KAKO DO BOLJ FLEKSIBILNE PREDSTAVITVE POSLOVNIH PRAVIL?

Jurij Jaklič, Janez Grad
Ekonomska fakulteta, Univerza v Ljubljani
e-pošta: jurij.jaklic@uni-lj.si

Povzetek

V prispevku predlagamo način predstavitve poslovnih pravil v informacijskih sistemih, ki omogoča večjo fleksibilnost poslovnih pravil, njihovo večkratno uporabljivost, hkrati pa približa delo z njimi tudi uporabniku. Predlagani način predstavitve temelji na trinivojskem pogledu na odločitvena pravila, s katerimi oblikujemo poslovna pravila, objektnem modelu ter logični odločitveni tabeli. Rešitev smo tudi praktično preverili na dva načina: z izdelavo prototipa, ki v celoti implementira trinivojski pogled na pravila, ter z analizo možne uporabe na konkretnem primeru.

Summary

In the paper we propose and describe a way of business rules representation within information systems that enables their greater flexibility and reusability. The representation is based on the three level view of decision rules, object oriented models and decision tables. The proposed approach of representation was carried through and tested in two ways: (i) by designing a prototype which fully implemented the three level view of decision rules, and (ii) by analyzing its possible application within a selected case problem.

1 PROBLEMI PREDSTAVITVE ODLOČITVENIH PRAVIL V INFORMACIJSKIH SISTEMIH

1.1 Problematika predstavitve poslovnih pravil

Vsak poslovni sistem se mora odzivati na hitre in neprestane spremembe v okolju, kar povzroča probleme pri vzdrževanju in upravljanju informacijske tehnologije v podjetjih [19]. Ker informatiki pogosto ne morejo pravočasno ugoditi zahtevam uporabnikov po novih rešitvah in spremembah obstoječih, se vedno pogosteje dogaja, da uporabniki (resda preprostejše) rešitve razvijejo samostojno s pomočjo sodobne, prijazne tehnologije, pri tem pa jim z nasveti lahko pomagajo tudi informatiki. Hkrati pridejo do rešitev tudi hitreje, saj poznajo področje, za katerega oblikujejo rešitev, in se s tem izognejo pogosti in intenzivni komunikaciji z analitiki in načrtovalci [13]. Seveda morajo imeti uporabniki na voljo ustrezno temeljno znanje informatike in do uporabnika prijazno in za uporabo enostavno informacijsko tehnologijo.

1.2 Dosedanji razvoj

V dosedanjem razvoju informacijske tehnologije je bil storjen velik napredek na področju predstavitve podatkov. Preprost relacijski podatkovni model, neodvisnost podatkov od programskih rešitev, ki jo zagotavljajo sistemi za upravljanje baz podatkov, logična neodvisnost podatkov, ki je dosežena s trinivojskim modelom, ter prijazni grafični uporabniški vmesniki – vse to omogoča, da lahko uporabnik s temeljnim znanjem informatike danes *samostojno* modelira podatke, zgradi preprosto bazo podatkov in jo uporablja. Zato je tudi večina poslovnih rešitev, ki jih izdelajo uporabniki samostojno, s področja manipulacije (poizvedovanje po bazi podatkov) ali analize podatkov (preglednice).

Objektni pristop je omogočil velik napredek tudi na področju izgradnje in vzdrževanja uporabniških vmesnikov (zaslonskih mask in poročil). Tako je mogoče danes hitro narediti grafične uporabniške vmesnike s postavljanjem objektov na delovno površino ter preprostim povezovanjem dogodkov in aktivnosti.

Tudi za poslovna pravila velja, da jih moramo, če želimo slediti in se prilagajati spremembam v okolju, neprestano spreminjati, saj predstavljajo del modela realnega sveta. Za uporabnika to pomeni težavo, saj v največ primerih tega ne zna in ne zmore storiti sam. Zato bi bilo zaželeno, da bi lahko uporabnik samostojno manipuliral (definiral, spreminjal, dodajal, brisal) tudi poslovna pravila.

1.2.1 Vizualizacija poslovnih pravil

V fazi analize sistema in načrtovanja informacijskega sistema moramo uporabiti za predstavitev odločitvenih pravil tehniko, ki je hkrati dovolj blizu naravnemu načinu razmišljanja (opisni pripovedi) in tudi dovolj formalna, da lahko pravila čim lažje kodiramo. Za predstavitev odločitvenih pravil poznamo danes več formalnih tehnik, med katerimi so najbolj znane: *pravila "če - potem", odločitvena drevesa, logične odločitvene tabele* in *logični diagrami poteka*. Njihova skupna značilnost je predvsem usmerjenost v dokumentacijo, torej zapis – prikaz poslovnih pravil, kar je posledica namena njihove uporabe. Zaradi težnje po približevanju uporabniku je poudarjena komponenta vseh tehnik predvsem vizualizacija poslovnih pravil, hkrati pa so te tehnike oddaljene od predstavitve istih pravil v informacijskih sistemih. To pomeni, da obstaja prepad med zunanjimi predstavitvami poslovnih pravil in njihovim kodiranjem.

Na področju primernosti različnih predstavitev odločitvenih pravil (logične odločitvene tabele, odločitvena drevesa, pravila "če - potem", strukturirana angleščina ...) je bilo opravljenih več empiričnih raziskav, na primer [15], [26] in [30], ki te predstavitve primerjajo med seboj z vidika uporabnika. Večina teh raziskav ni pokazala prednosti pri uporabi ene tehnike pred ostalimi predstavitvami ali pa si izsledki raziskav med seboj celo nasprotujejo. Zato lahko sklepamo, da je izbor primernega načina predstavitve odločitvenih pravil pogojen predvsem z osebnimi preferencami in z vrsto (kompleksnostjo) problema.

1.2.2 Logične odločitvene tabele

Logična odločitvena tabela je način prikaza odločitvenih pravil z uporabo tabelarične strukture. Zaradi zgoščenega in preglednega načina prikazovanja pogojev in aktivnosti [12], ki so vključene v odločitvah, so logične odločitvene tabele zelo primerne za prikazovanje, razumevanje in spremljanje logike dogajanja kompleksnih odločitvenih problemov. S povezovanjem več odločitvenih tabel lahko modeliramo poljubno kompleksno odločitveno situacijo.

Tako opredeljene logične odločitvene tabele so, kot že omenjeno, namenjene predvsem predstavitvi – vizualizaciji odločitvenih pravil. V nadaljnjem razvoju logičnih odločitvenih tabel pa se mnogo uporablja tudi *konceptualna definicija logične odločitvene tabele*, ki ne implicira načina predstavitve odločitvenih pravil, saj logično odločitveno tabelo definiramo [29] kot *funkcijo, ki preslika kartezični produkt stanj pogojev v kartezični produkt stanj aktivnosti*:

$$DT: CT_1 \times CT_2 \times ... \times CT_{cnum} \rightarrow V_1 \times AV_2 \times ... \times AV_{anum},$$

kjer sta CT_i urejena množica n_i *stanj pogojev* subjekta pogoja CS_i ter AV_j množica *vrednosti aktivnosti*.

Pri pridobivanju in vzdrževanju odločitvenih pravil se srečujemo s protislovji in pomanjkljivostmi v teh pravilih. Zato je izrednega pomena preverjanje in potrjevanje pravilnosti opisa odločitvene logike. V tem pogledu imajo logične odločitvene tabele pred ostalimi predstavitvenimi tehnikami pomembno prednost, saj je z njihovo uporabo mogoče na enostaven način preverjati želene lastnosti odločitvenih pravil: *popolnost, odsotnost protislovnosti* ter *odsotnost odvečnosti*.

Popolnost. Pravila so *popolna*, če lahko odgovorijo na vsa možna stanja v domeni. Z drugimi besedami, za vsako kombinacijo vrednosti parametrov mora biti s pravili določeno, kaj naj se zgodi. Popolnost torej ustreza robustnosti [8] Če so pravila predstavljena v odločitveni tabeli, je popolnost mogoče enostavno preverjati s časovno zahtevnostjo $O(n)$, kjer je n število pogojev. Časovno bolj zahtevno pa je iskanje manjkajočih pravil, to je $O(2^n)$ [8].

Dvoumnost, protislovnost in odvečnost. Kadar obstaja kombinacija vrednosti parametrov, ki zadošča pogojem dveh ali več pravil z različnimi množicami aktivnosti, pravimo, da so pravila *dvoumna*. Kadar se množice aktivnosti dvoumnih pravil med seboj izključujejo (ne morejo biti izvršene hkrati), so pravila *protislovna*. Če pa so množice aktivnosti ekvivalentne, pa so nekatera odločitvena pravila *odvečna*. Časovna zahtevnost znanih algoritmov za odkrivanje protislovnosti in odvečnosti je $O(n^2)$ [8].

V [22] avtorja razdelita uporabo logičnih odločitvenih tabel v dve fazi: *aktivno*, v kateri modeliramo odločitveni proces, in *pasivno*, v kateri le uporabljamo logične odločitvene tabele za odločanje. Očitno je, da so pri pasivni uporabi nepopolnost, protislovnost in odvečnost nezaželene lastnosti poslovnih pravil. Avtorja pa zagovarjata pozitivno vlogo omenjenih lastnosti v fazi modeliranja.

Logične odločitvene tabele so bile zaradi omenjenih možnosti preverjanja popolnosti in nedvoumnosti v ta namen že uporabljene v ekspertnih sistemih [16] ter za testiranje programov [23]. Njihova uspešnost se je pokazala tudi v raziskavi [31], v kateri so ugotovili precej nepopolnosti, protislovnosti in odvečnosti na primeru kliničnih navodil.

1.2.3 Kodiranje poslovnih pravil

Čeprav so poslovna pravila jedro kompleksnih poslovnih uporabniških rešitev, je bilo prav na tem področju narejenega najmanj, saj je potrebno danes veliko večino programske logike kodirati v programskih jezikih tretje generacije. Čeprav so bili tudi na tem področju v zadnjih letih narejeni določeni koraki, ki jih kratko predstavljamo v nadaljevanju, pa ostaja problem večje avtomatizacije kodiranja poslovne logike še vedno odprt [2].

Omejitve različnih vrst: referenčne, omejitve območja ali eksplicitne omejitve, ki jih je mogoče definirati v razširjenem relacijskem modelu (t.i. RM/T model [27]), predstavljajo določene vrste poslovnih pravil. Mnogo več poslovnih pravil je mogoče v okviru baze podatkov kodirati s sprožilci. Robustnost informacijskih sistemov se s tem poveča, saj se poveča konsistentnost poslovnih pravil v rešitveh, ki uporabljajo isto bazo podatkov.

Za objektni pristop je značilno, da so poslovna pravila kodirana v metodah objektov, z drugimi besedami na strani strežnikov. Tudi to ima za posledico lažje vzdrževanje, pravila so bolj konsistentna, razviti sistemi so bolj prožni ...

V obeh primerih: pri sprožilcih in pri metodah objektov pa so poslovna pravila še vedno kodirana v programskem jeziku tretje ali četrte generacije. Prednosti, ki jih prinaša objektni pristop, so predvsem posledica spremenjene topologije programa, manj pa posledica spremembe načina kodiranja poslovnih pravil.

1.3 Želene lastnosti predstavitve poslovnih pravil

Zdaj lahko opredelimo, kakšne zahteve bi moral izpolnjevati način predstavitve poslovnih pravil:

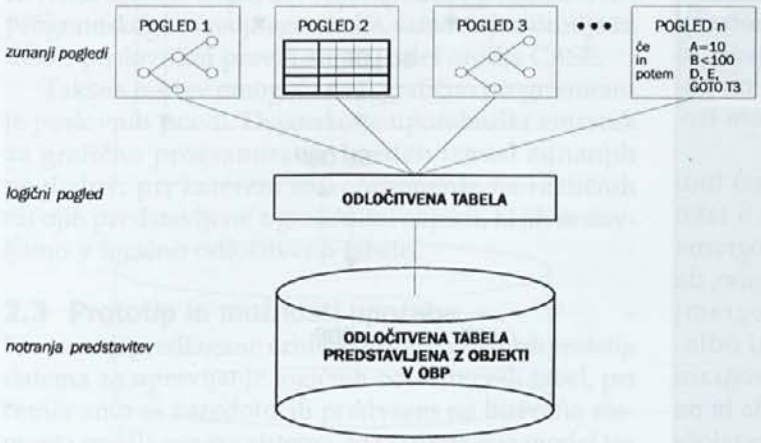
- Isto tehniko prikaza poslovnih pravil naj bi bilo mogoče uporabljati v vseh fazah razvoja informacijskega sistema, tudi pri kodiranju in vzdrževanju sistema. Z drugimi besedami, ne glede na način notranje predstavitve poslovnih pravil (kodiranja) naj si bo mogoče poslovna pravila ogledati in jih spreminjati z uporabo ene izmed tehnik, ki so namenjene predvsem vizualizaciji in so prijazne do uporabnika hkrati pa dovolj formalne za zapis v računalniško podprtem informacijskem sistemu.
- Orodje za kodiranje in ažuriranje poslovnih pravil naj bo vizualno.
- Vizualizacija naj bo fleksibilna, kar pomeni, da lahko uporabnik izbere tako tehniko za predstavitev poslovnih pravil, ki najbolj ustreza njemu in obravnavanemu problemu.
- Kodirana pravila naj bodo fleksibilna, kar pomeni, da naj jih bo mogoče spreminjati na enostaven način, spremembe pa naj se takoj odražajo v informacijskem sistemu.
- Želene lastnosti poslovnih pravil (popolnost, odsotnost odvečnosti in protislovnosti) naj se preverjajo avtomatično.
- Poslovna pravila in njihove komponente naj bo mogoče uporabiti večkrat, ne samo znotraj ene programske rešitve.

2 PREDLAGANA REŠITEV

Ugotovimo lahko, da že obstajajo tehnike in pristopi s posameznimi značilnostmi, ki smo jih opredelili kot želene lastnosti predstavitve poslovnih pravil. Na primer: mogoče je izbirati različne tehnike za vizualizacijo glede na uporabnikove preference in značilnosti problema, logične odločitvene tabele omogočajo avtomatično preverjanje lastnosti poslovnih pravil, objektni pristop zagotavlja fleksibilnost in ponovno uporabljivost ... Menimo, da je mogoče te tehnike in pristope ustrezno kombinirati tako, da hkrati dosežemo večino zastavljenih ciljev. V ta namen lahko uporabimo trinivojski pogled na poslovna pravila, ki ga predstavljamo v nadaljevanju.

2.1 Trinivojski pogled na poslovna pravila

Kombiniranje tehnik in pristopov omogoča trinivojski pogled na poslovna pravila, ki ločuje zunanje poglede na poslovna pravila, to je poglede uporabnikov, konceptualni pogled, na katerem so pravila predstavljena z logičnimi odločitvenimi tabelami, ter notranjo predstavitev, kjer so logične odločitvene tabele in vse njihove komponente shranjene kot objekti v objektni bazi podatkov[1] (slika 1).



Slika 1: Trinivojski pogled na poslovna pravila

V tem primeru uporabimo logične odločitvene tabele le kot *logično predstavitev* poslovnih pravil, kar omogoča avtomatsko in sprotno preverjanje popolnosti, protislovnosti in odvečnosti.

Uporabnik pa si sam izbere *zunanji pogled* na poslovna pravila, oziroma predstavitev le-teh. Ta pogled je lahko spet logična odločitvena tabela, lahko so

1 Seveda bi lahko uporabili tudi objektno relacijske (univerzalne) baze podatkov. Ker je mogoče objektni pristop uporabiti tudi v okolju relacijskih baz podatkov, je predlagani pristop primeren tudi za relacijska okolja.

odločitvena drevesa, pravila "če - potem" in podobno. Z neodvisnostjo med zunanjimi predstavitvami in logično predstavitvijo in s primernimi grafičnimi vmesniki za posamezne vrste zunanjih pogledov omogočimo uporabniku, da na enostaven in sebi prilagojen način kodira in uporablja kompleksna poslovna pravila.

Pri modeliranju odločitvenih pravil in odločanju je vizualizacija, kot ugotavljajo avtorji v [26], izrednega pomena. Poglavitni namen različnih tehnik predstavitve odločitvenih pravil je prav vizualizacija. Če (kadar) namen programske rešitve ni vizualizacija (npr. pri izvajanju), oblika predstavitve ni pomembna.

Če analiziramo predlagani trinivojski pogled na odločitvena pravila, ugotovimo, da ta popolnoma ustreza tem ugotovitvam, saj ločuje zunanji pogled (predstavitev, vizualizacijo) in logični ter notranji nivo, pri katerih vizualizacija ni pomembna, saj uporabnik teh nivojev "ne vidi". Na logičnem nivoju smo uporabili konceptualno definicijo logičnih odločitvenih tabel (glej podpoglavje 1.2.2), ki je neodvisna od predstavitve – vizualizacije.

Notranja predstavitev oziroma *implementacija* logičnih odločitvenih tabel je načeloma lahko poljubna, vendar predlagamo, da jih implementiramo z objekti v *objektni bazi podatkov*. Z analizo konceptualne definicije logične odločitvene tabele lahko ugotovimo, da je ta sestavljena iz podkomponent in te naprej iz podkomponent na nižjih nivojih: pravil, subjektov pogojev, vrednosti subjektov pogojev, subjektov aktivnosti ... Z objekti predstavimo tako logično odločitveno tabelo kot tudi vse njene komponente. S postavitvijo logičnih odločitvenih tabel v okolje objektno usmerjenih baz podatkov podedujemo pridobitve objektno usmerjenega pristopa: ponovno uporabljivost vseh elementov (od odločitvenih tabel do pravil in pogojev), možnost vzporednega izvajanja pravil brez dodatnega programiranja ...

Preslikave med različnimi nivoji omogočajo, da se vsaka uporabnikova sprememba poslovnih pravil, ki se zgodi na zunanjem nivoju, takoj odraža v notranji predstavitvi in s tem v vseh programskih rešitvah, ki ta poslovna pravila uporabljajo, vključno z vsemi zunanjimi pogledi. Posledično je mogoče v vsakem trenutku spremeniti pogled na pravila.

Na logične odločitvene tabele lahko gledamo tudi kot na način predstavitve programske kode. S tako implementacijo odločitvenih tabel torej del programa predstavimo kot objekte v bazi podatkov. Če vemo, da je mogoče vsak logični diagram poteka (program) emulirati z eno ali več povezanimi logičnimi odločitvenimi tabelami [20], potem lahko vsak program predstavimo v objektni bazi podatkov. Ideja, da bi na program gledali kot na objekte, ni nova, je pa obetajoča in še dokaj neraziskana [1].

2.2 Implementacija rešitve

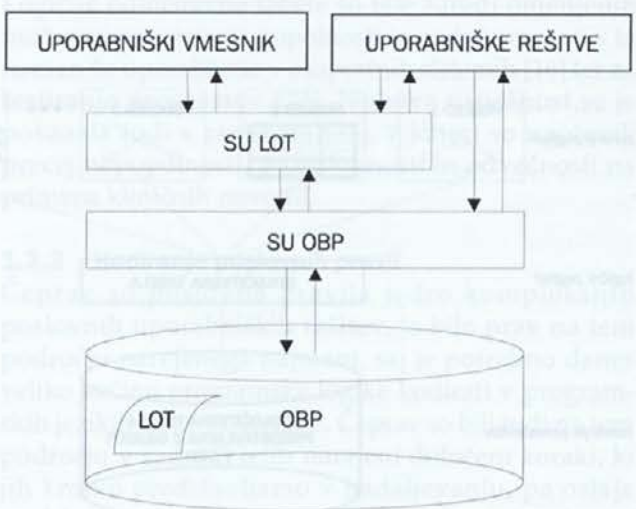
Predlagano rešitev implementiramo s *sistemom za upravljanje logičnih odločitvenih tabel*, s pomočjo katerega lahko:

- tvorimo nove komponente logične odločitvene tabele,
- iz obstoječih komponent tvorimo nove na višjih nivojih,
- tvorimo logično odločitveno tabelo iz obstoječih komponent,
- spreminjamo obstoječe komponente in logične odločitvene tabele ter
- izvajamo logične odločitvene tabele.

Del sistema za upravljanje logičnih odločitvenih tabel pa so tudi uporabniški vmesniki, ki ustrezajo zunanjim pogledom na poslovna pravila (slika 2).

Sistem za upravljanje logičnih odločitvenih tabel sestavljajo torej *predefinirani razredi objektov*, ki predstavljajo komponente logične odločitvene tabele, skupaj z implementacijo pripadajočih operacij ter *uporabniški vmesniki* za posamezne vrste zunanjih pogledov. Najpomembnejše metode posameznega razreda objektov so tiste, ki omogočajo delo z logičnimi odločitvenimi tabelami: povezovanje komponent v komponente na višjem nivoju, odstranjevanje komponent, spreminjanje, izvajanje – ovrednotenje komponent, preverjanje želenih lastnosti poslovnih pravil ... Poleg teh metod pripadajo posameznim razredom objektov še metode, ki omogočajo delo na nivoju zunanjih pogledov (izpis objekta, na primer pogoja, za različne poglede na logične odločitvene tabele).

Predlagamo sedem razredov objektov, ki omogočajo implementacijo logičnih odločitvenih tabel v objektni bazi podatkov: razred *logičnih odločitvenih tabel*,



Slika 2: Arhitektura sistema

razred *parametrov*, razred *pravil*, razred *pogvr*, razred *subjektov pogojev*, razred *operandov* ter razred *subjektov aktivnosti*.

Subjekti pogojev, ki nastopajo v logičnih odločitvenih tabelah, so v večini primerov neki relacijski izrazi, v katerih primerjamo med seboj dve ali več vrednosti. V objektnem okolju bodo te vrednosti svojstva (atributi) nekih objektov. Tudi kadar nastopajo v pogojih neke fiksne vrednosti, ki nastopajo le v logičnih odločitvenih tabelah, jih lahko obravnavamo kot atribute objektov. Objekte, katerih svojstva nastopajo v pogojih, bomo imenovali *parametri* logične odločitvene tabele. Sprememba operanda v subjektu pogoja je torej mogoča na dva načina: spremeni se vrednost atributa parametra ali pa zamenjamo parameter z drugim objektom istega razreda.

Objekti razreda *pogvr* predstavljajo povezavo med subjekti pogojev in pripadajočo (zahtevano) vrednostjo v nekem pravilu.

V objektnem okolju je logična odločitvena tabela povezana z drugimi objekti na dva načina:

- V subjektih pogojev uporablja vrednosti atributov objektov – parametrov logične odločitvene tabele.
- Z aktivnostmi spreminja stanje sistema. Subjekti aktivnosti torej predstavljajo zahtevo objektu naj izvede svojo metodo. Tudi ti objekti so parametri logične odločitvene tabele.

Izvajanje logične odločitvene tabele, predstavljene z objekti, temelji na klicih metod za ovrednotenje posameznih pravil, ki pripadajo tabeli. Postopek se nadaljuje po hierarhiji komponent vse do podkomponent na najnižjih nivojih. Izvajanje torej poteka prav tako modularno, kot je modularna struktura logične odločitvene tabele. Modularno izvajanje seveda pomeni, da je izvajanje tako predstavljenih poslovnih pravil časovno bolj zahtevno od enakovrednega programa, kodiranega v programskem jeziku tretje generacije.

Tako izdelan sistem za upravljanje logičnih odločitvenih tabel je lahko del objektne baze podatkov, del programskega razvojnega okolja, samostojno orodje za delo s poslovnimi pravili ali pa je del orodja CASE.

Takšna rešitev omogoča tudi grafično programiranje poslovnih pravil. Dejansko je uporabniški vmesnik za grafično programiranje le eden izmed zunanjih pogledov, pri katerem so komponente na različnih nivojih predstavljene z grafičnimi objekti, ki jih sestavljamo v logično odločitveno tabelo.

2.3 Prototip in možnosti uporabe

Na osnovi predlagane arhitekture smo izdelali *prototip* sistema za upravljanje logičnih odločitvenih tabel, pri čemer smo se osredotočili predvsem na bistvena elementa predlaganega sistema, to je trinivojski model ter predstavitev komponent z objekti, kar ima za posledico modularno sestavljivost odločitvenih pravil ter ponovno uporabljivost. Poglavitne cilje pri izdelavi prototipa lahko strnemo v naslednje:

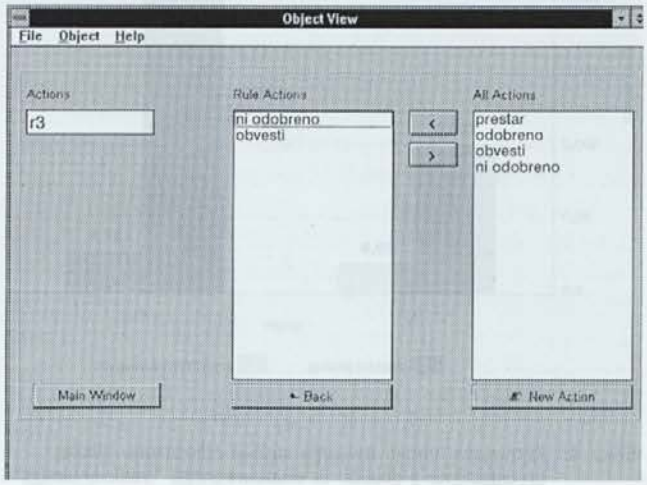
- Proučevati smo želeli predvsem funkcionalnost sistema, zato nismo posvečali posebne pozornosti uporabniškemu vmesniku, čeprav je ta izjemnega pomena, če želimo sistem nameniti uporabniku.
- Iz istega razloga tudi nismo implementirali optimizacijskih metod (npr. relativne frekvence pravil).
- Prototip naj upošteva trinivojski model, s katerim uporabniku ponujamo različne poglede na poslovna pravila, ki so na logičnem nivoju predstavljena z logičnimi odločitvenimi tabelami, le-te pa dejansko predstavljene v objektnem modelu z objekti, ki so komponente odločitvene tabele. Poseben poudarek smo dali spodnjemu nivoju, torej notranji predstavitvi z objekti.

V okviru analize izdelave in uporabe prototipa rešitve so nas posebej zanimali naslednji vidiki:

- težave pri implementaciji predlaganega modela,
- možnost uporabe objektnega pristopa, ki ga predvideva rešitev, v okoljih, kjer informacijski sistem temelji na relacijskem podatkovnem modelu, ter
- časovna zahtevnost izvajanja z objekti predstavljene logične odločitvene tabele.

V prototipu smo v celoti implementirali trinivojsko arhitekturo sistema. Preslikava med logičnim in notranjim nivojem je implementirana s povezavami med objekti, ki komponente povežejo v logično odločitveno tabelo, ter metodami objektov, ki omogočajo delo z logično odločitveno tabelo (tvorba, izvajanje, ažuriranje, preverjanje pravilnosti).

Na zunanjem – predstavitvenem nivoju lahko uporabnik izbere pogled na odločitvena pravila, ki mu najbolj ustreza. Izbere v pogovornem oknu tabelo in nato način prikaza pravil. V prototipu sta implementirana le dva pogleda: tabelarični in objektni.



Slika 3: Objektni pogled na pravila – "sestavljanje aktivnosti"

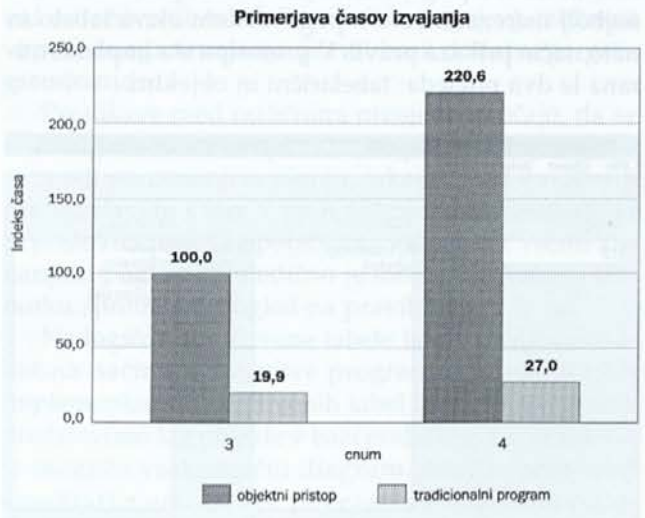
Objektni pogled je blizu notranji predstavitvi odločitvenih pravil, saj uporabnik vidi komponente in podkomponente, ki jih sestavljajo. Uporabnik lahko logično odločitveno tabelo sestavlja z že obstoječimi komponentami (ponovna uporabljivost) ali pa tvori nove. Nadaljnji razvoj tega pogleda bi nas pripeljal do vmesnika za grafično programiranje poslovnih pravil.

2.3.1 Hitrost izvajanja logičnih odločitvenih tabel, predstavljenih z objekti

Kot že povedano v predstavitvi predlagane rešitve, je pričakovana časovna zahtevnost izvajanja tako predstavljenih poslovnih pravil večja od enakovrednega programa, kodiranega v programskem jeziku tretje generacije.

Primerjavo učinkovitosti (hitrosti izvajanja) smo preverili z eksperimentom, v katerem smo merili hitrost izvajanja logične odločitvene tabele z omejenimi pogoji, predstavljene z objekti, in ekvivalentnega programa, kodiranega v programskem jeziku tretje generacije.

Na sliki 4 so prikazani dobljeni rezultati merjenja velikega števila ponovitev izvajanja logične odločitvene tabele z elementarnimi pravili in enostavnimi subjekti pogojev, predstavljene z objekti in s tradicionalnim programom, za število subjektov pogojev *cnum*=3 in *cnum*=4. Ker nas zanima primerjava časov, na grafikonu niso prikazane absolutne vrednosti, temveč le indeksi. Po pričakovanjih so bili časi izvajanja logične odločitvene tabele, predstavljene z objekti, daljši.



Slika 4: Primerjava hitrosti izvajanja logične odločitvene tabele, predstavljene z objekti, in enakovrednega "tradicionalnega" programa

Velik delež presežnega časa izvajanja gre na račun modularnosti in sestavljivosti logične odločitvene tabele, če je predstavljena z objekti v objektni bazi podatkov. Kljub temu lahko za pospešitev izvajanja uporabimo nekatere tehnike, ki so že znane za tradicionalne logične odločitvene tabele, npr. *relativno frekvenco pravil* [21], pri čemer naj bo frekvenca pravila eden od privatnih atributov objektov, ki predstavljajo pravila.

Eden izmed virov nepotrebnega podaljševanja časa izvajanja je večkratno vrednotenje istih komponent logične odločitvene tabele. Tako se npr. za vsako pravilo, v katerem nastopa neki subjekt pogoja (ne nastopa indiferentnost zanj), ta subjekt pogoja ovrednoti. To podvajanje lahko odpravimo tako, da se ob ovrednotenju komponente njena vrednost shrani kot vrednost privatnega atributa. Povprečni čas izvajanja logične odločitvene tabele se lahko na ta način bistveno zmanjša, predvsem v bolj statičnih okoljih, hkrati pa se poveča čas ob ažuriranju.

Posebej zanimiva se zdi možnost vzporednega izvajanja pravil logične odločitvene tabele, saj so sama objektna okolja izredno primerna za tako izvajanje [24]. Po drugi strani so tudi logične odločitvene tabele po svoji naravi vzporedne, saj je vrstni red pravil nepomemben[2]. Za večino drugih predstavitev odločitvenih pravil (logični diagram poteka, odločitvena drevesa) to ne velja.

2.3.2 Analiza možne uporabe rešitve

V nadaljevanju smo ugotavljali smiselnost praktične uporabe rešitve in težave, na katere bi utegnili naleteti pri njenem uvajanju v realno okolje. V ta namen smo analizirali možnost uporabe predlagane rešitve na realnem problemu izdelave informacijskega sistema referata za študijske zadeve Ekonomske fakultete v Ljubljani. Pri tem smo se naslonili na načrt [9], ki ga je že prej izdelala skupina študentov pod mentorstvom avtorja metode TAD, ki je bila pri izdelavi načrta uporabljena.

V prvi fazi smo skušali identificirati podprobleme znotraj celotnega načrta, ki bi bili zanimivi za realizacijo z odločitvenimi tabelami, predstavljenimi z objekti v objektni bazi podatkov. Pri identifikaciji podproblemov smo izhajali iz dejstva, da je uporaba predlagane rešitve smiselna predvsem pri kompleksnih odločitvenih problemih, to je takih, pri katerih je število subjektov pogojev in aktivnosti razmeroma veliko. Analiza načrta je pokazala, da sta primerni dve (približno enako kompleksni) nalogi:

- *vpis študenta*, kjer mora referent preveriti vse pogoje za vpis v višji letnik, ter

2 *Razen pri samem (zaporednem) izvajanju, ko je vrstni red pomemben pri minimizaciji časa, potrebnega za odgovor.*

- *prijava na izpit*, kjer mora referent preveriti vse pogoje za pristop k izpitu.

Za nadaljnjo obdelavo smo izbrali drugi problem. Natančnejšo analizo naloge *prijava na izpit* smo opravili pri referentkah v referatu za dodiplomski študij Ekonomske fakultete. Rezultat te analize je logična odločitvena tabela z enajstimi subjekti pogojev, štirinajstimi aktivnostmi in dvanajstimi (ne-elementarnimi) pravili, ki je prikazana na sliki 5.

Nato smo identificirali objekte, ki so povezani s tem odločitvenim problemom, in se lotili ugotavljanja problemov, na katere bi utegnili naleteti pri kodiranju obravnavanega odločitvenega problema.

Implementacija omenjene logične odločitvene tabele zahteva razmislek, saj smo hitro ugotovili, da oblika le enega subjekta pogoja ustreza klasični obliki subjekta pogoja, kjer z relacijskimi operatorji primerjamo dve numerični vrednosti (aritmetična izraza). Večina drugih subjektov pogojev zahteva preverjanje, ali v nekem razredu objektov obstaja objekt z določenimi vrednostmi atributov. Npr. subjekt pogoja "Ali je samo ena prijava vnaprej?" ustreza vprašanju "Ali v razredu objektov PRIJAVA obstaja objekt, ki je povezan s študentom (objektom razreda ŠTUDENT), ki se želi prijaviti, in z izpitnim rokom (objektom razreda ROK), za katerega se želi prijaviti?"

Realizacija takega subjekta pogoja je mogoča na dva načina:

- V definiciji razreda objektov POGOJ lahko dodamo definicijo operatorja *je_v* in implementiramo ustrezno metodo, ki ovrednoti ta pogoj.
- V sistemih, kjer tak operator ni definiran, lahko preverjanje realiziramo tako, da subjekte pogojev nadomestimo s klici logičnih odločitvenih tabel. Preverjanje prisotnosti objekta v razredu torej realiziramo z logično tabelo. V ta namen mora obstajati mehanizem posredovanja parametrov ene logične odločitvene tabele drugi.

Oglejmo si še aktivnosti. Večina aktivnosti je enostavnih (izpis ali zapis podatkov), le aktivnost *Sprejem prijave* je nekoliko zahtevnejša, saj skriva: tvorbo objekta razreda PRIJAVA, povezavo tega objekta z ustreznima objektoma razredov ŠTUDENT in ROK ter izpis ustreznega sporočila (npr. "Prijava sprejeta").

Ugotovitve analize možne uporabe rešitve v okviru obravnavanega problema lahko strnemo v nekaj točk.

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}
Datum prijave < datum roka – čas za prijavo ?	ne										da	da
Ali že obstaja prijava?		da									ne	ne
Ali je že opravil izpit?			da								ne	ne
Ali obstaja rok?				da							ne	ne
Ali ima študent predmet v predmetniku?					da						ne	ne
Ali so izpolnjeni predpogoji?						da					ne	ne
Ali je samo ena prijava vnaprej?							da				ne	ne
Ali je vpisana ocena za zadnjo prijavo?								da			ne	ne
Ali je št. prijav v študijskem letu preseženo?									da		ne	ne
Ali je skupno št. prijav preseženo?										da	ne	ne
Ali plača izpit?											da	ne
Izpis "Prepozna prijava"	X											
Izpis "Prijava že obstaja"		X										
Izpis "Izpit že opravljen"			X									
Izpis "Rok ne obstaja"				X								
Izpis "Predmeta ni v predmetniku"					X							
Izpis "Niso opravljeni predpogoji"						X						
Izpis "Prijava za ta rok že obstaja"							X					
Izpis "Ocena za prejšnji rok še ni vpisana"								X				
Izpis "Št. prijav v tem študijskem letu preseženo"									X			
Izpis "Skupno št. prijav preseženo"										X		
Izpis "Izpit je potrebno plačati"											X	
Izpis potrdila o plačilu											X	
Zabeleži podatke o neuspeli prijavi	X	X	X	X	X	X	X	X	X	X		
Sprejem prijave											X	X

Slika 5: Logična odločitvena tabela naloge prijava na izpit

- Pri implementaciji rešitve je potrebno nadaljnje razvojno delo na nekaterih delih rešitve. Tako je npr. potrebno implementirati možnost povezovanja logičnih odločitvenih tabel, različne operatorje, ki naj bi jih bilo mogoče uporabljati v subjektih pogojev, ter možnost uporabe klicev metod objektov v subjektih pogojev.
- Izkazalo se je, da bi bila rešitev za odločitveni problem preverjanja pogojev za prijavo na izpit izredno primerna, saj omogoča enostavno spreminjanje pogojev ob morebitnih (relativno pogostih) spremembah le-teh. To pomeni enostavnejše in s tem tudi cenejše vzdrževanje programske opreme.
- Hkrati bi bilo mogoče na enostaven način dodajati in spreminjati pogoje za pristop k izpitu, ki so specifični za posamezni predmet.

3 PREDNOSTI IN SLABOSTI PREDLAGANE REŠITVE

Predlagana rešitev ima nekaj pomembnih prednosti pri modeliranju, kodiranju in uporabi poslovnih pravil v informacijskih sistemih, med katerimi so najpomembnejše naslednje:

- Način dela s poslovnimi pravili je enak v vseh fazah razvoja informacijskega sistema, saj zunanji pogled, ki ga uporabljamo tudi pri kodiranju, ažuriranju in uporabi pravil, ustreza eni izmed tehnik za vizualizacijo poslovnih pravil.
- Na ta način predstavljena poslovna pravila zagotavljajo večjo fleksibilnost in robustnost sistema, saj so spremembe poslovnih pravil enostavne in ne vplivajo na ostale dele sistema.
- Ker so metode definirane pri razredih objektov, ki so predefinirani, ni potrebno programa ob tvorjenju ali spreminjanju logične odločitvene tabele prevajati. S tem je zagotovljena večja fleksibilnost poslovnih pravil.
- Predstavitev poslovnih pravil z objekti, ki predstavljajo komponente logičnih odločitvenih tabel poveča ponovno uporabljivost programske kode na dva načina:
  - Komponente lahko poljubno združujemo v pravila in tabele s tvorbo povezav med objekti. Velja pa seveda tudi obratno, saj lahko povezave tudi odstranimo.
  - Ker je med objekti mogoče tvoriti povezave tipa M:N, to pomeni, da lahko komponente na nižjem nivoju sodelujejo v več komponentah na višjem nivoju. Na primer, en subjekt aktivnosti lahko nastopa v več pravilih, eno pravilo pa v več logičnih odločitvenih tabelah. Na ta način je povečana ponovna uporabljivost znotraj uporabniške rešitve.
- Z razvojem ustreznega zunanjega pogleda je mogoče grafično programiranje poslovnih pravil. Pri tem gre za neposredno delo s poslovnimi pravili in ne za generiranje programske kode.
- Želene lastnosti poslovnih pravil se preverjajo avtomatično. Preverjanje mora biti opcijsko, saj ni vedno zaželeno, da poteka sprotno.

Poleg prednosti pa ima trinivojski pogled na poslovna pravila in notranja predstavitev le-teh z objekti tudi nekatere slabosti:

- Zunanji pogledi morajo biti predvideni vnaprej, kar pomeni, da si uporabnik ne more tvoriti nekega svojega zunanjega pogleda. Seveda mora sistem za upravljanje logičnih odločitvenih tabel omogočati, da si uporabniki čimbolj prilagodi vgrajene zunanje poglede svojim potrebam.
- Prav tako je potrebno vnaprej definirati operatorje (npr. za vektorske operande ali za preverjanje obstoja objekta z določeno lastnostjo v množici objektov), ki jih smemo uporabljati v subjektih pogojev.
- Časovna zahtevnost izvajanja tako predstavljenih poslovnih pravil se poveča.

4 SKLEP

V prispevku smo predstavili način predstavitve poslovnih pravil, ki temelji na trinivojskem pogledu na poslovna pravila, pri čemer so pravila na notranjem nivoju predstavljena z objekti v bazi podatkov, kar omogoča predvsem večjo fleksibilnost poslovnih pravil in enostavnejše delo tudi za uporabnika.

Z logičnimi odločitvenimi tabelami in posledično na predlagani način je mogoče predstaviti vsak logični diagram poteka. Vendar predlagani model ne rešuje vseh problemov predstavitev poslovnih pravil v informacijskih sistemih. Iz analize predlagane rešitve, prototipa in primera uporabe lahko sklepamo, da je uporaba smiselna predvsem v primerih, ko so prednosti bolj izrazite, to pa je predvsem v primeru kompleksnih poslovnih pravil, za katera sta značilna pogosto spreminjanje in potreba po preverjanju popolnosti in nedvoumnosti. Pri odločitvi za uporabo predlaganega načina predstavitve poslovnih pravil so torej odločilni faktorji predvsem:

- pogostost sprememb pravil,
- odzivni čas vzdrževalcev sistema,
- potrebe po avtomatičnem preverjanju lastnosti pravil,
- kompleksnost problema,
- možnosti za ponovno uporabo komponent poslovnih pravil ter
- pomembnost hitrosti izvajanja.

Za uporabo v realnih okoljih je potrebno nadaljnje raziskovalno in razvojno delo predvsem na naslednjih področjih:

- razvoj zunanjih pogledov,
- povezava z univerzalnimi (objektno relacijskimi) bazami podatkov,
- optimizacija izvajanja pravil,
- možnost dela z večdimenzionalnimi tabelami,
- razvoj za porazdeljena okolja ...

Sklenemo lahko, da bi bilo v mnogih kompleksnih odločitvenih problemih, kjer se odločitvena pravila pogosto spreminjajo, smiselno in smotrno uporabiti predlagano rešitev, kar bi omogočilo večjo fleksibilnost programskih rešitev in hkrati pocenilo vzdrževanje le-teh.

5 LITERATURA IN VIRI

[1] BANCILHON, François, DELOBEL, Claude, KANELLAKIS, Paris: Building an Object-Oriented Database System: The Story of O2, Morgan Kaufmann Publishers, San Mateo, California, 1992, ISBN 1-55860-169-4

[2] BAUM, David: The Right TOOLS for Coding BUSINESS Rules, Datamation, 1995 št. 4 (str. 36-38)

[3] CRAGUN, Brian J., STEUDEL, Harold J.:A decision-table-based processor for checking completness and consistency in rule-based expert systems, International Journal of Man-Machine Studies, 1987 št. 5 (str. 633-648), ISSN 0020-7373.

[4] DAMIJ Talib: Načrt informacijskega sistema za referat Ekonomske fakultete za dodiplomski študij izdelan po metodi TAD (Tabular Application Development). Ljubljana : Ekonomska fakulteta, 1997. 11 str.

[5] ELMASRI, Ramez, NAVATHE, Shamkant B.: Fundamentals of Database Systems, Addison-Wesley, Redwood City, 1989, ISBN 0-201-5309-2

[6] GRADIŠAR Miro, RESINOVIČ Gortan: Informatika v poslovnem okolju. Ljubljana: Ekonomska fakulteta v Ljubljani, 1996. 479 str.

[7] HALVERSON Richard Jr.: Programming the Hawaii Parallel Computer. Proceedings of the 2nd ACM International Workshop on FPGAs, Berkley, CA, 1994. 7 str.

[8] HICKS, Sam A., WASSERMAN William E.: Computers in Taxation, The Tax Adviser, 1991 št. 7 (str. 471-473)

[9] HOFFER J. A., GEORGE J. F., VALACHIC J. S.: *Modern System Analysis and Design*. Reading : The Benjamin/Cummings Publishing Company, 1996.

[10] JENKINS A. Milton: Evolution of Information Technology and the Problems and Opportunities it Presents to Business. 50th Anniversary of the Faculty of Economics, Ljubljana, Slovenia, International Conference, Summary of Abstracts, Ljubljana : Faculty of Economics, 1996, str. 572-575.

[11] LEW, Art: On the Emulation of Flowcharts by Decision Tables, Communications of the ACM, 1982 št. 12 (str. 895-905).

[12] LONDON, Keith R.: Decision Tables, Auerbach Publishers Inc., 1972, ISBN 0-87769-122-3.

[13] MAES, R., Van DIJK J. E. M.: On the Role of Ambiguity and Incompleteness in the Design of Decision Tables and Rule-Based Systems, The Computer Journal, 1988 št. 6 (str. 481-489)

[14] MAES, R.: On the representation of program structures by decision tables: a critical assessment, The Computer Journal, 1978 št. 4 (str. 290-295), ISSN 0010-4620.

[15] MARTIN, James.: Principles of Object-Oriented Analysis and Design, Prentice-Hall, Engelwood Cliffs, 1993, ISBN 0-13-720871-5.

[16] McDANIEL, Herman, ed.: Applications of Decision Tables, Brandon/System Press, Inc., Princeton, 1970

[17] TSICHRITZIS Dionysios C., LOCHOVSKY Frederick H.: Data Models. Englewood Cliffs : Prentice-Hall, Inc., 1982. 343 str.

[18] VANTHIENEN Jan, DRIES E.: Illustration of a Decision Table Tool for Specifying and Implementing Knowledge Based System. International Journal of Artificial Intelligence Tools, 3 (1994), 2, str. 267-288.

[19] VANTHIENEN Jan, WETS Geert: Integration of the Decision Table Formalism With a Relational Database Environment. Information Systems, 20 (1995), 7, str. 595-616.

[20] VESSEY Iris, WEBER Ron: Structured Tools and Conditional Logic: An Empirical Investigation. Communications of the ACM, 29 (1986), 1, str. 48-57.

[21] WEARS, Robert L. et al.: Using Decision Tables to Verify the Logical Consistency and Completness of Clinical Guidelines: Fevers Without Sources in Children Under Age Three Years, http://solaris.ckm.ucsf.edu/Originals/SAEMabs/z.abs103.html, University of Florida, 1996

◆

Jurij Jaklič je diplomiral iz uporabne matematike na Oddelku za matematiko in mehaniko Fakultete za naravoslovje in tehnologijo v Ljubljani. Magistriral je na Oddelku za računalništvo Univerze v Houstonu, ZDA. Doktorsko disertacijo je obranil leta 1997 na Ekonomski fakulteti v Ljubljani, kjer je tudi zaposlen kot asistent pri predmetih informatike. Ukvarja se predvsem s problemi s področja baz podatkov.

◆

Janez Grad je doktoriral iz matematičnih znanosti na Vseučilišču v Zagrebu. Od l. 1973 sodeluje kot učitelj za informatiko na Ekonomski fakulteti v Ljubljani, od leta 1985 dalje kot redni profesor. Ukvarjal se je s programiranjem na računalniku in z reševanjem problema lastnih vrednosti in vektorjev matrik, v zadnjih letih pa se ukvarja z reševanjem problemov s področja operacijskega raziskovanja in baz podatkov.