ЛЕКЦІЯ №2

ЕКСПЕРТНІ СИСТЕМИ РЕАЛЬНОГО ЧАСУ

ПЛАН

2.1 Ведення в експертні системи реального часу (ЕСРЧ).
2.2 Основні виробники ЕСРЧ.
2.3 Архітектура ЕСРЧ.
2.4 Основні компоненти ЕСРЧ.
2.5 Розробка експертної системи. Приклад.

Час: 2 години.
Література: [1-23]

2.1 Ведення в експертні системи реального часу (ЕСРЧ).

Серед спеціалізованих систем, заснованих на знаннях, найбільш значущі експертні системи реального часу, або динамічні експертні системи. На їх частку доводиться 70 відсотків цього ринку.

Значущість інструментальних засобів реального часу визначається не стільки їх бурхливим комерційним успіхом (хоч і це гідно ретельного аналізу), але, в першу чергу, тим, що тільки за допомогою подібних засобів створюються стратегічно значущі застосування в таких областях, як управління безперервними виробничими процесами в хімії, фармакології, виробництві цементу, продуктів харчування і т.п., аерокосмічні дослідження, транспортування і переробка нафти і газу, управління атомними і тепловими електростанціями, фінансові операції, зв'язок і багато інших.

Класи завдань, що вирішуються експертними системами реального часу, такі: моніторинг в реальному масштабі часу, системи управління верхнього рівня, системи виявлення несправностей, діагностика, складання розкладів, планування, оптимізація, системи-порадники оператора, системи проектування.

Статичні експертні системи не здатні вирішувати подібні задачі, оскільки вони не виконують вимоги, що пред'являються до систем, що працюють в реальному часі:

1. Представляти що змінюються в часі дані, що поступають від зовнішніх джерел, забезпечувати зберігання і аналіз даних, що змінюються.

2. Виконувати тимчасові міркування про декілька різних асинхронних процесів одночасно (т.е. планувати відповідно до пріоритетів обробку процесів, що поступили в систему).

3. Забезпечувати механізм міркування при обмежених ресурсах (час, пам'ять). Реалізація цього механізму пред'являє вимоги до високої швидкості роботи системи, здатності одночасно вирішувати декілька задач (т.е. операційні системи UNIX, VMS, Windows NT, але не MS-DOS).

4. Забезпечувати "передбаченість" поведінки системи, тобто гарантію того, що кожне завдання буде запущено і завершена в строгій відповідності з тимчасовими обмеженнями. Наприклад, дана вимога не допускає використання в ЕС реального часу механізму "збірки сміття", властивого мові Lisp.

5. Моделювати "навколишній світ", що розглядається в даному застосуванні, забезпечувати створення різних його станів.

6. Протоколювати свої дії і дії персоналу, забезпечувати відновлення після збою.

7. Забезпечувати наповнення бази знань для застосувань реального ступеня складності з мінімальними витратами часу і праці (необхідне використання об'єктно-орієнтованої технології, загальних правил, модульної і т.п.).

8. Забезпечувати настроювання системи на вирішувані завдання (проблемная/предметная орієнтованість).

9. Забезпечувати створення і підтримку призначених для користувача інтерфейсів для різних категорій користувачів.

10. Забезпечувати рівень захисту інформації (по категоріях користувачів) і запобігати несанкціонованому доступу.

Підкреслимо, що окрім цих десяти вимог засобу створення експертних систем реального часу повинні задовольняти і перерахованим вище загальним вимогам.

2.2 Основні виробники ЕСРЧ.

Інструментарій для створення експертних систем реального часу вперше випустила фірма Lisp Machine Inc в 1985 році. Цей продукт призначався для символьних ЭВМ Symbolics і носив назву Picon. Його успіх привів до того, що група ведучих його розробників утворила фірму Gensym, яка, значно розвинувши ідеї, закладені в Picon, випустила в 1988 році інструментальний засіб під назвою G2. Зараз працює його третя версія і підготовлена четверта.

З відставанням від Gensym на два-три роки ряд інших фірм почали створювати (або намагатися створювати) свої інструментальні засоби.

Назвемо ряд з них: RT Works (фірма Talarian, США), COMDALE/C (Comdale Techn., Канада), COGSYS (SC, США), ILOG Rules (ILOG, Франція). Порівняння двох найбільш просунутих систем, G2 і RT Works, яке проводилося шляхом розробки одного і того ж застосування двома організаціями, NASA (США) і Storm Integration (США), показала значна перевага першої.

2.3 Архітектура ЕСРЧ.

Специфічні вимоги, що пред'являються до експертної системи реального часу, призводять до того, що їх архітектура відрізняється від архітектури статичних систем. Не вдаючись до деталей, відзначимо появу двох нових підсистем - моделювання зовнішнього оточення і сполучення із зовнішнім світом (датчиками, контроллерами, СУБД і т.п.) - і значні зміни, яким піддаються підсистеми, що залишилися.

Для того, щоб зрозуміти, що вдає із себе середовище для створення експертних систем реального часу, опишемо нижчим життєвий цикл такої системи, а також її основні компоненти. Опис оболонки експертної системи реального часу приведемо на прикладі засобу G2, оскільки в нім повністю реалізовані можливості, які вважаються необхідними і доречними в подібних програмних продуктах.

Життєвий цикл застосування в G2 складається з ряду етапів.

Розробником зазвичай є фахівець в конкретній області знань. Він в ході обговорень з кінцевим користувачем визначає функції, що виконуються прототипом. При розробці прототипу не використовується традиційне програмування. Створення прототипу зазвичай займає від однієї до двох тижнів (за наявності у розробника досвіду по створенню застосувань в даному середовищі. Прототип, як і застосування, створюється на структурованій природній мові, з використанням об'єктної графіки, ієрархії класів об'єктів, правив, динамічних моделей зовнішнього світу. Багатослівність мови зведена до мінімуму шляхом введення операції клонування, бази знань, що дозволяє розмножити будь-яку суть.

Кінцевий користувач пропонує етапність проведення робіт, напрями розвитку бази знань, указує пропуски в ній. Розробник може розширювати і модифікувати базу знань у присутності користувача навіть в той момент, коли застосування виконується. В ході цієї роботи прототип розвивається до такого стану, що починає задовольняти уявленням кінцевого користувача. У крупних застосуваннях команда розробників може розбити застосування на окремі модулі, які інтегруються в єдину базу знань.

Можливий і альтернативний підхід до створення застосування. При цьому підході кожен розробник має доступ до бази знань, що знаходиться на сервері, за допомогою засобу, званого Telewindows, зазвичай розташованого на компьютереклиенте. В цьому випадку розробники можуть мати різні авторизовані рівні доступу до застосування.

Застосування може бути реалізоване не тільки на різних ЕОМ, але і з використанням декількох взаємодіючих оболонок G2.

Тестування застосування на наявність помилок

У G2 помилки в синтаксисі показуються безпосередньо при введенні конструкцій (структур даних і виконуваних тверджень) в базу даних; ці конструкції аналізуються інкрементно. Можуть бути введені тільки конструкції, що не містять синтаксичних помилок. Таким чином, відпадає ціла фаза відладки застосування (властива традиційному програмуванню), що прискорює розробку застосувань.

Розробник звільнений і від необхідності знати детальний синтаксис мови G2, оскільки при введенні в базу знань деякої конструкції йому у вигляді підказки повідомляється перелік всіх можливих синтаксично правильних продовжень.

Для виявлення помилок і неопределенностей реалізована можливість "Inspect", що дозволяє проглядати різні аспекти бази знань, наприклад, "показати всі твердження з посиланнями на невизначену суть (об'єкти, зв'язки, атрибути)", "показати графічно ієрархію заданого класу об'єктів", "показати всю суть, у якої

значення атрибуту Notes не ОК". (Даний атрибут є у всієї суті, уявної в мові G2; його значення - або ОК, коли немає претензій до суті, або опис реальних або потенційних проблем, наприклад, посилання на неіснуючий об'єкт, декілька об'єктів з одним ім'ям і т.п.)

Тестування логіки застосування і обмежень (за часом і пам'яті)

Блок динамічного моделювання дозволяє при тестуванні відтворити різні ситуації, адекватні зовнішньому світу. Таким чином, логіка застосування перевірятиметься в тих умовах, для яких вона створювалася. Кінцевий користувач може взяти безпосередню участь в тестуванні завдяки управлінню кольором (т.е. зміна кольору при настанні заданого стану або виконання умови) і анімації (т.е. переміщення/обертання суті при настанні стану/умови). Завдяки цьому він зможе зрозуміти і оцінити логіку роботи застосування, не аналізуючи правила і процедури, а розглядаючи графічне зображення керованого процесу, технічної споруди і т.п.

Для перевірки виконання обмежень використовується можливість "Meters", що обчислює статистику по продуктивності і використовуваній пам'яті.

Отримане застосування повністю переносимий на різні платформи в середу UNIX (SUN, DEC, HP, IBM і т.д.), VMS (DEC VAX) і Windows NT (Intel, DEC Alpha). База знань зберігається в звичайному ASCII-файлі, який однозначно інтерпретується на будь-якій з підтримуваних платформ. Перенесення застосування не вимагає його перекомпіляції і полягає в простому переміщенні файлів. Функціональні можливості і зовнішній вигляд застосування не зазнають при цьому ніяких змін. Застосування може працювати як в "повному" (т.е. призначеною для розробки) середовищі, так і під runtime, яка не дозволяє модифікувати базу знань.

Супровід застосування.

Не тільки сам розробник даного застосування, але і будь-який користувач може легко його зрозуміти і супроводжувати, оскільки всі об'єкти/класи, правила, процедури, функції, формули, моделі зберігаються в базі знань у вигляді структурованої природної мови і у вигляді графічних об'єктів. Для її перегляду використовується можливість "Inspect". Супровід спрощується за рахунок того, що різним групам користувачів видається не вся інформація, а тільки її частина, відповідна їх потребам.

2.4 Основні компоненти ЕСРЧ.

Експертна система реального часу складається з бази знань, машини висновку, підсистеми моделювання і планувальника.

Бази знань

Всі знання в G2 зберігаються в двох типах файлів: бази знань і бібліотеки знань. У файлах першого типу зберігаються знання про застосування: визначення всіх об'єктів, об'єкти, правила, процедури і т.п. У файлах бібліотек зберігаються загальні знання, які можуть бути використані більш, ніж в одному застосуванні, наприклад, визначення стандартних об'єктів. Файли баз знань можуть перетворитися в бібліотеки знань і назад.

В цілях забезпечення повторної використовуваної застосувань реалізований засіб, що дозволяє об'єднувати з поточним застосуванням раніше створені бази і бібліотеки знань. При цьому конфлікти в об'єднуваних знаннях виявляються і відображаються на дисплеї.

Знання структуруються: передбачені ієрархія класів, ієрархія модулів, ієрархія робочих просторів. Кожну з них можна показати на дисплеї.

Структури даних

Суть в базі знань з погляду їх використання можна розділити на структури даних і виконувані твердження. Прикладами перших є об'єкти і їх класи, зв'язки (connection), відносини (relation), змінні, параметри, списки, масиви, робочі простори. Прикладами других - правила, процедури, формули, функції.

Виділяють об'єкти (і класи), що вбудовані в систему і вводяться користувачем. При розробці застосування, як правило, створюються підкласи, що відображають специфіку даного застосування. Серед вбудованих підкласів об'єктів найбільший інтерес представляє підклас даних, що включає підкласи змінних, параметрів, списків і масивів.

Особлива роль відводиться змінним. На відміну від статичних систем змінні діляться на три види: власне змінні, параметри і прості атрибути. Параметри отримують значення в результаті роботи машини висновку або виконання якої-небудь процедури. Змінні представляють вимірювані характеристики об'єктів реального миру і тому мають специфічні риси: час життя значення і джерело даних. Час життя значення змінної визначає проміжок часу, протягом якого це значення актуальне, після закінчення цього проміжку змінна вважається такою, що не має значення.

Не дивлячись на те що продукційні правила забезпечують достатню гнучкість для опису реакцій системи на зміни навколишнього світу, в деяких випадках, коли необхідно виконати жорстку послідовність дій, наприклад, запуск або зупинку комплексу устаткування, переважніший процедурний підхід. Мова програмування, використовувана в G2 для представлення процедурних знань, є достатньо близьким родичем Паскаля. Окрім стандартних конструкцій, що управляють, мова розширена елементами, що враховують роботу процедури в реальному часі: очікування настання подій, дозвіл іншим завданням переривати її виконання, директиви, задаючі послідовне або паралельне виконання операторів. Ще одна цікава особливість мови - ітератори, що дозволяють організувати цикл безліччю екземплярів класу.

Головний недолік прямого і зворотного висновку, використовуваних в статичних експертних системах, - непередбачуваність витрат часу на їх виконання. З погляду динамічних систем, повний перебір можливих до застосування правив - недозволена розкіш. У зв'язку з тим, що G2 орієнтована на застосування, що працюють в реальному часі, в машині висновку повинні бути засоби для скорочення перебору, для реакції на непередбачені події і т.п. Для машини виведення G2 характерний багатий набір способів збудження правив. Передбачено дев'ять випадків:

Розглянуті тенденції розвитку штучного інтелекту дозволяють стверджувати, що одним з основних напрямів в цій області є ЕС реального часу.

Розгляд проведений на прикладі оболонки експертних систем реального часу G2, що є самодостатнім середовищем для розробки, впровадження і супроводу застосувань в широкому діапазоні галузей. G2 об'єднує 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 природна мова для представлення бази знань), а також інтегрує технології систем, заснованих на знаннях з технологією традиційного програмування (з пакетами програм, з СУБД, з контроллерами і концентраторами даних і т.д.).

Все це дозволяє за допомогою даної оболонки створювати практично будь-які великі застосування значно швидше, ніж з використанням традиційних методів програмування, і понизити трудовитрати на супровід готових застосувань і їх перенесення на інші платформи.

2.5 Розробка експертної системи. Приклад (в програмі Mini Expert System).

Програма Mini Expert System є простою експертною системою, використовуючою байесовскую системою логічного висновку. Вона призначена для проведення консультації з користувачем в якій-небудь прикладній області (на яку налаштована завантажена база знань) з метою визначення вірогідності можливих результатів і використовує для цього оцінку правдоподібності деяких передумов, що отримується від користувача.

На першому етапі створення бази знань необхідно сформулювати знання про дану область у вигляді двох наборовши: Q = {qj} – набір питань (симптомів, свідоцтв) і V = {vi} - набір варіантів результату (варіантів рішення), а також двох матриць вірогідності: Py = {pyij} і Pn = {pnij} розміром mn, де pyij – вірогідність отримання позитивної відповіді на j-й питання, якщо i-й результат вірний; pnij – вірогідність отримання негативної відповіді на j-й питання, якщо i-й результат вірний; n і m – кількості питань і результатів відповідно. Крім того, кожному результату ставиться у відповідність апріорна вірогідність даного результату P, тобто вірогідність результату у разі відсутності додаткової інформації.

В процесі роботи ЕС вирішувач, користуючись даними наборами і матрицями і теоремою Байеса, визначає апостеріорну вірогідність кожного результату, тобто вірогідність, скоректовану відповідно до відповіді користувача на кожне питання:

- при позитивній відповіді $P_{апостер.} = \frac{Py_{ij} \cdot P_i}{Py_{ij} \cdot P_i + Pn_{ij} \cdot (1 - P_i)}$,
- при негативній відповіді $P_{àïîñòåð.} = \frac{(1 - Py_{ij}) \cdot P_i}{(1 - Py_{ij}) \cdot P_i + (1 - Pn_{ij}) \cdot (1 - P_i)}$,
- при відповіді «не знаю» апостеріорна вірогідність рівна апріорною.

Тобто вірогідність здійснення якоїсь гіпотези за наявності певних підтверджуючих свідоцтв обчислюється на основі апріорної вірогідності цієї гіпотези без підтверджуючих свідоцтв і вірогідності здійснення свідоцтв за умов, що гіпотеза вірна або невірна.

Початкова інформація оформляється у вигляді текстового файлу з розширенням .DAT з наступною структурою:

Опис бази знань, ім'я автора, коментар і т.д.
(можна в декілька рядків; ця інформація виводиться після завантаження бази знань; дана секція закінчується після першого порожнього рядка)

Питання № 0 (будь-який текст, що закінчується перенесенням рядка)
Питання № 1
Питання № 2
...
Питання № N (після останнього питання слідує один порожній рядок, і друга секція закінчується)

Результат № 0, P [, i, Py, Pn]
Результат № 1, P [, i, Py, Pn]
Результат № 2, P [, i, Py, Pn]
...
Результат № M, P [, i, Py, Pn]

У останній секції перераховуються результати і відповідні ним елементи матриць вірогідності. Кожен результат задається в окремому рядку, перерахування закінчується з кінцем файлу.

На початку опису правила висновку задається результат, вірогідність якого міняється відповідно до даного правила. Це текст, що включає будь-які символи, окрім ком. Після коми указується апріорна вірогідність даного результату P. Після цього через кому йде ряд полів, що повторюються, з трьох елементів. Перший елемент i – номер відповідного питання. Наступні два елементи Pyij і Pnij – відповідно вірогідності отримання відповіді «Та» на це питання, якщо можливий результат вірний і невірний. Ці дані указуються для кожного питання, пов'язаного з даним результатом.

Примітка: P <= 0.00001 вважається рівною нулю, а P >= 0.99999 – одиниці, тому не слід указувати такі значення – результат з подібною апріорною вірогідністю оброблятися не буде.

Наприклад:

Грип, 0.01, 1,0.9,0.01, 2,1,0.01, 3,0,0.01

Тут сказано: існує апріорна вірогідність P = 0,01 того, що будь-яка навмання узята людина хворіє на грип.

Першому питанню (i = 1) відповідає запис «1,0.9,0.01». Звідси слідують значення Pyi1 = 0,9 і Pni1 = 0,01, які означають, що якщо у пацієнта грип, то він в дев'яти випадках з десяти відповість «Та» на це питання, а якщо у нього немає грипу, він відповість «Та» лише в одному випадку із ста (т.е. даний симптом зустрічається досить рідко при інших хворобах). Відповідь «Та» підтверджує гіпотезу про те, що у нього грип. Відповідь «Ні» дозволяє припустити, що людина на грип не хворіє.

При позитивній відповіді «Та» (+5) на перше питання апостеріорна вірогідність для даного прикладу складе:

$$P_{àïîñòåð.} = \frac{Py_{ij} \cdot P_i}{Py_{ij} \cdot P_i + Pn_{ij} \cdot (1 - P_i)} = \frac{0.01 * 0.9}{0.01 * 0.9 + (1 - 0.01) * 0.01} = 0{,}47619.$$

При негативній відповіді «Ні» (-5) на перше питання апостеріорна вірогідність для даного прикладу складе:

$$P_{àïîñòåð.} = \frac{(1 - Py_{ij}) \cdot P_i}{(1 - Py_{ij}) \cdot P_i + (1 - Pn_{ij}) \cdot (1 - P_i)} =$$

$$= \frac{(1 - 0.9) * 0.01}{(1 - 0.9) * 0.01 + (1 - 0.9) * (1 - 0.01)} = 0{,}00102.$$

При відповіді «Не знаю» (0) апостеріорна вірогідність результату рівна апріорною: $P_{àïîñòåð.} = P_i$.

При проміжній відповіді *h* (від -5 до 0 і від 0 до +5) апостеріорна вірогідність розраховується з урахуванням ступеня упевненості приналежності ознаки і розраховується лінійною інтерполяцією від значень ствердних відповідей «Так», «Ні», «Не знаю».

При негативній відповіді (-5;0):

$$P_{àïîñòåð.} = P_i + (P_i - P_{àïîñòåð.(íåÅ)}) \cdot \frac{h}{5}.$$

Наприклад, при відповіді *h* = -3:

$$P_{àïîñòåð.} = 0.01 + (0.01 - 0.00102) \cdot \frac{-3}{5} = 0{,}00461.$$

При негативній відповіді (0;+5):

$$P_{àïîñòåð.} = P_i + (P_{àïîñòåð.(Å)} - P_i) \cdot \frac{h}{5}.$$

Наприклад, при відповіді *h* = +3:

$$P_{àïîñòåð.} = 0.01 + (0.47619 - 0.01) \cdot \frac{3}{5} = 0{,}28971.$$

Для другого питання маємо запис «2,1,0.01». Тобто, якщо у людини грип, то цей симптом обов'язково повинен бути присутнім (Pyi2 = 1) і він обов'язково відповість «Так». Відповідний симптом може мати місце і за відсутності грипу (Pni2 = 0,01), але це маловірогідно.

Примітка: При великій кількості питань немає необхідності в кожному рядку останньої секції перераховувати їх все, тим більше якщо відповідь на яке-небудь питання не впливає на вірогідність даного результату.