ЛЕКЦІЯ №6

ІНСТРУМЕНТАЛЬНІ ЗАСОБИ ПРОЕКТУВАННЯ БЗ ТА ЕС

ПЛАН

6.1 Введення.
6.2 Аналіз традиційних мов програмування та представлення знань.
6.3 Інструментальний комплекс для створення експертних систем

Час: 2 години.
Література: [1-23]

6.1 Введення

Програмні засоби інженерії знань і реалізації інтелектуальних інформаційних систем (ІІС) можна розділити на наступні групи: універсальні мови програмування (у тому числі традиційні), універсальні мови вистави знань і оболонки.

Вище вже говорилося, що ІІС являють собою деякий програмний комплекс, що дозволяє вирішувати виробничий і економічні завдання на рівні людини – оператора або керівника (експерта). Однак очевидно, що будь-яку програму можна написати на машинно – орієнтованій мові (асемблері) або універсальною мовою високого рівня (ПЛ/1, Си, Бейсик, Алгол, Ада, Фортран, Паскаль і т.ін.). У цьому зв'язку виникає цілком слушне питання: навіщо розглядати спеціалізовані засоби, для вивчення яких потрібне певний час, якщо універсальною мовою високого рівня (або мовою асемблера) володіє практично будь-який програміст? Відповідь на це питання взятий із практики: процес програмування систем ШІ на спеціалізованих засобах займає в 2-3 рази менше часу, чому на універсальні. Однак слід завжди пам'ятати, що параметри ефективності (обсяг пам'яті й швидкодія) ІІС, реалізованих на базі спеціалізованих засобів, у більшості випадків нижче, чим при реалізації ІІС на універсальних засобах.

Ще одним фактором, істотним для вибору ІІС інструментальних програмних засобів при розробці ІІС, є потенційна можливість взаємодії із програмними засобами, використовуваними на різних рівнях ієрархії інтегрованих корпоративних інформаційних систем.

У цьому зв'язку оптимальним розв'язком завдання вибору програмних засобів для реалізації ІІС випливає, по – видимому, уважати наступне: перший прототип (або прототипи: дослідницький, демонстраційний) реалізується на спеціалізованих засобах. У випадку достатньої ефективності цих засобів на них можуть бути написані діючий прототип, і навіть промислова система.

Однак у більшості випадків прототип і навіть промислова система. Однак у більшості випадків прототип випливає «переписати» на традиційних програмних засобах.

Зложилася певна технологія розробки ЕС, яка включає наступні шість етапів: ідентифікація, концептуалізація, формалізація, виконання, тестування й експериментальна експлуатація.

Інструментальні засоби ЕС.

По своєму призначенню й функціональним можливостям інструментальні засоби, застосовувані при проектуванні ЕС можна розділити на 4 категорії**.**

- Оболонки експертних систем. EMYCIN з MYCIN
- Мови програмування високого рівня. OPS5 CLIPS
- Середовище програмування підтримуюче кілька парадигм (KEE, Knowlegecraft, ART).
- Додаткові модулі.

Розглянемо найбільш відомі й широко застосовувані програмні засоби інтелектуальних систем.

6.2 Аналіз традиційних мов програмування та представлення знань.

Спеціалізована мова LISP.

Одним із самих популярних мов програмування в системах ШІ є мова LISP. Ця мова була створена в 60-х роках американським ученим Дж. Маккарти і його учнями. На сьогоднішній день існує близько 20 діалектів цієї мови. Найбільш відомими є INTERLISP, FRANZLISP, QLISP, COMMONLISP. У Радянському Союзі розроблені також кілька версій мови LISP. Мовою LISP написано багато експертних систем (MYCIN, INTERNIST, KEE і ін.), системи природно – язикового спілкування (MARGIE, SHRDLU, ДИЛОС і ін.), інтелектуальні операційні системи (FLEX).

Популярність мови LISP у першу чергу пояснюється тим, що він за допомогою досить простих конструкцій дозволяє писати складні й витончені системи обробки символьної інформації. На жаль, майже всі існуючі LISP – системи мають низьку обчислювальну ефективність. Саме це не дає можливість мові LISP вийти за рамки «академічних» експериментальних систем. Однак бурхливе підвищення продуктивності сучасних комп'ютерів, а також розробка LISP – машин типу З/330, SYMBOLICS і т.ін. вселяє оптимізм відносно майбутньої мови.

Мова LISP має дуже простий синтаксис, оскільки можливі тільки дві його конструкції: атом і список.

Атом – елементарна конструкція мови LISP, характеризуемая своїм іменем і значенням. У деяких LISP – системах з атомом зв'язується також певний список властивостей. Прикладами атомів можуть служити: A, B, A1, ЧАС ВИЛЬОТУ, ВИПУСК, АИ-93 і т.ін.

Список – конструкція LISP, що полягає з безлічі атомів і подсписков. В LISP прийнята скобкова нотація опису списків. Приклади списків (A1, A2,…,АК), (A,B) (ЧАС ВИЛЬОТУ, 15_40) (ВИПУСК АИ-93).

Істотною особливістю мови LISP є те, що тут «дані» і «програми» зовні нічим не відрізняються друг від друга. Це дає можливість писати на LISP «програми», що маніпулюють не тільки даними, але й «програмами». Саме дана властивість дозволяє LISP стати витонченим засобом програмування систем ШІ. Поняття «дані», і «програма» в LISP не використовуються, їх заміняють такі поняття, як вираження й функція.

LISP – функціональна мова. Усі процедури обробки інформації оформляються у вигляді функцій.

Завдяки стандартному набору системних функцій, LISP може бути «розширений» за рахунок користувацьких функцій. Системні функції діляться на арифметичні, рядкові функції, функції введення – висновку, предикати й ін.

LISP – це рекурсивна мова, тобто забезпечує можливість визначення функцій за допомогою самих себе.

Рекурсивність LISP зручна при розв'язку дуже популярної в штучному інтелекті завдання «пошуку по дереву», яка є досить узагальненою й охоплює широкий клас конкретних завдань, починаючи шаховими й кінчаючи завданнями «прийняття розв'язків» або керування складними об'єктами. У кожному конкретному випадку вершини дуги будуть мати свою семантику. Наприклад, при розв'язку шахового завдання вершинам можуть відповідати можливі позиції, а дугам ті або інші ходи, що приводять до цих позицій. Послідовність ходів, що обов'язково приводять до «виграної» позиції, і буде розв'язком даного завдання. Деревоподібна структура дуже часто має також і сценарій діалогу людину з ЕОМ. У цьому випадку з вершинами співвідносяться стани (кроки) діалогу, а з дугами – можливі переходи з одного стану в інше.

Фрейм – орієнтована мова FRL.

Одним з відомих мов вистави знань є мова FRL (Frame Representation Language)[92] фрейм, що ставиться до класу, - орієнтованих. Основна одиниця знання в таких мовах – фрейм, що представляє собою інформаційну модель (або опис) деякої стереотипної ситуації. «Стереотипна ситуація», є узагальненням таких понять, як дії, процеси, події, об'єкти, властивості, модифікатори і т.ін.

Фрейм в FRL – це сукупність пойменованих, асоціативних списків, що містить до п'яти рівнів підструктур. Підструктурами фреймів можуть бути слоти, аспекти, дані, коментарі й повідомлення.

Основною структурною одиницею у фреймі є слоти, що відбивають взаємозв'язки між поняттями предметної області. Слот характеризується своїм іменем і значенням. Імена слотів призначаються проектувальниками БЗ. Однак FRL має також і зарезервовані імена слотів: AKO, INSTANSE, CLAS SIFICATON. У якості значення слотів можуть виступати числа, символи, імена інших фреймів, імена процедур.

Фрейми в FRL будуються за допомогою процедури FASSERT.

В FRL є сім зарезервованих аспектів: ¤VALUE, ¤DEFAULT, ¤IF-NEEDED, ¤IF-ADDED, ¤IF-REMOVED, ¤IF-INSTANTIAD, ¤REQUIRE. Дані з аспекту ¤VALUE інтерпретуються як значення слота, а з аспекту ¤DEFAULT – як значення за замовчуванням. Інші п'ять аспектів зв'язують із фреймом процедуральні знання. Процедури з аспекту ¤IF-ADDED активізуються в тому випадку, якщо в слот додане нове дані; з аспекту ¤IF-REMOVED – якщо зі слота віддаляється те або інше дані. Процедури з аспекту ¤IF-NEEDED запускаються при створенні екземплярів фрейму. Аспект ¤REQUIRE містить процедури, які обмежують значення слота.

Важливою властивістю FRL є наявність у ньому вбудованого механізму «спадкування властивостей». Суть цього механізму полягає в наступному. Усі поняття предметної області в БЗ організовуються у вигляді ієрархічної класифікаційної системи, де кожне поняття зв'язується за допомогою спеціальних відкладань із більш конкретними. Для реалізації цих відкладань існують слоти AKO й INSTANSE. Слот AKO зв'язує поняття з більш загальним (родовим). Слот INSTANSE зв'язує поняття з більш конкретним (видом). Властивості властиві всьому класу, описують тільки у фреймі класу, а інші фрейми цього класу можуть успадковувати цю властивість у випадку потреби.

Процедури обробки FRL підрозділяються на незалежні й приєднані. Незалежно від типу ці процедури пишуться звичайно мовою реалізації самого FRL. На сьогоднішній день більшість FRL – систем написані на LISP.

Мова логічного програмування PROLOG.

Останнім часом до розробки ЭС усе частіше став залучатися мова програмування Пролог [30]. Своє найменування Пролог одержав від скорочення «Програмування логіки» (Programming in Logic). Математичною основою Прологу є вирахування предикатів переважно першого порядку, метод резолюції Робинсона, теорія рекурсивних функцій.

Основною конструкцією мови (у формі, прийнятої для Прологу), є імплікація:

$A \leftarrow B_1, B_2, \ldots, B_n$, називана правилом, де $A_1, B_1, B_2 \ldots, B_n$ – предикати.

Зміст її такий «А істинно, якщо істинно B_1 і істинно B_2 і … і істинно B_n». Наприклад;

РЕЖИМ_УСТАНОВКИ («РЕЖИМ_1») ТИП_НАФТИ
(«ЗАХІДНО-СИБІРСЬКА»)
КІЛЬКІСТЬ_НАФТИ (X),
БІЛЬШЕ (X,0)

Зміст цього правила: «ЯКЩО тип нафти – західно – сибірська й кількість нафти більше нуля, те технологічна установка працює на першому режимі».

З наведеної імплікації випливає два вирожденних випадку:

1) $A\leftarrow$ - така конструкція називається фактором і має сенс: «*A* істинно завжди». Наприклад: ВХОДИТЬ («УСТАНОВКА - 26», «ЦЕХ № 2»), тобто установка 26 входить у цех № 2.

2) $\leftarrow B_1, B_2, \ldots, B_n$ *– дана* конструкція зветься «питання» і означає: «Істинно чи $B_1 \wedge B_2 \ldots \wedge B_n$*?* (знак $\wedge$ - конъюнкція)». Наприклад: ← РЕЖИМ_УСТАНОВКИ («РЕЖИМ_2»), тобто слід перевірити чи працює установка в другому режимі?

Предикати Прологу мають вигляд:

<ІМ'Я предиката> (<аргумент 1>, … <аргумент *ДО*>)

Ім'я предиката можна розглядати як найменування відносини на безлічі аргументів. Аргументом є або константа (атом, атомічне вираження), або змінна. Змінні використовуються для побудови більш складних залежностей:

ВХОДИТЬ (X, «ЗАВОД НПЗ1») ←ВХОДИТЬ (X, «ЦЕХ_№-2»)

Тобто якщо деякий об'єкт (X) входить цех №2, то він входить у НПЗ1;

ВХОДИТЬ (X,B) ← ВХОДИТЬ (X, Z), ВХОДИТЬ (Z,Y).

Тут X входить в B, якщо X входить в Z, а Z входить в У.

При використанні змінних « у питанні» можна одержати їхні значення як результат. Наприклад: РЕЖИМ_УСТАНОВКИ (X). Якщо режим установки відомий і визначений, то в якості відповіді одержимо логічне «так» або «істинно», а в якості побічного ефекту одержимо значення змінної X, тобто діючий режим роботи установки, наприклад X= «РЕЖИМ_2».

Невід'ємним елементом Прологу є рекурсія. Зокрема, наведене вище визначення предиката ВХОДИТЬ є рекурсивним, тобто ім'я предиката, що коштує в лівій частині, збігається з іменем хоча б одного із предикатів, що коштують у правій частині. За допомогою рекурсії можна реалізувати також циклічні процеси обчислень.

Існуючі системи програмування Прологу мають великий набір «вбудованих» предикатів (тобто предикатів, що розуміються самим Прологом), які забезпечують виконання арифметичних операцій, строкову (символьну) обробку, функції введення-висновку й цілий ряд специфічних функцій. За рахунок наявності вбудованих предикатів мова Пролог можна віднести до універсальних мов програмування й навіть до мов системного програмування.

Найважливішою особливістю мови Пролог є наявність реляційної бази даних, причому доступ і робота з реляційними відносинами занурені в сам Пролог. Для користувача ці відносини існують лише у вигляді предикатів.

Відзначена властивість робить Пролог дуже зручним засобом для опису організаційних і технологічних структур.

Так, на Пролозі ефективно реалізуються завдання підсистеми «Кадри», що видають усіляку інформацію про кадрову структуру підрозділів і про підприємство в цілому. Зручно використовувати Пролог і для опису технологічної схеми виробництва з безліччю взаємозв'язків окремих вузлів (установок) по матеріальних, енергетичним, інформаційним і іншим потокам.

Тут не переслідувалася мета дати повний опис мови Пролог, тому багато елементів мови (такі, як зіставлення, backtracking і ін.) нами не розглядалися.

У цей час створене велике число різних по ефективності й потужності Пролог – систем, кожна з яких пропонує свій синтаксис мови й свій набір вбудованих предикатів. Синтаксис мови визначений формою запису Пролог – конструкцій: фактів, правил, питань, предикатів, атомів, змінних, виражень і т.ін.

Продукционна мова OPS.

Мова ставиться до числа продукційних. Будучи універсальною мовою програмування, він у першу чергу призначений для розробки систем ШІ, і, зокрема експертних систем. Ідеологія мови OPS знайшла відбиття в цілому ряді практичних реалізацій, що досить сильно відрізняються друг від друга. Однієї з перших і найбільш відомої є реалізація OPS-5, виконана на одній з версій Лиспа (Franz LISP). Тому синтаксис OPS –5 максимально наближений до синтаксису Лиспа. Мовою OPS – 5 створений ряд промислово експлуатованих експертних систем для фірми DEC з обсягом баз знань від 1000 до 5000 правил. Однієї з останніх, але вже досить широко відомою реалізацією є OPS-83[90]. Особливості цієї реалізації – наявність деяких конструкцій, характерних для процедурних мов програмування, а також сильна типізація даних.

Говорячи про загальні відмітні риси сімейства мов OPS, необхідно відзначити наявність:

- програмного керування стратегій виведення розв'язків;
- розвитому структури даних і принципової ефективності реалізації.

Мова OPS має типову для продукционных систем архітектуру, що включає в себе базу правил, робочу пам'ять і механізм висновку. База правил складається з неупорядкованої сукупності правил, робоча пам'ять – з дискретних об'єктів, називаних елементами робочої пам'яті. Елемент робочої пам'яті може бути доданий у робочу пам'ять, вилучений з неї або модифікований. Механізм виведення є стандартним для системи продукцій циклом керування. На першій фазі циклу вибираються всі правила, ліві частини яких зіставилися із умістом

робочої пам'яті. На другій фазі правило виконується. Вбудований в OPS механізм виведення безпосередньо підтримує тільки прямий висновок, однак у мові є засоби для організації зворотного й змішаного виведення.

У мові OPS допускається використання зовнішніх процедур, реалізованих на інших мовах програмування. Ефективність у мові OPS досягається, в – перших, за рахунок використання спеціального алгоритму швидкого зіставлення, а по-друге, за рахунок схеми, що компілює, застосовуваної замість більш традиційної для продукционных мов інтерпретуючої. Застосування алгоритму швидкого зіставлення накладає ряд обмежень, однак досвід експлуатації мови OPS показав досить високу адекватність засобів мови для розробки експертних систем.

Програма мовою OPS складається з декларативної й продукционной частин. Загалом кажучи, мова OPS дуже простий: у ньому всього три види операторів: оператор опису типів даних TYPE, оператор опису класів CLASS і оператор опису правил RULE. Декларативна частина програми містить опис типів даних і класів елементів робочої пам'яті. Елемент робочої пам'яті (клас) є єдино можливою виставою даних в OPS програмі. Він являє собою фіксовану структуру, що полягає із сукупності пар «атрибут - значення» виду:

Клас
    Атрибут – 1
значення
    Атрибут –2
значення

Наприклад, елемент пам'яті «Установка 1 підприємства» має вигляд:

ОБ'ЄКТ
    Ім'я
Установка 1
Входить в
Цех 1
Складається з
(колона 1, колона 2, колона 3)
Одержує сировину від
Резервуар 5
Тип сировини
Нафта
Виробляє
Гас
Поставляє продукцію
(установка 3, установка 4).

Елементарним об'єктом даних, що мають значення є атрибут. Атрибути локалізовано в межах одного класу, деякі атрибути можуть не мати значення. Кожний

Елемент робочої пам'яті ставиться до певного класу. Припустимі класи елементів робочої пам'яті повинні бути попередньо описані в розділі визначення класів.

Визначення класу задає його структуру, типи припустимих значень атрибутів, що входять у клас, і, якщо це необхідно, початкові значення атрибутів (значення за замовчуванням).

Відзначимо, що всі використовувані типи даних (крім вбудованих) повинні бути явно описані в операторах TYPE або CLASS; особливо зручними при програмуванні експертних систем є довільні типи даних, які дозволяють стежити за правильністю припустимих значень, що вводяться користувачем даних і, у процесі логічного висновку.

Перейдемо тепер до розгляду продукционній частини OPS програми – роздягнула правил. Він записується після декларативної частини і являє собою сукупність правил. Правило OPS складається із заголовка й тіла правила. Заголовок правила починається зі слова RULE, за яким іде ім'я правила й опис змінних (якщо вони використовуються). Тіло правила складається з лівої частини, що задає умова застосовності правила, і правої частини, що містить послідовність виконуваних дій. Ліва частина починається словами IF, роздільником між лівої й правої частинами служить слово THEN. Нижче наведений приклад правила OPS:

```
RULE ЗУПИНКА УСТАНОВКИ 1
IF
  ОБ'ЄКТ
       ІМ'Я          УСТАНОВКА_1
       СТАН          ЗУПИНИЛАСЯ
AND
   ОБ'ЄКТ
       ІМ'Я          УСТАНОВКА_2
       СТАН          ПРАЦЮЄ
AND
ОБ'ЄКТ
       ІМ'Я          УСТАНОВКА_3
       СТАН          ПРАЦЮЄ
THEN
WRITE ( Установка 1 зупинилася(
WRITE ( Розподілити її сировина між установкою 2 і установкою 3 )
```

Ліва частина правила складається з одного або більш умовних елементів. Кожний умовний елемент задає зразок, який зіставляється з елементами, що втримуються в робочій пам'яті. Ліва частина вважається зіставленої, якщо одночасно зіставилися всі вхідні в неї умовні елементи. Роздільником між умовними елементами служить слово AND або ANDNUT. У лівій частині можуть бути використані наступні елементарні предикатне оператори: «не рівно»,

«більше», «менше», «не більше», «не менше», «входить», «не входить у список», «має/не має значення» і ін. Права частина правила складається з послідовності імперативних тверджень, називаних діями.

Дії, наявні в OPS, розділяються на дві групи: елементарн розв'язку, що забезпечують висновок, і допоміжн сервісні можливості, що забезпечують уведення й інші. До елементарних дій ставляться:

MARE – створення нового елемента робочої пам'яті;

REMOVE – видалення елемента з робочої пам'яті;

MODIFY – зміна значень атрибутів, що вже перебувають у робочій пам'яті

До допоміжних дій ставляться:

HALT – явне припинення програми;

WRITE – видача повідомлення на термінал;

PRINT – печатка повідомлення на друкувальному пристрої;

DISPLAY – висновок на екран термінала інформації з бібліотеки.

BIND – виклик функцій або модулів на інших мовах програмування;

SET – динамічна зміна стратегії висновку розв'язків або подробиці пояснень розв'язків.

Розглянемо сучасні розробки засобів побудови інтелектуальних систем.

6.3 Інструментальний комплекс для створення експертних систем

Інструментальний комплекс для створення експертних систем реального часу (на прикладі інтегрованого середовища g2-gensym corp.)

Історія розвитку ІС для створення ЕС реального часу почалася в 1985 р., коли фірма Lisp Machine Inc. випустила систему Picon для символьних ЕОМ Symbolics. Успіх цього ІС привів до того, що група провідних розроблювачів Picon в 1986 р. утворювала приватну фірму Gensym, яка, значно розвивши ідеї, закладені в Picon, в 1988 р. вийшла на ринок з ИС за назвою G2, версія 1.0. У цей час функціонує версія 4.2 і готується до випуску версія 5.0.

Основне призначення програмних продуктів фірми Gensym (США) - допомогти підприємствам зберігати й використовувати знання й досвід їх найбільш талановитих і кваліфікованих співробітників в інтелектуальних системах реального часу, що підвищують якість продукції, надійність і безпека виробництва, що й знижують виробничі витрати. Про те, як фірмі Gensym вдається впоратися із цим завданням, говорить хоча б те, що сьогодні їй належать 50% світового ринку експертних систем, використовуваних у системах керування.

З відставанням від Gensym на 2 - 3 року інші фірми почали створювати свої ІС для ЕС РВ. З погляду незалежних експертів NASA, що проводили комплексне дослідження характеристик і можливостей деяких з перерахованих систем, у цей час найбільш просунутим ІС, безумовно, залишається G2 (Gensym, США);

наступні місця зі значним відставанням (реалізоване менш 50% можливостей G2) займають Rtworks - фірма Talarian (США), COMDALE/C (Comdale Techn. - Канада), COGSYS (SC - США), ILOG Rules (ILOG - Франція).

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.

Інструментальні засоби фірми Gensym є еволюційним кроком у розвитку традиційних експертних систем від статичних предметних областей до динамічних. Чималу частку успіху фірмі Gensym забезпечують основні принципи, яких вона дотримується у своїх нових розробках:

- проблемно/предметна орієнтація;
- проходження стандартам;
- незалежність від обчислювальної платформи;
- сумісність знизу-нагору з попередніми версіями;
- універсальні можливості, не залежні від розв'язуваного завдання;
- забезпечення технологічної основи для прикладних систем;
- комфортне середовище розробки;
- пошук нових шляхів розвитку технології;
- розподілена архітектура клієнт-сервер;
- висока продуктивність.

Основною гідністю оболонки експертних систем G2 для російських користувачів є можливість застосовувати її як інтегруючий компонент, що дозволяє за рахунок відкритості інтерфейсів і підтримки широкого спектра обчислювальних платформ легко об'єднати вже існуючі, розрізнені засоби автоматизації в єдину комплексну систему керування, що охоплює всі аспекти виробничої діяльності - від формування портфеля замовлень до керування технологічним процесом і відвантаження готової продукції. Це особливо важливо для вітчизняних підприємств, парк технічних і програмних засобів яких формувався по більшій частині безсистемно, під впливом різких коливань в економіці.

Крім системи G2, як базового засобу розробки, фірма Gensym пропонує 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 фірми Gensym, згрупований по проблемній орієнтації, охоплює всі стадії виробничого процесу й виглядає в такий спосіб:

- інтелектуальне керування виробництвом - G2, G2 Diagnostic Assistant (GDA), Neuron-line (NOL), Statistical Process Control (SPC), Batchdesign_Kit;
- оперативне планування - G2, G2 Scheduling Toolkit (GST), Dynamic Scheduling Packadge (DSP);
- розробка й моделювання виробничих процесів - G2, Rethink, Batchdesign_Kit;
- керування операціями й корпоративними мережами - G2, Fault Expert.

Незважаючи на те, що перша версія системи G2 з'явилася не дуже давно - в 1988 р., її навіть у багатій Америці ніхто не назве дешевої. G2 можна назвати бестселером на ринку програмних продуктів - на початок 1996 р. у світі було встановлено більш 5000 її копій. Фірма Gensym обслуговує більш 30 галузей - від аерокосмічних досліджень до виробництва харчових продуктів. Список користувачів G2 виглядає як довідник Who-Is-Who у світовій промисловості. 25 найбільших індустріальних світових корпорацій використовують G2. На базі G2 написане більш 500 діючих додатків.

Чим же пояснюється успіх інструментального комплексу G2? Насамперед G2 - динамічна система в повному змісті цього слова. G2 - це об'єктно-орієнтоване інтегроване середовище для розробки й супроводу додатків реального часу, що використовують бази знань. G2 функціонує на більшості існуючих платформ (табл.9.1). 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 простім переписуванні файлів. Функціональні можливості й зовнішній вигляд додатка не перетерплюють при цьому ніяких змін.

Таблиця – Платформи, на яких функціонує G2

Фирма-производитель	Вычислительная система	Операционная среда
Digital Equipment	VAX 3xxx,4xxx,6xxx, 7xxx, 8xxx,9xxx	VMS
	DECstation 3xxx, 6xxx	ULTRIX
	DEC Alpha APX	Open VMS, OSF/1, Windows NT
SUN Microsystems	SUN=4	Sun OS
	SPARC 1,2, 10, LX, Classic	Sun OS/Solaris 1, Solaris 2.x
Hewlett Packard	HP9000/4xx, 7xx, 8xx	HP-UX
IBM	RISC 6000	AIX
Data General	AViiON	DG/UX
Silicon Graphics	IRIS, INDIGO	IRIX
ПЭВМ	Intel 486/Pentium	Windows NT, Windows-95
Motorola	Motorola 88000	UNIX
NEC	EWS 4800	EWS-UX/V