у рамках одного набору правил. Як стратегіями висновку, так і цільовими змінними можна управляти динамічно в процесі консультації.

GURU забезпечує потужні засоби керування обробкою факторів упевненості, що відбивають ступінь неточності й нечіткості виражених у правилах евристичних знань. Для надання такої нечіткості в GURU з кожним значенням змінної може бути зв'язаний числовий коефіцієнт від 0 до 100. Система надає розроблювачеві вибір більш ніж з 30 різних формул, що дозволяють управляти обробкою факторів упевненості під час висновку.

Корисними є такі додаткові засоби керування логічним висновком, як установка ступеня «точності» висновку значення для деякої змінної, зміна прийнятого за замовчуванням порядку перегляду правил.

Ефективність машини логічного висновку багато в чому залежить від того, як вона здійснює пошук у наборі правил, коли шукає правила, які можна виконувати. На відміну від традиційного програмного забезпечення, що використовує принципи штучного інтелекту, система GURU надає розширені кошти керування настроюванням, зокрема підтримує до 50 різних стратегій пошуку. Ефективність також залежить від кількості й складу правил у наборі правил. Оскільки система GURU надає різноманітні можливості створення наборів правил, те можна значно скоротити кількість правил, необхідних для охоплення всіх знань і досвіду в конкретній проблемній області. Це приводить до прискорення процесу одержання логічних висновків, а також до спрощення керування цими правилами.

Інтегрована система GURU намагається перетворити потенційні переваги ЕС у реальність, полегшити користувачеві процес створення ЕС, зробити його прямим, ефективним і природнім.

7.5 Інтегроване інструментальне середовище G2 для створення інтелектуальних систем реального часу.

В 1986 р. фірма Gensym вийшла на ринок з інструментальним засобом G2, версія 1.0. У цей час функціонує вже версія 5.2.

Основне призначення програмних продуктів фірми Gensym (США) допомогти підприємствам зберігати й використовувати знання й досвід найбільш кваліфікованих співробітників в інтелектуальних системах реального часу, І підвищувальних якість продукції, надійність і безпека виробництва, що й знижують виробничі витрати.

Класи завдань, для яких призначена G2 і подібні її системи:

– моніторинг у реальному масштабі часу;
– системи керування верхнього рівня;
– системи виявлення несправностей;
– діагностика;
– складання розкладів;
– планування;
– оптимізація;
– системи - порадники оператора;
– системи проектування.

Інструментальний комплекс G2 є еволюційним кроком у розвитку традиційних експертних систем від статичних предметних областей до динамічних.

Основні принципи, які закладені в G2:

– проблемно/предметна орієнтація;
– проходження стандартам;
– незалежність від обчислювальної платформи;
– універсальні можливості, не залежні від розв'язуваного завдання;
– забезпечення технологічної основи для прикладних систем;
– комфортне середовище розробки;
– розподілена архітектура клієнт-сервер;
– висока продуктивність.

Основною гідністю оболонки експертних систем G2 є можливість застосовувати її як інтегруючий компонент, що дозволяє за рахунок відкритості інтерфейсів і підтримки широкого спектра обчислювальних платформ об'єднати вже існуючі, засобу автоматизації в єдину комплексну систему керування, що охоплює все -аспекти виробничої діяльності - від формування портфеля замовлень до керування технологічним процесом і відвантаження готової продукції [100, 58].

На основі базового засобу G2– фірма Gensym розробила комплекс проблемно/предметно-орієнтованих розширень для швидкої реалізації складних динамічних систем на основі спеціалізованих графічних мов, що включають параметризуемые операторные блоки для вистави елементів технологічного процесу й типових завдань обробки інформації. Набір інструментальних

середовищ**,** згрупований по проблемній орієнтації, охоплює всі стадії виробничого процесу й виглядає в такий спосіб:

– інтелектуальне керування виробництвом - G2 Diagnostic Assistant (GDA), Statistical Process Control (SPC).

– оперативне планування - G2 Scheduling Toolkit (GST), Dynamic Scheduling Packadge (DSP);

– розробка й моделювання виробничих процесів - G2, Rethink.

– керування операціями й корпоративними мережами - Fault Expert [93].

G2 динамічна система в повному змісті цього слова.

Це об'єктно-орієнтоване інтегроване середовище для розробки й супроводу додатків реального часу, що використовують бази знань. G2 функціонує на більшості існуючих платформ: Solaris 1 and 2, Unix, Openvms, Windows NT / 2000 Professional / XP. База знань G2 зберігається у звичайному Ascii-Файлі, який однозначно інтерпретується на кожній з підтримуваних платформ. Перенос додатка не вимагає його перекомпіляції й полягає в простому переписування файлів. Функціональні можливості й зовнішній вигляд додатка не перетерплюють при цьому ніяких змін.

G2 - середовище для розробки й розгортання інтелектуальних динамічних систем керування. Прикладні програми, написані в середовищі G2, можуть суттєво підвищити ефективність виконання операцій, завдяки наступним факторам

– безперервному контролю над потенційними проблемами перш, ніж вони виявлять несприятливий вплив;

– прийняття комплексних оперативних розв'язків на основі інформації, отриманої за допомогою міркувань і аналізу даних, що втримуються в інтелектуальній моделі процесу;

– діагностування основних випадків виникнення проблем, критичних вчасно виконання й вироблення послідовності правильних дій;

– підтримка оптимальних робочих умов;

– координування дій і інформації у виконанні складних оперативних процесах.

Використання потужності об'єктно-орієнтованого програмування

Об'єкти в G2 - це інтуїтивний спосіб вистави матеріальних і абстрактних сутностей у прикладних програмах.

Потужність об'єктно-орієнтованого підходу до розробки програм, дозволяє швидко й легко:

– вмонтувати модулі й об'єкти з інших прикладних програм;

– графічно визначити об'єкти, їх властивості й дії;

– створити нові зразки об'єктів, імітуючи існуючі об'єкти.

Об'єкти або клас об'єктів певні один раз можуть використовуватися багаторазово. Будь-який об'єкт або група об'єктів можуть мати кілька екземплярів. Кожний екземпляр успадковує всі властивості й поведінка первісного об'єкта (ов). Об'єкти, правила, і процедури можна групувати в бібліотеки, які будуть загальними для всіх прикладних програм. Для моделювання широкої різноманітності дій типу виробничих процесів, мережних топологий, інформаційних маршрутизації, або логічних потоків об'єкти можна поєднувати графічно на екрані дисплея.

Представлення знань, правила, процедури й моделі.

В G2 можна ефективно створювати й застосовувати загальні знання, створюючи універсальні правила, процедури, формули й залежності, які є застосовними для повних класів об'єктів. У результаті скорочується час на розробку й збільшується ефективність додатків.

Для вистави знань використовується структурна природня мова, що дозволяє полегшити читання, редагування й підтримку баз знань. Це полегшує використання й редагування додатків користувачем непрограмістом. Для створення й редагування баз знань використовується Редактор Баз Знань.

Для представлення знань експерта про проблемну область використовуються правила. Правила можуть бути як загальними, тобто стосовним до всього класу, так і специфічн, що ставляться до конкретних екземплярів класу. Правила збуджуються автоматично в наступних випадках:

– з'явилися нові дані (висновок від фактів до мети);
– потрібно знайти дані (висновок від мети до фактів) для автоматичного виклику інших правил, процедур, або формул;
– потрібно визначити значення змінних;
– кожні *n* секунд для оцінювання правила в зазначеному інтервалі часу.

Подібно правилам, процедури виконуються в реальному часі. Процедури, правила, і моделі виконуються одночасно відповідно до їхніх пріоритетів. Процедури можуть використовуватися для ефективної вистави поведінки об'єктів. Для процедур можна визначити стану очікування й виконувати обробку інформації паралельно. У результаті це дозволяє формувати потужні прикладні системи реального часу простіше й швидше, чим за допомогою традиційних інструментальних засобів програмування.

Робота в реальному часі

Робота в реальному часі, операційні розв'язки й реакції найчастіше повинні бути виконані миттєво. Прикладні програми в G2 можуть одночасно виконувати міркування відносно багаторазово виконуваних дій у реальному масштабі часу,

переробляючи тисячі правил, виконуючи процедури й моделі згідно їх пріоритетів. Для зберігання хронологий даних і подій і для міркування щодо поведінки через якийсь час використовуються змінні типу час.

G2 графіка може моделювати знання, представляючи об'єкти, зв'язки й залежності між об'єктами. Вона може міркувати в термінах зв'язку, випливаючи мережі зв'язаних об'єктів для визначення причин і результатів. Графічна связность об'єктів дозволяє розширити прикладну програму використовуючи графічне об'єднання аналогів. Графіка включає вбудовані діаграми **(**графіки**),** таблиці й малюнки і т.ін.

G2 також працює з утилітами графічного інтерфейсу Windows. Ці утиліти використовують усі переваги об'єктно-орієнтованих можливостей G2.

Динамічне моделювання й моделювання для аналізу в умовах "ЩО- ЯКЩО"

G2 дозволяють динамічно моделювати системи й процеси, використовуючи об'єкти, правила, процедури й формули. Під час розробки моделі використовуються замість об'єктів реального миру, що дозволяє безупинно перевіряти прикладні програми протягом них розробки. Моделі можуть використовуватися на етапі експлуатації як частина прикладної програми G2 для порівняння фактичних і модельних знань.

Моделі можна також використовувати для проведення аналізу й відповіді на запитання "ЩО- ЯКЩО", визначення, наприклад, кращих робочих умов або кращих проектів. Моделі можна також використовувати для пророкування важливих параметрів дій у реальному часі, наприклад, якість виробу або витрат.

G2 поєднує в собі як універсальні технології побудови сучасних інформаційних систем (стандарти відкритих систем, архітектура клієнт/сервер, об'єктно-орієнтоване програмування, використання ОС, що забезпечують паралельне виконання в реальному часі багатьох незалежних процесів), так і спеціалізовані методи (міркування, засновані на правилах, міркування, засновані на динамічних моделях, або імітаційне моделювання, процедурні міркування, активна об'єктна графіка, структурована природня мова для вистави бази знань), а також інтегрує технології систем, заснованих на знаннях з технологією традиційного програмування (з пакетами програм, із СУБД, з контролерами й концентраторами даних і т.ін.).

Усе це дозволяє за допомогою даної оболонки створювати практично будь-які більші додатки значно швидше, чим з використанням традиційних методів програмування, і знизити працезатрати на супровід готових додатків і їх перенос на інші платформи.