KLAIPĖDOS UNIVERSITETAS
GAMTOS IR MATEMATIKOS MOKSLŲ FAKULTETAS
INFORMATIKOS KATEDRA

ROBERTAS STRIGŪNAS
Informatikos studijų studentas

VERSLO TAISYKLIŲ RINKINIO DARNOS UŽTIKRINIMAS LOGINIO IŠVEDIMO MAŠINA

Baigiamasis magistro darbas

Mokslinis darbo vadovas: prof. dr. Olegas Vasilecas
Konsultantas: lekt. Aidas Šmaižys

Klaipėda, 2007

Klaipėdos universitetas
Gamtos ir matematikos mokslų fakultetas
Informatikos katedra

Baigiamasis magistro darbas
Verslo taisyklių rinkinio darnos užtikrinimas loginio išvedimo mašina
Robertas Strigūnas

Anotacija

Šiandienos įmonės susiduria su labai greitai besikeičiančia aplinka ir negali laikytis numatyto ilgalaikio veiklos modelio. Įmonės veikla ir jos veiklos modelis turi būti dinamiški. Viena didžiausių problemų verslo taisyklių panaudojimui yra jų kiekis ir sunkiai nuspėjama tarpusavio sąveika. Taisyklėms veikiant kartu atsiranda įvairūs konfliktai. Taisyklių konfliktų aptikimui ir logiškumui bei nuoseklumui patikrinti galima naudoti logika pagrįstus mechanizmus tokius kaip išvedimas. Darbe atlikta dalykinės srities literatūros apžvalga, suformuluotas metodas, leidžiantis XML atvaizduotas verslo taisykles transformuoti į predikatus bei komponuoti į verslo taisyklių rinkinius siekiant tuos rinkinius panaudoti duomenų analizei programų sistemoje. Verslo taisyklių rinkinys turi būti išsamus ir neprieštaraujantis. Tam naudojama išvedimo mašina.

Reikšminiai žodžiai: verslo taisyklės, išvedimo mašina, verslo taisyklių transformavimas, predikatai, darna, XML.

Klaipėda University
Faculty of Natural Sciences and Mathematics
Computer Science Department

MSc thesis
Inference engine driven business rule set consistency check
Robertas Strigūnas

Annotation

One of the greatest problems using business rules are their quantity and difficulty in foreseeing of their interplay. Conflicts become apparent when rules are used in conjunction. A logic based derivation mechanism can be used for detection of rule incongruence and to analyze their logical and chronological sequence. This paper presents a review of related works and method for transformation of the business rules represented in XML into predicate set for use in inference engine and business rules driven rule set consistency check and use of such a complete rule set in data analysis system.

Keywords: Business rules, business rules representation, transformation of business rules, inference engine, derivation, consistency, XML.

TURINYS

4. Naujai gauti faktai RuleML formate.
5. Prieštaraujančių taisyklių rinkinys RuleML.
6. Kompaktinis diskas.

PAVEIKSLŲ RODYKLĖ

ĮVADAS

Temos naujumas ir aktualumas

Verslo taisyklių panaudojimo informacinių sistemų kūrimui tema šiuo metu yra labai populiari ir plačiai nagrinėjama moksliniuose straipsniuose. Šiandienos sparčiai besivystantis verslas reikalauja lanksčių verslo valdymo sistemų, kurios sugebėtų vystytis sparta, atitinkančia veiklos pokyčius. Organizacija, kurdama verslo valdymo sistemą, siekia įgyti konkurencinį pranašumą, sumažinti veiklos kaštus, pagerinti procesų valdymą. Šiuolaikinės įmonės veiklos esmine komponente yra informacinė technologija, vartotojo požiūriu vadinama kompiuterizuota informacine sistema [3].

Informacinių sistemų kūrimo metodams iki šiol būdingas „išorinis požiūris“ į organizacijos veiklos procesą – informacinės sistemos kūrimas pradedamas nuo vartotojo poreikių ar analitiko pastebėjimų, formuojama informacinės sistemos specifikacija taip, „kaip reikia” vartotojui ir projektuotojui. Informacinių sistemų inžinerijai būtinas „vidinis požiūris“ į veiklos procesus, grindžiamas priežastiniais veiklos elementų ryšiais – vidine veiklos (vadybos ir produkto gamybos) procesų logika. Taip pat reikia formalizuoti veiklos modelius, skirtus kompiuterizuotam informacinės sistemos kūrimui, kurie aprašo priežastinius (esminius, dėl vidinių veiklos savybių egzistuojančius) veiklos elementų ryšius. Šioms problemoms spręsti buvo sukurtas iš esmės naujas požiūris į sistemos kūrimą – verslo taisyklių požiūris [25].

Efektyvus verslo taisyklių panaudojimas informacinių sistemų kūrimui leidžia ne tik paspartinti patį informacinių sistemų kūrimo procesą, bet ir išlaikyti taisyklių verslo sistemoje ryšį su informacinės sistemos modeliu kūrimo metu, taisyklių transformavimo procesų pasėkoje, atsiradusiais naujais tiek informacinės sistemos, tiek programų sistemos objektais ir naujomis taisyklėmis. Verslo taisyklių požiūris teigia, kad veiklos vadovas ar atstovas, priimantis strateginius sprendimu ir žinantis bendrą organizacijos veiklos koncepciją, turi pateikti ir turėti galimybę keisti išskirtas veiklos taisykles sukurtoje sistemoje [3, 27].

Kad verslo taisykles būtų lengva keisti, reikia, kad jos būtų logiškai atskirtos nuo duomenų bei programos struktūros. Dažniausiai jos įsimenamos taisyklių repozitoriuje, kuris yra priemonė, leidžianti peržiūrėti, pakeisti ar tvarkyti tas taisykles. Tačiau viena didžiausių problemų verslo taisyklių panaudojimui yra jų kiekis ir sunkiai nuspėjama tarpusavio sąveika. Taisyklėms veikiant kartu atsiranda įvairūs konfliktai, kai viena iš dviejų vykdomų taisyklių priima užduotį, o kita atmeta ir t.t.

Tyrimo objektas

Verslo taisyklių sistema, kurią po taisyklių formalizacijos ir transformacijos procedūrų galima panaudoti informacijos analizės bei sprendimų priėmimo automatizavimui.

Problematika

Dėl didelio verslo taisyklių skaičiaus atsirandančių sunkiai nuspėjamų tarpusavio sąveikų tyrimas ir jų konfliktų sprendimas siekiant gauti darnią taisyklių sistemą ir ją panaudoti informacijos analizės bei sprendimų priėmimo automatizavimui.

Darbo tikslas ir uždaviniai

Darbo tikslas - išplėtoti metodą, leidžiantį užtikrinti XML atvaizduotų verslo taisyklių rinkinio darną, siekiant tuos rinkinius panaudoti duomenų analizei programų sistemoje. Tikslui įgyvendinti didžiausias dėmesys skiriamas išvedimo mašinų panaudojimui.

Darbo uždaviniai:

- Išanalizuoti šiuo metu naudojamus verslo taisyklių klasifikavimo ir vaizdavimo metodus.
- Išanalizuoti verslo taisyklių atvaizdavimą XML kalba ir šių taisyklių galimas transformacijas į programų sistemos taisykles.
- Išanalizuoti verslo taisyklių transformacija į predikatų logikos sakinius.
- Išnagrinėti verslo taisyklių valdymo metodus.
- Verslo taisyklių tarpusavio sąveikos realizavimas išvedimo mašinose.
- Suformuluoti ir aprašyti metodą, kuris leistų XML atvaizduotas verslo taisykles transformuoti į predikatus bei komponuoti į prasminiais ryšiais susietus verslo taisyklių rinkinius
- Realizuoti pasiūlytą metodą informacinių sistemų prototipe ir atlikti aprašyto metodo eksperimentinius bandymus.

Tyrimo metodika

Tyrimas atliktas naudojant literatūros apžvalgos, lyginamosios analizės, sisteminės analizės ir eksperimentinės analizės metodus.

Mokslinė darbo vertė

Darbe pasiūlytas metodas, leidžiantis užtikrinti verslo taisyklių rinkinio darną. Metodas leidžia XML atvaizduotas verslo taisykles transformuoti į predikatus bei komponuoti į išsamius ir neprieštaraujančius verslo taisyklių rinkinius, panaudojant loginio išvedimo mašiną.

Viešas darbo rezultatų pristatymas

- Publikuotas straipsnis KTU mokslinėje konferencijoje "Informacinės technologijos'2007**”** pavadinimu „Verslo taisyklių rinkinio užtikrinimas loginio išvedimo mašina“.
- Svarbiausi darbo rezultatai pateikti pranešime, skaitytame 2007 m. balandžio 13 d., Vilniaus Gedimino technikos universitete vykusioje 10-ojoje jaunųjų mokslininkų konferencijoje “Lietuva be mokslo – Lietuva be ateities”. Pagal šį pranešimą parengtas straipsnis.

Darbo rezultatai

- Išanalizuotas verslo taisyklių panaudojimas informacinėse sistemose.
- Pasiūlytas metodas skirtas verslo taisyklių transformacijai į predikatus.
- Pasiūlytas metodas verslo taisyklių darnos užtikrinimui išvedimo mašina.
- Atlikta pasiūlyto metodo eksperimentinė analizė.
- Sukurtas programų sistemų prototipas, skirtas pasiūlytam metodui realizuoti. Eksperimentas parodo XML užrašyto taisyklių rinkinio transformaciją į predikatus bei panaudojimui išvedimo mašinoje.

1. VERSLO TAISYKLĖS INFORMACINĖSE SISTEMOSE

Projektuojant verslo sistemas, pagrindine to proceso dalimi tampa verslo procesų modeliavimas. Proceso samprata tampa vis labiau esminiu principu organizuojant verslą, nei funkcinė hierarchija. Taigi verslo procesų modeliavimas tampa dar labiau populiaresnis. Informacinių technologijų ir verslo inžinerijos sričių ekspertai yra padarę vieningą išvadą, kad sėkmingos sistemos pradėjo nuo verslo procesų supratimo ir reikšmingumo organizacijoje: verslo procesų modelio. Inžinerijos procese išaiškėjus naujoms aplinkybėms yra keičiami visi modeliai, bet kuo tolimesniame etape pastebėti neatitikimai, tuo sudėtingiau ir brangiau juos atlikti, nes reikia sugrįžti atgal ir įvertinti visus pasikeitimus nuo pat pradžių, t.y. nuo verslo modelio [12].

Šioms problemoms spręsti buvo sukurtas iš esmės naujas požiūris į sistemos kūrimą – verslo taisyklių požiūris. Efektyvus verslo taisyklių panaudojimas informacinių sistemų kūrimui leidžia ne tik paspartinti patį informacinių sistemų kūrimo procesą, bet ir išlaikyti taisyklių verslo sistemoje ryšį su informacinės sistemos modeliu kūrimo metu, taisyklių transformavimo procesų pasėkoje, atsiradusiais naujais tiek informacinės sistemos, tiek programų sistemos objektais ir naujomis taisyklėmis. Verslo taisyklių požiūris teigia, kad veiklos vadovas ar atstovas, priimantis strateginius sprendimu ir žinantis bendrą organizacijos veiklos koncepciją, turi pateikti ir turėti galimybę keisti išskirtas veiklos taisykles sukurtoje sistemoje [3, 5].

1.1. Verslo taisyklių samprata ir paskirtis

Verslo sistemoje verslo taisyklės apibrėžiamos kaip verslo politika, praktika, aiškumas, kur yra gerai suprantamos ir vykdomos kaip organizacijos vertybių rinkinys. Verslo taisyklės suprantamos kaip pagrindinė sąlyga verslui plėtoti. Verslo taisyklė negali būti suskaidyta arba detalizuota į smulkesnes taisykles.

Pagal Business Rules Group [3], verslo taisyklė:

- tai nuostata, kuri apibrėžia arba apriboja tam tikrą verslo požiūrį;
- apibrėžia verslo struktūrą, kontroliuoja arba įtakoja verslo elgseną.

Verslo taisyklės yra pagrįstos verslo taisyklių formuluotėmis (angl. *statements*), kurios toliau grindžiamos verslo politika. Verslo taisyklės formuluotė gauta iš verslo atstovo yra nevienareikšmiška, dviprasmiška. Konkrečioje situacijoje kiekviena verslo taisyklės formuluotė galima faktiškai suskaidyti į diskrečias taisykles. Taisyklės suskaidytos į elementarias formas vadinamos atominėmis. Verslo taisyklė neturi valdymo operatorių, kurie dažniausiai sutinkami programose kaip pranešimai, duomenų bazės atnaujinimai (angl.: *updates*), sekos. Atominė verslo taisyklė, užrašyta deklaratyvia forma, naudojant natūralią kalbą, kad verslo atstovai galėtų lengvai

suprasti, nėra dviprasmiška. Verslo įmonės apima tūkstančius verslo taisyklių kombinacijų, kurios veikia operaciniame lygmenyje. Verslo taisyklės apibrėžia ir kontroliuoja produkto ir paslaugų gyvavimo ciklą [27].

Mums svarbesnis požiūris iš informacinių sistemų perspektyvos, kur verslo taisyklės suprantamos kaip nesudėtingas teiginys, tikrinantis duomenų teisingumą, atliekantis būtinus paskaičiavimus ar valdantis duomenų pavaizdavimą [2, 3, 4]. Programuotojas sukuria šių taisyklių rinkinį, aprašantį įstaigos veiklą. Toks principas labai priimtinas, kai greitai keičiasi veiklos sąlygos. Juk užtenka pakeisti tik taisykles ir nereikia liesti ir keisti pačios duomenų bazių sistemos (skirtingai nuo procedūrinio programavimo) [32].

Atitinkamai, verslo taisyklė išreiškia specifinius apribojimus duomenų kūrimui, atnaujinimui, pašalinimui informacinėje sistemoje. Taisyklės nėra nei procesas, nei procedūra, todėl neturėtų būti nė vieno iš jų sudėtinė dalis. Tiesiogiai susijusiose veiklos srityse gali būti panaudoti panašūs verslo taisyklių rinkiniai. Taisyklės gali būti susistemintos pagal ECA techniką aktyviose duomenų bazėse, kaip ryšys tarp įvykių, būsenų ir veiksmų.

Apibendrinant galima konstatuoti, kad verslo sistema veikia pagal verslo taisykles. Žvelgiant globaliai, verslo taisyklės yra fundamentali verslo politika ir to verslo apribojimai. Kitaip tariant, tai yra teiginiai, kontroliuojantys arba veikiantys verslo aplinką. Taigi verslo taisyklės išplaukia iš verslo strategijos. Taip tiesiogiai užtikrinama verslo tikslų ir uždavinių realizavimas, įgyvendinimas.

1.2. Verslo taisyklių klasifikavimas

Kai kalbama apie verslo taisyklę, dažniausia turime omenyje taisyklių rinkinį su tam tikra jų hierarchija ir prioritetais. Dėl jų įvairumo nėra ir negali vieno visiems atvejams tinkamo taisyklių šablono ar jų vaizdavimo formos. Todėl siekiant maksimaliai supaprastinti darbą su sudėtingomis taisyklių sistemomis jos yra skirstomos remiantis įvairiais kriterijais. Taisykles galima suskirstyti į rūšis, tipus ir klases. Kiekvienai klasei galima nustatyti atitinkamą šabloną, kurio pagalba būtų vaizduojamos tai klasei priklausančios taisyklės. Taisyklių klasifikavimui gali būti pasitelkiami įvairiausi kriterijai: struktūros tipas, semantinės savybės, paskirtis, šaltinis, realizacijos tipas, ir kt. Šiai dienai yra sukurta gausybė įvairiausių verslo taisyklių klasifikavimo taksonomijų.

Šiame skyriuje pateikiamos detaliausiai išnagrinėtos ir labiausiai paplitusios verslo taisyklių klasifikavimo schemos.

Verslo taisyklių grupė (angl.: *Business Rules Group*) [5] verslo taisykles skirsto į:

- Išvestis - sakinys, išreiškiantis žinias, kurios savo ruožtu išvedamos iš kitų veiklos žinių. Tai matematinis skaičiavimas, loginė išvada.

- Struktūrinius teiginius - apibrėžta mintis arba fakto konstatavimas, išreiškiantis tam tikrą organizacijos sandaros aspektą. Gali būti sąvoka, faktas.
- Veiksmo teiginius - yra apribojimą ar sąlygą reiškiantis sakinys, kuris nustato arba valdo organizacijos veiksmus. Gali būti: sąlyga, apribojimas, leidimas, įgaliotojas, laikmatis, vykdymo, veiksmą valdantis, veiksmą veikiantis.

Kitas verslo taisyklių klasifikavimas pateiktas Ronald'o Ross'o (2001). Kur verslo taisyklės skirstomos į faktus, terminus, apribojimus, kilmę, išvadas, nustatytą laiką (angl.: *timing*), seką, euristiką (angl.: *heuristics*). Pagrindinė šio klasifikavimo idėja – suskirstyti taisykles į standartinius, aukšto lygio taisyklių tipus, kurie nurodo, kokį patikrinimą turi atlikti taisyklės [27].

Barbara von Halle knygoje *„Business Rules Applied: Building Better Systems Using the Business Rules Approach“* pateikia verslo taisyklių klasifikavimo schemą, kur taisyklės skirstomos į terminus, faktus, veiksmo teiginius (angl. rule) [3].

- Terminas – tai žodis ar frazė , kuris turi specifinę reikšmę veiklai (veiklos sąvokos, bendros sąvokos).
- Faktai parodo asociaciją tarp dviejų ar daugiau terminų, kitaip tariant, išreiškia ryšį. Faktas įtraukia du ar daugiau terminų, o terminas gali būti viename ar daugiau faktų.
- Veiksmo teiginiai – tai sakiniai, kurie turi ryšį su kokiu nors dinaminiu veiklos aspektu. Jis apibrėžia rezultato, kurį duoda veiksmas, apribojimus. Apribojimai aprašomi neprocedūriškai, kitų nedalomų verslo taisyklių terminais.

USoft kompanija siūlo verslo taisykles klasifikuoti į:

- Apribojimo taisyklės: verslo apribojimai saugomai informacijai.
- Elgsenos taisyklės: sistemos elgsena tam tikrose numatytose situacijose, sistemos automatiniai veiksmai nenumatytose situacijose.
- Išvesties taisyklės: sistemos logikos valdymas, sistemos skaičiavimų valdymas.
- Atvaizdavimo taisyklės - kaip sistema sąveikauja su vartotoju, veiksmų ir užduočių organizavimo tvarka.
- Nurodymų taisyklės: kaip vartotojas turi valdyti sistemą tam tikrose situacijose.

Kituose šaltiniuose verslo taisyklės skirstomos į dvi kategorijas: darnos ir veikos. Darnos taisyklės nurodo duomenų darnos reikalavimus, kurie gali būti užtikrinti pasyviais ir aktyviais darnos ribojimais. Pasyvūs ir aktyvūs darnos ribojimai skiriasi veikimo strategija: aktyvūs ribojimai užtikrina duomenų darną vykdant duomenų keitimo operacijas, o pasyvūs ribojimai uždraudžia vykdyti operacijas, kurios pažeistų duomenų darną. Verslo taisyklės savo ruožtu nusako veiksmus ar operacijas, kurios turi būti įvykdytos [22].

Apibendrinant galima teigti, kad verslo taisyklių klasifikavimo modelio pasirinkimas pirmiausia priklauso nuo tikslo, kuriam tarnaus. Racionaliausia naudoti skirstymą į klases pagal

struktūrą, į rūšis pagal semantines savybes ir į tipus pagal kitus labiausiai konkrečiai situacijai tinkančius kriterijus.

1.3. Verslo taisyklių vaizdavimas

Verslo taisyklės turi būti suprantamos ir informacinės sistemos kūrėjams, ir verslo atstovams. Pats paprasčiausias būdas užrašyti taisyklę, be abejo, yra paprastas tekstas natūralia kalba. Tačiau siekiant ją panaudoti kuriant informacines sistemas, verslo taisyklę reikia užrašyti formalia forma. Iš čia seka, kad verslo taisyklėms reikia skirtingo formalumo lygio specifikacijų, kurios būtu suprantamos visiems su verslo taisyklėmis dirbantiems žmonės.

Formali taisyklė (angl.: *formal rule*), tai taisyklė išdėstyta pagal specifinę (apibrėžtą) formą ar sintaksę. Dažniausiai visais atvejais, formali taisyklė reikalauja formalaus žodyno, kuris apibrėžia pagrindinius terminus, naudojamus taisyklėse. Formalios taisyklės dažniausiai yra deklaratyvios ir atominės. Dirbant prie didesnio produkto ir verslo taisyklių skaičiui didėjant, kyla poreikis veiklos taisykles skirstyti į kategorijas, verifikuoti jų atitikimą veiklos atstovo išreikštai formai [1].

Ko gero pats seniausias formalus taisyklių vaizdavimo būdas yra predikatų logikos sakiniai. Toks taisyklių užrašymo būdas nėra labai informatyvus, ypač sudėtinga interpretuoti tokiu būdu užrašytas taisykles, kai sakinių yra daug. Tačiau tai kol kas vienintelis būdas kaip taisykles panaudoti loginių išvadų darymui, naudojant automatizuotus procesus [32].

Vieni pirmųjų žinių, o tuo pačiu ir taisyklių vaizdavimo kalbos kūrimu susidomėjo „The Knowledge Sharing Effort“ konsorciumas. Šis konsorciumas buvo specialiai įkurtas tam, kad ištirtų galimybes žinias vaizduoti taip, kad jos galėtų būti pakartotinai panaudotos. Šios iniciatyvos didžiausias indėlis buvo sukurtos KQML, Ontolingua ir KIF.

KIF (angl.: *knowledge interchange format*) formatu galima vaizduoti paprastus duomenis, sudėtingą informaciją ir logiką. KIF yra labai turtinga kalba, tačiau būtent dėl šios priežasties ją ir yra labai sudėtinga naudoti, kuriant informacines sistemas. Todėl vėliau atsirado efektyvesnės kalbos, kurios nors ir yra labiau apribotos, bet gali būti panaudojamos žymiai efektyviau. Be to, KIF buvo sukurta prieš atsirandant XML, todėl nėra ir KIF aprašymo skirto XML. Tačiau “kuriant KIF buvo padarytas didžiulis įdirbis šioje srityje ir ji buvo panaudota kaip pagrindas kai kurioms dažniau sutinkamoms taisyklių vaizdavimo ir apsikeitimo formatų kalboms (RuleML) [33].

Grafinį taisyklių vaizdavimą, panaudojus taisyklių vaizdavimo šablonus, galima užrašyti natūralia kalba. Tokios taisyklės galėtų būti vaizduojamos UML diagramomis. Ypač tam tinka būsenų perėjimo diagramos. Grafinio modeliavimo privalumai – standartiniai žymėjimai, vieninga terminologija, intuityvus supratimas. Pagrindinės verslo procesų modeliavimo kalbos: UML 2.0,

BPMN (angl.: *Business process modelling notation*) bei BPEL4WS (angl.: *Business Process Execution Language for Web Services*) [35].

Šiuo metu yra sukurta daug verslo taisyklių ir žinių vaizdavimo kalbų XML pagrindu. Kiekvieną tokią kalbą aprašo specialus DTD dokumentas, kuriame išvardintos visos galimos struktūros jų atributai ir apribojimai. Populiariausios iš jų yra RuleML, BRML ir SRML, kurios panaudotos eilėje komercinių produktų skirtų verslo taisyklėmis pagrįstų informacinių sistemų kūrimui. BRML buvo sukurta IBM korporacijos, tačiau nors ji taip pat remiasi pirmos eilės predikatų logika ji dar yra labai ankstyvoje kūrybos stadijoje ir dar neužregistruota W3C.

RuleML iniciatyva siūlo hierarchinę taisyklių taksonomiją, kurios viršuje yra būsenos pakeitimo taisyklės, kurias nusako sandaros apribojimai ir būsenos išsaugojimo sąlygos, kai tuo tarpu faktai yra traktuojami kaip tam tikra būsenos išsaugojimo taisyklių rūšis. RuleML taip pat tinka ir Horno logikos sakinių vaizdavimui.

Toliau pateiktas trumpas RuleML notacijos pritaikymas taisyklei.

Taisyklė: *„Jei pirkėjas yra nuolatinis, tada pirkėjui suteikiama nuolaida prekėms„*

Taisyklė RuleML formate:

```
<imp>
   <_head>
      <atom>
         <_opr> <rel>nuolaida</rel> </opr>
         <var>pirkėjas</var>
      </atom>
   </_head>
   <body>
      <atom>
         <_opr> <rel>nuolatinis</rel></_opr>
            <var>pirkėjas</var>
      </atom>
   </_body>
</imp>
```

Apibendrinant nagrinėta literatūra, galima išskirti pagrindinius verslo taisyklių vaizdavimo metodus: taisyklių vaizdavimas “žmonių kalba”, logikos konstrukcijomis, įvairiais tekstiniais formatais, žymėmis (BRML, RuleML, XML), grafinių vaizdavimu (UML), sprendimų lentelėmis.

1.5. Verslo taisyklių repozitorius

Dažnai kyla klausimas - kur padėti verslo taisykles, kad jas būtų galima bet kada nesunkiai surasti, o susietą programinį kodą bet kada panaudoti verslo procesams vykdyti ar aprašyti?

Paprastai VT yra organizuojamos į verslo taisyklių saugyklą, dar vadinama repozitoriumi. Verslo taisyklių repozitoriaus tikslas yra patenkinti visų suinteresuotų pusių verslo taisyklių informacinius poreikius (juos įtraukiant tiesiogiai arba paveikiant netiesiogiai) pradiniame sistemų kūrimo procese ir nuolatos tobulinant bei keičiant sistemą per visą gyvavimo ciklą [15]. Tokia saugykla turi leisti:

- aprašyti (vaizduoti) verslo taisykles panaudojant šablonus;
- kurti šablonus taisyklių vaizdavimui, apibrėžiant taisyklių struktūrą;
- tikrinti (testuoti) pavaizduotas taisykles;
- susieti pavaizduotą taisyklę su joje panaudotais kitų modelių objektais;
- pateikti taisyklę verslo vartotojui suprantama kalba (arba grafiniu pavidalu).

Dauguma sistemų naudoja skirtingų formatų duomenis, todėl atsiranda įvairių problemų bei sunkumų perduodant duomenis iš vienos sistemos į kitą. Šios problemos sprendimas yra naudoti tokį duomenų struktūros aprašymą, kurį suprastų visos sistemos. Toks aprašymas turėtų būti universalus. Šiuo metu labiausiai išvystytas ir plačiai naudojamas standartas duomenų aprašymui yra XML (angl. *eXtensible Markup Language*). XML yra standartinis formatas, kurį įvairios programos supranta vienodai, todėl duomenys gali būti transformuoti į XML ir laisvai perduoti kitai sistemai ar programai [7].

Pastebėta, kad vis dažniau naudojamas XML, kaip standartas verslo taisyklių saugojimui ir apsikeitimui. Bandoma naudoti tas pačias kalbas taisyklių užrašymui [6]. Verslo taisyklių saugyklos reikalavimų sukūrimas atskira problema, kuri darbe plačiau nenagrinėjama.

Kaip teigia Paul Ford, XML duomenų bazės turi šiuos privalumus prieš reliacines ar objektines duomenų bazes:

- XML duomenys į duomenų bazę yra pakraunami tiesiogiai, jais nereikia niekaip manipuliuoti ar ištraukti iš dokumentu prieš saugojant.
- Užklausos grąžina XML dokumentus arba jų fragmentus, todėl išlaikoma hierarchinė XML dokumento struktūra.

2. VERSLO TAISYKLIŲ RINKINIO KONFLIKTŲ SPRENDIMAS

Įgyvendinus verslo taisyklių sistemą, dažniausiai susiduriama su problemomis, susiejusiomis su logika ir informatyvumu. Verslo taisyklių sistema gali neturėti privalomų duomenų arba turėti klaidingus duomenis, ar klaidingus faktus. Taip pat sistema gali neturėti reikalingų taisyklių arba jos bus klaidingos, kurios duos nepageidaujamą išvadą. Viena didžiausių problemų verslo taisyklių panaudojimui yra jų kiekis ir sunkiai nuspėjama tarpusavio sąveika. Taisyklėms veikiant kartu atsiranda įvairūs konfliktai.

Šiame skyriuje apžvelgsime verslo taisyklių valdymo sistemas, problemas, susijusias su taisyklių panaudojimu bei esamus šių problemų sprendimo būdus.

2.1. Verslo taisyklių valdymo sistemos

Verslo taisyklių valdymo sistemos naudojamos tam, kad taisyklių kūrimo, modifikavimo bei vykdymo procesas būtu greitesnis ir efektyvesnis. Tokios sistemos naudoja sudėtingus algoritmus kompiuterinių resursų optimizavimui vykdant verslo taisykles. Integruojant verslo taisyklių valdymo sistemą į verslo sistemą kiekvieną taisyklė užrašoma kaip nepriklausomas teiginys, aprašantis kažkokį verslo aspektą ir verslo taisyklės tokiu atveju dirba kaip autonominis vienetas atskirtas nuo taikomosios programos logikos [25].

Verslo taisyklių valdymo sistemas rekomenduojama naudoti kai:

- programa apima sprendimų priėmimą;
- taisyklės yra sudėtingos;
- taisyklės yra dažnai keičiamos;
- taisyklės turi būti naudojamos keliose programose ir/arba organizacijose.

Verslo taisyklių valdymo sistemas nerekomenduojama naudoti kai:

- taisyklės yra statines ir išskaičiuojamos;
- taisyklės yra paprastos, vienodos, kartojamos;
- greitis ir našumas yra svarbesnis už lankstumą ir palaikymo kaštus.

Bendru požiūriu, visos programinės įrangos naudoja taisykles duomenų valdymui ir modifikavimui. Taisyklės visada yra apibrėžtos, kurios kontroliuoja automatizuotus procesus.

Taigi verslo taisyklės gali būti panaudotos:

- taisyklės taikomosiose programose;
- taisyklės duomenų bazėse ir valdomos DBVS;
- taisyklės „workflow“ sistemose;
- taisyklės verslo taisyklių valdymo įrankiuose (angl.: *Business Rule Management tool*).

Prieš analizuodami verslo taisyklių valdymo sistemas pabandysime apibrėžti kokius reikalavimus tūri tenkinti, kokis funkcijos tūri atlikinėti tokios sistemos.

- verslo taisyklių repozitorijus;
- ryšiai su kitais objektais;
- užtikrinta taisyklių rinkinio darna.

Egzistuoja labai daug įvairiausiu verslo taisyklių valdymo sistemų. Jos atlieka skirtingas funkcijas, daro verslo sprendimus įvairiausiuose lygmenyse, jų paskirtis yra skirtinga. Bet verslo taisyklių valdymo sistemas galima klasifikuoti pagal tai, kokių tipų taisykles jos gali valdyti [18]:

- Paprastos verslo taisyklės (angl.: *simple business rules*) – taisyklės, kurios gali būti išreikštos naudojant paprastą žodyną ir iškviečiami verslo proceso arba programos kintamumo taškuose. Šiai grupei galima priskirti tokius produktus kaip ILOG, BlazeAdvizor, BRBeans, QuickRules ir kt.
- Dirbtinio intelekto taisyklės (angl.: *artificial intelligence rules*) – taisyklės, kurios įtakoja algoritmus DI ir duomenų kasimo sistemose. Šiai grupei galima priskirti tokius produktus kaip Intelligent Mineramp, DB2Amp ir kt.
- Įvykių apibendrinimo taisyklės (angl.: *event correlation rules*) – taisyklės, kurios yra naudojamos įvykių apibendrinimui. Kaip gerą tokių sistemų pavyzdį galima paminėti tinklų valdymo sistemos. Tinklų įrenginiai generuoją didžiulius kiekius įvykiu, kuriuos apibendrina tokio tipo sistemos. Šiai grupei galima priskirti tokius taisyklių sistemų valdymo produktus kaip Tivoliamp, Eventconsole, HP OpenView ir kt.
- Taisyklės orientuotos į duomenys (angl.: *data-centric rules*) – taisyklės, valdančios duomenų užklausimą ir atnaujinimą. Tokios taisyklės nustato kaip duomenys turi būti transformuojami, kas gali prie jų prieiti, užtikrina duomenų darną ir vientisumą Šiai grupei galima priskirti tokius sistemų valdymo produktus kaip Versta.

Pagal funkcionalumą, kuris reikalingas Verslo taisyklių valdymo proceso palaikymui, verslo taisyklių įrankiai gali būti klasifikuoti į tris grupes. Kiekviena grupė suteikia tam tikrus, reikalingus taisyklių valdymo procesus [2].

- Verslo taisyklių įrankiai, skirti taisyklėmis grindžiamų informacinių sistemų kūrimui.
- Verslo taisyklių įrankiai, skirti kurti žiniomis grindžiamas dalykines programas.
- Verslo taisyklių įrankiai, skirti visos įmonės verslo taisyklių valdymui .

Įrankiai iš pirmos grupės yra nukreipti kurti taisyklėmis grindžiamą dalykinę programą. Jie siūlo įvairias savybes verslo taisyklės įgijimui, formalizavimui, modeliavimui ir realizavimui. Šie

įrankiai pirmiausia skirti kūrėjams (angl.: *developers*). Labai dažnai, įrankiai nenumato palaikymo verslo žmonėms. Daugelis įrankių rinkoje priklauso šiai grupei.

Programinės įrangos kūrėjai:

- Usoft (Ness Technologies, Inc.);
- Versata Logic;
- CDM RuleFrame sąjungoje su Oracle Designer (Oracle corporation);
- kt.

Antros grupės įrankiai palaiko žiniomis grindžiamos dalykinės kūrimą. Šie įtrankai išplaukia iš ekspertinių sistemų paradigmos. Jie dažnai grindžiami rezultatais iš turtingo ekspertinių sistemų technologijų palikimo. Taisyklių apdorojimui, jie naudoja specialias technikas, taip vadinamas išvedimo mašinos (angl. *inference engines*). Įrankiai paprastai yra lanksčios priemonės siūlančios didelę įvairovę savybių, kurios naudojamos verslo taisyklių valdyme verslo informacinių sistemų lygyje. Pagal „Gartner Group“, jų ateities ketinimai yra daug žadantys. Daug verslo taisyklių įrankių prisijungė prie šios grupės. Susijungime su verslo procesų valdymų, kuris paplitęs rinkoje šiandien, jų pasisekimas tikėtinas.

Programinės įrangos kūrėjai:

- ILOG Business Rules;
- Blaze Advisor (Fair Isaac Corporation);
- CleverPath Aion Business Rule (Computer Associates International, Inc.);
- Visual Rule Studio (RuleMachines Corporation);
- kt.

Paskutinė grupė įrankių skirta valdyti verslo taisyklėms iš verslo perspektyvos, nepriklausomai nuo tam tikros realizavimo aplinkos. Šios grupės įrankiai specializuojasi verslo taisyklių įgijime, apimant ir įgijimą iš įvairų verslo informacijos dalių. Tai padeda patalpinti verslo taisykles į kontekstą. Šių įrankių rinkoje nėra daug. Jų didžiausias trūkumas, kad jie nepalaiko taisyklių realizavimo. Šios grupės tipinis atstovas yra BRS RuleTrack.

2.2. Verslo taisyklių tarpusavio sąveikos realizavimas išvedimo mašinose

Gerokai paprastesnės ekspertinės sistemos buvo sukurtos problemoms, kurios reikalavo nuo 50 iki 500 taisyklių. Kitos – valdyti tūkstančius taisyklių. Labiau sudėtingesni verslo taisyklių valdymo įrankiai naudoja objektinius metodus, apibrėžti sąvokas, kurias naudoja taisyklės. Verslo taisyklių valdymo įrankiai taisykles saugo repozitoriuje, deklaratyvia forma, ir naudoja atskirtą

išvados darymo mašiną, kad apdoroti taisykles. Šis būdas garantuoja, kad kiekviena taisyklė gali būti saugoma kaip nepriklausomas vienetas ir pakeista arba pašalinta iš taisyklių saugyklos nekeičiant pačios sistemos [17].

Duomenų valdomuose samprotavimuose taisyklė yra išrenkama vykdymui, kai jos prielaidos yra tenkinamos. Judama į priekį nuo duomenų link problemos sprendimo. Toks samprotavimas yra plačiai taikomas tokiose srityse, kaip projektavimas, konfigūravimas, planavimas, ir tvarkaraščių sudarymas. Šiose problemų srityse duomenys valdo sprendimo pasiekimą, ir potencialiai egzistuoja daugybė skirtingų, bet lygiai priimtinų sprendimų. Tikslo valdomame samprotavime, išrenkamas tikslas ir sistema tikrina jo pagrįstumą surandant patvirtinančius įrodymus. šis požiūris yra idealiai tinkamas diagnostinėms problemoms, kurios turi nedidelį išvedamų išvadų skaičių. Taisyklių konfliktų aptikimas būtent ir yra tokia problema. Kai surandamas atsakymas, nutraukiamas vykdymas, arba einama link sekančio tikslo patenkinimo.

Viena iš labiausiai paplitusių verslo taisyklių valdymo sistemų yra ILOG Rules [20].

Verslo taisyklių valdymo sistemos ILOG Rules (C++) ir ILOG Jrules (Java) yra produktai kompanijos ILOG [5]. ILOG Riles (C++) naudoja C++ klasių bibliotekos, o ILOG Jrules (Java) – Java klasių bibliotekos. Sistema gali valdyti iki 50000 taisyklių.

ILOG taisyklės yra ECA tipo taisyklės, ir turi tokią pat struktūra kaip ir ECA taisyklės. ILOG taisyklė sudaryta iš trijų dalių: antraštės, sąlygos ir veiksmo. Bendras taisyklės užrašymo formatas yra toks:

```
rule ruleName { [ priority = value; ]
[ packet = packetName; ]
when { conditions ... }
then { [ actions ... ] }
};
```

Taisyklės antraštė aprašo taisyklės pavadinimą, prioritetą ir paketo pavadinimą. Sąlyga prasideda raktiniu žodžiu `when` ir po jo užrašomos sąlygos atskirtos kabliataškiais. Veiksmo dalys prasideda raktiniu žodžiu `then`, ir po jo aprašomas veiksmas, kuris yra vykdomas, kada visos sąlygos yra išpildytos.

Taisyklės prioritetas kontroliuoja taisyklių paleidimo eilės tvarką. Paketo pavadinimas suriša taisyklę su taisyklių paketu. Naudojant paketus susietos taisyklės yra grupuojamos.

ILOG Rules priemonėje verslo taisykles galima aprašinėti ir kitais budais: naudojant verslo veiksmų kalba - BAL (angl.: *business action language*), intuityvus IF-THEN-ELSE taisyklės formatas ir TRL (angl.: *technical rule language*), be to naudojamas ir “sprendimų lentelės formatas.”

Funkcinės savybės:

- Versijų kontrolė: Versijų kontrolė padeda sekti taisyklių istoriją, o būtent koks vartotojas, kada, kurią taisyklę ir kaip pakeitė. Galima sulyginti tuo pačiu metu vykdomas taisykles,
- Verslo naudotojui draugiška verslo taisyklių kalba: ILOG Rules verslo taisykles gali įrašinėti keliais būdais, kaip ir programuotojui skirta kalba, taip ir verslo naudotojui suprantama kalba. Tai leidžia verslo naudotojams valdyti verslo procesus.
- ILOG Rules API: ILOG Rules API leidžia nesunkiai kontroliuoti verslo taisyklių vykdymą, ir integruoti jas į C++ ir Java taikomąsias programas. Tai padaryti galima COM, COBRA arba IBM MQSeries technologijų pagalbą.
- Taisyklių derinimo programa: Naudojama įvykių, vykstančių taisyklių vykdymo metu kontrolei. Yra kontroliuojamas taisyklių vykdymo procesas. Be to galima inspektuoti objektų, su kuriais taisyklė yra susijusi, būsena, peržiūrėti visų įvykių istoriją.

ILOG komponentės:

- Rule Builder - pilnai integruota grafinė sąsaja verslo taisyklių kūrimui, derinimui bei diegimui.
- ILOG Rules verslo taisykles ir visą su jais susijusią verslo informaciją saugo verslo taisyklių saugykloje. Jame yra saugojami taikomųjų programų projektai, verslo taisyklės, verslo objektų modeliai, verslo taisyklių šablonai ir kita informacija.
- ILOG Rules Business Rule Editor skirtas verslo taisyklų kodo rašymui. Pagrindinė naudojama kalba – BAL, yra galimybė pasirinkti ir kitas kalbas.
- Įrankis, skirtas priėjimui prie verslo taisyklių, verslo taisyklių statusų, versijų valdymo. Jo pagalbą valdomi verslo taisyklių versijos, saugojama jų keitimo istorija.

2.3. Verslo taisyklių rinkinio konfliktų sprendimo būdai

Viena didžiausių problemų verslo taisyklių panaudojimui yra jų kiekis ir sunkiai nuspėjamos tarpusavio sąveikos. Taisyklėms veikiant kartu atsiranda įvairūs konfliktai, kai viena iš dviejų vykdomų taisyklių priima užduotį, o kita atmeta ir t.t. Šiame skyriuje nagrinėjamos problemos, susijusios su verslo taisyklių rinkinio darnos užtikrinimu, bei esami sprendimo būdai.

Vienas iš galimų sprendimo būdų, tai perdavimo algoritmas (angl.: *propagation algorithm*), skirtas statinių ECA taisyklių analizei. Analizės technika yra paremta bendro pritaikymo algoritmu, kuris yra naudojamas nuspręsti, kada vienos taisyklės veiksmas gali paveikti kitos taisyklės būseną ir nuspręsti, kada taisyklių veiksmai persijungia (angl:. *commute*). Algoritmas perduoda vienos taisyklės veiksmus per kitos taisyklės būseną arba veiksmą, kad būtų galima nuspręsti, kaip veiksmas gali paveikti būseną arba kitą veiksmą. Perdavimo algoritmas yra naudingas analizuojant

taisyklių užbaigimą, nes gali nuspręsti, kada viena taisyklė gali aktyvuoti kitą taisyklę. Taip pat jis yra naudingas analizuojant vientisumą, nes gali nuspręsti, kada dviejų taisyklių vykdymo eilės tvarka turi didelę svarbą [11].

Taisyklės gali būti susistemintos pagal ECA techniką aktyviose duomenų bazėse, kaip ryšys tarp įvykių, būsenų ir veiksmų. Buvo pastebėta, kad dvi svarbios ir pageidaujamos aktyvių taisyklių elgesio savybės yra užbaigimas (*angl. termination*) ir vientisumas (*angl. confluence*). Taisyklių rinkinys garantuotai užbaigiamas, kai kiekvienai duomenų bazės būsenai taisyklių apdorojimas negali tęstis be galo (taisyklės negali aktyvuoti viena kitą neribotą laiką). Taisyklių rinkinys yra vientisas, jeigu kiekvienai duomenų bazės būsenai, galutinė būsena po taisyklių apdorojimo yra nepriklausoma nuo to, kokia eilės tvarka taisyklės buvo įvykdytos [10].

Kitas problemos sprendimo būdas galėtų būti paremtas verslo taisyklių prioritetų ir jų vykdymo strategijos nustatymu. Tokiu atveju, atsiradus konfliktams, žemesnį rangą turinti taisyklė galėtų būti tiesiog pašalinama. Dažnai minima sprendimo galimybė yra taisyklių ir su jomis susijusių procesų hierarchizacija, kai žemesnio lygmens procesai naudojantys taisykles esančias aukštesniame lygmenyje tampa tų procesų subprocesais. Kadangi verslo procesai gali būti specifikuojami verslo taisyklėmis, o verslo taisyklės gali būti detalizuojamos procesais, įvedus subprocesus taisyklės tuo pačiu gali įgyti tam tikrą hierarchinę struktūrą [24].

Dažniausiai taisyklių validavimui yra naudojama speciali simuliacija, kai tiesiog stebima ar esant tam tikroms sąlygoms sistema reaguoja adekvačiai.

Mano nuomone, pagrindinė problema yra ne standartinė taisyklių kalba, trūkumas yra taisyklių variklių standartinio veikimo semantikoje. Verslo taisyklių variklio taisyklių išreiškimo būdas sudaro programavimo kalbą. Kiekviena programavimo kalba turi sintaksę ir semantiką (ką programa iš tikro daro). Taip pat taisyklių rinkiniai gali būti transformuoti į duomenų struktūras, pritaikytas produkcinėse sistemose šablonų tikrinimo algoritmui RETE, kuris turi vykdymo semantiką taisyklių kalbai.

Taisyklių variklis įvertina taisyklių rinkinį, taigi yra tikimybė kad nors viena taisyklė bus aktyvuota. Kaip pasirinkti tinkama taisyklę? Yra keletas skirtingų konfliktų sprendimo algoritmų, kur kiekvienas yra skirtas pasiekti skirtingą efektą.

- Būdingo bruožo, prioritetų (angl.: s*alience*). Kai taisyklės autorius priskiria prioritetą tam tikrai taisyklei.
- Trukmės palyginimas (angl.: *longest matching*). Ši strategija vykdo taisykles, kurios turi daugiausiai sąlygos išraiškų. Tikslas vykdyti tą taisyklę, kuri turi daugiausia specifinių sąlygų (angl.: *clauses*).
- Naujausių faktų. Kai taisyklė buvo iškviesta naujo fakto arba faktas suteikia pirmumą. Panašiai kaip ir tiesioginio išvedimo metu, paieška vyksta į gylį.

- Nesirinkti iškarto prieš tai vykdytos taisyklės. Jei jau taisyklė atitiko reikiamus faktus ir buvo jau vykdyta, rekomenduojama jos antrą kartą nebepaleisti vykdymui. Idėja - visada vykdyti naujus dalykus.

Tai tik keletas bendriausių konfliktų sprendimo būdų, kuriuos galima sujungti ir naudoti kartu. Pagrindinė esmė yra tai, kad skirtingos konfliktų sprendimų strategijos gali įtakoti taisyklių vykdymo eiliškumą. Pasikeitimai, kai keli faktai gali įtakoti taisykles, kurios prieš tai buvo prieštaraujančios, bet nepastebėtos konflikto sprendimo strategijos, nevykdomos visai.

Dažniausiai sudarant taisyklių rinkinius, neatsižvelgiama į vykdymo eiliškumą. Praktikoje nepatartinas toks taisyklių rinkinio sudarymo požiūris. Dažniausiai taisyklės tiesiogiai išplaukia iš verslo procesų, kur jų vykdymo eiliškumas jau yra išlanksto apibrėžtas [12].

Verslo taisyklių valdymo sistemos dažniausiai yra sudėtingos sistemos su komplikuota elgsena. Turėti skirtingas konfliktų sprendimų strategijas yra dažnai neišvengiama ir pageidautina.

Produkcinėse sistemos taip pat gali skirtis taisyklių paleidimas ir vykdymas paskutinėje stadijoje. Taisyklių rinkinys gautas iš ankstesnio tikrinimo algoritmo yra vadinamas konfliktų rinkiniu, o išrinkimo algoritmas vadinamas konfliktų sprendimo strategija. Tokios strategijos gali būti įvairios, nuo paprasčiausios, kur naudojamas taisyklių užrašymo eiliškumas; priskirti svorius ar prioritetus ir t.t. Bet kokiu atveju konfliktų sprendimo strategijos yra įgyvendinamos verslo taisyklių valdymo sistemose

Konfliktų sprendimo strategijos taikomos produkcinėse sistemose, kur nusprendžiama kuria tinkamiausią taisyklę vykdyti. Šios strategijos naudojimas reikalingas, kai dviejų ar daugiau taisyklių sąlygos yra patenkintos šiuo metu žinomu faktu.

Konflikto sprendimas skirstomas į tris pagrindines kategorijas:

- Tikslumas, specifiškumas (angl.: s*pecificity*) – kai dviejų ar daugiau taisyklių sąlygos yra patenkinamos, pasirenkama taisyklė su tiksliausiai apibrėžta sąlyga. Šis sprendimo būdas dažnai vadinamas „specializacijos laipsniu“ (angl.: *"degree of specialisation"*);
- Naujumas (angl.: *„recency“*) – kai faktai yra pažymimi, parodant kaip seniai jie yra pridėti į darbinę atmintį. Kai dvi ar daugiau taisyklių gali būti pasirinkta, pirmenybė teikiama tai, kuri paskutinė pridėjo naują faktą. Tai padeda panaudoti tinkamiausius faktus;
- Refraktorius (angl.: *refractoriness*) – kai taisyklės sąlyga yra patenkinta, bet prieš tai buvusi taisyklės sąlyga buvo patenkinta to paties fakto, tai taisyklė tiesiog ignoruojama. Tai padeda išvengti įeinamo begalinio ciklo sistemoje.

Taisyklių konfliktų aptikimui ir logiškumui bei nuoseklumui patikrinti galima naudoti logika pagrįstus mechanizmus, tokius kaip rezoliucija arba išvedimas, panaudojant modus ponens, atbulinį išvedimą (angl.: *backward chaining*), tiesioginį išvedimą (angl.: *forward chaining*). Šie metodai

dažniausia realizuojami išvedimo mašinose (angl.: *inference engine*), dirbančiose pagal Rete algoritmą arba specialiose ekspertinėse sistemose. Algoritmai, naudojami įvertinti taisyklės elgseną duotoje situacijoje vadinami išvados darymo (angl.: *inference*) arba verslo taisyklių mašina (angl.: *business rule engine*). Esmė ta, kai taisyklių saugykla yra išanalizuota, išvedimo mašina paima įėjimo duomenys ir tada patikrina taisykles, esančias taisyklių repozitoriuje, kad nustatyti kas jau yra padaryta. Skirtingų tipu išvados darymo mašinos dirba geriau su skirtingomis sprendimų darymo užduotimis. Išvedimo mašina nežino nieko apie taisykles, ji tik įgyvendina sistemingą taisyklių paiešką. Logiškai taisyklių tvarka nedaro jokios įtakos veikimui, ir taisyklių kiekis nėra svarbus. Dauguma atveju, visos taisyklės nebus reikalingos tikslui pasiekti ar išvadai gauti, bet jų buvimas nedaro jokios įtakos reikalingai išvadai gauti. Taisyklių atskirimas nuo išvedimo mašinos, kuri ieško išvadų, kurios patenkina tikslą, įgalina sistemos efektyvumą saugoti ir atlikti taisyklių paiešką. Keičiant tikslą, gaunami skirtingi rezultatai [17].

Šiuo metu dažniausiai naudojami tiesioginio ir atbulinio išvedimo metodai. Tiesioginis išvedimas prasideda su jau turimais duomenimis ir naudoja išvadų darymo taisykles, kad gauti daugiau duomenų kol tikslas yra pasiektas. Tiesioginio išvedimo samprotavimo procesas tęsiasi tol, kol pasiekiamas tikslas arba kol nebenustatoma daugiau naujų faktų (procesas taip pat gali tęstis amžinai). *Faktai* yra vaizduojami ir saugomi darbinėje atmintyje (angl. *working memory*), kuri yra pastoviai atnaujinama. *Taisyklės* pateikia galimą veiksmą, specifikuotos būsenos. Duomenų valdymo (angl.: *data driven*) samprotavimo procese taisyklės sąlygos (angl.: *conditions*) yra lyginamos su faktais iš faktų saugyklos. Jei sąlygos yra patenkinamos, tada veiksmas yra vykdomas. Atbulinis išvedimas yra tikslo valdymo samprotavimo procesas. Procesas prasideda nuo tikslo kuris turėtų būti pasiektas, naujos hipotezės (potikslis) yra gaunamos naudojant taisykles, kurios yra paleidžiamos jei jų išvada atitinka tikslą arba potikslį. Jei yra atitikimas, tai taisyklės būsenos (angl.: *conditions*) tampa naujomis hipotezėmis. Procesas pabaigiamas kai hipotezės yra jau žinomi faktai arba kai naujų hipotezių nerandama.

Apibendrinant tiesioginio ir atbulinio išvedimo metodus galima teigti, kad tiesioginis išvedimas – duomenų valdymo procesas, išvados gaunamos iš faktų. Naudojamas kai norima nustatyti pasikeitimų prasmę duomenų reikšmėms arba kai norima forma pagrįstos sąveikos, bendravimo (angl.: *form - based type of interaction*). Atbulinis išvedimas - tikslo siekiantis procesas, samprotavimas nuo hipotezių iki faktų (faktų radimas), taisyklės naudojamos pagrįsti tikslo teisingumą arba ne. Naudojamas kai norima nustatyti, kuri reikšmė yra sprendimas iš viso rinkinio arba kai norima klausimas – atsakymas tipo bendravimo (angl.: *question-and-answer dialog type of interaction*) [6].

Algoritmas, naudojamas aktyvuoti taisykles yra toks pats ir tiesioginiame, ir atbuliniame išvedime, ir turi tris žingsnius: identifikacija: kokios taisyklės gali būti paleistos ir vykdomos (angl.: *fired*), taisyklės parinkimas ir vykdymas.

Egzistuoja samprotavimais pagrįsta žinių bazių struktūra – išvadų tinklas. Išvadų tinklą galima pavaizduoti grafu, kurio mazgai reiškia faktų turimus parametrus, kaip pradinius duomenis ar išvestus iš kitų duomenų (t.y. tarpinių parametrų). Parametras gali būti kitos taisyklės prielaida ar išvada. Kiekvienas iš jų gali turėti vieną ar daugiau reikšmių, turinčių savo tikimybę. Grafe taisyklės vaizduojamos, kaip įvairių mazgų jungtys. Visos jos išvadų tinkle yra žinomos prieš sistemos vykdymą. Todėl jungtys gali būti suformuotos iš anksto, tuo būdu prielaidų tikrinimo metu minimizuojant faktų paiešką duomenų bazėje. Jos taip pat supaprastina išvadų mechanizmo sudarymą bei paaiškinimų apdorojimą. Kadangi žinomos jungtys, konfliktų sprendimo problema sprendžiama patikrintų taisyklių sąrašo sudarymu. Lengvai nustatoma, kurie veiksmai reikalingi atitinkamiems faktams išvesti. MYCIN, PROSPECTOR ir GenAID yra didelės žinių bazių sistemos, kuriuose sėkmingai pritaikoma išvadų tinklo architektūra. Jos panaudojamos diagnostikoje (MYCIN ir GenAID) arba mokslo bei inžinerijos žinių klasifikavimui.

Dažnai minimos ir šablonų tikrinimo sistemos.Kur tikrinant taisykles vykdoma išplėsta paieška, įjungiant sudėtingus šablonus, prielaidų parametrams nustatomos leidžiamos reikšmės bei ieškoma taisyklių, kurios atitiktų šias prielaidas. Ryšiai tarp taisyklių ir faktų šablonuose suformuojami dinamiškai. Šablonų tikrinimo sistemoje tikrinamos esamų taisyklių prielaidos. Taisyklės vykdymo metu naujai išvesti faktai įterpiami į žinių bazę. Taisyklių prielaidos yra paprasti arba sudėtingi šablonai. Paprasti šablonai yra ekvivalentiški išvadų tinklų parametrams, tačiau sudėtingų, kelių kintamųjų laukų šablonų tikrinimo sistemų architektūra žymiai skiriasi nuo išvadų tinklo. Šablonų tikrinimas remiasi automatizuotu samprotavimu, kuris panaudojamas ir PROLOG'e. Šablonų tikrinimas samprotavimų sistemose gali tapti labai sudėtingu, jei taisyklių prielaidoms bus naudojamos išplėstos konstrukcijos. Galima specifikuoti sudėtingus šablonus, kurie faktų bazėje neturi identiško atitikimo

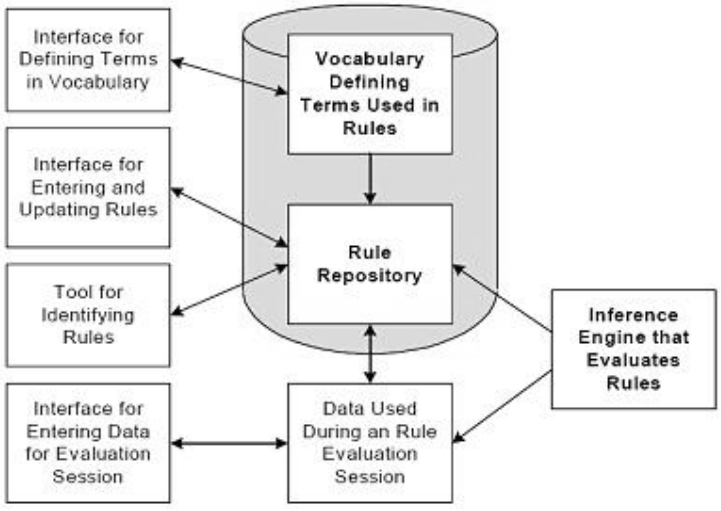
Yra labai daug darbų susijusių su taisyklių varikliu ir verslo logika. Produktai kaip „Jrules“ ir „QuickRules“ turi didelę paklausą, integruojant taisyklių variklius su taikomųjų programų sistemomis.

Šie varikliai yra parašyti Java kalba ir integruoti kartu su taikomųjų programų serveriu, kuriame išdėstytos nepriklausomos sesijos.. Tai leidžia taisyklių varikliui betarpiškai bendrauti su taikomųjų programų serveriu. Tačiau tai turi savo trūkumų. Taisyklių variklio efektyvumas yra sumažinamas ir yra priklausomas nuo taikomųjų programų serverio. Dauguma taikomųjų programų serverių apibrėžia panašius gijų prioritetus visoms sesijoms. Kol šios sesijos neturi papildomų privilegijų, jos turėtų charakteristikos trūkumų, jeigu jos siektų tikslo kaip atskiri serveriai.

Vartotojui naudojantis sąsaja, šios taikomosios programos naudojamos su natūralios kalbos apdorojimo supratimu, kurios įgalina dalykinės srities ekspertą ar sistemos kūrėją pradėti naudoti naujas taisykles be sistemos. Tačiau šie produktai nesuteikia galimybės vartotojui pačiam kurti ir naudoti taisykles. Taikomosiose programose kur vartotojams reikalinga, kad taisyklės būtų keičiamos dinamiškai, būtina, kad jie bendrautų su dalykinės srities ekspertu ir atliktų visus darbus per jį. Atsižvelgiant į taikomąsias programas internete, kur gali būti šimtai potencialių vartotojų, tai būtų neįgyvendinama. Taigi kompleksinės sąsajos buvimas sukelia sunkumų projektuojant paprastą vartotojo sąsają [5].

2.4. Išvedimo mašina

Algoritmai, naudojami įvertinti taisyklės elgseną duotoje situacijoje vadinami išvados darymo (angl.: *inference*) arba tiesiog išvedimo mašina (angl.: *inference engine*) [17]. Esmė ta, kai taisyklių saugykla yra išanalizuota, išvedimo mašina paima įėjimo duomenys ir tada patikrina taisykles, esančias taisyklių repozitoriuje, kad nustatyti kas jau yra padaryta. Skirtingų tipu išvados darymo mašinos dirba geriau su skirtingomis sprendimų darymo užduotimis. Tipinis verslo taisyklių variklis pavaizduotas 1 paveikslėlyje.



1 pav. ***Išvedimo mašinos architektūra*** [1]

Paveikslėlyje iliustruoti pagrindiniai verslo taisyklių mašinos elementai. Taisyklės ir žodynas, kuris apibrėžia taisyklų elementus dažniausiai yra saugomi duomenų bazėje arba repozitoriuje. Kai taisyklės yra įvertintos (angl.: *evaluated*), išvados darymo variklis naudoja duomenis, apibrėžtus vartotojo arba išgautus iš duomenų saugyklos, ir tada ieško reikiamų taisyklių išvadai gauti.

Pabandykime panagrinėti trivialų pavyzdį kaip taisyklėmis grindžiama sistema galėtų panaudoti 3 taisykles, kad pasiekti sprendimą. Šiuo atveju naudojamas atbulinis išvedimas (angl.: *backward chaining*), t.y. išvadų gavimas pradedamas nuo tikslo ir judama atgal, kad nustatyti tikslo vertę.

Pvz.

```
Tikslas: Spausdintuvas = ?

Taisyklė 1.
Jei kaina = žema
Tada spausdintuvas = rašalinis

Taisyklė 2.
Jei kaina = aukšta
Tada spausdintuvas = lazerinis

Taisyklė 3.
Jei turimos lėšos = mažiau negu $350
Tada kaina = žema
Kitu atveju kaina = aukšta
```

Išvedimo mašina pradeda paiešką taisyklių saugykloje ir ieško taisyklės, kuri užbaigiama tikslu *spausdintuvas*. Surandama Taisyklė 1. *spausdintuvas = rašalinis*. Paieškos procesas persikelia į Taisyklę 1., kur nustatomas naujas tikslas – rasti kainą. Procesas prasideda vėl iš naujo nuo taisyklių saugyklos ir ieškoma taisyklės, kuri daro išvadas apie kainą. Toliau nustatomas naujas tikslas – *turimos lėšos*.

Baigus paiešką taisyklių saugykloje trečią kartą ir neradus taisyklių, susijusių su turimom lėšom, sistema automatiškai generuoja klausimą vartotojui arba kreipiasi į duomenų bazę. Išvedimo mašina gavusi atsakymą, kad turimos lėšos yra *$300*, padaro išvadą, kad turimos lėšos yra mažiau už *$350* ir *kaina = žema*. Šiuo atveju išvedimo mašina remiasi savo žiniomis ir nustato, kad prielaida apie Taisyklę 1. yra teisinga. Taigi išvada – naudoti rašalinį spausdintuvą.

Faktiškai, išvedimo mašina patikrins ir kitas taisykles, kurios susijusios su spausdintuvo kaina. Radusi Taisyklę 2., bus bandoma patvirtinti, bet nepavykus, paieškos procesas baigiamas rekomendacija naudoti rašalinį spausdintuvą.

Kaip ir minėjau, tai labai paprastas pavyzdys. Pagrindinė esmė gerokai toli, ar šis požiūris veiks taip pat gerai su šimtais tokių taisyklių. Išvedimo mašina nežino nieko apie taisykles, ji tik įgyvendina sistemingą taisyklių paiešką. Logiškai taisyklių tvarka nedaro jokios įtakos veikimui, ir taisyklių kiekis nėra svarbus. Dauguma atveju, visos taisyklės nebus reikalingos tikslui pasiekti ar išvadai gauti, bet jų buvimas nedaro jokios įtakos reikalingai išvadai gauti. Taisyklių atskirimas nuo

išvedimo mašinos, kuri ieško išvadų, kurios patenkina tikslą, įgalina sistemos efektyvumą saugoti ir atlikti taisyklių paiešką. Keičiant tikslą, gaunami skirtingi rezultatai.

Duomenų valdomuose samprotavimuose taisyklė yra išrenkama vykdymui, kai jos prielaidos yra tenkinamos. Judama į priekį nuo duomenų link problemos sprendimo. Toks samprotavimas yra plačiai taikomas tokiose srityse, kaip projektavimas, konfigūravimas, planavimas, ir tvarkaraščių sudarymas. Šiose problemų srityse duomenys valdo sprendimo pasiekimą, ir potencialiai egzistuoja daugybė skirtingų bet lygiai priimtinų sprendimų. Tikslo valdomame samprotavime, išrenkamas tikslas ir sistema tikrina jo pagrįstumą surandant patvirtinančius įrodymus (supporting evidence). šis požiūris yra idealiai tinkamas diagnostinėms problemoms, kurios turi nedidelį išvedamų išvadų skaičių. Taisyklių konfliktų aptikimas būtent ir yra tokia problema. Kai surandamas atsakymas, nutraukiamas vykdymas, arba einama link sekančio tikslo patenkinimo [7].

2.5. Išvedimo metodai

Išvados gavimas – tai procesas, naudojamas žynių sistemose, kad gauti naują informaciją iš jau žinomos informacijos. Išvadoms gauti reikalingos žinios, faktai ir problemos sprendimo strategijos [22].

Išvedimo mašinose dažniausia naudojami du išvadų gavimo būdai:

1. Tiesioginis išvedimas (angl. *forward chaining*). Tai duomenų valdymo procesas,

2. Išrinkti galimas išvadas ir bandyti patvirtinti jų pagrįstumą patvirtinančiais įrodymais (tikslo valdymas arba atbulinis išvedimas).

Egzistuoja ir kiti metodai, tokie kaip Rete tinklas, modus ponens, išvadų tinklas, šablonų tikrinimo metodas ir kita.

Toliau panagrinėsime labiausiai naudojamus išvadų gavimo metodus: tiesioginį išvedimą, atbulinį išvedima ir Rete algoritmą.

2.5.1. Tiesioginis išvedimas

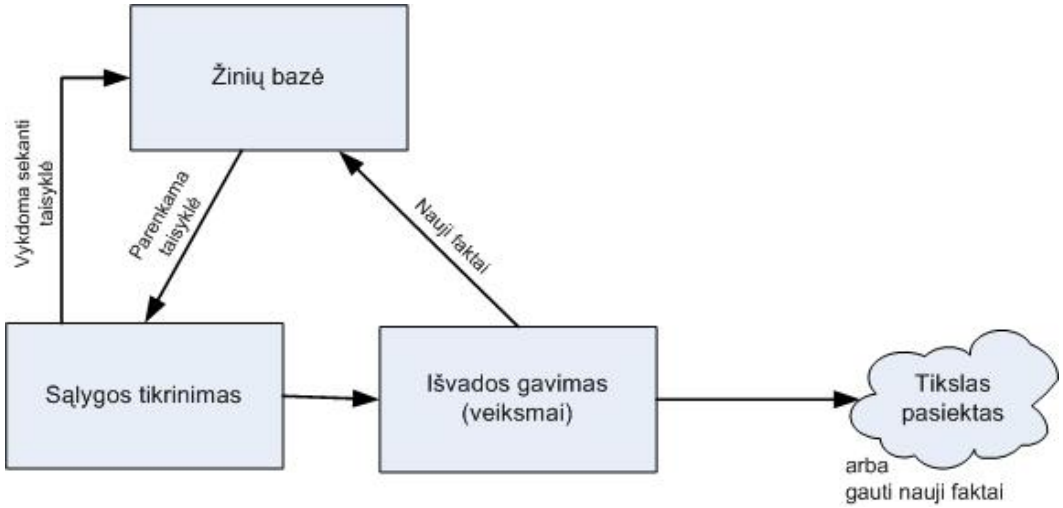
Tiesioginis išvedimas prasideda su jau turimais duomenimis ir naudoja išvadų darymo taisykles, kad gauti daugiau duomenų kol tikslas yra pasiektas.

Duomenų valdymo (angl.: *data driven*) samprotavimo procese taisyklės sąlygos (angl.: *conditions*) yra lyginamos su faktais iš faktų saugyklos. Jei sąlygos yra patenkinamos, tada veiksmas yra vykdomas.

Tiesioginio išvedimo samprotavimo procesas tęsiasi tol, kol pasiekiamas tikslas arba kol nebenustatoma daugiau naujų faktų (procesas taip pat gali tęstis amžinai). *Faktai* yra vaizduojami ir

saugomi darbinėje atmintyje (angl. *working memory*), kuri yra pastoviai atnaujinama. *Taisyklės* pateikia galimą veiksmą, specifikuotos būsenos.

Tiesioginio išvedimo samprotavimo algoritmas pateiktas 7 paveikslėlyje.



2 pav. ***Tiesioginis išvedimas***

1. Jei tikslas yra pasiektas, tada procesas sustoja. Kitu atveju procesas toliau vyksta nuo „2“ žingsnio
2. Taisyklių rinkinys yra inicijuojamas visų taisyklių pagalba, nauji faktai yra klaidingi.
3. Jei taisyklių rinkinys yra neužpildytas ir nauji faktai klaidingi, tada procesas sustabdomas. Jei taisyklių rinkinys yra neužpildytas, bet nauji faktai teisingi, tada procesas tęsiasi nuo „2“ žingsnio.
4. Pirmos taisyklė sąlyga tikrinama su modelio faktais:

4.1. Jei visos sąlygos yra tinkamos, tada taisyklės veiksmas yra vykdomas. Jei nauji faktai yra gaunami, tai nauji faktai yra nustatomi kaip teisingi ir procesas tęsiasi, „5“ žingsnis.

4.2. Jei nors viena sąlyga neatitinka, tada taisyklių rinkinys toliau tikrinamas be šios taisyklės ir procesas tęsiasi toliau, „3“ žingsnis.

4.3. Jei viena sąlyga nėra žinoma, tada:

4.3.1. Jei yra galimybė rinkti informacija apie šią sąlygą, tada informacija yra surinkta ir tikrinimas kartojamas.

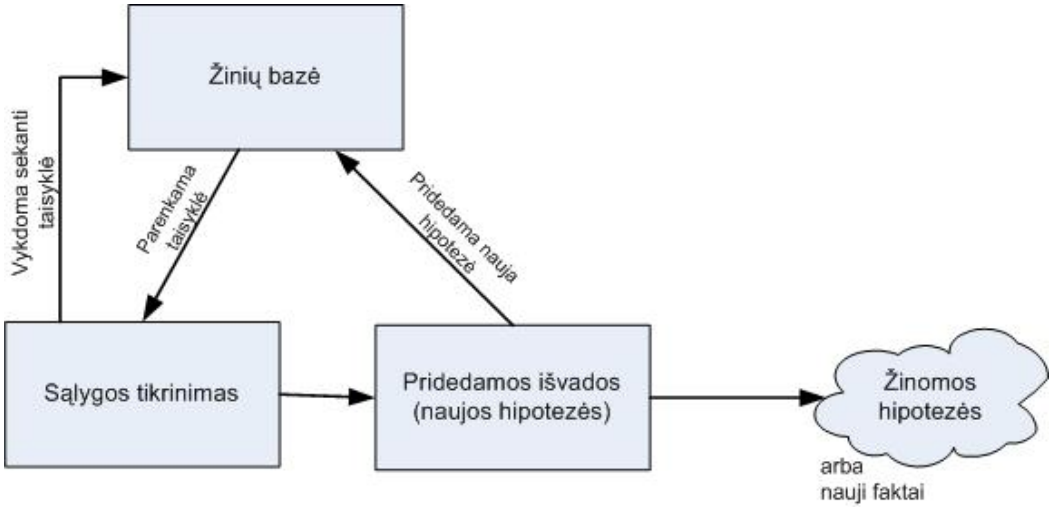
4.3.2. Jei nėra galimybės rinkti informacija apie šią sąlygą, tada taisyklių rinkinys tikrinimo metu vykdomas be šios taisyklės, „3“ žingsnis.

5. Kada tikslas pasiektas, t.y. taisyklė atitinka visas sąlygas, tai procesas sustoja. Kitu atveju rinkinys tikrinamas be šios taisyklės ir procesas tęsiasi toliau, „3“ žingsnis.

2.5.2. Atbulinis išvedimas

Atbulinis išvedimas yra tikslo valdymo samprotavimo procesas. Procesas prasideda nuo tikslo kuris turėtų būti pasiektas, naujos hipotezės (potikslis) yra gaunamos naudojant taisykles, kurios yra paleidžiamos jei jų išvada atitinka tikslą arba potikslį. Jei yra atitikimas, tai taisyklės būsenos (angl.: *conditions*) tampa naujomis hipotezėmis. Procesas pabaigiamas kai hipotezės yra jau žinomi faktai arba kai naujų hipotezių nerandama [17].

Atbulinio išvedimo samprotavimo algoritmas pateiktas 8 paveikslėlyje.



3 pav. ***Atbulinis išvedimas***

Toliau pateiktas pilnas proceso algoritmas. Taisyklių rinkinys pateikia taisykles kurios turi būti patikrintos naujais faktais ar jie yra išvesti ar ne.

1. Jei tikslas pasiekiamas iš karto, tada procesas yra užbaigiamas. Kitu atveju procesas tęsiasi, „2“ žingsnis.
2. Taisyklių rinkinys yra inicializuojamas, nauji faktai – klaidingi.
3. Jei taisyklių rinkinys yra neužpildytas ir nauji faktai klaidingi, tada procesas užbaigiamas. Jei taisyklių rinkinys neužpildytas, bet nauji faktai tinkami – procesas tęsiasi toliau, „2“ žingsnis.
4. Jei išvada (veiksmas) pirmos taisyklės prisideda prie siekiamo tikslo, tada bandoma patikrinti šios taisyklės sąlygas (angl.: *conditions*) su modelio faktais.
   - 4.1 Kai visos sąlygos yra patikrintos ir tinkamos, pradedamas taisyklės vykdymas. Jei gaunami nauji faktai, tai tie faktai nustatomi kaip teisingi ir tinkami. Procesas tęsiasi „5“ žingsnyje.
   - 4.2 Jei nors viena sąlyga neatitinka, tada taisyklių rinkinys toliau vykdomas be šios taisyklės ir procesas tęsiasi toliau (“3” žingsnis).
   - 4.3 Jei nors viena sąlyga nėra žinoma, tada:

4.3.1 Jei yra galimybė rinkti informacija apie šią sąlygą, tada informacija yra renkama ir tikrinimas kartojamas iš naujo.

4.3.2 Jei nėra galimybės gauti informacijos apie šią sąlygą, tai naujos problemos sprendimo procesas pradedamas nuo šios sąlygos tikslo gavimo.

4.3.2.1 Jei sąlyga vis dar nežinoma po šio proceso, tai taisyklių rinkinys toliau tikrinamas be šios taisyklės, ir procesas toliau vykdomas nuo "žingsnio".

4.3.2.2 Jei sąlyga žinoma po šio proceso, tai šios sąlygos tikrinimas kartojamas iš naujo.

5. Jei tikslas pasiektas, tai procesas yra baigiamas. Kitu atveju taisyklių rinkinys toliau tikrinamas be šios taisyklės ir procesas toliau prasideda nuo "3" žingsnio.

Apibendrinant tiesioginio ir atbulinio išvedimo metodus galima teigti:

- Tiesioginis išvedimas – duomenų valdymo procesas, išvados gaunamos iš faktų. Naudojamas kai norima nustatyti pasikeitimų prasmę duomenų reikšmėms arba kai norima forma pagrįstos sąveikos, bendravimo (angl.: *form - based type of interaction*).
- Atbulinis išvedimas - tikslo siekiantis procesas, samprotavimas nuo hipotezių iki faktų (faktų radimas), taisyklės naudojamos pagrįsti tikslo teisingumą arba ne. Naudojamas kai norima nustatyti, kuri reikšmė yra sprendimas iš viso rinkinio arba kai norima klausimas – atsakymas tipo bendravimo (angl.: *question-and-answer dialog type of interaction*).

Algoritmas, naudojamas aktyvuoti taisykles yra toks pats ir tiesioginiame, ir atgaliniame išvedime ir turi tris žingsnius:

- identifikacija: kokios taisyklės gali būti paleistos ir vykdomos (angl.: *fired*);
- taisyklės parinkimas;
- vykdymas.

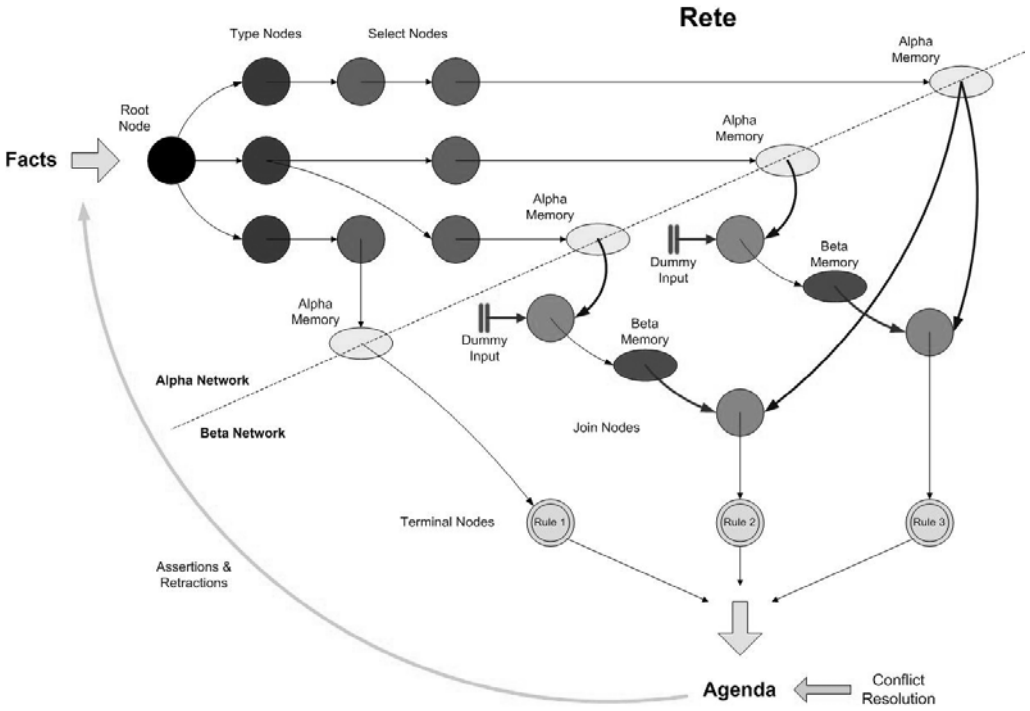
Atbulinio išvedimo mašinos yra labiau ribotų galimybių negu tiesioginio. Yra įvairaus tipo užduočių kurios gali būti sprendžiamos lengvai tiesioginiu išvedimu, kur atbulinio išvedimo mašina yra sunku arba visai neįmanoma išspręsti.

Atbulinio išvedimo sistemos yra tinkamos diagnostinėms ir klasifikavimo užduotims, bet mažiau tinkamos planavimui, konstravimui, proceso stebėjimui ir keletui kitų užduočių. Kai tiesioginis išvedimas lengvai susitvarko su šiom užduotim.

2.5.3. Rete tinklas

Rete – išvados darymo mechanizmas, kuris suriša pradžioje nesusijusias taisykles į loginį tinklą. Rete paieškos pirmyn algoritmo esmė, kad iš faktų ir jų derinių yra sukuriamas grafas – medis. Jame faktai išdėstomi ir sujungiami taip, kad pagal taisyklių prielaidų šablonus, būtų vykdoma kuo efektyvesnė peržiūra/paieška, t.y. medis atsimena surastus faktų derinius ir taip sutaupo paieškos laiko (kai tų derinių dažnai ieškoma). Atsiradus naujam faktui, medis minimaliai reorganizuojamas [31].

Žemiau pavaizduota diagrama iliustruoja pagrindinę Reta topografiją ir parodo ryšius tarp skirtingų mazgų ir atminties.



4 pav. ***Rete tinklo topografija***

2.5.4. Sprendimų lentelės ir medžiai

Sprendimų lentelė supaprastina pasikartojančių taisyklių vaizdavimą, kur tokių pačių parametrų reikšmės skirtingai sąlygoją rezultatą. Sprendimų lentelės taip pat leidžia laisvą konteksto pasirinkimą. Skirtingai nuo sprendimų medžio, nebūtina nuosekliai laikytis kelio siekiant tikslo. Tiktai įėjimo reikšmės privalomiems parametrams išilgai ašių yra reikalingi.

Toliau pateikiamos rekomendacijos kada naudoti ir kada ne sprendimų lenteles:

- Sprendimų lentelės dažniausiai naudojamos pastoviems parametrų rinkiniams, kur galimas didelis kiekis parametrų reikšmių.

- Lentelė su daug skirtingų parametrų greitai tampa sudėtinga ir mažinamas jos naudingumas kaip supaprastinimo koncepcija.
- Verslo analizė dažnai naudoja sprendimų lenteles atvaizduoti detalius reikalavimus, taigi replikuojant šią struktūrą taisyklių variklyje yra panaudojama jau žinoma koncepcija.
- Sprendimų lentelės gali greitai parodyti kur trūksta rezultato ar sprendimo.

Taip pat ne visais atvejais, visi parametrai turi tinkamus sprendimų rezultatus visais atvejais. Verslas gali labiau reikalauti apibrėžti taisykles, kurios valdo matricą. Lentelė gali mėginti naudoti per daug parametrų vieno sprendimo priėmimo metu. Tuo atveju lentelė turi būti padalinta į kelias ir sujungtos taisyklių srautu. Lentelės struktūra ne visada gali būti geriausias būdas pateikti tam tikrus taisyklių rinkinius ir taisyklės turėtų būti paverstos į individualios taisyklės formuluote.

Sprendimų medžiai padeda sistemai pasiekti prognozuojamą sprendimą, kuris priklauso nuo turimų žinių kelio iki siekiamos išvados. Kiekvienas analitikas ar taisyklių autorius planuoja sprendimų kelią su išsišakojančiomis sąlygomis, vedančiomis prie skirtingų rezultatų. Vykdymo metu sprendimų medžiai labai dažnai būna panašus į struktūrinį programavimą. Nuosekliai parengtas išsišakojimų formuluotės gali būti palygintos su „if“, „then“, „else“ sakiniais tradiciniame kode.

Priešingai sprendimų lentelėms, sprendimų medžiai gali turėti nenuoseklų parametrų rinkinį kuris nepadaro jo sudėtingu. Pavyzdžiui, kiekvienam šakos taškui nereikia turėti susiejimą su tuo pačiu parametru. Medžiai pasidaro labiau komplikuoti, kai kiekvienas parametras potencialiai gali turėti dideli skaičių skirtingų reikšmių. Kiekviena galima parametro reikšmė medyje pasidaro išsišakojimo mazgu (angl.: *node*). Paprastai medžiai tampa sudėtingais ir nesuvaldomais kai išsišakojimų lygiai didėja.

Nerekomenduojama naudoti medžių taisyklių įgyvendinimui, nebent yra labai geros priežastys apeiti išvedimą (išvados gavimą) ir priversti vienos taisyklės vykdymą prieš tai esančia taisyklės. Sprendimų medžiai nerekomenduojami dėl kelių priežasčių:

- Sprendimų medžiai labai nepatvarūs, kai taisyklės keičiasi ir reikalinga jų priežiūra.
- Sprendimų medžiai labai greitai tampa sudėtingi ir sunkiai valdomi.
- Taisyklių varikliai panaudoja išvedimą nustatyti taisyklių vykdymo eiliškumą, nuoseklumą.

Vientisas sprendimo priėmimo procesas yra sukeltas pasirenkant vieną parametrą (pagrindinį elementą). Naudojant išvedimą, taisyklių variklis gali prasidėti nuo bet kurios vietos ir veikti link to paties atsakymo iš bet kurios pusės, kai tik įvykdo taisykles.

Nustatant vykdymo tvarką mažam taisyklių skaičiui rinkinyje, geriau naudoti prioritetų nustatymui taisyklėms svorius negu medžius.

Bet kokiu atveju rekomenduojama naudoti sprendimų medžius taisyklių analizavimo procese dėl kelių priežasčių:

- Sprendimų medžiai padės analitikui identifikuoti taisyklių grupes, kurios dažniausiai paleidžiamos vykdymui kartu, kartu identifikuoti taisyklių rinkinius.
- Identiškos mažesnio išsišakojimo struktūros skirtingose didelio medžio vietose parodo taisyklių rinkinio panaudojimo galimybes taisyklių vykdymo sraute.
- Analizė, naudojant sprendimų medžius, gali parodyti verslo procesus paslėptus didelėje taisyklių grandinėje.

2.6. Predikatų logika

Formalioji, pavyzdžiui, predikatų, logika turi esminę reikšmę tiek taisyklių išraiškų veiklos terminais taisyklingumui, tiek jų įgyvendinimo technologijoms. Predikatų skaičiavimas pateikia daugiau lankstumo ir tikslesnį žinių pateikimą. Predikatų skaičiavimo nedalomi sintaksės elementai yra simboliai: jie negali būti padalinti į jų komponentų dalis naudojant kokius nors veiksmus.

Predikatų skaičiavimo semantika pateikia formalius pagrindus tam kad nustatyti tiesos reikšmę, taisyklingai suformuotoms išraiškoms. O išraiškos teisingumas priklauso nuo konstantų, kintamųjų, predikatų ir funkcijų sujungimo į objektus bei ryšių tarp jų, duotoje apibrėžimo srityje [4].

Konstantos aprašomos tiksliai įvardinant jų reikšmes. Teigiamojo pobūdžio sakinys su n kintamųjų, kuris virsta teiginiu, kai vietoje visų kintamųjų įstatomos konkrečios reikšmės, vadinamas n-viečiu predikatu. Funkcijos naudojamos įvertinti kintamųjų reikšmes, pvz.: ūgis(AsmuoX). Kintamieji tai tam tikros aibės elementai. Aibę galima nusakyti išvardijant jos elementus, arba predikatu nurodant jos elementams būdingą požymį, dėsningumą. Tam tikrais atvejais aibės apibrėžimas predikatu yra tikslesnis, nei elementų vardinimas, ypač begalinės aibės atveju. Kiekvienas predikate naudojamas kintamasis turi tam tikrą apibrėžimo sritį, kuri konkrečiu atveju nurodoma arba yra aiški iš konteksto. Ne su visomis kintamųjų reikšmėmis predikatai virsta prasmingais teigiamojo pobūdžio sakiniais, kurių teisingumą galima nustatyti, todėl iš anksto reikia apibrėžti kintamųjų reikšmių aibes. Aibė $A:=\{(x1,x2,...,xn)|P(x1,x2,...,xn)=1\}$ yra vadinama predikato teisingumo aibe. Predikatas virsta teisingu arba klaidingu teiginiu, visus kintamuosius pakeitus konkrečiomis (galimomis ir priklausančiomis predikato apibrėžimo sričiai) reikšmėmis ir panaudojus visuotinumo bei egzistencijos kvantorius, kurie atitinkamai reiškia žodžius “su kiekvienu” bei “egzistuoja” ir žymimi simboliais [31].

Kiekviena taisyklė gali būti išreikšta formule arba predikatu. Pavyzdžiui, sakinys “Įmonės darbuotojas turi būti vyresnis negu 18 metų amžiaus” yra vienvietis predikatas. Šis sakinys yra

teigiamojo pobūdžio, bet nėra teiginys, nes nežinoma jo teisingumo reikšmė. Sakinys „Įmonės darbuotojas Jonas Petraitis yra vyresnis negu 18 metų“ arba sakinys „Įmonės darbuotojas Petras Jonaitis yra jaunesnis negu 18 metų amžiaus“ yra teiginiai ir pirmasis teiginys teisingas, o antrasis klaidingas. Tai reiškia, kad su skirtingomis kintamojo reikšmėmis taisyklė galioja arba yra pažeidžiama. Taisyklės galiojimas tikrinamas vertinant teiginių atitikimą suformuluotam predikatui. Pateikto predikato teisingumo aibė yra įmonės darbuotojų aibė. Arba kalbant apie sistemos modelį, tai yra visi analizuojamos sistemos klasių modelio klasės „Darbuotojas“ egzemplioriai.

Pavyzdžiui taisyklė, kad nuolatiniam klientui taikoma 7.5% procentų nuolaida, gali būti užrašyta tokiu predikatų logikos sakiniu:

```
nuolaida(klientas, nuolatinis, 7.5%)
```

3. VERSLO TAISYKLIŲ RINKINIO DARNOS UŽTIKRINIMAS LOGINIO IŠVEDIMO MAŠINA METODO FORMULAVIMAS

Viena didžiausių problemų verslo taisyklių panaudojimui programų inžinerijoje, yra jų kiekis ir sunkiai nuspėjama tarpusavio sąveika. Taisyklėms veikiant kartu atsiranda įvairūs konfliktai.

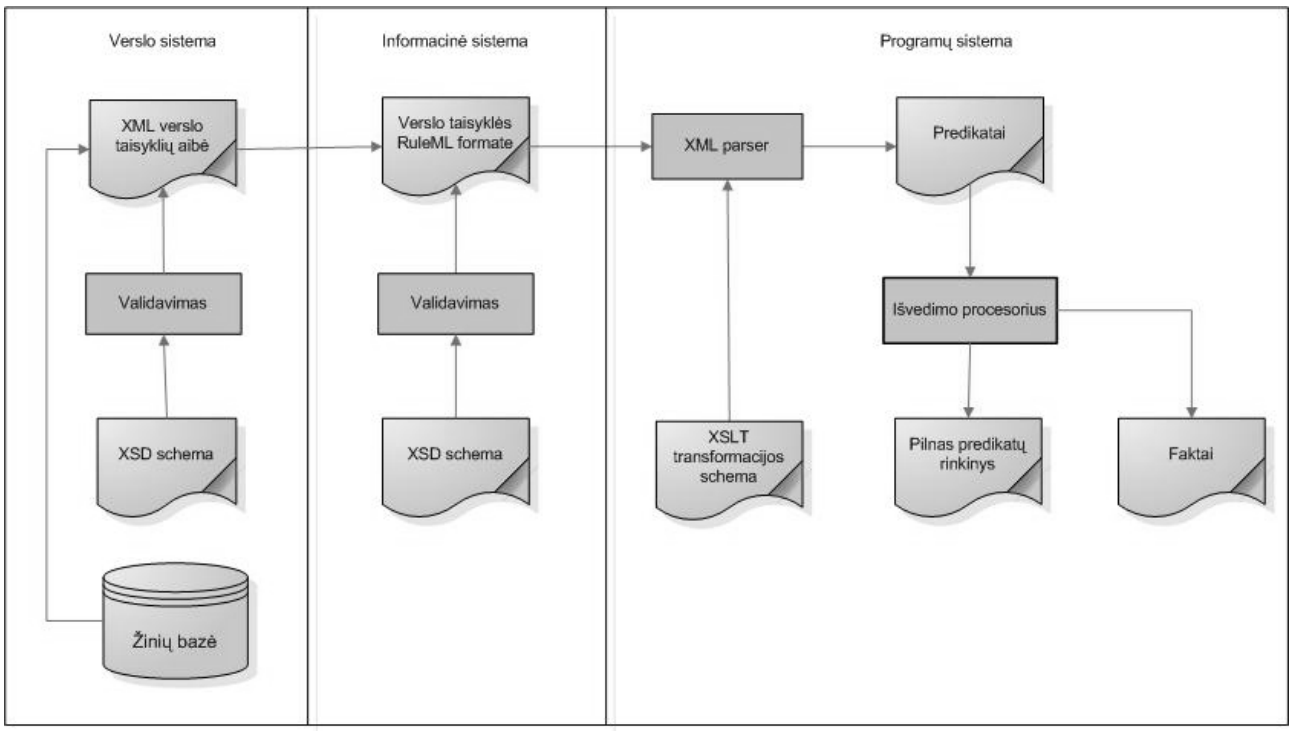
Taisyklių konfliktų aptikimui ir logiškumui bei nuoseklumui patikrinti galima naudoti logika pagrįstus mechanizmus. Šie metodai dažniausia realizuojami išvedimo mašinose (angl.: *inference maschine*).

Pabandysime suformuluoti metodiką, kuri leistų XML atvaizduotas verslo taisykles transformuoti į predikatus bei komponuoti į verslo taisyklių rinkinius (angl.: *rule set*), siekiant tuos rinkinius panaudoti duomenų analizei programų sistemoje. Šis verslo taisyklių rinkinys turi būti maža to dar ir išsamus (angl.: *complete*) ir neprieštaraujantis. Tam ir bus naudojama išvedimo mašina.

Verslo taisyklių rinkinio darnos užtikrinimas loginio išvedimo mašina (angl.: *inference engine*) ir jos panaudojimas sprendimų priėmimo automatizavimui metodas detalizuojamas:

- Verslo taisykles pavaizduotas XML formatu transformuoti į predikatų logikos sakinius ir patikrinti išvedimo mašinoje.
- Verslo taisykles atvaizduotas XML forma, sujungti į nepriklausomus, prasminiais ryšiais susietus taisyklių rinkinius.
- Panaudojus loginį išvedimą automatizuoti informacijos analizės strategijos parinkimo sprendimus, panaudojant verslo taisyklių rinkinius kartu su faktais, gautais iš verslo duomenų ir atspindinčiais esamą verslo sistemos būseną arba įvedus faktus atspindinčius siekiamą verslo sistemos būseną.

Verslo taisyklių rinkinio darnos užtikrinimo problemos sprendimas galėtų būti realizuotas žemiau suformuluotu metodu, naudojant verslo taisyklių transformaciją ir išvedimo logiką (5 pav.).



5 pav. ***Grafinis metodo vaizdavimas***

Pasiūlytame metode siūloma klasifikuoti verslo taisykles ir vaizduoti XML forma, sujungtas į specialius taisyklių rinkinius kartu su faktais, gautais iš esamų verslo duomenų (žinių bazė). Kiekvienai taisyklių grupei turi būti sukuriama XSD schema (DTD dokumentas). XSD schema arba DTD dokumentas skirti validuoti taisyklėms, t.y. turi būti patikrinama, ar taisyklė atitinka tai taisyklių klasei skirtą DTD dokumentą ar XSD schemą. Sekančiame žingsnyje duotasis verslo taisyklių rinkinys transformuojama į informacijos apdorojimo taisykles. Toliau šis taisyklių rinkinys transformuojamas į predikatus ir užkraunamas išvedimo mašinoje. Taip patikrintas rinkinio funkcionalumas, ko pasėkoje būtų gaunamos naujos išvestinės taisyklės bei patikrinamas atskirų taisyklių vientisumas, t.y. ar nėra tarp jų prieštaravimų. Taisyklių atskirimas nuo išvedimo mašinos, kuri ieško išvadų, kurios patenkina tikslą, įgalina sistemos efektyvumą saugoti ir atlikti taisyklių paiešką. Keičiant tikslą, gaunami skirtingi rezultatai.

XML kalba yra labai lanksti transformacijų požiūriu. Transformacijoms aprašyti yra sukurta XSLT kalba, kuria remiantis galime daryti prielaidą, kad verslo taisyklės pavaizduotos XML gali būti transformuojamos naudojant XSLT schemas į bet kokį pavidalą. Transformacijos procese dalyvauja trijų tipų dokumentai: XML, XSD ir XSLT. XML dokumente pateikiama pavaizduota verslo taisyklė, XSD dokumente apibrėžta verslo taisyklių vaizdavimo XML dokumente kalba ir žodynas, o XSLT dokumentas yra naudojamas kaip schema, pagal kurią verslo taisyklė transformuojama iš formalaus atvaizdo XML dokumente į naują pavidalą, šiuo atveju į predikatus [20].

Kita labai svarbi transformacijų taikymo sritis yra verslo taisyklių perkėlimas. Kadangi šiuo metu kuriamos informacinės sistemos dažnai yra heterogeninės, jose naudojami įvairiausi tiek specifikavimo, projektavimo ar programavimo metodai, priemonės ir kalbos. Siekiant pakartotinio verslo taisyklių panaudojimo informacinėse sistemose, iškyla labai svarbi tokių taisyklių perkėlimo problema, nes yra naudojami skirtingi jų formalizavimo metodai, kalbos ir dokumentų formatai.

Dauguma taisyklių naudojimu pagrįstų sistemų turi galimybę įkelti XML kalba pavaizduotas taisykles, todėl lieka vienintelė problema – išversti iš vieno formato XML kalbos į kitą. Tam gali būti sėkmingai panaudojami XSD dokumentai, kuriuose aprašytos XML dokumente panaudota kalba, jos konstrukcijos ir žodynas.

Tokio perkėlimo mechanizmo panaudojimas yra atskira taisyklių transformavimo rūšis. Šio proceso metu taip pat sukuriama XSLT schema. Skiriasi tik jos sukūrimo mechanizmas. Šiuo atveju yra žinomas tiek šaltinio, tiek paskirties formatas aprašytas XSD (arba DTD) schemomis. Todėl naudojant standartines XSLT konstrukcijas vienu formatu užrašyto dokumento laukai gali būti tiesiog surišami su atitinkamais kito formato dokumento laukais. Be to, galimos įvairios laukų kombinacijos, aritmetiniai veiksmai, filtravimo operacijos ir papildomų konstantų įvedimas [21].

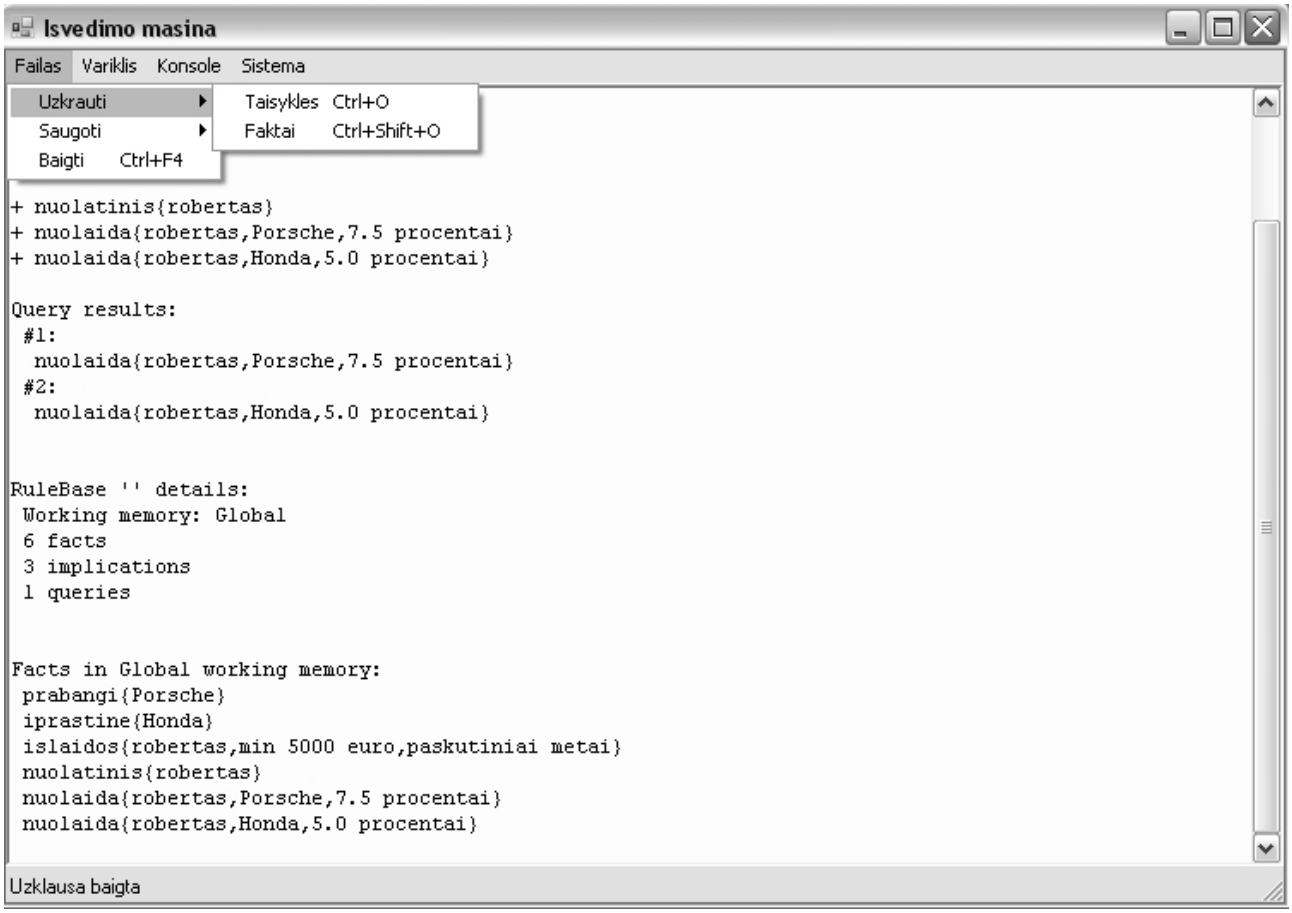
Išvedimo mašiną galima naudoti ir taisyklių sistemos darnos testavimui. Tam reiktų sukurti tam tikrus faktus, užrašyti juos predikatais ir fiksuoti rezultatus, kurie turi gautis. Įvedus naują taisyklę į taisyklių rinkinį reiktų užkrauti visas taisyklių rinkinio taisykles kartu su naująja ir patikrinti ar gaunamas rezultatas atitinka testuojamąjį. Toks būdas bent jau iš dalies padėtų išvengti akivaizdžių klaidų.

Išvedimo metu gauti nauji faktai, patalpinami žinių bazėje, kurie toliau gali būti panaudoti duomenų analizėje, atspindinčius siekiamą verslo sistemos būseną.

4. PROTOTIPO APRAŠYMAS

Suformuluotam metodui realizuoti buvo sukurtas prototipas (6 pav.). Prototipas skirtas verslo taisyklių rinkinio darnos testavimui, naujų faktų gavimui.

Prototipas sukurtas MS Visual Studio .NET platformoje, C# programavimo kalba, pasinaudojus jau esamu išvedimo proceso algoritmu, suteikiant reikiamą funkcionalumą metodo realizavimui.



6 pav. ***Prototipo grafinė vartotojo aplinka***

Prototipo funkcinės savybės:

1. Prototipas leidžia užkrauti taisykles, atvaizduotas predikatų logikos sakiniais, išvedimo mašinoje.
2. Atlikus taisyklių užkrovimą, galima atlikti loginį tiesioginio išvedimo procesą, atvaizduoti atminties būseną konsolėje, atlikti specializuotas užklausas ir pateikti naujai gautus faktus.
3. Galimybė išsaugoti gautus naujus faktus RuleML formate.
4. Galimybė išsaugoti verslo taisyklės, atlikus išvedimą, RuleML formate.

5. VERSLO TAISYKLIŲ RINKINIO DARNOS UŽTIKRINIMAS LOGINIO IŠVEDIMO MAŠINA METODO EKSPERIMENTINĖ ANALIZĖ

Šiame skyriuje aprašomo eksperimento metu siekiama patikrinti aukščiau aprašytą metodą. Tam, pasitelkiamos atskiros metodo realizacijos, suskaidant problemą ir ją sprendžiant dalimis. Toliau pateikiama eksperimentinė metodo panaudojimo analizė.

5.1. Eksperimento formulavimas

Eksperimento uždavinys yra verslo taisykles pavaizduotas XML formatu transformuoti į predikatų logikos sakinius ir patikrinti išvedimo mašinoje. Gautas išvestines taisykles sujungti į nepriklausomus, prasminiais ryšiais susietus taisyklių rinkinius.

Eksperimento planas:

- Užrašytą verslo taisyklę XML kalba validuoti pagal RuleML XSD dokumentą.
- Sukurti XSD dokumentą taisyklei užrašyti RuleML.
- Sukurti XSLT schemą taisyklės transformavimui į predikatus.
- Taisyklių rinkinį, išreikštą predikatais patikrinti išvedimo mašinoje.
- Į taisyklių rinkinį įvesti akivaizdžiai prieštaraujančią taisyklę ir šį rinkinį patikrinti išvedimo mašina.

Taisyklės:

- Pirkėjui taikoma 5.0 procentų nuolaida, jei pirkėjas yra nuolatinis ir prekė įprastinė.
- Pirkėjui taikoma 7.5 procentų nuolaida, jei pirkėjas yra nuolatinis ir prekė prabangi.
- Pirkėjas laikomas nuolatiniu, jei per paskutinius metus išlaidos buvo mažiausiai 5000 eurų.

Faktai:

- Porsche yra prabangi prekė
- Honda yra įprastinė prekė.
- Robertas per paskutinius metus išleido 5000 eurų.

Užklausa:

- Pateikti visų pirkėjų nuolaidas perkant bet kokią prekę.

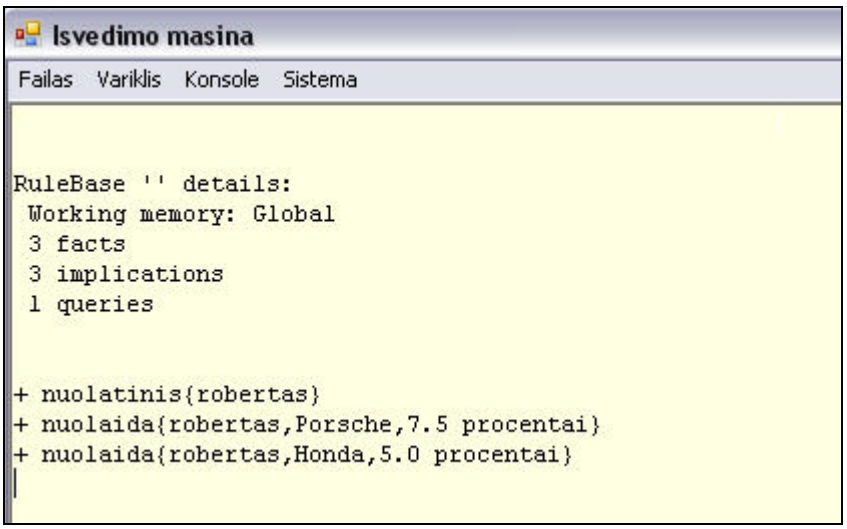
5.2. Eksperimento eiga

Su Altova™ XMLSpy užrašome taisykles XML kalba ir validuojame ją pasinaudodami RuleML kalbos XSD dokumentu (žr. 2 priede).

Toliau pagal tam tikrą sugeneruotą RuleML XSD dokumentą ir XML pavaizduotas taisykles sukuriama XSLT schema, reikalinga taisyklių transformacijai į predikatų logikos sakinius. XSLT schemos generavimui naudojame Altova® StyleSheet Designer programą. Visos operacijos atliekamos vizualioje grafinėje aplinkoje, o schemos dinaminės dalies generavimui naudojamas tas pats XSD dokumentas, kaip ir XML byloje. Tai leidžia išvengti klaidų ir supaprastina laukų įterpimo procesą. Pagal XSLT schemą sugeneruojami predikatų logikos sakiniai. Toliau pateikiamas vienos taisyklės fragmentas RuleML formate.

```
<rulebase>
  <imp>
    <_head>
      <atom>
        <_opr>
          <rel>nuolaida</rel>
        </_opr>
        <var>pirkejas</var>
        <var>preke</var>
        <ind>5.0 procentai</ind>
      </atom>
    </_head>
    <_body>
      <and>
        <atom>
          <_opr>
            <rel>nuolatinis</rel>
          </_opr>
          <var>pirkejas</var>
        </atom>
        <atom>
          <_opr>
            <rel>iprastine</rel>
          </_opr>
          <var>preke</var>
        </atom>
      </and>
    </_body>
  </imp>
```

Taisyklės, atvaizduotos predikatais, užkraunamos išvedimo mašinoje ir atliekamas loginis išvedimo procesas. Taisyklių atvaizdavimas prototipo konsolėje pavaizduotas 7 paveiksle.

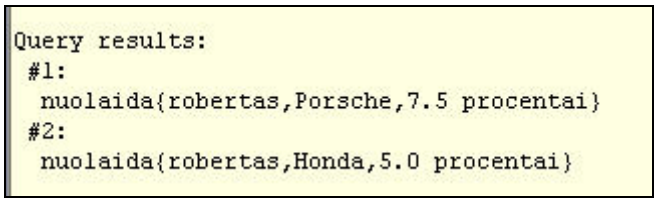


7 pav. ***Predikatų logikos sakiniai programos konsolėje***

Kaip matome, gavome taisykles, išreikštas predikatais:

- nuolatinis{robertas}
- nuolaida{robertas,Porsche,7.5 procentai}
- nuolaida{robertas,Honda,5.0 procentai}

Kadangi klientas atitinka abiejų taisyklių sąlygos dalis, tai atlikus užklausą, gauname visus galimus klientui taikomus nuolaidų variantus (8 pav.).



8 pav. ***Užklausos grąžinamas rezultatas***

Tiesioginio išvedimo procesas pradedamas nuo surinktų duomenų peržiūros. Tikrinama kiekviena taisyklė ir žiūrima, ar stebimi duomenys tenkina šios taisyklės prielaidas. Jei taisyklė tenkina prielaidas, tai ji vykdoma, išvedant naujus faktus, kurie gali būti naudojami kitose taisyklėse išvedant dar kitus faktus. Taisyklių patikrinimo procesas, tikrinant ar jos tenkina sąlygas, vadinamas taisyklių interpretavimu.

Atlikus tiesioginio išvedimo procesą, gaunami papildomi faktai (9 pav.).

```
Facts in Global working memory:
 prabangi{Porsche}
 iprastine{Honda}
 islaidos{robertas,min 5000 euro,paskutiniai metai}
 nuolatinis{robertas}
 nuolaida{robertas,Porsche,7.5 procentai}
 nuolaida{robertas,Honda,5.0 procentai}
```

9 pav. ***Nauji faktai***

Naujai gauti faktai, kurios prototipas leidžia išsaugoti atskirame faile (žr. 4 priede):

- prabangi{Porsche}
- iprastine{Honda}
- islaidos{robertas,min 5000 euro,paskutiniai metai}
- nuolatinis{robertas}
- nuolaida{robertas,Porsche,7.5 procentai}
- nuolaida{robertas,Honda,5.0 procentai}

Toliau eksperimente į išvedimo mašiną užkrauname sąmoningai sudarytą prieštaraujančių taisyklių rinkinį (žr. 5 priede) ir patikriname išvedimo mašinoje.

Įvedame naują taisykle, kuri prieštarautų nuolatinio kliento apibrėžimui:

- Pirkėjas laikomas nuolatiniu, jei per paskutinius metus išlaidos buvo mažiausiai 50 eurų.

Patikrinus šį taisyklių rinkinį išvedimo mašinoje, buvo gauti tie patys nauji faktai. Kaip ir minėjome tikrinama kiekviena taisyklė ir žiūrima, ar turimi pradiniai faktai šios taisyklės prielaidas. Jei taisyklė tenkina prielaidas, tai ji vykdoma, išvedant naujus faktus. Kadangi naujai įvesta taisyklė netenkino pradinių faktų (nuolatinis klientas mažiausiai turi būti išleidęs 5000 eurų), tai ji nebuvo panaudota išvedimo procese. Prototipo funkcijos leidžia išsaugoti tiktai tas taisykles, kurios buvo panaudotos procese. Taip išrenkamos tik susietos taisyklės, kurios sudaro prasmingą rinkinį.

5.3. Eksperimento rezultatai

- Eksperimento metu verslo sistemos taisyklė užrašyta XML kalba transformuota į predikatus, sukurta transformavimo schema. Taisyklės, išreikšos predikatų logikos sakiniai patikrintos išvedimo mašinoje.
- Išvedimo metu gauti nauji faktai, kurie gali būti atskirai išsaugomi ir panaudoti duomenų analizėje, atspindinčioje siekiamą verslo sistemos būseną.
- Eksperimento metu gautos tarpusavyje prasmingais ryšiais susietos taisyklės ir sudarančios vientisą rinkinį.

IŠVADOS

Verslo taisyklių panaudojimo informacinių sistemų kūrimui tema šiuo metu yra labai populiari ir plačiai nagrinėjama moksliniuose straipsniuose. Nagrinėjant mokslinę literatūrą, galima išskirti tokias pagrindines problemas, susijusias su verslo taisyklių panaudojimu: sudėtingas verslo taisyklių išgavimas iš verslo žmonių ir pateiktų verslo dokumentų; sudėtingas verslo taisyklių vaizdavimas; sudėtingas verslo taisyklių sistemos pilnumo užtikrinimas; verslo taisyklių sudėtingas darnos palaikymas. Atlikus detalesnę analizę:

1. Darbe pateikti verslo taisyklių apibrėžimai ir jų vaidmuo verslo bei informacinėse sistemose.
2. Pateikiamos detaliausiai išnagrinėtos ir labiausiai paplitusios verslo taisyklių klasifikavimo schemos bei vaizdavomo būdai. Pastebėta, kad verslo taisyklių klasifikavimo modelio pasirinkimas pirmiausia priklauso nuo tikslo, kuriam tarnaus. Racionaliausia naudoti skirstymą į klases pagal struktūrą, į rūšis pagal semantines savybes ir į tipus pagal kitus labiausiai konkrečiai situacijai tinkančius kriterijus.
3. Aptartas verslo taisyklių repozitorius ir jo privalomos funkcijos. Pastebėta, kad vis dažniau naudojamas XML, kaip standartas verslo taisyklių saugojimui ir apsikeitimui, bandoma naudoti tas pačias kalbas, taisyklių užrašymui. Panaudojant XSLT transformavimo schemas verslo taisyklės gali būti transformuojamos į įvarius informacinės sistemos ir programų sistemos objektus.
4. Atlikus verslo taisyklių valdymo programinės įrangos analizę, aprašytas verslo taisyklių sąveikos realizavimas išvedimo mašinose, išvadų darymo metodai. Galima teigti, kad verslo taisyklių valdymo sistemos dažniausiai yra sudėtingos sistemos su tam tikra komplikuota elgsena. Turėti skirtingas konfliktų sprendimų strategijas yra dažnai neišvengiama ir pageidautina.
5. Pasiūlytas verslo taisyklių komponavimo į prasmingus rinkinius metodas, atsižvelgiant į tai, jog viena didžiausių problemų verslo taisyklių panaudojimui yra jų kiekis ir sunkiai nuspėjama tarpusavio sąveika. Taisyklėms veikiant kartu atsiranda įvairūs konfliktai. Šis verslo taisyklių rinkinys turi būti maža to dar ir išsamus (angl.: *complete*) ir neprieštaraujantis. Tam naudojama išvedimo mašina.
6. Sukurtas programų sistemos prototipas, reikalingas pasiūlyto metodo eksperimentams.
7. Atlikus pasiūlyto metodo eksperimentinę analizę, pasiūlytas metodas leido XML kalba užrašytas taisykles transformuoti į predikatus bei patikrinti taisyklių rinkinio vientisumą. Buvo gauti nauji faktai, kurie gali būti atskirai išsaugomi ir panaudoti duomenų analizėje, atspindinčioje siekiamą verslo sistemos būseną.

Tolimesniuose darbuose numatoma, panaudojus loginį išvedimą, automatizuoti informacijos analizės strategijos parinkimo sprendimus, panaudojant verslo taisyklių rinkinius kartu su faktais, gautais iš verslo duomenų ir atspindinčiais esamą verslo sistemos būseną arba įvedus faktus atspindinčius siekiamą verslo sistemos būseną. Galibybė panaudoti agentus ir dirbtinį intelektą verslo taisyklėmis pagrįstose programų sistemose.

TERMINŲ IR SANTRAUKŲ ŽODYNĖLIS

ADBVS	–	aktyvi duomenų bazių valdymo sistema
Aktyvi duomenų bazių valdymo sistema	–	tai duomenų bazių valdymo sistema, kuri automatiškai gali reaguoti į veiksmus atliekamus su duomenų bazėse saugomais duomenimis.
DB	–	duomenų bazė
DBVS	–	duomenų bazių valdymo sistema
DDL (Data Definition Language)	–	duomenų aprašymo kalba.
Duomenų bazė	–	tai tarpusavyje susijusių duomenų rinkinys. Duomenys gali būti įvairūs: tekstai, paveikslai, garsai. Juos tvarko duomenų bazės valdymo sistema (DBVS).
Duomenų bazių valdymo sistema	–	tai programų sistema, skirta patikimai ir veiksmingai kurti ir valdyti dideles, integruotas ir daugelio vartotojų duomenų bazes.
DTD (Document Type Definition)	–	Dokumento tipo apibrėžimas.
ECA (Event-Condition-Action)	–	taisyklė įvykis-sąlyga-veiksmas angl.
ERA	–	Esybių-Ryšių-Atributų diagramos tipas.
Informacinė organizacijos sistema	–	tai organizacinio viene-to informacijos saugojimo, paieškos, perdavimo ir apdorojimo sistema.
Išvedimo mašina	–	algoritmai, naudojami įvertinti taisyklės elgseną duotoje situacijoje.
IS	–	Informacinė sistema.

Iteracija	–	Procesas, kurio metu nuosekliai atliekamos visos Iteracinio proceso stadijos.
Iteracinis procesas	–	Procesas, kurio metu praeinamos visos proceso stadijos nuo pradžios iki galo, jų metu sukuriami nauji artefaktai ir procesas prasideda vėl iš pradžių.
KIF (Knowledge Interchange format)	–	Apsikeitimo žiniomis formatas.
Kompiuterizuota informacinė sistema	–	Jeigu bent vienas informacinės sistemos posistemis yra kompiuterizuotas, tai sistema yra vadinama kompiuterizuota.
Programų sistema	–	tai integruota programų, rinkmenų, duomenų bazių ir žinių bazių visuma, skirta tam tikros klasės uždaviniams spręsti arba tam tikriems įrenginiams ar procesams valdyti.
Programų sistemos gyvavimo ciklo rėmimo sistema (CASE sistema)	–	tai instrumentinė sistema, skirta visoms programų sistemos gyvavimo ciklo modelyje numatytoms veikoms kompleksiškai automatizuoti.
RuleML (Rule Markup Language)	–	Taisyklių vaizdavimo žymėmis kalba.
SRML (Simple Rule Markup Language)	–	Paprasta žymių kalba taisyklėms vaizduoti.
UML (Unified Modeling Language)	–	unifikuota modeliavimo kalba.
Verslo taisyklė (business rule)	–	formali procedūra arba metodika, konkrečioje situacijoje atliekanti tipišką valdymo sprendimą.
Verslo taisyklės (business rules)	–	tai visuma visų taisyklių, kurios reglamentuoja veiklos vykdymą (įstatymai, politika, vidaus tvarka ir t.t.). Šis taisyklių rinkinys apima statinius (struktūrinius) ir dinaminius (funkcinius, elgsenos) organizacijos veiklos aspektus.

VS	–	verslo sistema
VT	–	verslo taisyklė
XML (eXtensible Markup Language)	–	Išplečiama žymių kalba skirta dokumentams su struktūrizuota informacija.
XSL (eXtensible Stylesheet Language)	–	Išplečiama šablonų kalba skirta dokumentams su struktūrizuota informacija transformuoti (vaizduoti).
XSLT (eXtensible Stylesheet Language Transformations)	–	Išplečiamos šablonų kalbos transformacija.
W3C (Wide Web Consortium)	–	Organizacija tvarkanti ir kurianti Internete naudojamus standartus.

LITERATŪRA

[1] A. Cawsey. Knowledge Representation and Inference. [žiūrėta 2006-12-14] http://www.cee.hw.ac.uk/~alison/ai3notes/chapter2_4.html.

[2] B. Andziulienė. Mokslinių darbų rašymo metodiniai patarimai. *Klaipėda: KU leidykla*, 2003.

[3] Barbara von Hale. Business Rules Applied: Building Better Systems Using the Business Rules Approach. *John Wiley & Sons, Inc., New York*, 2001.

[4] B. Beckert. First-order Logic (Logic, Deduction, Knowledge Representation), 2003. [žiūrėta 2006-11-15] http://www.uni-koblenz.de/~beckert/Lehre/Einfuehrung-KISS2003/folien08.pdf.

[5] Business Rules Group. Defining Business Rules: What is a Business Rule?, 2004, [žiūrėta 2005-01-17], http://www.BusinessRulesGroup.org

[6] C. McClintock, C. A. Berlioz. Implementing Business Rules in Java [interaktyvus]. *Java developers journal*. May 1, 2000 [žiūrėta 2006-04-01], http://www.sys-con.com/story/?storyid=36608

[7] D. Rosca, S. Greenspan, C. Wild. Enterprise Modeling and Decision-Support for Automating the Business Rules Lifecycle, Automated Software Engineering, v. 9, n. 4, 2002, 361 psl.

[8] D. Rosca, C. Wild. Business rules in the real world: a decision support approach. Journal: *Software Engineering and Knowledge Engineering*. pp., 1996, 121-128.

[9] E. Kock. Decentralizing the codification of rules in a decision support expert knowledge base. *University of Pretoria, Faculty of Engineering, Built Environment and Information Technology*, 2003, Master thesis, Promoter prof. Bishop J.M.

[10] G. Wagner. *Agent-Oriented Enterprise Modeling Based on Business Rules*. Proc. of 20th Int. Conf. on Conceptual Modeling (ER2001), November 2001, Yokohama, Japan, Springer-Verlag LNCS 2224.

[11] I. Graham. Service oriented business rules management systems. *Trireme*, 2006.

[12] J. Faget, M. Marin, P. Megard. Business Processes and Business Rules: Business Agility becomes Real. Workflow Handbook. 2003.

[13] J. Sinur. The Business Rule Engine. Magic Quadrant. 2003.

[14] J. Vanthienen. Quality by Design: Using Decision Tables in Business Rules, *Business Rules Journal, Vol. 5, Issue 2* (Feb. 2004), ISSN: 1538-6325, 7 pp.

[15] K. Kapočius, R. Butleris. Repository for Business Rules Based IS Requirements. *Informatica*, Vol.17, No.4, Institute of Mathematics and Informatics, Vilnius, 2006, pp. 503-518.

[16] K. D. Wilson. Business Rules, Platforms, and Inferencing. *Business Rules Journal*, 2003, 10. [žiūrėta 2006-10-21] http://www.BRCommunity.com/a2003/b169.html.

[17] N. L. Griffin and F. D. Lewis. A Rule-Based Inference Engine which is Optimal and VLSI Implementable, [žiūrėta 2007-02-15], http://cs.engr.uky.edu/~lewis/papers/inf-engine.pdf.

[18] M. Bajec, M. Krisper. A Methodology and Tool Support for Managing Business Rules in Organisations. *Information Systems* 30, *no. 6,* 2004, 423-443 psl.

[19] M. Bajec, M. Krisper. Issues and challanges in business rule-based Information systems development. *ECIS 2005, Information systems in a rapidly changing economy***.** Regensburg: Institute for Management of Information Systems, 2005, pp. 1-12.

[20] M. Thorpe. Business Rule Exchange - the Next XML Wave [interaktyvus]. *In XML Europe 2001. Internationales Congress Centrum (ICC), Berlin, Germany*, 21-25 May 2001 [žiūrėta 2007-04-01], http://www.gca.org/papers/xmleurope2001/papers/html/s15-2.html

[21] N. Prakash, S. Srivastava. Engineering Methods For Schema Transformation: Application to XML. IFIP TC8/WG8.1 Working Conference on Engineering Information Systems in the Internet Context, *Kanazawa, Japan*. *Kluwer Academic publishers,* 2002, 153 - 175 psl.

[22] O. Vasilecas, A. Smaizys. The framework: an approach to support business rule based data analysis. *In O. Vasilecas et al. (eds.), Proc. of 7th International IEEE Baltic Conference on Databases and Information Systems,* July 3, 2006, 141 – 147 psl.

[23] O. Vasilecas, A. Smaizys. Verslo taisyklių panaudojimas duomenų analizei ir informacijos pateikimui. *Informacinės technologijos 2005 (Information technologies 2005), Kaunas: Technologija*, 2005, 655-663 psl.

[24] O. Vasilecas, A. Smaizys. Business rule based knowledge integrated intelligent system framework. *Informacijos mokslai, Vilniaus universiteto leidykla*, 2005, 34 tomas, 195-200 p.

[25] P. Dorsey. *The Business Rules Approach to Systems Development* [interaktyvus]. Dulcian Inc. July 27, 2002 [žiūrėta 2007-05-01] http://www.dulcian.com/BRIM

[26] P. Harmon, C. Hall. Intelligent Software Systems Development: An IS Manager's Guide. *John Wiley*, 1993.

[27] R. G. Ross. Principles of the Business Rule Approach. *Addison-Wesley*. 2003. pp. 400

[28] R. Ross, K. Healy. Organizing Business Plans. The Standard Model for Business Rule Motivation. *Business Rules Group*. 2000.

[29] R. Strigūnas, A. Šmaižys, O. Vasilecas. Verslo taisyklių rinkinio darnos užtikrinimas loginio išvedimo mašina. *Informacinės technologijos 2007*, *Konferencijos pranešimo medžiaga*. Kaunas: Kauno technologijos universitetas, 2007, pp. 202-206.

[30] Rima A.; Šmaižys A.; Vasilecas O. Verslo taisyklių panaudojimas duomenų analizės metamodelių transformacijų pagrindu. *10-osios Lietuvos jaunųjų mokslininkų konferencijos „Mokslas – Lietuvos ateitis,* 2007.

[31] S. Staab, H. P. Schnurr. Knowledge and Business Processes: Approaching an Integration. *Knowledge Management and Organizational Memories*. Kluwer, 2002, [žiūrėta] www.aifb.uni-karlsruhe.de/~sst/Research/Publications/staabschnurr-om99.pdf

[32] The Object Management Group. Semantics of Business Vocabulary and Business Rules Specification. *Interim Convenience Document,* 2006.

[33] The Rule Markup Initiative. [žiūrėta] http://www.ruleml.org/

[34] T. Morgan. Business Rules and Information Systems:Aligning IT with Business Goals. *Addison Wesley Pub Co*, 2002.

[35] T. ŽOBAKAS. Biznio taisyklų modeliavimas, informacinės technologijos. Kaunas: *Technologija*, 1997, p. 61 -67

PRIEDAI

1 priedas. Straipsnis, publikuotas KTU mokslinėje konferencijoje "Informacinės technologijos'2007"

VERSLO TAISYKLIŲ RINKINIO DARNOS UŽTIKRINIMAS LOGINIO IŠVEDIMO MAŠINA

Robertas Strigūnas, Aidas Šmaižys, Olegas Vasilecas
Klaipėdos universitetas, H. Manto g. 84, Klaipėda

Šiandienos įmonės susiduria su greitai besikeičiančia verslo aplinka ir negali laikytis pastovaus ilgalaikio veiklos modelio. Įmonės veikla ir jos veiklos modelis turi būti dinamiški. Viena didžiausių problemų verslo taisyklių panaudojimui yra jų kiekis ir sunkiai valdomos tarpusavio sąveikos. Taisyklėms veikiant kartu galimi įvairūs jų sąveikos konfliktai. Taisyklių potencialių konfliktų aptikimui ir logiškumui bei nuoseklumui patikrinti galima naudoti logika pagrįstus mechanizmus, tokius kaip išvedimas. Darbe atlikta susijusių darbų apžvalga, pasiūlytas metodas, leidžiantis atvaizduotas XML verslo taisykles transformuoti į predikatus bei komponuoti į verslo taisyklių rinkinius, kuriuos būtų galima panaudoti OLAP sistemoje.

Įvadas

Verslo taisyklių panaudojimo informacinių sistemų kūrimui tema šiuo metu yra labai populiari ir plačiai nagrinėjama moksliniuose straipsniuose. Šiandienos sparčiai besivystantis verslas reikalauja lanksčių verslo valdymo sistemų, kurios sugebėtų vystytis sparta, atitinkančia veiklos pokyčius. Organizacija, kurdama verslo valdymo sistemą, siekia įgyti konkurencinį pranašumą, sumažinti veiklos kaštus, pagerinti procesų valdymą. Tačiau ne mažiau svarbu suprasti, kad nauja informacinė sistema savaime esamų valdymo problemų neišspręs. Reikalinga ir pačių procesų reinžinerija. Šiuolaikinės įmonės veiklos esmine komponente yra informacinė technologija, vartotojo požiūriu vadinama kompiuterizuota informacine sistema [2, 3].

Informacinių sistemų kūrimo metodams iki šiol būdingas „*išorinis požiūris"* į organizacijos veiklos procesą – informacinės sistemos kūrimas pradedamas nuo vartotojo poreikių ar analitiko pastebėjimų, formuojama informacinės sistemos specifikacija taip, „kaip reikia" vartotojui ir projektuotojui. Informacinių sistemų inžinerijai būtinas „*vidinis požiūris"* į veiklos procesus, grindžiamas priežastiniais veiklos elementų ryšiais – vidine veiklos (vadybos ir produkto gamybos) procesų logika. Taip pat reikia formalizuoti veiklos modelius, skirtus kompiuterizuotam informacinės sistemos kūrimui, kurie aprašo priežastinius (esminius, dėl vidinių veiklos savybių egzistuojančius) veiklos elementų ryšius. Šioms problemoms spręsti buvo sukurtas iš esmės naujas požiūris į sistemos kūrimą – verslo taisyklių požiūris [1].

Efektyvus verslo taisyklių panaudojimas informacinių sistemų kūrimui leidžia ne tik paspartinti patį informacinių sistemų kūrimo procesą, bet ir išlaikyti taisyklių verslo sistemoje ryšį su informacinės sistemos modeliu kūrimo metu, taisyklių transformavimo procesų pasėkoje, atsiradusiais naujais tiek informacinės sistemos, tiek programų sistemos objektais ir naujomis taisyklėmis. Verslo taisyklių požiūris teigia, kad veiklos vadovas ar atstovas, priimantis strateginius sprendimu ir žinantis bendrą organizacijos veiklos koncepciją, turi pateikti ir turėti galimybę keisti išskirtas veiklos taisykles sukurtoje sistemoje [3, 14].

Kad verslo taisykles būtų lengva keisti, reikia, kad jos būtų logiškai atskirtos nuo duomenų bei programos struktūros. Dažniausiai jos įsimenamos taisyklių repozitoriuje, kuris yra priemonė, leidžianti peržiūrėti, pakeisti ar tvarkyti tas taisykles. Tačiau viena didžiausių problemų verslo taisyklių panaudojimui yra jų kiekis ir sunkiai nuspėjama tarpusavio sąveika. Taisyklėms veikiant kartu atsiranda įvairūs konfliktai, kai viena iš dviejų vykdomų taisyklių priima užduotį, o kita atmeta ir t.t. Šiame straipsnyje nagrinėjamos problemos, susijusios su verslo taisyklių panaudojimu duomenų analizei. Straipsnyje apžvelgta verslo taisyklių samprata bei jų galimos klasifikavimo schemos, reikalingos sudaryti verslo taisyklių rinkinius (angl.: *rulesets*). Nagrinėjami logika pagrįsti išvedimo būdai, kurie panaudoti verslo taisyklių rinkinių konfliktų pašalinimui. Suformuluotas metodas, leidžiantis XML atvaizduotas verslo taisykles transformuoti į predikatus bei komponuoti į išsamius ir neprieštaraujančius verslo taisyklių rinkinius, siekiant tuos rinkinius panaudoti OLAP duomenų analizei programų sistemoje.

Susijusių darbų apžvalga

Verslo sistemoje verslo taisyklės apibrėžiamos kaip verslo politika, praktika, aiškumas, kur yra gerai suprantamos ir vykdomos kaip organizacijos vertybių rinkinys. Verslo taisyklės suprantamos kaip pagrindinė sąlyga verslui plėtoti.

Pagal Business Rules Group [3], verslo taisyklė:

- tai nuostata, kuri apibrėžia arba apriboja tam tikrą verslo požiūrį.

- apibrėžia verslo struktūrą, kontroliuoja arba įtakoja verslo elgseną. Verslo taisyklė negali būti suskaidyta arba detalizuota į smulkesnes taisykles.

Verslo taisyklės yra pagrįstos verslo taisyklių formuluotėmis (angl. *statements*), kurios toliau grindžiamos verslo politika. Verslo taisyklės formuluotė gauta iš verslo atstovo yra nevienareikšmiška, dviprasmiška. Konkrečioje situacijoje kiekviena verslo taisyklės formuluotė galima faktiškai suskaidyti į diskrečias taisykles. Taisyklės suskaidytos į elementarias formas vadinamos atominėmis. Verslo taisyklė neturi valdymo operatorių, kurie dažniausiai sutinkami programose kaip pranešimai, duomenų bazės atnaujinimai (angl.: *updates*), sekos. Atominė verslo taisyklė, užrašyta deklaratyvia forma, naudojant natūralią kalbą, kad verslo atstovai galėtų lengvai suprasti, nėra dviprasmiška.

Verslo įmonės apima tūkstančius verslo taisyklių kombinacijų, kurios veikia operaciniame lygmenyje. Verslo taisyklės apibrėžia ir kontroliuoja produkto ir paslaugų gyvavimo ciklą. Mums svarbesnis požiūris iš informacinių sistemų perspektyvos, kur verslo taisyklės suprantamos kaip nesudėtingas teiginys, tikrinantis duomenų teisingumą, atliekantis būtinus paskaičiavimus ar valdantis duomenų pavaizdavimą [2, 3, 4]. Programuotojas sukuria šių taisyklių rinkinį, aprašantį įstaigos veiklą. Toks principas labai priimtinas, kai greitai keičiasi veiklos sąlygos, - juk užtenka pakeisti tik taisykles ir nereikia liesti ir keisti pačios DB sistemos (skirtingai nuo procedūrinio programavimo) [4].

Atitinkamai, verslo taisyklė išreiškia specifinius apribojimus duomenų kūrimui, atnaujinimui, pašalinimui informacinėje sistemoje [3]. Taisyklės nėra nei procesas, nei procedūra, todėl neturėtų būti nė vieno iš jų sudėtinė dalis. Taisyklės veikia procesus ir procedūras. Tiesiogiai susijusiose veiklos srityse gali būti panaudoti panašūs verslo taisyklių rinkiniai.

Taisyklės gali būti susistemintos pagal ECA techniką aktyviose duomenų bazėse, kaip ryšys tarp įvykių, būsenų ir veiksmų. Buvo pastebėta, kad dvi svarbios ir pageidaujamos aktyvių taisyklių elgesio savybės yra užbaigimas (*angl. termination*) ir vientisumas (*angl. confluence*). Taisyklių rinkinys garantuotai užbaigiamas, kai kiekvienai duomenų bazės būsenai taisyklių apdorojimas negali tęstis be galo (taisyklės negali aktyvuoti viena kitą neribotą laiką). Taisyklių rinkinys yra vientisas, jeigu kiekvienai duomenų bazės būsenai, galutinė būsena po taisyklių apdorojimo yra nepriklausoma nuo to, kokia eilės tvarka taisyklės buvo įvykdytos [10].

Kai kalbama apie verslo taisyklę (angl. *business rule*) dažniausia turime omenyje taisyklių rinkinį (angl.: *ruleset*) su tam tikra jų hierarchija ir prioritetais. Dėl jų įvairumo nėra ir negali būti vieno visiems atvejams tinkamo taisyklių šablono ar jų vaizdavimo formos [12].

Toliau pateikiamos detaliausiai išnagrinėtos ir labiausiai paplitusios verslo taisyklių klasifikavimo schemos, reikalingos taisyklių rinkinių sudarymui (1 lentelė).

1 lentelė. Verslo taisyklių klasifikavimas.

Šaltinis	**Taksonomija**
Guide Business Rules Project	Struktūriniai teiginiai (terminai, faktai) Veiksmo teiginiai (apribojimai, sąlygos) Išvestis (matematinis skaičiavimas, loginė išvada)
Ron Ross, Database Research Group	Terminai (veiklos sąvokos, bendros sąvokos) Faktai (asociacija tarp dviejų ar daugiau terminų) Taisyklės
Barbara von Halle	Terminai; Faktai; Apribojimai; Kilmė; Išvados
Usoft Corporation	Apribojimai; Dedukcijos; Elgsenos; Atvaizdavimo
Vision Software	Validavimo; Kilmės; Sąryšio; Sąlygos veiksmo

Apibendrinant galima teigti, kad verslo taisyklių klasifikavimo modelio pasirinkimas pirmiausia priklauso nuo tikslo, kuriam tarnaus.

Viena didžiausių problemų verslo taisyklių panaudojimui yra jų kiekis ir sunkiai nuspėjamos tarpusavio sąveikos. Vienas iš galimų sprendimo būdų, tai perdavimo algoritmas (angl.: *propagation algorithm*), skirtas statinių ECA taisyklių analizei. Analizės technika yra paremta bendro pritaikymo algoritmu, kuris yra naudojamas nuspręsti, kada vienos taisyklės veiksmas gali paveikti kitos taisyklės būseną ir nuspręsti, kada taisyklių veiksmai persijungia (angl:. *commute*). Algoritmas perduoda vienos taisyklės veiksmus per kitos taisyklės būseną arba veiksmą, kad būtų galima nuspręsti, kaip veiksmas gali paveikti būseną arba kitą veiksmą. Perdavimo algoritmas yra naudingas analizuojant taisyklių užbaigimą, nes gali nuspręsti, kada viena taisyklė gali aktyvuoti kitą taisyklę. Taip pat jis yra naudingas analizuojant vientisumą, nes gali nuspręsti, kada dviejų taisyklių vykdymo eilės tvarka turi didelę svarbą [7].

Kitas problemos sprendimo būdas galėtų būti paremtas verslo taisyklių prioritetų ir jų vykdymo strategijos nustatymu. Tokiu atveju, atsiradus konfliktams, žemesnį rangą turinti taisyklė galėtų būti tiesiog pašalinama. Dažnai minima sprendimo galimybė yra taisyklių ir su jomis susijusių procesų hierarchizacija, kai žemesnio lygmens procesai naudojantys taisykles esančias aukštesniame lygmenyje tampa tų procesų subprocesais. Kadangi verslo procesai gali

būti specifikuojami verslo taisyklėmis, o verslo taisyklės gali būti detalizuojamos procesais, įvedus subprocesus taisyklės tuo pačiu gali įgyti tam tikrą hierarchinę struktūrą [11].

Dažniausiai taisyklių validavimui yra naudojama speciali simuliacija, kai tiesiog stebima ar esant tam tikroms sąlygoms sistema reaguoja adekvačiai.

Taisyklių konfliktų aptikimui ir logiškumui bei nuoseklumui patikrinti galima naudoti logika pagrįstus mechanizmus, tokius kaip rezoliucija arba išvedimas, panaudojant modus ponens, atbulinį išvedimą (angl.: *backward chaining*), tiesioginį išvedimą (angl.: *forward chaining*). Šie metodai dažniausia realizuojami išvedimo mašinose (angl.: *inference engine*), dirbančiose pagal Rete algoritmą arba specialiose ekspertinėse sistemose. Algoritmai, naudojami įvertinti taisyklės elgseną duotoje situacijoje vadinami išvados darymo (angl.: *inference*) arba verslo taisyklių mašina (angl.: *business rule engine*). Esmė ta, kai taisyklių saugykla yra išanalizuota, išvedimo mašina paima įėjimo duomenys ir tada patikrina taisykles, esančias taisyklių repozitoriuje, kad nustatyti kas jau yra padaryta. Skirtingų tipu išvados darymo mašinos dirba geriau su skirtingomis sprendimų darymo užduotimis. Išvedimo mašina nežino nieko apie taisykles, ji tik įgyvendina sistemingą taisyklių paiešką. Logiškai taisyklių tvarka nedaro jokios įtakos veikimui, ir taisyklių kiekis nėra svarbus. Dauguma atveju, visos taisyklės nebus reikalingos tikslui pasiekti ar išvadai gauti, bet jų buvimas nedaro jokios įtakos reikalingai išvadai gauti. Taisyklių atskirimas nuo išvedimo mašinos, kuri ieško išvadų, kurios patenkina tikslą, įgalina sistemos efektyvumą saugoti ir atlikti taisyklių paiešką. Keičiant tikslą, gaunami skirtingi rezultatai [5].

Šiuo metu dažniausiai naudojami tiesioginio ir atbulinio išvedimo metodai. Tiesioginis išvedimas prasideda su jau turimais duomenimis ir naudoja išvadų darymo taisykles, kad gauti daugiau duomenų kol tikslas yra pasiektas. Tiesioginio išvedimo samprotavimo procesas tęsiasi tol, kol pasiekiamas tikslas arba kol nebenustatoma daugiau naujų faktų (procesas taip pat gali tęstis amžinai). *Faktai* yra vaizduojami ir saugomi darbinėje atmintyje (angl. *working memory*), kuri yra pastoviai atnaujinama. *Taisyklės* pateikia galimą veiksmą, specifikuotos būsenos. Duomenų valdymo (angl.: *data driven*) samprotavimo procese taisyklės sąlygos (angl.: conditions) yra lyginamos su faktais iš faktų saugyklos. Jei sąlygos yra patenkinamos, tada veiksmas yra vykdomas. Atbulinis išvedimas yra tikslo valdymo samprotavimo procesas. Procesas prasideda nuo tikslo kuris turėtų būti pasiektas, naujos hipotezės (potikslis) yra gaunamos naudojant taisykles, kurios yra paleidžiamos jei jų išvada atitinka tikslą arba potikslį. Jei yra atitikimas, tai taisyklės būsenos (angl.: *conditions*) tampa naujomis hipotezėmis. Procesas pabaigiamas kai hipotezės yra jau žinomi faktai arba kai naujų hipotezių nerandama.

Apibendrinant tiesioginio ir atbulinio išvedimo metodus galima teigti, kad tiesioginis išvedimas – duomenų valdymo procesas, išvados gaunamos iš faktų. Naudojamas kai norima nustatyti pasikeitimų prasmę duomenų reikšmėms arba kai norima forma pagrįstos sąveikos, bendravimo (angl.: *form - based type of interaction*). Atbulinis išvedimas - tikslo siekiantis procesas, samprotavimas nuo hipotezių iki faktų (faktų radimas), taisyklės naudojamos pagrįsti tikslo teisingumą arba ne. Naudojamas kai norima nustatyti, kuri reikšmė yra sprendimas iš viso rinkinio arba kai norima klausimas – atsakymas tipo bendravimo (angl.: *question-and-answer dialog type of interaction*) [6].

Algoritmas, naudojamas aktyvuoti taisykles yra toks pats ir tiesioginiame, ir atbuliniame išvedime, ir turi tris žingsnius: identifikacija: kokios taisyklės gali būti paleistos ir vykdomos (angl.: *fired*), taisyklės parinkimas ir vykdymas.

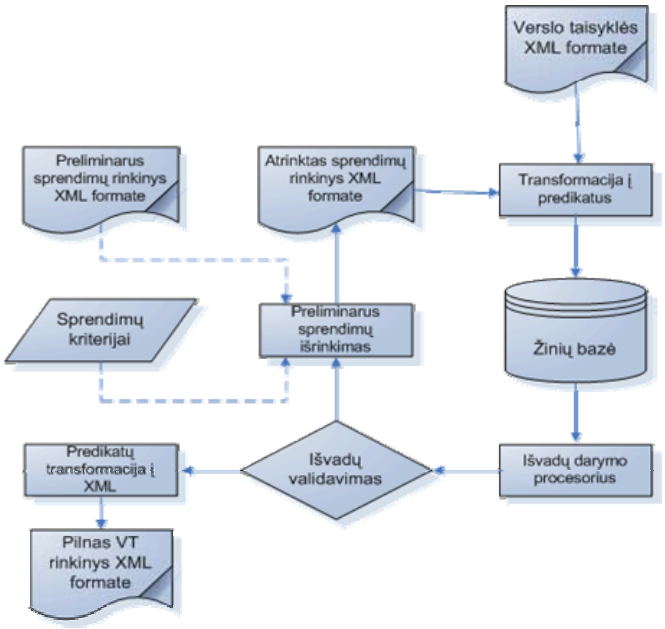
Dažnai straipsniuose sutinkamas ir Rete algoritmas. Tai išvados darymo mechanizmas, kuris suriša pradžioje nesusijusias taisykles į loginį tinklą. Rete tiesioginio paieškos algoritmo esmė, kad iš faktų ir jų derinių yra sukuriamas grafas – medis. Jame faktai išdėstomi ir sujungiami taip, kad pagal taisyklių prielaidų šablonus, būtų vykdoma kuo efektyvesnė peržiūra/paieška, t.y. medis atsimena surastus faktų derinius ir taip sutaupo paieškos laiko (kai tų derinių dažnai ieškoma). Atsiradus naujam faktui, medis minimaliai reorganizuojamas [9].

Yra labai daug darbų susijusių su taisyklių varikliu ir verslo logika. Produktai kaip „Jrules“ ir „QuickRules“ turi didelę paklausą, integruojant taisyklių variklius su taikomųjų programų sistemomis.

Šie varikliai yra parašyti Java kalba ir integruoti kartu su taikomųjų programų serveriu, kuriame išdėstytos nepriklausomos sesijos.. Tai leidžia taisyklių varikliui betarpiškai bendrauti su taikomųjų programų serveriu. Tačiau tai turi savo trūkumų. Taisyklių variklio efektyvumas yra sumažinamas ir yra priklausomas nuo taikomųjų programų serverio. Dauguma taikomųjų programų serverių apibrėžia panašius gijų prioritetus visoms sesijoms. Kol šios sesijos neturi papildomų privilegijų, jos turėtų charakteristikos trūkumų, jeigu jos siektų tikslo kaip atskiri serveriai. Vartotojui naudojantis sąsaja, šios taikomosios programos naudojamos su natūralios kalbos apdorojimo supratimu, kurios įgalina dalykinės srities ekspertą ar sistemos kūrėją pradėti naudoti naujas taisykles be sistemos. Tačiau šie produktai nesuteikia galimybės vartotojui pačiam kurti ir naudoti taisykles. Taikomosiose programose kur vartotojams reikalinga, kad taisyklės būtų keičiamos dinamiškai, būtina, kad jie bendrautų su dalykinės srities ekspertu ir atliktų visus darbus per jį. Atsižvelgiant į taikomąsias programas internete, kur gali būti šimtai potencialių vartotojų, tai būtų neįgyvendinama. Taigi kompleksinės sąsajos buvimas sukelia sunkumų projektuojant paprastą vartotojo sąsają [5].

VT rinkinio darnos užtikrinimo loginio išvedimo mašina metodikos formulavimas

Verslo taisyklių rinkinio darnos užtikrinimo problemos sprendimas galėtų būti realizuotas žemiau suformuluotu metodu, naudojant verslo taisyklių transformaciją ir išvedimo logiką (1 pav.). Verslo taisyklių rinkinio darnos užtikrinimas susideda iš dviejų procesų: išvedimo ir preliminaraus sprendimo išrinkimo.



1 pav. Verslo taisyklių rinkinio darnos užtikrinimo loginio išvedimo mašina metodo schema.

Pasiūlytame metode verslo taisykles vaizduojame XML forma, sujungtas į specialius taisyklių rinkinius kartu su faktais, gautais iš esamų verslo duomenų (žinių bazė). Faktai saugojami atskirame XML formato faile. Duotasis verslo taisyklių rinkinys transformuojama į predikatus ir užkraunamas į išvedimo mašiną. Taip patikrintas rinkinio funkcionalumas, ko pasėkoje būtų gaunamos naujos išvestinės taisyklės bei patikrinamas atskirų taisyklių vientisumas, t.y. ar nėra tarp jų prieštaravimų. Taisyklių atskirimas nuo išvedimo mašinos, kuri ieško išvadų, kurios patenkina tikslą, įgalina sistemos efektyvumą saugoti ir atlikti taisyklių paiešką. Keičiant tikslą, gaunami skirtingi rezultatai.

Eksperimento uždavinys yra verslo taisykles pavaizduotas XML formatu transformuoti į predikatų logikos sakinius ir patikrinti išvedimo mašinoje. Eksperimento metu panaudojome nemokamą, kompanijos *Sun Microsystems* Java pagrindu sukurtą *„Jess“* išvedimo mašiną, kuri veikia Rete algoritmo pagrindu.

Pirmiausia pagal tam tikrą sugeneruotą SRML DTD dokumentą ir XML pavaizduotas taisykles sukuriama XSLT schema, reikalinga taisyklių transformacijai į predikatų logikos sakinius. XSLT schemos generavimui naudojame Altova® StyleSheet Designer programą. Visos operacijos atliekamos vizualioje grafinėje aplinkoje, o schemos dinaminės dalies generavimui naudojamas tas pats DTD dokumentas, kaip ir XML byloje. Tai leidžia išvengti klaidų ir supaprastina laukų įterpimo procesą. Pagal XSLT schemą sugeneruojami predikatų logikos sakiniai, kurie užkraunami į išvedimo mašiną.

Išvedimo mašina pradžioje suriša nesusijusias taisykles į loginį tinklą. Rete tiesioginio paieškos algoritmo esmė ta, kad iš faktų ir jų derinių yra sukuriamas grafas – medis. Jame faktai išdėstomi ir sujungiami taip, kad pagal taisyklių prielaidų šablonus, būtų vykdoma kuo efektyvesnė peržiūra/paieška, t.y. medis atsimena surastus faktų derinius ir taip sutaupo paieškos laiko (kai tų derinių dažnai ieškoma). Atsiradus naujam faktui, medis minimaliai reorganizuojamas.

Daugelį atvejų atlikus loginius išvedimus, turime išbaigtus sprendimus bei verslo taisyklių rinkinį, išreikštą predikatais. Išvedimo mašina gautus išsamius rinkinius būtų galima panaudoti OLAP duomenų analizėje. Toliau šis taisyklių rinkinys gali būti konvertuojamas atgal į XML formatą ir XSLT schemos pagalba transformuojamas į MDX užklausas.

OLAP technologija leidžia greitai ir efektyviai peržiūrėti ir analizuoti didelius informacijos kiekius, atlikti prognozavimą, kur duomenų atvaizdavimui naudojamas daugiamatis duomenų modelis, atspindintis realų kompanijos vaizdą, lengvai suprantamą vartotojui.

Taip pat išvedimo mašiną galima naudoti ir taisyklių sistemos darnos testavimui. Tam reiktų sukurti tam tikrus faktus, užrašyti juos predikatais ir fiksuoti rezultatus, kurie turi gautis. Įvedus naują taisyklę į taisyklių rinkinį reiktų užkrauti visas taisyklių rinkinio taisykles kartu su naująja ir patikrinti ar gaunamas rezultatas atitinka testuojamąjį. Toks būdas bent jau iš dalies padėtų išvengti akivaizdžių klaidų.

Išvados

Atlikus eksperimentą, pasiūlytas metodas leido verslo taisykles atvaizduotas XML forma, sujungti į nepriklausomus, tačiau susietus taisyklių rinkinius kartu su faktais, gautais iš žinių bazės, panaudojus loginį išvedimą. Eksperimento metu duotas verslo taisyklių rinkinys transformuotas į predikatus, jis užkrautas į išvedimo mašiną ir patikrintas jo funkcionalumas. Remiantis eksperimento rezultatais galima teigti, kad pasiūlytas metodas daugeliu atveju leidžia užtikrinti verslo taisyklių rinkinio darną.

Išvedimo mašiną galima naudoti ir taisyklių sistemos darnos testavimui. Tam reiktų sukurti tam tikrus faktus, užrašyti juos predikatais ir fiksuoti rezultatus, kurie turi gautis. Prie šio uždavinio sugrįšime tolimesniuose darbuose.

Literatūros sąrašas

[36] **Barbara von Hale.** Business Rules Applied: Building Better Systems Using the Business Rules Approach. *John Wiley & Sons, Inc., New York*, 2001.

[37] **Business Rules Group**. Defining Business Rules: What is a Business Rule?, 2004, [žiūrėta 2005-01-17], http://www.BusinessRulesGroup.org

[38] **D. Rosca, S. Greenspan, C. Wild.** Enterprise Modeling and Decision-Support for Automating the Business Rules Lifecycle, Automated Software Engineering, v. 9, n. 4, 2002, 361 psl.

[39] **I**. **Graham.** Service oriented business rules management systems. *Trireme*, 2006.

[40] **J. Sinur.** The Business Rule Engine. Magic Quadrant. 2003

[41] **M. Bajec, M. Krisper.** A Methodology and Tool Support for Managing Business Rules in Organisations. *Information Systems* 30, *no. 6,* 423-443 psl.

[42] **N. Prakash, S. Srivastava**. Engineering Methods For Schema Transformation: Application to XML. IFIP TC8/WG8.1 Working Conference on Engineering Information Systems in the Internet Context, *Kanazawa, Japan*. *Kluwer Academic publishers,* 2002, 153 - 175 psl.

[43] **O. Vasilecas, A. Smaizys.** The framework: an approach to support business rule based data analysis. *In O. Vasilecas et al. (eds.), Proc. of 7th International IEEE Baltic Conference on Databases and Information Systems,* July 3, 2006, 141 – 147 psl.

[44] **O. Vasilecas, A. Smaizys.** Verslo taisyklių panaudojimas duomenų analizei ir informacijos pateikimui. *Informacinės technologijos 2005 (Information technologies 2005), Kaunas: Technologija*, 2005, 655-663 psl.

[45] **O. Vasilecas, A. Smaizys.** Business rule based knowledge integrated intelligent system framework. *Informacijos mokslai, Vilniaus universiteto leidykla*, 2005, 34 tomas, 195-200 psl.

[46] **P. Harmon, C. Hall**. Intelligent Software Systems Development: An IS Manager's Guide. *John Wiley*, 1993.

[47] **R. Ross, K. Healy.** Organizing Business Plans. The Standard Model for Business Rule Motivation. *Business Rules Group*. 2000.

[48] **R. Ross.** Principles of the Business Rule Approach. *Addison Wesley Professional*, 2003.

[49] **The Object Management Group.** Semantics of Business Vocabulary and Business Rules Specification. *Interim Convenience Document,* 2006.

Inference engine driven business rule set consistency check

Contemporary companies contend with a rapidly changing environment and can not maintain long term operational models. A company's business process and process model has to be dynamic. One of the greatest problems using business rules are their quantity and difficulty in foreseeing of their interplay. Conflicts become apparent when rules are used in conjunction. A logic based derivation mechanism can be used for detection of rule incongruence and to analyze their logical and chronological sequence. This paper presents a review of related works and method for transformation of the business rules represented in XML into predicate set for use in inference engine and business rules driven rule set consistency check and use of such a complete rule set in data analysis OLAP system.

2 priedas. RuleML XSD schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.ruleml.org/0.86/xsd"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ruleml.org/0.86/xsd">
    <xs:attributeGroup name="rulebase.attlist">
        <xs:attributeGroup ref="direction.attrib"/>
    </xs:attributeGroup>
    <xs:group name="rulebase.content">
        <xs:choice>
            <xs:sequence>
                <xs:element ref="_rbaselab"/>
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                    <xs:element ref="fact"/>
                    <xs:element ref="query"/>
                    <xs:element ref="imp"/>
                </xs:choice>
            </xs:sequence>
            <xs:sequence minOccurs="0">
                <xs:choice maxOccurs="unbounded">
                    <xs:element ref="fact"/>
                    <xs:element ref="query"/>
                    <xs:element ref="imp"/>
                </xs:choice>
                <xs:element ref="_rbaselab" minOccurs="0"/>
            </xs:sequence>
        </xs:choice>
    </xs:group>
    <xs:complexType name="rulebase.type">
        <xs:group ref="rulebase.content" minOccurs="0"/>
        <xs:attributeGroup ref="rulebase.attlist"/>
    </xs:complexType>
    <xs:element name="rulebase" type="rulebase.type"/>
    <xs:attributeGroup name="direction.attrib">
        <xs:attribute name="direction" use="optional" default="bidirectional">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="forward"/>
                    <xs:enumeration value="backward"/>
                    <xs:enumeration value="bidirectional"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:attributeGroup>
    <xs:attributeGroup name="fact.attlist"/>
    <xs:group name="fact.content">
        <xs:choice>
            <xs:sequence>
                <xs:element ref="_rlab"/>
                <xs:element ref="_head"/>
```

```
        </xs:sequence>
        <xs:sequence>
            <xs:element ref="_head"/>
            <xs:element ref="_rlab" minOccurs="0"/>
        </xs:sequence>
    </xs:choice>
</xs:group>
<xs:complexType name="fact.type">
    <xs:group ref="fact.content"/>
    <xs:attributeGroup ref="fact.attlist"/>
</xs:complexType>
<xs:element name="fact" type="fact.type"/>
<xs:attributeGroup name="imp.attlist"/>
<xs:group name="imp.content">
    <xs:choice>
        <xs:sequence>
            <xs:element ref="_rlab"/>
            <xs:choice>
                <xs:sequence>
                    <xs:element ref="_head"/>
                    <xs:element ref="_body"/>
                </xs:sequence>
                <xs:sequence>
                    <xs:element ref="_body"/>
                    <xs:element ref="_head"/>
                </xs:sequence>
            </xs:choice>
        </xs:sequence>
        <xs:sequence>
            <xs:element ref="_head"/>
            <xs:choice>
                <xs:sequence>
                    <xs:element ref="_rlab"/>
                    <xs:element ref="_body"/>
                </xs:sequence>
                <xs:sequence>
                    <xs:element ref="_body"/>
                    <xs:element ref="_rlab" minOccurs="0"/>
                </xs:sequence>
            </xs:choice>
        </xs:sequence>
        <xs:sequence>
            <xs:element ref="_body"/>
            <xs:choice>
                <xs:sequence>
                    <xs:element ref="_rlab"/>
                    <xs:element ref="_head"/>
                </xs:sequence>
                <xs:sequence>
                    <xs:element ref="_head"/>
                    <xs:element ref="_rlab" minOccurs="0"/>
                </xs:sequence>
```

```
            </xs:choice>
        </xs:sequence>
    </xs:choice>
</xs:group>
<xs:complexType name="imp.type">
    <xs:group ref="imp.content"/>
    <xs:attributeGroup ref="imp.attlist"/>
</xs:complexType>
<xs:element name="imp" type="imp.type"/>
<xs:attributeGroup name="query.attlist"/>
<xs:group name="query.content">
    <xs:choice>
        <xs:sequence>
            <xs:element ref="_rlab"/>
            <xs:element ref="_body"/>
        </xs:sequence>
        <xs:sequence>
            <xs:element ref="_body"/>
            <xs:element ref="_rlab" minOccurs="0"/>
        </xs:sequence>
    </xs:choice>
</xs:group>
<xs:complexType name="query.type" mixed="true">
    <xs:group ref="query.content"/>
    <xs:attributeGroup ref="query.attlist"/>
</xs:complexType>
<xs:element name="query" type="query.type"/>
<xs:attributeGroup name="_rbaselab.attlist"/>
<xs:group name="_rbaselab.content">
    <xs:choice>
        <xs:element ref="ind"/>
    </xs:choice>
</xs:group>
<xs:complexType name="_rbaselab.type">
    <xs:group ref="_rbaselab.content"/>
    <xs:attributeGroup ref="_rbaselab.attlist"/>
</xs:complexType>
<xs:element name="_rbaselab" type="_rbaselab.type"/>
<xs:attributeGroup name="_rlab.attlist"/>
<xs:group name="_rlab.content">
    <xs:choice>
        <xs:element ref="ind"/>
    </xs:choice>
</xs:group>
<xs:complexType name="_rlab.type">
    <xs:group ref="_rlab.content"/>
    <xs:attributeGroup ref="_rlab.attlist"/>
</xs:complexType>
<xs:element name="_rlab" type="_rlab.type"/>
<xs:attributeGroup name="_head.attlist"/>
<xs:group name="_head.content">
    <xs:choice>
```

```
        <xs:element ref="atom"/>
    </xs:choice>
</xs:group>
<xs:complexType name="_head.type">
    <xs:group ref="_head.content"/>
    <xs:attributeGroup ref="_head.attlist"/>
</xs:complexType>
<xs:element name="_head" type="_head.type"/>
<xs:attributeGroup name="_body.attlist"/>
<xs:group name="_body.content">
    <xs:choice>
        <xs:element ref="atom"/>
        <xs:element ref="and"/>
        <xs:element ref="or"/>
    </xs:choice>
</xs:group>
<xs:complexType name="_body.type">
    <xs:group ref="_body.content"/>
    <xs:attributeGroup ref="_body.attlist"/>
</xs:complexType>
<xs:element name="_body" type="_body.type"/>
<xs:attributeGroup name="and.attlist"/>
<xs:group name="and.content">
    <xs:choice>
        <xs:element ref="atom"/>
        <xs:element ref="or"/>
    </xs:choice>
</xs:group>
<xs:complexType name="and.type">
    <xs:group ref="and.content" minOccurs="0" maxOccurs="unbounded"/>
    <xs:attributeGroup ref="and.attlist"/>
</xs:complexType>
<xs:element name="and" type="and.type"/>
<xs:attributeGroup name="or.attlist"/>
<xs:group name="or.content">
    <xs:choice>
        <xs:element ref="atom"/>
        <xs:element ref="and"/>
    </xs:choice>
</xs:group>
<xs:complexType name="or.type">
    <xs:group ref="or.content" minOccurs="0" maxOccurs="unbounded"/>
    <xs:attributeGroup ref="or.attlist"/>
</xs:complexType>
<xs:element name="or" type="or.type"/>
<xs:group name="atom.extend">
    <xs:sequence>
        <xs:choice maxOccurs="unbounded">
            <xs:element ref="ind"/>
            <xs:element ref="var"/>
        </xs:choice>
    </xs:sequence>
```

```
</xs:group>
<xs:attributeGroup name="atom.attlist"/>
<xs:group name="atom.content">
   <xs:choice>
      <xs:sequence>
         <xs:element ref="_opr"/>
         <xs:element ref="_slot" minOccurs="0" maxOccurs="unbounded"/>
         <xs:sequence minOccurs="0">
            <xs:group ref="atom.extend"/>
            <xs:element ref="_slot" minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:sequence>
      <xs:sequence>
         <xs:choice>
            <xs:sequence>
               <xs:element ref="_slot" maxOccurs="unbounded"/>
               <xs:sequence minOccurs="0">
                  <xs:group ref="atom.extend"/>
                  <xs:element ref="_slot" minOccurs="0" maxOccurs="unbounded"/>
               </xs:sequence>
            </xs:sequence>
            <xs:sequence>
               <xs:group ref="atom.extend"/>
               <xs:element ref="_slot" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
         </xs:choice>
         <xs:element ref="_opr"/>
      </xs:sequence>
   </xs:choice>
</xs:group>
<xs:complexType name="atom.type">
   <xs:group ref="atom.content"/>
   <xs:attributeGroup ref="atom.attlist"/>
</xs:complexType>
<xs:element name="atom" type="atom.type"/>
<xs:attributeGroup name="_opr.attlist"/>
<xs:group name="_opr.content">
   <xs:sequence>
      <xs:element ref="rel"/>
   </xs:sequence>
</xs:group>
<xs:complexType name="_opr.type">
   <xs:group ref="_opr.content"/>
   <xs:attributeGroup ref="_opr.attlist"/>
</xs:complexType>
<xs:element name="_opr" type="_opr.type"/>
<xs:attributeGroup name="rel.attlist"/>
<xs:group name="rel.content">
   <xs:sequence/>
</xs:group>
<xs:complexType name="rel.type" mixed="true">
   <xs:group ref="rel.content"/>
```

```
    <xs:attributeGroup ref="rel.attlist"/>
</xs:complexType>
<xs:element name="rel" type="rel.type"/>
<xs:attributeGroup name="_slot.attlist">
    <xs:attributeGroup ref="name.attrib"/>
    <xs:attributeGroup ref="card.attrib"/>
    <xs:attributeGroup ref="weight.attrib"/>
</xs:attributeGroup>
<xs:group name="_slot.content">
    <xs:choice>
        <xs:element ref="ind"/>
        <xs:element ref="var"/>
    </xs:choice>
</xs:group>
<xs:complexType name="_slot.type">
    <xs:group ref="_slot.content"/>
    <xs:attributeGroup ref="_slot.attlist"/>
</xs:complexType>
<xs:element name="_slot" type="_slot.type"/>
<xs:attributeGroup name="name.attrib">
    <xs:attribute name="name" type="xs:string" use="required"/>
</xs:attributeGroup>
<xs:attributeGroup name="card.attrib">
    <xs:attribute name="card" type="xs:nonNegativeInteger" use="optional"/>
</xs:attributeGroup>
<xs:attributeGroup name="weight.attrib">
    <xs:attribute name="weight" use="optional">
        <xs:simpleType>
            <xs:restriction base="xs:decimal">
                <xs:minInclusive value="0"/>
                <xs:maxInclusive value="1"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:attributeGroup>
<xs:attributeGroup name="type.attrib">
    <xs:attribute name="type" type="xs:anyURI" use="optional"/>
</xs:attributeGroup>
<xs:attributeGroup name="ind.attlist">
    <xs:attributeGroup ref="type.attrib"/>
</xs:attributeGroup>
<xs:group name="ind.content">
    <xs:sequence/>
</xs:group>
<xs:complexType name="ind.type" mixed="true">
    <xs:group ref="ind.content"/>
    <xs:attributeGroup ref="ind.attlist"/>
</xs:complexType>
<xs:element name="ind" type="ind.type"/>
<xs:attributeGroup name="var.attlist">
    <xs:attributeGroup ref="type.attrib"/>
</xs:attributeGroup>
```

```
    <xs:group name="var.content">
        <xs:sequence/>
    </xs:group>
    <xs:complexType name="var.type" mixed="true">
        <xs:group ref="var.content"/>
        <xs:attributeGroup ref="var.attlist"/>
    </xs:complexType>
    <xs:element name="var" type="var.type"/>
</xs:schema>
```

3 priedas. Taisyklės išreikštos RuleML formatu

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rulebase SYSTEM "ruleml-0_8-datalog.dtd">
<rulebase>
   <imp>
      <_head>
         <atom>
            <_opr>
               <rel>nuolaida</rel>
            </_opr>
            <var>pirkejas</var>
            <var>preke</var>
            <ind>5.0 procentai</ind>
         </atom>
      </_head>
      <_body>
         <and>
            <atom>
               <_opr>
                  <rel>nuolatinis</rel>
               </_opr>
               <var>pirkejas</var>
            </atom>
            <atom>
               <_opr>
                  <rel>iprastine</rel>
               </_opr>
               <var>preke</var>
            </atom>
         </and>
      </_body>
   </imp>
   <imp>
      <_head>
         <atom>
            <_opr>
               <rel>nuolaida</rel>
            </_opr>
            <var>pirkejas</var>
            <var>preke</var>
            <ind>7.5 procentai</ind>
         </atom>
      </_head>
      <_body>
         <and>
            <atom>
               <_opr>
                  <rel>nuolatinis</rel>
               </_opr>
               <var>pirkejas</var>
            </atom>
```

```
            <atom>
               <_opr>
                  <rel>prabangi</rel>
               </_opr>
               <var>preke</var>
            </atom>
         </and>
      </_body>
   </imp>
   <imp>
      <_head>
         <atom>
            <_opr>
               <rel>nuolatinis</rel>
            </_opr>
            <var>pirkejas</var>
         </atom>
      </_head>
      <_body>
         <atom>
            <_opr>
               <rel>islaidos</rel>
            </_opr>
            <var>pirkejas</var>
            <ind>min 5000 euro</ind>
            <ind>paskutiniai metai</ind>
         </atom>
      </_body>
   </imp>
   <fact>
      <_head>
         <atom>
            <_opr>
               <rel>prabangi</rel>
            </_opr>
            <ind>Porsche</ind>
         </atom>
      </_head>
   </fact>
   <fact>
      <_head>
         <atom>
            <_opr>
               <rel>iprastine</rel>
            </_opr>
            <ind>Honda</ind>
         </atom>
      </_head>
   </fact>
   <fact>
      <_head>
         <atom>
```

```
                <_opr>
                    <rel>islaidos</rel>
                </_opr>
                <ind>robertas</ind>
                <ind>min 5000 euro</ind>
                <ind>paskutiniai metai</ind>
            </atom>
        </_head>
    </fact>
    <query>
        <_body>
            <atom>
                <_opr>
                    <rel>nuolaida</rel>
                </_opr>
                <var>pirkejas</var>
                <var>preke</var>
                <var>islaidos</var>
            </atom>
        </_body>
    </query>
</rulebase>
```

4 priedas. Naujai gauti faktai RuleML formate

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<RuleML xmlns="http://www.ruleml.org/0.9/xsd"
xsi:schemaLocation="http://www.ruleml.org/0.9/xsd ruleml-0_9-nafdatalog.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <!--Facts-->
 <Assert>
  <Atom>
   <op>
    <Rel>luxury</Rel>
   </op>
   <Ind>Porsche</Ind>
  </Atom>
  <Atom>
   <op>
    <Rel>regular</Rel>
   </op>
   <Ind>Honda</Ind>
  </Atom>
  <Atom>
   <op>
    <Rel>spending</Rel>
   </op>
   <Ind>Peter Miller</Ind>
   <Ind>min 5000 euro</Ind>
   <Ind>previous year</Ind>
  </Atom>
  <Atom>
   <op>
    <Rel>premium</Rel>
   </op>
   <Ind>Peter Miller</Ind>
  </Atom>
  <Atom>
   <op>
    <Rel>discount</Rel>
   </op>
   <Ind>Peter Miller</Ind>
   <Ind>Porsche</Ind>
   <Ind>7.5 percent</Ind>
  </Atom>
  <Atom>
   <op>
    <Rel>discount</Rel>
   </op>
   <Ind>Peter Miller</Ind>
   <Ind>Honda</Ind>
   <Ind>5.0 percent</Ind>
  </Atom></Assert></RuleML>
```

5 priedas. Prieštaraujančių taisyklių rinkinys RuleML

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rulebase SYSTEM "ruleml-0_8-datalog.dtd">
<rulebase>
   <imp>
      <_head>
         <atom>
            <_opr>
               <rel>nuolaida</rel>
            </_opr>
            <var>pirkejas</var>
            <var>preke</var>
            <ind>5.0 procentai</ind>
         </atom>
      </_head>
      <_body>
         <and>
            <atom>
               <_opr>
                  <rel>nuolatinis</rel>
               </_opr>
               <var>pirkejas</var>
            </atom>
            <atom>
               <_opr>
                  <rel>iprastine</rel>
               </_opr>
               <var>preke</var>
            </atom>
         </and>
      </_body>
   </imp>
   <imp>
      <_head>
         <atom>
            <_opr>
               <rel>nuolaida</rel>
            </_opr>
            <var>pirkejas</var>
            <var>preke</var>
            <ind>7.5 procentai</ind>
         </atom>
      </_head>
      <_body>
         <and>
            <atom>
               <_opr>
                  <rel>nuolatinis</rel>
               </_opr>
               <var>pirkejas</var>
            </atom>
```

```
            <atom>
               <_opr>
                  <rel>prabangi</rel>
               </_opr>
               <var>preke</var>
            </atom>
         </and>
      </_body>
   </imp>
   <imp>
      <_head>
         <atom>
            <_opr>
               <rel>nuolatinis</rel>
            </_opr>
            <var>pirkejas</var>
         </atom>
      </_head>
      <_body>
         <atom>
            <_opr>
               <rel>islaidos</rel>
            </_opr>
            <var>pirkejas</var>
            <ind>min 5000 euro</ind>
            <ind>paskutiniai metai</ind>
         </atom>
      </_body>
   </imp>
   <imp>
      <_head>
         <atom>
            <_opr>
               <rel>nuolatinis</rel>
            </_opr>
            <var>pirkejas</var>
         </atom>
      </_head>
      <_body>
         <atom>
            <_opr>
               <rel>islaidos</rel>
            </_opr>
            <var>pirkejas</var>
            <ind>min 50 euro</ind>
            <ind>paskutiniai metai</ind>
         </atom>
      </_body>
   </imp>
   <fact>
      <_head>
         <atom>
```

```
            <_opr>
               <rel>prabangi</rel>
            </_opr>
            <ind>Porsche</ind>
         </atom>
      </_head>
   </fact>
   <fact>
      <_head>
         <atom>
            <_opr>
               <rel>iprastine</rel>
            </_opr>
            <ind>Honda</ind>
         </atom>
      </_head>
   </fact>
   <fact>
      <_head>
         <atom>
            <_opr>
               <rel>islaidos</rel>
            </_opr>
            <ind>robertas</ind>
            <ind>min 5000 euro</ind>
            <ind>paskutiniai metai</ind>
         </atom>
      </_head>
   </fact>
   <query>
      <_body>
         <atom>
            <_opr>
               <rel>nuolaida</rel>
            </_opr>
            <var>pirkejas</var>
            <var>preke</var>
            <var>islaidos</var>
         </atom>
      </_body>
   </query>
</rulebase>
```

6 priedas. Kompaktinis diskas