INSTRUMENTE PENTRU VERIFICAREA SI TESTAREA SISTEMELOR EXPERTE DEZVOLTATE IN MEDII DE PROGRAMARE ORIENTATE SPRE OBIECTE

Ing. Stefan Trăușan-Matu
Ing. Mihai Bărbuceanu

Institutul de Cercetări în Informatică

Prof. dr. Ing. Jaak Tepandi

Universitatea Tehnică din Tallin

Rezumat

Lucrarea își propune să analizeze disponibilitățile mediilor de programare orientate spre obiecte în verificarea și testarea sistemelor experte. De asemenea, vor fi propuse soluții pentru realizarea modulelor specializate în verificare și testare. Considerăm că pot fi construite sisteme experte pentru verificare și testare care, folosind reprezentarea prin obiecte structurate, pot ridica substanțial calitatea sistemelor dezvoltate. In acest sens, lucrarea se plasează în contextul preocupărilor pentru asigurarea de instrumente destinate sprijinirii întregului ciclu de viață al sistemelor experte /1/. Totodată, își propune ca instrumentele dezvoltate să aibă aspectul unor "asistenți inteligenți", lucru posibil datorită abordării bazate pe cunoștințe în dezvoltarea lor.

1. Verificarea și testarea sistemelor experte

O aplicație remarcabilă a cercetărilor în domeniul inteligenței artificiale este elaborarea de programe care pot rezolva probleme la nivelul unui expert uman într-un anumit domeniu. Aceste programe sînt denumite sisteme experte datorită "competenței" lor. Ele sînt caracterizate de o metodologie de dezvoltare evolutivă care are ca scop, pentru expertul uman, conștientizarea cunoștințelor pe care le folosește în rezolvarea problemelor, iar pentru elaboratorul sistemului expert (denumit inginer de cunoștințe), transferul cunoștințelor într-o formă interpretabilă de calculator (și înmagazinate într-o bază de cunoștințe) /2/. Caracterul evolutiv și iterativ al procesului de dezvoltare a acestor programe este impus de dificultatea întîmpinată de experții umani într-un anumit domeniu de a explica formal toate considerentele, regulile, procedurile, planurile etc., într-un cuvînt toate cunoștințele pe care le folosesc în rezolvarea unei probleme. Calea prin care se poate evita această dificultate este elaborarea rapidă a unei variante simple a programului, a unui prototip, care ulterior se va îmbunătăți permanent. Îmbunătățirea se face prin achiziția de noi cunoștințe sau prin restructurarea celor existente.

Un rol important în dezvoltarea sistemelor experte îl au activitățile de verificare, validare și testare, acestea constituind punctul de plecare pentru detectarea lipsurilor și inconsistențelor în baza de cunoștințe /3/. Verificarea este procesul prin care se determină dacă rezultatele unei anumite faze din ciclul de dezvoltare al unui produs program satisfac cerințele stabilite în faza anterioară ("dezvolt bine programul?"). Validarea este procesul de evaluare a unui produs program pentru a stabili dacă satisface cerințele urmărite ("am dezvoltat programul pe care mi l-am propus?"). Testarea este procesul căutarii erorilor în program, necesitînd execuția programului și compararea rezultatelor cu cele furnizate de un "oracol". Prin "oracol" înțelegem în acest context o specificație, o persoană, un program, o bază de date de fapte sau orice altă modalitate de a stabili rezultatele programului testat în funcție de datele de intrare.

În prezenta lucrare ne vom concentra asupra instrumentelor și tehnicilor utile inginerului de cunoștințe pentru verificarea și testarea sistemelor experte, dezvoltate în medii de programare orientate spre obiecte. Am considerat că aspectele legate de validare țin mai mult de interfața cu expertul în domeniu și cu utilizatorii sistemului și mai puțin de activitatea inginerului de cunoștințe și de aceea nu le vom trata în această lucrare. Pe plan mondial, cercetările referitoare la verificarea și testarea sistemelor experte sînt încă la început și, în general, au avut în vedere cunoștințele reprezentate cu reguli de producție.

2. Medii de programare orientate spre obiecte structurate

Programarea orientată spre obiecte oferă o combinație a tipurilor de date abstracte, cu facilități de definire a relațiilor de moștenire. Un obiect are o mulțime de componente (slot-uri) și poate efectua prelucrări ca răspuns la trimiterea unui mesaj. Pentru fiecare mesaj la care poate răspunde, un obiect are atașată o metodă (o procedură). Obiectele pot fi în relații de moștenire prin care componente și metode ale unui "super" obiect pot fi moștenite de un "sub" obiect. Trimiterea unui mesaj către un obiect este o generalizare a apelului unei proceduri (din limbajele "tradiționale" de programare). Diferența care apare este că, la transmiterea unui mesaj, nu se știe exact codul care va rezolva acest mesaj, datorită posibilităților de moștenire a metodelor. Abstractizarea datelor este realizată prin mecanismul trimiterii de mesaje (în cazul în care starea unui obiect - valorile componentelor - este modificată numai de metodele activate ca rezultat al trimiterii unui mesaj).

Mediile de programare orientate spre obiecte, destinate dezvoltării de sisteme experte, integrează mai multe tehnici de reprezentare a cunoștințelor: obiecte structurate (frame, obiecte care au drept componente alte obiecte), reguli de producție, programare logică, reprezentare procedurală /4/. Aceste tehnici pun la

dispoziția utilizatorului module pentru diverse regimuri de rezolvare a problemelor, diverse utilitare (editoare specializate, browsere, generatoare de liste de referințe încrucișate etc.) și chiar sisteme generice pentru diverse clase de probleme (de exemplu, sisteme generice de diagnosticare) /5/.

Este natural ca, în astfel de medii de programare, să existe și instrumente specializate, destinate asistării, verificării și testării sistemelor experte dezvoltate. Acest fapt este cu atît mai mult justificat de complexitatea programelor care se pot scrie în aceste medii. De asemenea, datorită metodologiei evolutive, nedetectarea unor erori în primele versiuni poate duce la pierderi semnificative de timp și de alte resurse pentru dezvoltarea unor module neviabile. Pe de altă parte însă, facilitățile de structurare oferite de programarea orientată spre obiecte au un impact pozitiv asupra posibilităților efectuării de verificări de semantică, cu un grad înalt de complexitate. Mai mult, posibilitățile oferite de reprezentarea cunoștințelor prin obiecte structurate permit integrarea în mediile de programare a unor module "inteligente" de asistare a verificării și testării programelor dezvoltate.

În capitolele următoare vom trata în primul rînd verificările statice (care nu necesită rularea sistemului expert), oprindu-ne apoi la testarea sistemelor expert bazate pe obiecte. Vom avea în vedere numai aspectele legate de modulele de programare orientată spre obiecte și nu pe cele legate de reguli de producții, programare logică etc., dintr-un mediu de programare multiparadigmă. Vom analiza rolul instrumentelor existente în majoritatea mediilor de programare orientate spre obiecte, în verificarea sistemelor experte dezvoltate. Vom prezenta modulele specializate pe care le considerăm utile pentru verificare și testare. In analizele pe care le vom face vom avea tot timpul în vedere două aspecte ale obiectelor: relațiile (de moștenire și obiect-slot-obiect component) între obiecte și configurația mesaje-metode (atașarea metodelor pentru mesaje și trimiterea de mesaje din metode).

3. Verificarea relațiilor dintre obiecte

În acest capitol ne propunem să analizăm instrumentele care pot fi utile celor care dezvoltă sisteme experte, pentru detectarea unor nereguli în utilizarea relațiilor dintre obiecte, fără a implica execuții ale sistemului, ci numai analiza textului său. Vor fi luate în considerare relațiile "moștenește pe", "are ca parte pe" și combinații ale acestora.

Limbajele orientate spre obiecte facilitează structurarea bazei de cunoștințe. Relațiile de moștenire și relațiile stabilite între un obiect și obiectele care sînt valori ale slot- urilor sale (relația "are ca parte pe") permit reprezentarea cunoștințelor într-o formă apropiată de structura conceptuală a expertizei specialistului în domeniu. O consecință a acestui fapt este și posibilitatea efectuării unor verificări semantice, conceptuale, ale structurii bazei de cunoștințe.

O serie de instrumente existente în multe medii de programare orientate spre obiecte permit efectuarea de verificări manuale sau automate ale bazei de cunoștințe. Verificările manuale pot fi efectuate folosind instrumente care oferă posibilitatea vizualizării unor aspecte ale structurii unei baze de cunoștințe:

- lista obiectelor din baza de cunoștințe;
- lista slot-urilor (proprii sau moștenite) unui obiect;
- relațiile dintre obiecte ("moștenește pe" sau "are ca parte pe"); unele medii de programare oferă posibilitatea vizualizării grafurilor corespunzătoare relațiilor de interes;
- listarea de referințe încrucișate între definiri și referiri de obiecte și slot-uri;
- parafrazări în limbaj natural ale bazelor de cunoștințe.

Pentru verificarea bazelor de cunoștințe se poate apela și la metodele utilizate în tehnologia verificării programelor, cum ar fi inspecțiile și parcurgerile (walkthroughs) /6/. Acestea pot fi făcute manual, dar și în mod automat, prin dezvoltarea, de exemplu, a unui sistem expert în inspecția sistemelor experte implementate cu obiecte.

Verificări automate pot fi efectuate cu ajutorul unor proceduri specializate în detectarea unor violări ale unor restricții care privesc relațiile dintre obiecte. Cel mai simplu exemplu este detectarea ciclurilor în relația de moștenire (avînd în vedere că este o relație de ordine parțială). Astfel de verificări pot fi introduse în procedurile de creare și actualizare ale rețelei de obiecte sau pot fi efectuate la cerere, fiind obiectul unei componente specializate. În cele ce urmează ne vom referi mai pe larg la ceea ce considerăm că ar trebui să conțină o componentă specializată de verificare a bazei de cunoștințe.

Considerăm că, pentru componenta de verificare a bazei de cunoștințe poate fi dezvoltat un sistem expert. In acest mod, inginerului de cunoștințe îi este pus la dispoziție un asistent inteligent, capabil de verificări complexe, atît sintactice, cît și semantice. In baza de cunoștințe a acestuia vor fi înmagazinate cazuri de erori (virtuale) și sugestii pentru rezolvare. Un caz de eroare este un fragment generic de structură de obiecte. Baza de cunoștințe a sistemului expert verificat este explorată de către sistemul expert care constituie componenta de verificare. În cazul în care în baza de cunoștințe a sistemului analizat este găsită o configurație similară cu un caz de eroare, este anunțat cel care face verificarea și, eventual, îi este propusă și o rezolvare.

Cazurile de eroare pot fi clasificate în funcție de natura regulilor violate. Pot fi astfel erori de utilizare a limbajului de reprezentare pe obiecte sau erori legate de semantica domeniului pentru care este dezvoltat sistemul expert verificat.

Cazurile de erori de utilizare a limbajului sînt generale,

valabile pentru orice aplicație. Cunoștințele referitoare la aceste situații vor fi furnizate o dată cu sistemul expert de verificare. Se prezintă în cele ce urmează un exemplu de astfel de eroare (virtuală). Obiectele 01 și 02 se află în relația "este un" ("moștenește pe"); ambele obiecte au slot-ul S1. Valoarea slot-ului S1 din 01 este obiectul 03, iar valoarea slot-ului S1 din 02 este obiectul 04. Dacă 03 nu este în relația "este un" cu 04, este semnalată o posibilă eroare /3/. In baza de cunoștințe a sistemului expert de verificare poate fi înmagazinată și o sugestie de corectare a erorii. In acest caz, se poate sugera ca 03 să fie printre obiectele care moștenesc de la 04 sau, invers, 04 să fie printre obiectele de la care 03 moștenește (utilizatorului îi vor fi afișate și listele corespunzătoare de obiecte).
Erorile de semantică a domeniului sînt proprii unei aplicații sau unui grup de aplicații. De aceea, în cazul în care în mediul de programare sînt puse la dispoziție aplicații generice /3/, (de exemplu, sisteme experte cadru de diagnosticare), acestea trebuie să conțină și cunoștințe specifice de verificare, care vor fi cunoștințe utilizate de sistemul expert de verificare. O altă situație este cea în care este dezvoltată o aplicație, independent de orice altă aplicație generică. În acest caz, cunoștințele referitoare la eventualele erori trebuie furnizate de inginerul de cunoștințe pe baza informațiilor de la expertul în domeniu.
În finalul acestui capitol vrem să mai adăugăm faptul că, în afara situațiilor de eroare (care sînt cunoștințe superficiale), sistemul expert de verificare poate folosi și cunoștințe (de adîncime) referitoare la modul cum trebuie să fie folosite corect relațiile dintre obiecte. In acest sens, se poate folosi și un "model" al felului în care trebuie să "arate" un program corect.
În situația în care în dezvoltarea sistemului expert se folosește și un nivel conceptual (la care este elaborat un model conceptual al bazei de cunoștințe /7/), specificațiile de la acest nivel pot fi folosite pentru verificarea bazei de cunoștințe. O soluție în acest caz este implicarea sistemului expert pentru verificare în validarea bazei de cunoștințe în raport cu modelul conceptual.

4. Verificarea transmiterii de mesaje

Similar verificării relațiilor dintre obiecte și în cazul verificării protocolului de transmitere de mesaje, se pot folosi instrumente de vizualizare a unor aspecte ale bazei de cunoștințe. In acest sens sînt utile:

- lista mesajelor la care poate răspunde un obiect;
- lista metodelor unui obiect;
- lista perechilor mesaj-metodă pentru un obiect;
- lista mesajelor ce se pot transmite obiectelor dintr-o bază de cunoștințe.

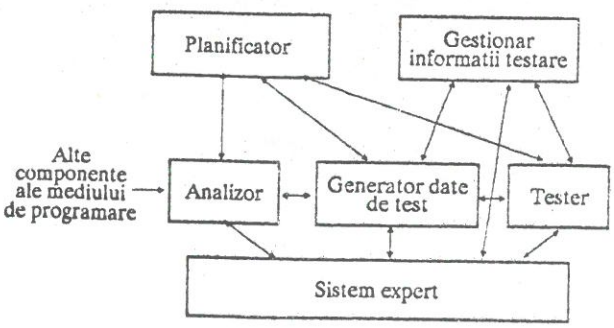
Pentru verificarea existenței obiectelor cărora li se trimit mesaje și a posibilității acestora de a răspunde la mesaje se pot implementa instrumente automate. Astfel, în urma verificării se obține lista obiectelor cărora li s-a transmis un mesaj la care nu au metode asociate sau moștenite și lista obiectelor nedefinite în baza de cunoștințe, cărora li s-a transmis un mesaj. Trebuie remarcat totodată că, aceste verificări pot fi făcute doar în cazul în care numele mesajului și obiectul căruia i se transmite sînt cunoscute static, fără a executa programul.
Ca și în cazul verificării relațiilor dintre obiecte, pot fi folosite metode cunoscute din verificarea programelor în general, cum sînt inspecțiile și parcurgerile /6/.

5. Testarea sistemelor expert scrise în medii de programare orientate spre obiecte

În acest capitol vom analiza caracteristicile pe care trebuie să le aibă un asistent "inteligent" pentru testare /3, 8, 9/. Datorită facilităților oferite de mediile de programare orientate spre obiecte, considerăm că se poate dezvolta ușor un modul bazat pe cunoștințe (un sistem expert) pentru asistarea testării sistemelor experte. Folosind pentru implementarea acestui modul tot reprezentarea prin obiecte, se poate obține o integrare ușoară a sa în mediul de programare. Considerăm util, în acest sens, ca modulul de testare să fie cuplat cu alte module ale mediului de programare, cum ar fi componentele de verificare și cele de achiziție a cunoștințelor.
După cum am spus mai sus, ne propunem să dezvoltăm un modul de testare "inteligent". In acest sens, el trebuie să permită direcționarea testării, să memoreze informațiile legate de testare și, bineînțeles, să efectueze testarea. Pentru a realiza toate aceste deziderate propunem ca modulul de testare să aibă următoarele componente:

- analizorul sistemului expert din punct de vedere al testării sale;
- planificatorul testării;
- generatorul de date de test;
- testerul;
- gestionarul informațiilor referitoare la testare.

Schema interacțiunii componentelor modulului de testare este dată în figura de mai jos.
După cum se știe, nu se poate stabili dacă un program nu are erori, ci numai dacă are. De aceea, procesul de testare trebuie să fie efectuat pînă cînd, pe baza unei analize a programului, se va obține o încredere suficientă în gradul de calitate a programului (în sensul



unui număr de erori sub o limită). Planificarea testării, direcționarea testării spre o zonă sau alta a sistemului expert, precum și decizia de continuare sau oprire a testării sînt efectuate de componenta denumită în figură "planificator". Acesta funcționează pe baza:

- cunoștințelor referitoare la testarea programelor orientate spre obiecte;
- cunoștințelor referitoare la sistemul expert dezvoltat, obținute prin intermediul analizorului.

Analizorul extrage din sistemul expert informațiile utile testării, cum ar fi de exemplu, porțiunile (obiectele) cu cea mai mare probabilitate de a conține erori. Tot analizorului îi revine sarcina de a stabili gradul de calitate a sistemului dezvoltat. Analizorul, în afara codului sistemului expert, poate folosi și informații furnizate de alte module ale mediului de programare. In acest sens sînt utile listele de obiecte, slot-uri, mesaje, referințe încrucișate avute în vedere în capitolele anterioare. De asemenea, pot fi utile informațiile furnizate de componenta de verificare. De exemplu, unele erori potențiale semnalate de verificator pot fi un punct de plecare în testare, pentru a clarifica prezența sau absența unei erori.

Analizorul poate avea ca rezultat o listă de fapte sau chiar un model din punct de vedere al testării sistemului expert analizat. Faptele, respectiv modelul, sînt actualizate după efectuarea unui set de teste.

Generatorul de date de test este o componentă care stabilește, pe baza analizei sistemului dezvoltat, seturi de date care exersează diverse zone cu erori potențiale.

Testerul efectuează testarea propriu-zisă. In acest scop este necesară existența unui "oracol" în sensul discutat anterior. De aceea, considerăm testerul ca o componentă interactivă, în care utilizatorul furnizează rezultatele așteptate în urma testării. Testerul trebuie să realizeze execuția on-line, creînd contextul necesar testării pentru fiecare set al datelor de test.

Gestionarul informațiilor de testare memorează seturile datelor de test exersate și rezultatele obținute. Rațiunea acestei activități constă în facilitarea generării de noi date de test și a reluării testelor făcute în momentul modificării bazei de cunoștințe a sistemului expert analizat. Informațiile referitoare la testele efectuate pot fi memorate chiar în obiectele sistemului expert. Astfel, în eventualitatea reutilizării unor obiecte pentru alte aplicații, pot fi refolosite și testele efectuate.

6. Concluzii

În lucrare sînt analizate posibilitățile de integrare într-un mediu de programare orientat spre obiecte, a unor module destinate verificării și testării sistemelor experte dezvoltate. Avînd în vedere experiența obținută în proiectarea și implementarea mediului XRL /10/, se propune realizarea acestor module pe baza tehnologiei sistemelor experte, în reprezentarea bazată pe obiecte oferită în mediu. In acest sens, se dorește obținerea de asistenți inteligenți pentru verificare și testare, apreciindu-se că aceștia vor accelera dezvoltarea de programe bazate pe cunoștințe (sisteme experte). De asemenea, automatizarea celei mai mari părți a activităților de verificare și testare are consecințe directe asupra calității programelor rezultate.

Pasul următor pe care ni-l propunem este validarea ideilor prezentate prin implementarea unor module de verificare și testare în mediul XRL. De asemenea, ne vom preocupa și de alte aspecte ale verificării și testării, în special de impactul utilizării unui nivel conceptual în dezvoltarea de sisteme experte și în achiziția cunoștințelor.

Bibliografie

1. WEITZEL, J., KERSCHBERG, L.: *Developing Knowledge-based Systems: Reorganizing the System Development Life Cycle.* In: Communications of the ACM, aprilie, 1989, vol. 32, nr. 4, pp. 482-488.
2. DOYLE, J.: *Expert Systems and the 'Myth' of Symbolic Reasoning.* In: IEEE Transactions on Software Engineering, noiembrie, 1985, vol. SE-11, pp. 1386-1390.
3. SCHOEN, E., SMITH, R. G., BUCHANAN, B. G.: *Design of Knowledge-based Systems with a Knowledge-based Assistant.* In: IEEE Transactions on Software Engineering, vol.14, nr. 12, decembrie, 1988.
4. RICHER, H.: *An Evaluation of Expert Eystem Development Tools,* raport KSL 85- 19, Stanford, iunie, 1986.
5. ERMAN, L. D., LARK, J. S., HAYES-ROTH, F.: *ABE: An Environment for Engineering Intelligent Systems.* In: IEEE Transactions on Software Engineering, vol. 14, nr. 12, decembrie, 1988, pp. 1758-1770.
6. EWICS: *Techniques for Verification and Validation of Software- related Software.* In: Computers & Standards 4, 1985, pp. 101-112.
7. BENBASAT, I., DHALIWAL, J. S.: *The Validation of Knowledge Acquisition Methodology and Techniques.* In: Lucrările EKAW '89, pp. 60-74.
8. TEPANDI, J.: *A Knowledge-based Approach to the Specification-based Testing.* In: Computers and Artificial Intelligence, vol. 7, nr. 1, 1988, pp. 39-48.
9. SEDLMEYER, R., THOMSON, W., JOHNSON, P.: *Knowledge-based Fault Localization in Debugging.* In: Journal of Systems and Software, 3, 1983, pp. 301-307.
10. BĂRBUCEANU, M., TRĂUȘAN-MATU, S., MOLNAR, B.: *The XRL2 Manual,* ITCI-București, 1988.