Koncepcja narzędzi CAD/CASE do wspomagania projektowania systemów regułowych *

Antoni Ligęza, Igor Wojnicki, Grzegorz J. Nalepa
Katedra Automatyki AGH
ali@ia.agh.edu.pl, wojnicki@agh.edu.pl, gjn@agh.edu.pl

Streszczenie

Z implementacją systemów regułowych związane są dwie główne trudności. Pierwsza z nich to tzw. problem pozyskiwania wiedzy, druga natomiast dotyczy weryfikacji wiedzy. Artykuł przedstawia próbę rozwiązania tych problemów tzw: *Modularny System Tabelowy.* Modularny System Tabelowy składa się z pewnej ilości określonych tabel (modułów wiedzy). Każda tabela opisuje zestaw reguł działający w określonym kontekście. Połączenia pomiędzy tabelami określają możliwe ścieżki wnioskowania. Model Modularnego Systemu Tabelowego można zastosować dla wielu atrybutowych systemów regułowych. Stanowi on uniwersalną metodologię strukturalnej reprezentacji wiedzy. Zaproponowane w niniejszej pracy podejście bazuje na podejściu narzędzi wspomagających projektowanie (CAD/CASE) oraz graficznej i algebraicznej reprezentacji wiedzy. Pozwala to rozwiązać problem akwizycji wiedzy poprzez systematyczne projektowanie reguł z użyciem modułowej i hierarchicznej struktury, której dostarcza MTS. Co więcej proponowane rozwiązanie umożliwia weryfikację własności projektowanych reguł takich jak zupełność, spójność i determinizm.

1. Wstęp

Wiele systemów z bazą wiedzy oraz systemów przetwarzających informacje stosuje mechanizmy wnioskowania oparte na przetwarzaniu reguł. Jest to bardzo uniwersalny model wnioskowania, a sam sposób wnioskowania wydaje się być najbardziej zbliżonym do rozumowania naturalnego. Systemy regułowe dostarczają prostego, a zarazem potężnego modelu programowania. Mogą być używane w dynamicznych bazach danych, dedukcyjnych bazach danych, systemach ekspertowych, systemach

*Praca realizowana w ramach grantu KBN nr: 8 T11C 019 17.

wspomagających podejmowanie decyzji, systemach monitorowania i nadzoru, oraz systemach inteligentnego przetwarzania informacji.

Regułę wnioskowania ogólnie można przedstawić jako:

$$R_i: \quad p_1 \wedge p_2 \wedge \ldots \wedge p_n \longrightarrow h, \tag{1}$$

gdzie $p_1, p_2, \ldots, p_n$ są warunkami aktywacji (*odpalenia* reguły), a h jest wnioskiem. Zarówno warunki jak i wniosek mogą być wyrażone przy użyciu różnych języków służących do reprezentacji wiedzy. Systemy regułowe funkcjonalnie zbudowane są z reguł oraz podsystemu wnioskującego. Reguły tworzą program, zaś podsystem wnioskujący sprawuje nadzór nad jego realizacją. Więcej informacji na temat systemów regułowych oraz podsumowanie aktualnego stanu wiedzy można znaleźć w pracy [8].

Chociaż regułowy model programowania jest prosty pod względem formy, to jednak implementacja systemu regułowego jest zadaniem żmudnym i pracochłonnym. Powstają dwa główne problemy. Pierwszy z nich związany jest z trudnością w uzyskaniu precyzyjnej specyfikacji wiedzy. Drugi, to problem weryfikacji i walidacji wiedzy. Weryfikacja i walidacja sprowadza się do sprawdzenia określonych charakterystyk tworzonego systemu oraz jego poprawności. Zadanie to jest na tyle złożone, że w większości przypadków jest najbardziej kosztowną fazą wdrażania systemu.

Jak dotąd, proponowano różne metodologie oraz narzędzia usprawniające proces projektowania [5]. Najlepsze ze znanych rozwiązań to systemy (metodologie) KADS oraz CommonKADS [8]. Narzędzia te wspierają poszczególne etapy projektowania koncepcyjnego. Takie podejście umożliwia projektowanie złożonych systemów, jednak wsparcie dla fazy implementacji często ogranicza się jedynie do prostego sprawdzania poprawności składniowej. Aby zapewnić bezpieczeństwo i wydajność systemu regułowego należy przeprowadzić analizę wybranych własności utworzonych reguł [1, 3, 2, 4, 15, 17, 18]. Do istotnych własności należy zaliczyć zupełność, spójność oraz determinizm. Należy przy tym zwrócić uwagę, że analiza tych własności (po zaprojektowaniu reguł) pociąga za sobą olbrzymi nakład pracy, związany z ewentualnym wykryciem i usuwaniem błędów i braków w projekcie bazy wiedzy.

Niniejsza praca przedstawia kontynuację badań opisanych w [13]. proponując następujące najważniejsze rozszerzenia:

- wprowadzono koncepcję modułowej reprezentacji wiedzy w oparciu rekonfigurowalne moduły w formie tabel,
- zaproponowano bardziej ogólna strukturę powiązań pomiędzy modułami, tzn. w formie grafu zamiast – jak poprzednio – drzewa,
- dopuszczono bardziej złożone wartości atrybutów (w szczególności mogą nimi być zbiory, przedziały oraz elementy tworzące strukturę kraty),
- zaproponowano koncepcję definiowania poprawności projektowanej bazy wiedzy w formie logicznych ograniczeń lokalnych, nakładanych na moduły i powiązania pomiędzy nimi, a realizowanych za pomocą środków algebraicznych.

Praca dotyczy zarówno wspomagania projektowania, jak i weryfikacji systemów regułowych, które można opisać za pomocą zaproponowanej tu koncepcji *Modularnego Systemu Tabelowego (MTS)*[1]. Systemy takie to te, które funkcjonują na bazie rozszerzonych atrybutowych tablic decyzyjnych. Mogą one operować na faktach reprezentowanych jako atrybuty w relacji z wartościami. Przy czym, wartości atrybutów mogą mieć charakter nieatomiczny tzn. mogą to być również przedziały, zbiory, czy też elementy zbiorów.

Modularny System Tabelowy składa się z pewnej ilości określonych tabel (modułów wiedzy). Każda tabela opisuje zestaw reguł działający w określonym kontekście. W ramach tabeli wszystkie reguły zbudowane są przy użyciu tych samych atrybutów (co implikuje identyczny schemat reguł). Połączenia pomiędzy tabelami określają możliwe ścieżki wnioskowania. Atrybuty mogą być porównywane z wartościami nieatomicznymi, włączając w to podzbiory oraz zakresy. Model Modularnego Systemu Tabelowego można zastosować dla wielu atrybutowych systemów regułowych. Stanowi on uniwersalną metodologię strukturalnej reprezentacji wiedzy. Zaproponowane w niniejszej pracy podejście bazuje na podejściu narzędzi wspomagających projektowanie (CAD/CASE) oraz graficznej i algebraicznej reprezentacji wiedzy. Pozwala to rozwiązać problem akwizycji wiedzy poprzez systematyczne projektowanie reguł z użyciem modułowej i hierarchicznej struktury, której dostarcza MTS. Co więcej proponowane rozwiązanie umożliwia weryfikację własności projektowanych reguł takich jak zupełność, spójność i determinizm. Prowadzi to do przesunięcia weryfikacji reguł do fazy ich projektowania. Pozwala to na stopniowe udoskonalanie projektu systemu zamiast standardowego cyklu projektowanie-weryfikacja-modyfikacja. Prototyp graficzo-algebraicznego narzędzia CAD/CASE ogólnego zastosowania, wspierającego projektowanie reguł i umożliwiającego weryfikację został zaimplementowany i jest również przedstawiony w niniejszej pracy. Narzędzie używa nowej metodologii tworzenia reguł bazującej na MTS o roboczej nazwie *drzewa tabelowe* (tab-trees, tabular-trees, tree-tables), które łączą w sobie zalety atrybutowych tabel decyzyjnych oraz drzew decyzyjnych. W dalszej części opisana jest teoria drzew tablicowych oraz zaprezentowany jest system *Osiris*. Osiris został początkowo zaprojektowany jako narzędzie CASE do projektowania reguł dla systemu ekspertowego Kheops, obecnie jest narzędziem uniwersalnym, niezależnym od docelowego systemu regułowego. Jego aktualna wersja rozwojowa stanowi ilustrację koncepcji wspomagania budowy modułowej struktury bazy wiedzy z inkrementalną weryfikacja wybranych własności. Prezentowany system stanowi więc narzędzie o charakterze generycznym, a z uwagi na zastosowane rozwiązania może być łatwo dostosowany do szczegółowych wymagań projektowanego systemu regułowego.

2. Podstawy inżynierii wiedzy z wykorzystaniem narzędzi CAD/CASE

Projektowanie bazy wiedzy dla systemów regułowych sprowadza się do bezpośredniego użycia graficznych i algebraicznych narzędzi CASE. Narzędzia takie mogą

[1]MTS: Modular Tabular System – (ang.) Modularny System Tabelowy

wspomagać proces projektowania oraz pozyskiwania wiedzy udostępniając odpowiednie metody je reprezentacji, bazujące na programowaniu graficznym. Warstwa logiczna, w oparciu o którą dokonywana jest specyfikacja wymaganych własności nie jest bezpośrednio widoczna dla użytkownika.

W niniejszej pracy przedstawiony jest system CAD/CASE wspierający zarówno proces projektowania reguł jak również umożliwiający weryfikację. Narzędzie zostało zbudowane bazując na czterech nowych propozycjach z dziedziny systemów regułowych:

- moduły FBM [2] – są to podstawowe moduły logiczne,
- elementy IBP [3] – reprezentują bezpośrednio ścieżki wnioskowania systemu regułowego jako połączenia pomiędzy FBM, które tworzą
- Tab-Trees (albo Tabular-Trees), które reprezentują strukturę MTS oraz
- proste modele algebraiczne określające specyfikację logiki reguł w formie tabel decyzyjnych.

FBM umożliwia projektowanie złożonych systemów, reprezentując nawet bardzo skomplikowane reguły (operujące na wielu atrybutach). Ponadto moduł może być wykorzystany wielokrotnie (ponowne użycie wiedzy), połączony z innymi modułami, lub o ile będzie to konieczne rozdzielony na kilka mniejszych modułów. Bezpośrednie graficzne odwzorowanie IBP powoduje, że projektowanie reguł staje się prostsze i bardziej przejrzyste. Aby zdefiniować regułę użytkownik łączy odpowiednie FBMy. Zarówno wejście, wyjście jak i własności reguł są natychmiast widoczne i łatwe do zinterpretowania, a graficzny interfejs czyni projektowanie wysoce intuicyjnym. Drzewa tablicowe (Tab-Trees) stanowią nowe graficzne narzędzie do reprezentacji wiedzy. Drzewo składa się z wielu FBMów połączonych za pomocą IBPsów. Drzewa tablicowe łączą w sobie zalety drzew decyzyjnych, atrybutowych tabel decyzyjnych oraz tzw. ψ-drzew [9, 10]. Więcej informacji na ten temat można znaleźć w [20].

Model algebraiczny w formie systemu tabelowego pozwala na efektywną weryfikację własności systemu. Podobieństwo do tabel relacyjnych baz danych i atrybutowych tabel decyzyjnych powoduje, że projektowanie i analiza stają się bardzo intuicyjne. Szczegóły techniczne odnośnie weryfikacji w systemach tabelowych można znaleźć w [12].

W ramach projektu pod nazwą *Osiris* zostało implementowane uniwersalne narzędzie do projektowania reguł dla systemów regułowych. Pierwotnie *Osiris* wspomagał projektowanie reguł dla systemów zbliżonych do Kheops (szczegółowe informacje na temat systemu Kheops można znaleźć w [6]). Aktualnie jest uniwersalnym, niezależnym od systemu narzędziem wspomagającym reprezentowanie, pozyskiwanie i weryfikację wiedzy.

[2] FBM: Flexible Branching Modules
[3] IBP: Inference Branching Paths – (ang.) Ścieżki Wnioskowania

3. Podstawy teoretyczne

3.1. Systemy tabelowe: projektowanie i weryfikacja

Jedną z głównych idei usprawniających projektowanie systemów regułowych jest przesunięcie weryfikacji do fazy projektowania, jak również wspomaganie samego projektowania za pomocą wygodnego i intuicyjnego graficznego środowiska programowania typu CAD/CASE. Aktualne badania są rozwinięciem tez przedstawionych w [9, 10], gdzie zaproponowano tzw. ψ-drzewa jako sposób reprezentacji wiedzy, wspomagający proces logicznego projektowania zupełnych i deterministycznych systemów regułowych. W badaniach wykorzystane zostały również propozycje odnośnie systemów tabelowych [11, 12], które prezentowały podejście zbliżone do atrybutowych tabel decyzyjnych. Atrybutowe tabele decyzyjne są łatwe w interpretacji, a ich struktura przypominająca strukturę relacyjnych baz danych może być używana zarówno do reprezentacji danych jak i wiedzy.

Modularny System Tabelowy (MTS), jest zbliżony do modelu relacyjnych baz danych (RDB) i posiada podobne zalety. System tablicowy może określać fakty albo reguły; w takim przypadku część atrybutów odpowiada warunkom reguł, podczas gdy reszta odpowiada hipotezom i decyzjom. Takie systemy można tworzyć z wykorzystaniem struktury RDB, ale w tym przypadku wszystkie wartości atrybutów muszą być atomiczne. W przypadku MTS dopuszczalne są wartości nieatomiczne; mogą to być podzbiory, przedziały itp. System tabelowy może składać się z jednej lub więcej tabel połączonych w strukturę.

W rezultacie powstało graficzne środowisko wspierające projektowanie i implemetujące częściową weryfikację podczas projektowania. Prototypowy system został nazwany *Osiris*.

3.2. Kheops

Początkowo projekt *Osiris* był zorientowany jako narzędzie CAD/CASE do tworzenia reguł dla systemu *Kheops*. *Kheops* [6] jest zaawansowanym systemem regułowym czasu rzeczywistego. Bazuje na metodologii wnioskowania w przód, oraz charakteryzuje się deterministycznym czasem reakcji (poniżej 15 milisekund) co umożliwia jego zastosowanie jako system czasu rzeczywistego. Kheops kompiluje reguły do postaci drzewa decyzyjnego, co pozwala na automatyczną weryfikację niektórych własności (np. zupełność) oraz obliczenie czasu reakcji. Ponadto Kheops implementuje mechanizmy temporalne oraz dopuszcza obliczanie wyrażeń (C-expression). Szczegółowe informacje na temat Kheopsa można znaleźć w [6].

4. Własności systemów regułowych

4.1. Pochłanianie reguł

Rozważmy ogólny przypadek pochłaniania reguł; niektóre definicje znajdują się w [1, 15, 17] oraz z uwzględnieniem systemów tabelowych w [11, 12]. Reguła zawiera inną regułę jeżeli zachodzą warunki: warunek pierwszej reguły jest *słabszy* (bardziej ogólny) niż warunek reguły zawieranej, wniosek pierwszej reguły jest *silniejszy* (bardziej szczegółowy) niż wniosek reguły zawieranej.

Niech reguły $R: \phi \longrightarrow h$ i $R': \phi' \longrightarrow h'$, spełniają: $\phi' \models \phi$ i $h \models h'$. Reguła zawierana R' może zostać usunięta zgodnie z poniższym schematem:

$$\begin{array}{cccc} R: & \phi & \longrightarrow & h \\ R': & \phi' & \longrightarrow & h' \\ \hline R: & \phi & \longrightarrow & h \end{array}$$

Intuicyjnie reguła zawierana może być usunięta ponieważ jej skutkiem są słabsze rezultaty i potrzebuje do zadziałania silniejszych warunków. Zatem ten sam rezultat może zostać osiągnięty z użyciem reguły pochłaniającej. Używając notacji tabelowej mamy:

rule	A_1	A_2	...	A_j	...	A_n	H
R	t_1	t_2	...	t_j	...	t_n	h
R'	t'_1	t'_2	...	t'_j	...	t'_n	h'

rule	A_1	A_2	...	A_j	...	A_n	H
R	t_1	t_2	...	t_j	...	t_n	h

Warunkiem zawierania, w przypadku powyższych tabel jest $t'_j \subseteq t_j$, dla $j = 1, 2, \ldots, n$ i $h' \subseteq h$ (w większości przypadków $h' = h$ są wartościami atomicznymi). W aktualnej wersji systemu *Osiris* zawieranie się reguł jest automatycznie eliminowane dzięki podziałowi wartości, jakie może przyjmować atrybut, na rozłączne przedziały.

4.2. Determinizm

Zbiór reguł jest *deterministyczny* jeżeli żadne dwie reguły nie są spełnione dla tego samego stanu. Zbiór reguł, który nie jest deterministyczny jest również określany jako *ambiwalentny* [15]. Determinizm można uzyskać eliminując pokrywające się reguły. Z czysto logicznego punktu widzenia system jest deterministyczny jeżeli prewarunki $\phi \wedge \phi'$ nie są spełnione. Obliczenie $\phi \wedge \phi'$ jest następujące: dla każdego atrybutu A_j jest wartość atomiczna postaci: $A_j = t_j$ w ϕ i $A_j = t'_j$ w ϕ', $i = 1, 2, \ldots, n$. Jeżeli przecięcie t i t' dla pewnego j (przynajmniej jednego) jest puste (np. dwie różne wartości; bardziej ogólnie $t_j \cap t'_j = \emptyset$), wtedy reguły są rozłączne. W aktualnej wersji *Osirisa* determinizm może zostać osiągnięty za pomocą podziału dziedzin atrybutów na niepokrywające się przedziały (lub zbiory).

4.3. Zupełność systemu regułowego

Intuicyjnie system regułowy jest *zupełny* jeżeli dla dowolnej, przewidzianej sytuacji wejściowej istnieje spełniona przynajmniej jedna reguła. Istnieje kilka możliwości aby zdefiniować zupełność [1, 15, 17], a w przypadku systemów tabelowych [11, 12].

Logiczna zupełność jest rozważana dla zbioru reguł jako:

$$\begin{array}{llll} R_1: & \phi_1 & \longrightarrow & h_1 \\ R_2: & \phi_2 & \longrightarrow & h_2 \\ \vdots & \vdots & \vdots & \vdots \\ R_m: & \phi_m & \longrightarrow & h_m \end{array}$$

Powyższa propozycja pochodzi z czysto logicznej analizy [11, 12] i nie wymaga dodatkowego ekstensjonalnego wyliczania możliwych przypadków. Algebraiczny sposób przetwarzania wiedzy przedstawiony jest poniżej [12].

Rozważmy złączenie rozłącznych formuł warunków reguły w postaci: $\Phi = \phi_1 \vee \phi_2 \vee \ldots \phi_m$. Warunkiem logicznej zupełności powyższego systemu jest:

$$\models \Phi \tag{2}$$

co oznacza, że Φ jest tautologią. W proponowanym rozwiązaniu, zamiast dowodów logicznych użyta została metoda algebraiczna oparta na podziale dziedziny atrybutów [11, 12]. W obecnej wersji system *Osiris* umożliwia sprawdzanie zupełności reguł. Wyszukiwanie brakujących warunków reguł realizowane jest automatycznie i dostępne w każdym momencie projektowania systemu. Dzięki temu niezupełność systemu może być eliminowana etapami już w fazie projektowania.

5. Graficzna reprezentacja

Reprezentacja i pozyskiwanie wiedzy są wciąż otwartymi tematami w dziedzinie inżynierii wiedzy. Istnieje wiele podejść mających zarówno wady jak i zalety, w zależności od celu do jakiego zostały zaprojektowane [8]. W niniejszej publikacji zaprezentowane jest nowe uniwersalne podejście bazujące na Modularnych Systemach Tabelowych (MTS). Wydaje się że MTS udostępnia przejrzysty i efektywny mechanizm reprezentacji struktur logicznych. Reprezentacja ta jest zwana Tree-Table (reprezentacja Drzewiasto-Tabelowa) i jest opisana poniżej.

5.1. Tabele decyzyjne

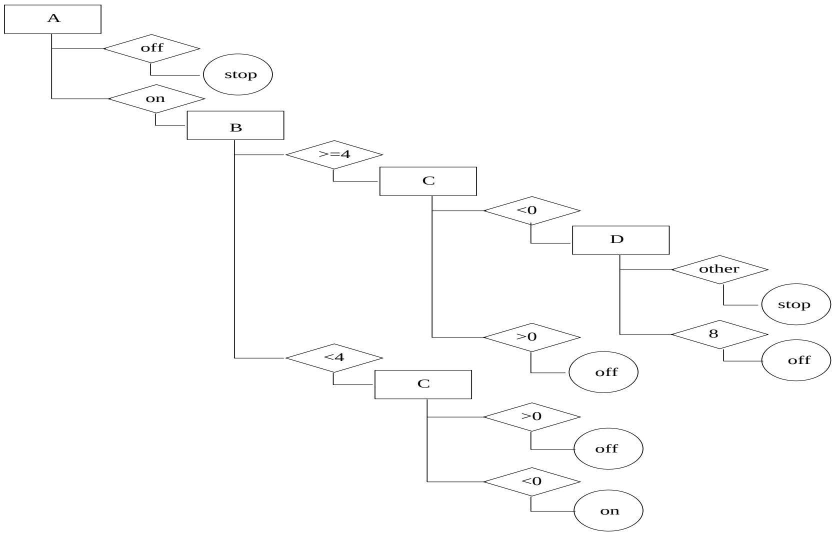
Tabele decyzyjne są sposobem reprezentacji reguł. Zarówno warunki jak i akcje są umieszczone w tabeli. Klasyczna tabela decyzyjna używa logiki dwustanowej, rozszerzonej o wartość "nieistotne". Główną zaletą tabel decyzyjnych jest ich prostota oraz intuicyjna interpretacja. Główną wadą natomiast jest ograniczenie do logiki dwustanowej. W niektórych przypadkach bardziej odpowiednie jest użycie wartości atrybutów. Rozszerzonymi tabelami decyzyjnymi są tabele OAV: Object-Attribute-Value. Przykład takiej tabeli pokazany jest w tabeli 5.1.. Wiersze określają dla jakich wartości atrybutów realizowane są określone akcje. Zarówno `v_ij` jak i `w_kl` mogą przyjmować różne wartości, nie tylko *prawda*, *fałsz*, *nieistotne*.

attrib_1	attrib_2	...	action_1	action_2	...
v_11	v_12	...	w_11	w_12	...
v_21	v_22	...	w_21	w_22	...
⋮	⋮	⋱	⋮	⋮	⋱

Tablica 1: Tabela OAV

5.2. Drzewa decyzyjne

Reprezentacje drzewiaste są łatwe do zrozumienia. Korzeniem drzewa jest węzeł podstawowy. Od każdego węzła, na kształt gałęzi, rozchodzą się połączenia. Wybór połączenia zależy od warunku przyporządkowanego do węzła. Obliczenie warunku określa wybór połączenia. Drzewo jest przemierzane w kierunku z góry na dół, liście określają końcowe decyzje, akcje. Przykład drzewa decyzyjnego dany jest na rysunku 1. Okręgi reprezentują akcje, prostokąty symbolizują atrybuty, a równoległoboki określają relacje i wartości.



Rysunek 1: Przykład drzewa decyzyjnego

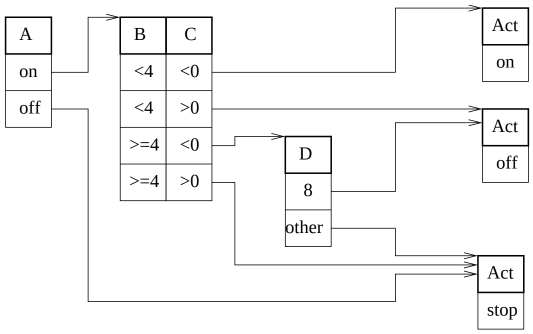
Istnieją również drzewa decyzyjne zwane ψ-drzewami [9, 10]; w takim przypadku pojedynczy węzeł może mieć więcej niż dwa połączenia, co lepiej odzwierciedla proces podejmowania decyzji i pozwala porównywać atrybuty z wieloma różnymi wartościami w tym samym węźle.

5.3. Nowe podejście: reprezentacja drzewiasto-tabelowa

Reprezentacja drzewiasto-tabelowa (Tree-Tab albo Tab-Tree) jest zbudowana jako hierarchia tabel FBM [9, 10]. Hierarchia jest skonstruowana za pomocą ψ-drzewa.

Każdy wiersz tabeli FBM jest połączony z prawej strony z inną tabelą FBM. Takie połączenie odwzorowuje koniunkcję logiczną pomiędzy tabelami. Tabele FBM mogą być łatwo dzielone pionowo na mniejsze komponenty bądź łączone – stanowi to ich podstawową zaletę, umożliwia bowiem ponowne wykorzystanie już zdefiniowanych modułów.

Moduły FBM używane w reprezentacji drzewiasto-tabelowej podzielone są na dwa rodzaje: tabele atrybutów oraz tabele akcji. Tabele atrybutów są częścią atrybutową klasycznej tabeli OAV, natomiast tabele akcji są częścią akcyjną. Podczas gdy tabele atrybutowe mogą mieć dowolną liczbę wierszy (ilość kolumn zależy od ilości użytych atrybutów), tabele akcji mogą posiadać tylko jeden wiersz, oznacza to, że określona akcja (lub zbiór akcji, o ile w tabeli jest więcej niż jedna kolumna) może posiadać ustawioną tylko jedną wartość.



Rysunek 2: Przykład reprezentacji drzewiasto-tabelowej

Przykład reprezentacji drzewiasto-tabelowej jest pokazany na rysunku 2. Należy zauważyć, że reprezentacja drzewiasto-tabelowa jest podobna do modelu relacyjnych baz danych. Głównymi jej zaletami są: prostota, hierarchiczna struktura, efektywny sposób wizualizacji z dużym zagęszczeniem informacji, zalety tabel decyzyjnych, analogie do RDB.

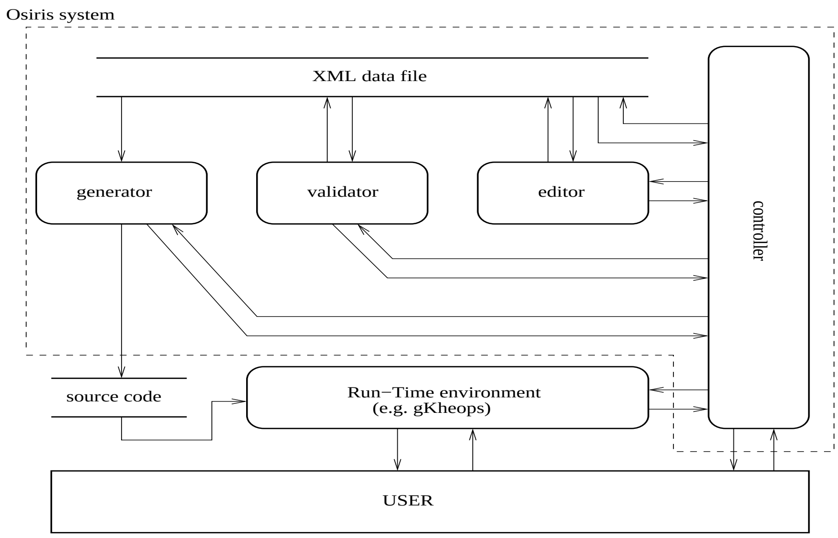
Obecnie, w stosunku do pierwotnego projektu (por. [13, 20]) reprezentacja drzewiasto-tabelowa została rozszerzona. Dotychczasową drzewiastą strukturę powiązań pomiędzy modułami zastąpiono grafem. Umożliwia to m.in. wykorzystanie tych samych modułów wiedzy na różnych ścieżkach wnioskowania (np. wielokrotne użycie modułów akcji `Act` na rys. 2). Modyfikacja ta w istotny sposób zmniejsza nakład pracy potrzebny do wykonania projektu systemu regułowego.

6. System Osiris

Głównym celem systemu Osiris było stworzenie interfejsu do wspomagania projektowania reguł dla systemów regułowych. Początkowo jak docelowy system regułowy został wybrany Kheops [6]. Jednak Osiris jest tak zaprojektowany aby być środowiskiem uniwersalnym. Może być z powodzeniem stosowany dla dowolnego systemu regułowego bez większych modyfikacji. Głównymi celami są:

- graficzny edytor reguł bazujący na logice,
- reprezentacja drzewiasto-tabelowej jako głównej metody projektowania,
- podsystem kontroli poprawności reguł, sprawdzanie zupełności, co umożliwia przesunięcie weryfikacji do fazy projektowania,
- intuicyjny graficzny interfejs użytkownika,
- generacja kodu dla systemu regułowego Kheops,
- integracja z graficznym interfejsem użytkownika gKheops,
- możliwość rozbudowy i modyfikacji (generacja kodu dla różnych systemów).

Osiris jest systemem wielomodułowym zaprojektowanym do pracy w środowisku Unix (testowany dla: Debian GNU/Linux i Solaris). Składa się z: graficznego środowiska do komputerowego wspomagania projektowania reguł, generatora kodu dla systemu Kheops, walidatora, który umożliwia sprawdzanie zupełności i środowisk uruchomieniowego (gKheops) dla stworzonych reguł. Architektura systemu Osiris pokazana jest na rys. 3.



Rysunek 3: Architektura systemu Osiris

Początkowo jako docelowy system regułowy został wybrany Kheops, jednakże struktura Osirisa jest na tyle elastyczna, że można go przekonfigurować do pracy z niemalże dowolnym systemem regułowym. Głównymi modułami są: Editor oraz Controller. Pozostałe moduły mogą być dołączana w trakcie działania systemu. Do wizualizacji tworzonych struktur logicznych została opracowana reprezentacja drzewiasto-tabelowa.

Należy zauważyć, że moduły do generacji kodu, weryfikacji jak również środowisko uruchomieniowe (odpowiednio Generator, Validator, gKheops) są oddzielnymi aplikacjami. Generator i Validator korzystają z teorii kompilatorów przetwarzając plik danych opisujących tworzone reguły. Odpowiednio generują kod dla systemu Kheops oraz sprawdzenie zupełności. Wszystkie moduły są zarządzane poprzez graficzny interfejs. Warto zauważyć, że moduł Controller może, w razie potrzeby, uruchomić dowolną aplikację do celów generacji kodu, bądź też sprawdzenia poprawności tworzonych reguł. Taka modularność umożliwia użycie Osirisa jako narzędzie uniwersalne i niezależne od systemu regułowego.

Do przechowywania projektowanych reguł Osiris używa pliku danych w formacie dokumentu XML. Format posiada dobrze zdefiniowaną gramatykę, co ułatwia generacje kodu dla dowolnego systemu regułowego oraz ewentualną implementację, nawet złożonych algorytmów kontrolujących spójność reguł.

Jedną z najbardziej istotnych cech systemu Osiris jest fakt, że udostępnia on mechanizmy sprawdzania poprawności tworzonych reguł (moduł Validator). Przenosi to proces weryfikacji do fazy projektowania. Validator jest odrębną aplikacją, zatem ewentualne modyfikacje algorytmów kontroli poprawności reguł mogą być dokonane bez konieczności zmian w innych modułach systemu.

Jako środowisko uruchomieniowe, kontrolujące wykonanie reguł, został wybrany gKheops. Wymaga podkreślenia, że środowiskiem uruchomieniowym może być dowolna aplikacja (system regułowy) kontrolująca proces odpalania reguł.

7. gKheops

gKheops [14] jest graficznym interfejsem użytkownika dla systemu ekspertowego Kheops. Jego główną funkcją jest uruchamianie i kontrola Kheopsa jak również wspomaganie tworzenia i testowania reguł. gKheops posiada wbudowany edytor dla plików Kheopsa jak również moduł sprawdzania poprawności składniowej. Interfejs jest łatwy w użyciu, intuicyjny oraz posiada wbudowaną dokumentację.

gKheops jest niezależną od Kheopsa aplikacją. Komunikacja pomiędzy interfejsem a Kheopsem jest możliwa dzięki mechanizmom komunikacji międzyprocesowej. gKheops został zaprojektowany jako aplikacja działająca pod kontrolą systemów operacyjnych Unix.

System gKheops upraszcza w znacznym stopniu tworzenie systemów ekspertowych w środowisku Kheops, udostępniając spójny, graficzny interfejs. Przyspiesza proces uruchamiania i kontroli nad systemem Kheops i jest niezmiernie użyteczny przy debugowaniu oraz testowaniu systemu ekspertowego.

gKheops jest zaprojektowany jako uniwersalny graficzny interfejs użytkownika. Został początkowo zaprojektowany dla system Kheops, ale jego generyczny charakter i modułowa budowa pozwalają na jego integrację z innymi narzędziami. Połączone systemy Osiris-gKheops tworzą zintegrowane środowisko (IDE) dla rozwijania technologii systemów ekspertowych. W przyszłości środowisko to może być rozszerzone o inne narzędzia takie jak inteligentne edytory, oraz moduły wspierające generację kodu dla różnych systemów ekspertowych.

8. Podsumowanie

Niniejsza publikacja prezentuje koncepcję narzędzia CAD/CASE pracujące w środowisku graficznym, wspierające proces projektowania systemów regułowych bazując na metodologii MTS. Głównym celem proponowanego rozwiązania jest stworzenie intuicyjnego, przyjaznego dla użytkownika środowiska, które usprawnia procesy pozyskiwania, projektowania i weryfikacji wiedzy. W aktualnej fazie prototypu, system umożliwia sprawdzanie zupełności projektowanych reguł, natomiast problemy determinizmu, czy też zawierania się reguł rozwiązane są na poziomie reprezentacji wiedzy: MTS.

System Osiris bazuje na nowej reprezentacji wiedzy tzw. reprezentacji drzewiasto-tabelowej, która zbudowana jest z FBMów połączonych za pomocą modułów IBP. Jest to zbiór generycznych, ogólnego przeznaczenia, intuicyjnych narzędzi, przeznaczonych głównie dla ludzi zaznajomionych z relacyjnymi bazami danych i tabelami decyzyjnymi. Należy zauważyć, że istnieje niewiele narzędzi wspierających w podobny sposób proces projektowania reguł (por. [16, 7]). Połączone systemy Osiris i gKheops znajdują zastosowanie, jako uniwersalne i rozwojowe środowisko CASE dla wielu różnych systemów regułowych.

Bibliografia

[1] Ed P. Andert. Integrated knowledge-based system design and validation for solving problems in uncertain environments. *Int. J. of Man-Machine Studies*, 36:357–373, 1992.

[2] B. Eaglestone Coenen, F. and M. Ridley. Validation, verification and integrity in knowledge and data base systems: future directions. *In [18]*, pages 297–311, 1999.

[3] F. Coenen. Verification and validation in expert and database systems: The expert systems perspective. *A Keynote presentation in [19]*, pages 16–21, 1998.

[4] F. Coenen. Validation and verification of knowledge based systems: Report on eurovav'99. *Knowledge Engineering Review 15:2*, pages 187–196, 2000.

[5] R. de Hoog. *[8]*, chapter Methodologies for Building Knowledge-Based Systems: Achievements and Prospectus.

[6] Jean-Paul Gouyon. Kheops users's guide. *Report of Laboratoire d'Automatique et d'Analyse des Systemes*, (92503), 1994.

[7] AITECH Katowice. Sphinx 2.3. *http://www.aitech.gliwice.pl/.*

[8] J. Liebowitz. *The Handbook of Applied Expert Systems.* CRC Press, Boca Raton, 1998.

[9] A. Ligęza. Towards design of complete rule-based control systems. In R. Karba J. Kocijan, editor, *IFAC/IMACS International Workshop on Artificial Intelligence in Real-Time Control*, pages 189–194. IFAC, Bled, Slovenia, 1995.

[10] A. Ligęza. Logical support for design of rule-based systems. reliability and quality issues. In M.C. Rousset, editor, *ECAI-96 Workshop on Validation, Verification and Refinment of Knowledge-based Systems*, volume W2, pages 28–34. ECAI'96, Budapest, 1996.

[11] A. Ligęza. Towards logical analysis of tabular rule-based systems. *In [19]*, pages 30–35, 1998.

[12] A. Ligęza. Toward logical analysis of tabular rule-based systems. *International Journal of Intelligent Systems*, 16:333–360, 2001.

[13] Antoni .J Ligęza, Igor Wojnicki, and Grzegorz Nalepa. Design and implementation support of rule-based systems (in polish). In K. Zieliński, editor, *II Polish National Conference on Software Engineering*, pages 229–236. AGH, Zakopane/Cracow, 2000.

[14] Grzegorz Nalepa. Graphical user interface to kheops rule-based expert system. Master's thesis, Academy of Mining and Metallurgy, 1999.

[15] Alun D. Preece. A new approach to detecting missing knowledge in expert system rule bases. *Int. J. of Man-Machine Studies*, 38:161–181, 1993.

[16] Attar Software. Xpertrule 3.0. *http://www.attar.com/pages/info_xr.htm.*

[17] A. Vermesan. *A chapter in [8]*, chapter Foundation and Application of Expert System Verification and Validation.

[18] A. Vermesan and F. Coenen (Eds.). *Validation and Verification of Knowledge Based Systems – Theory, Tools and Practice.* Kluwer Academic Publishers, Boston, 1999.

[19] R. R. (ed.) Wagner. Database and expert systems applications. In *Proceedings of the Ninth International Workshop, Vienna; IEEE Computer Society*, Los Alamitos, CA, 1998.

[20] Igor Wojnicki. Design and implementation of a graphical user interface for computer aided logical design of kheops knowledge based system. Master's thesis, Academy of Mining and Metallurgy, 2000.