Metody realizacji systemów ekspertowych w środowisku systemu PC-Shell

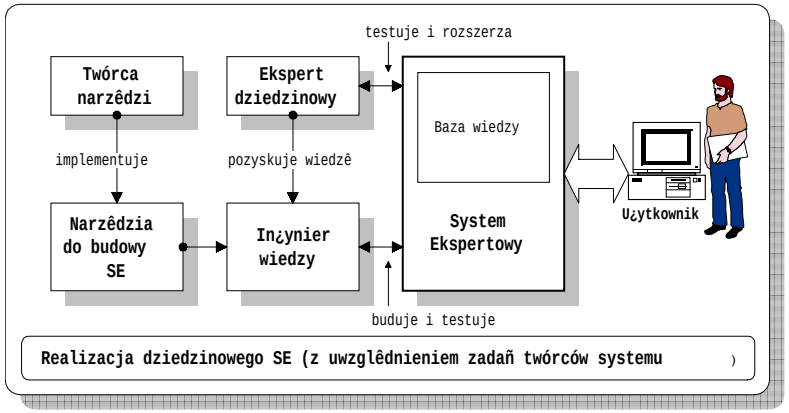
Systemy ekspertowe — charakterystyka ogólna

Właściwości:

- są narzędziem kodyfikacji wiedzy eksperckiej,
- mają zdolność rozwiązywania problemów specjalistycznych, w których duża rolę odgrywa doświadczenie a wiedza ekspercka jest dobrem rzadkim i kosztownym.
- zwiększają dostępność ekspertyzy,
- zapewniają możliwość prowadzenia jednolitej polityki przez centralę firm mających wiele oddziałów,
- poziom ekspertyzy jest stabilny - jej jakość nie zależy od warunków zewnętrznych i czasu pracy systemu,
- jawna reprezentacja wiedzy w postaci zrozumiałej dla użytkownika końcowego,
- zdolność do objaśniania znalezionych przez system rozwiązań,
- możliwość przyrostowej budowy i pielęgnacji bazy wiedzy.

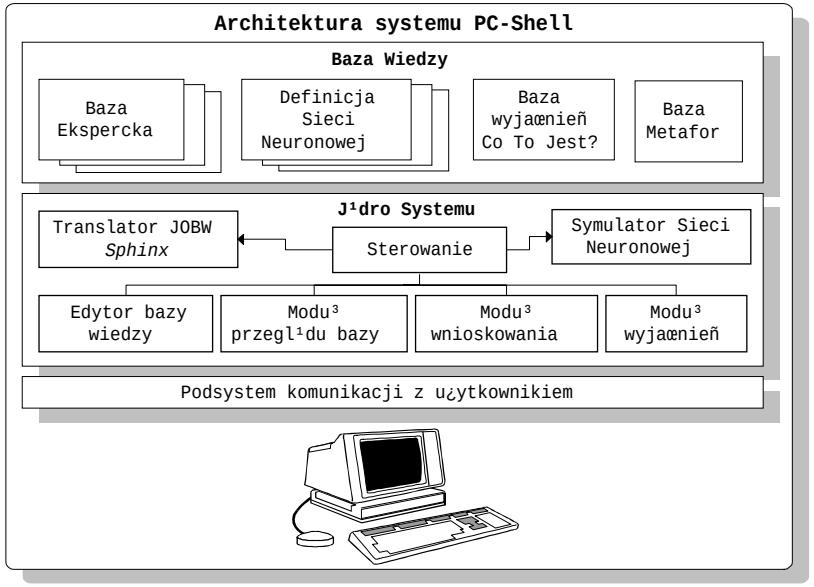
Zastosowania:

- analiza ryzyka,
- ocena wniosków kredytowych, uczestników przetargów,
- monitorowanie, diagnostyka, predykcja,
- wspomaganie procesów diagnostycznych,
- analiza i interpretacja danych,
- instruktaż, dydaktyka, szkolenia.



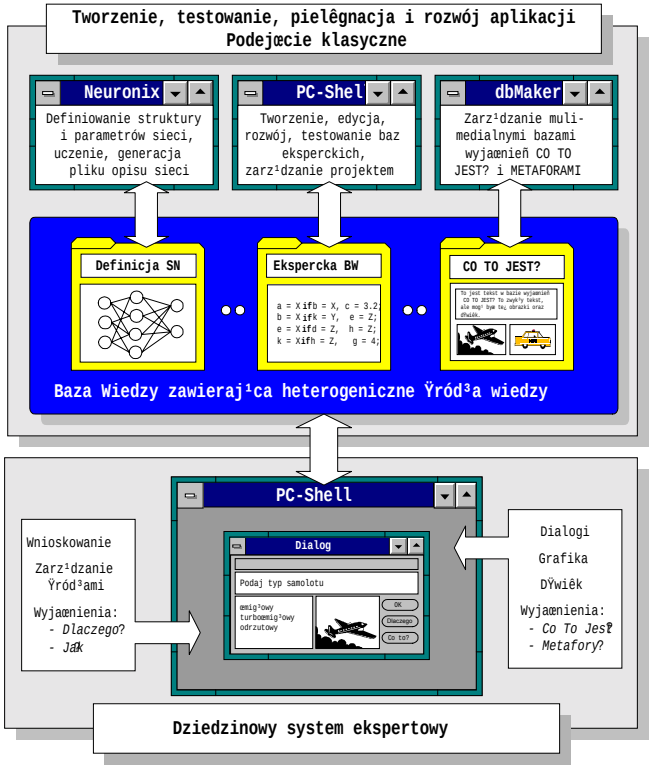
Ogólna charakterystyka szkieletowego systemu ekspertowego PC-Shell

- PC–Shell jest podstawowym elementem pakietu sztucznej inteligencji *Sphinx*®
- PC–Shell jest dziedzinowo niezależnym narzędziem do budowy systemów ekspertowych, posiada właściwości hybrydowe, wykorzystuje elementy architektury tablicowej;
- wykorzystuje różne metody reprezentacji wiedzy:
  - ⇨ deklaratywna w postaci reguł i faktów,
  - ⇨ wiedza rozproszona w sieci neuronowej,
  - ⇨ imperatywna w formie programu algorytmicznego,
  - ⇨ faktograficzna w formie tekstów, grafiki, dźwięku, sekwencji wideo;
- system zapewnia wyjaśnienia:
  - ⇨ jak (ang. *how*),
  - ⇨ dlaczego (ang. *why*),
  - ⇨ co to jest (ang. *what is*),
  - ⇨ metafory (ang. *metaphor*),
  - ⇨ opisu faktów;
- wykorzystywane jest wnioskowanie wstecz (z nawrotami),
- bazy wiedzy mogą być parametryzowane,
- system ma możliwość bezpośredniego pozyskiwania informacji z baz danych (ODBC), wykorzystuje mechanizm DDE,
- system PC–Shell współpracuje z innymi elementami pakietu – systemem Neuronix przeznaczonym do tworzenia sieci neuronowych, systemem CAKE przeznaczonym do wspomagania pracy inżyniera wiedzy oraz realizującym funkcje systemu dbMaker, zarządzającego bazami wyjaśnień.



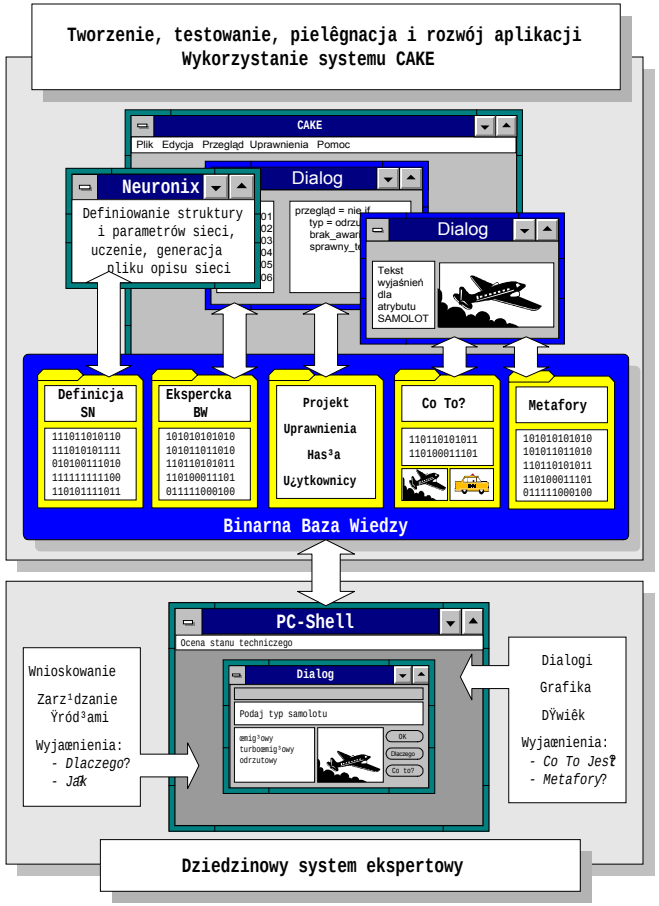
Realizacja aplikacji w środowisku systemu PC-Shell

PC-Shell jest dziedzinowo—niezależnym, szkieletowym systemem ekspertowym o właściwościach hybrydowych. Dzięki zastosowaniu elementów architektury tablicowej, bazę wiedzy podzielić można na pewną ilość heterogenicznych źródeł wiedzy. Realizacja dziedzinowego systemu ekspertowego polega na stworzeniu stosownej bazy wiedzy. Klasyczne podejście do tego procesu polega na zapisie wiedzy w sformalizowanej postaci, do czego wykorzystywany jest język reprezentacji wiedzy. Do definiowania sieci neuronowej wykorzystuje się system Neuronix, tworzenie i zarządzanie bazami wyjaśnień umożliwia system dbMaker. Umieszczony niżej rysunek ilustruje proces realizacji hybrydowej bazy wiedzy.



Pakiet *Sphinx*® zawiera system **CAKE** (ang. ***C**omputer **A**ided **K**nowledge **E**ngineering*), który jest przeznaczony do wspomagania procesu realizacji dziedzinowych aplikacji szkieletowego systemu ekspertowego PC–Shell. Dzięki wykorzystaniu systemu CAKE można realizować aplikacje systemu PC–Shell bez dokładnej znajomości języka opisu bazy wiedzy. Na każdym z etapów pracy system oferuje wygodne narzędzia wspomagające, eliminujące konieczność żmudnego wprowadzania kodu. Zapis baz wiedzy
w postaci binarnej zapewnia z jednej strony ochronę zgromadzonej wiedzy przed niepowołanym dostępem, z drugiej zaś strony poprawia efektywność wykonania aplikacji w

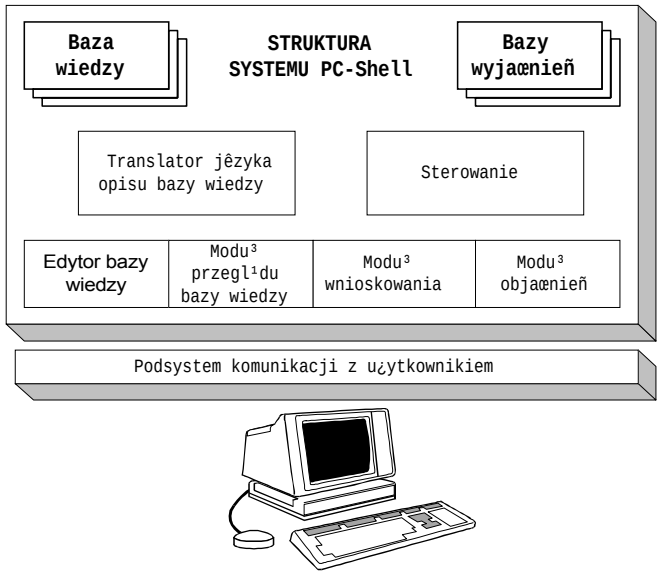
środowisku systemu PC–Shell. Dzięki systemowi uprawnień i haseł można ograniczyć dostęp do aplikacji zarówno na etapie jej tworzenia jak również na etapie jej wykonywania. Umieszczony poniżej rysunek ilustruje przebieg procesu realizacji aplikacji eksperckiej w środowisku systemu CAKE.



Struktura bazy wiedzy systemu PC-Shell

Baza wiedzy systemu zapisywana jest przy użyciu wyspecjalizowanego *języka opisu bazy wiedzy SPHINX*. Integruje on w sobie *deklaratywny język reprezentacji wiedzy* oraz *strukturalny język programowania*. Baza wiedzy zapisywana jest w postaci pliku (lub plików) tekstowych poddawanych procesowi translacji na początku każdej sesji konsultacyjnej.

Baza wiedzy ma strukturę blokową. Ogólną syntaktykę baz wiedzy oraz funkcje poszczególnych bloków przedstawia umieszczona poniżej tabela.



Struktura bazy wiedzy systemu PC-Shell

Struktura bazy wiedzy	Opis funkcji bloków bazy wiedzy
knowledge base *nazwa*	
sources *opis_plików* **end;**	Definicja plików zawierających źródła wiedzy: ekspercka baza wiedzy, definicja sieci neuronowej, baza wyjaśnień.
facets *opis_faset* **end;**	Definicja atrybutów - ich typów i właściwości, ustalenie wartości przełączników sterujących wnioskowaniem.
rules *opis_reguł* **end;**	Blok opisu reguł zapisanych w postaci klauzul Horna.
facts *opis_faktów* **end;**	Blok opisu faktów zapisanych w postaci trójek Obiekt-Atrybut-Wartość.
control *program* **end;**	Blok programu - sterowanie wnioskowaniem i aktywacją źródeł, pozyskiwanie i wstępne przetwarzanie danych, dostęp do plików baz danych, dynamiczna wymiana danych itp.
end;	

Blok deklaracji źródeł wiedzy

Format opisu źródła:

nazwa_źródła **:**
type { kb | **metaphor** | **what_is };**
file *łańcuch_znaków;*

gdzie **type** służy do specyfikacji typu źródła, jednego z poniższych:

- **kb** - eksperckie bazy wiedzy,
- **neural_net** - sieci neuronowe,
- **metaphor** - bazy danych zawierające wyjaśnienia typu *metafory*,
- **what_is** - bazy danych zawierające wyjaśnienia typu *co to jest,*

natomiast **file** *łańcuch_znaków* określa plik, w którym przechowywane jest źródło wiedzy.

Przykładowa deklaracja bloku źródeł*:*

```
sources
        decyzja_kredytowa :
                type kb
                file "c:\\bazy\\bw\\decyzja.zw";
        prognoza_finansowa :
                type neural_net
                file "c:\\bazy\\sieci\\prognoza.def";
        metafory :
                type metaphor
                file "c: \\bazy\\bw\\kredyt.dbm";
        coto :
                type what_is
                file "c: \\bazy\\bw\\kredyt.dbw";
end;
```

Ogólna struktura bloku faset

Fasetami określa się tu zbiór deklaracji odnoszących się do wybranych ***atrybutów***. Blok faset zawiera wykaz wszystkich atrybutów używanych w bazie wiedzy, wraz z przypisanymi do nich fasetami.

```
facets
  opis_faset
end;

opis_faset:
        atrybut1 [ deklaracje_faset1 ];
        -
        atrybutn [ deklaracje_fasetn ];
```

Rodzaje faset opisujących atrybuty:

ask - określa czy system może stawiać pytania dotyczące danego atrybutu; system zadaje pytania jedynie w sytuacji, gdy nie potrafi potwierdzić warunku reguły lub hipotezy wykorzystując fakty i reguły zawarte w bazie wiedzy;

query- umożliwia zdefiniowanie przez użytkownika własnej treści zapytań o wartość atrybutu, generowanych przez system;

unit - umożliwia zadeklarowanie jednostki miary, w której wyrażane są wartości danego atrybutu, podczas wyświetlania informacji zawierającej dany atrybut, dodatkowo - po wartości - będzie pojawiał się tekst zadeklarowany jako *jednostka_miary*.

val - określa zbiór dopuszczalnych wartości danego atrybutu. Wartości mogą być numeryczne lub symboliczne; do określenia dozwolonych lub niedozwolonych wartości służą następujące deklaracje związane z fasetą ***val* : *oneof***, ***someof***, ***range***, ***except;***

param - faseta ta umożliwia zadeklarowanie tzw. *zmiennych parametrycznych* i przypisanie im wartości domyślnych;

picture, sound - fasety te umożliwiają związanie rysunków i/lub dźwięku z atrybutem lub jego wartościami; rysunek jest automatycznie pokazywany, np. gdy pojawia się zapytanie dotyczące atrybutu z którym związany jest rysunek; dźwięk można odtworzyć po wybraniu odpowiedniego przycisku.

Przykłady definicji atrybutów z użyciem faset :

```
facets
 temperatura_ciała :
  query "Podaj temperaturę ciała:";
  unit "°C";
  val range < 36, 42 >;
  param { NORMALNA = 36.7, STAN_PODGOR = 37.5 };
 -
 -
 kolor_nadwozia :
  query "Proszę podać kolor nadwozia samochodu:" ;
  val oneof { "biały", "czerwony", "niebieski" };

 pojemność_silnika :
  query "Proszę podać pojemność silnika:" ;
  unit "cm3";
  val except { < MIN, 600 ), ( 4000, MAX > };
 -
 -
 grzyb :
  val oneof { "pieczarka", "muchomor",    "maślak" }
  picture   { "piecz.bmp", "muchomor.bmp","maslak.bmp"};
  sound     { "ok.bmp",    "alarm.bmp",   "ok.bmp" };
 -
 -
 ilość_pamięci_RAM :
  query "Podaj przewidywaną ilość pamięci RAM:";
  unit "MB";
  val range < 1, 128 >
 -
 -
end;
```

Bloki opisu faktów

facts
opis_faktów
end;

opis_faktu: trójka_OAW; lub not *trójka_OAW*;

Przykłady poprawnie zapisanych faktów:

```
facts
  temperatura_ciała( kowalski ) = 37.5;
  kolor_nadwozia = "biały";
  pojemnosc_silnika = 2500;
  grzyb = "maślak";
end;
```

Bloki opisu reguł

rules
opis_reguł
end;

opis_reguły:
[*nr_reguły* :] *konkluzja* if *warunek_1* &
warunek_2 &
...&
warunek_n;

Przykłady poprawnie zapisanych reguł:

```
typModemu = "Zoltrix 56000 Cobra Voice V.90 PCI" if
        producentModemu = "Zoltrix",
        jakiModem = "modem wewnętrzny",
        szybkośćModemu = "56000";

typDrukarki = "Hewlett-Packard Desk Jet 1120C" if
        rodzajDrukarki = "drukarka atramentowa",
        producentDrukarki = "Hewlett-Packard",
        zastosowanieDrukarki = "wydruk tekstu i rysunków",
        jakośćWydruku = "jak najlepsza";

ryzyko_rozwoju_choroby_wieńcowej = "wysokie" if
        skłonności_dziedziczne_choroby_wieńcowej = "występują",
        miażdżyca_tętnic_wieńcowych = "występuje";

 gatunek = "DOOM METAL" if
        instrument = "gitara elektryczna",
        rytm = "umiarkowany",
        rodzaj_wokalu = "mroczny, ponury",
        klimat = "ponury",
        tempo = "wolne";
```


Blok sterujący

control
program
end;

program:
[deklaracje zmiennych]
instrukcje

Typy danych

- całkowite (int, longint),
- rzeczywiste (float, double),
- tablicowe (jedno i dwuwymiarowe),
- rekordowe.

Instrukcje programowania (podział ogólny)

- instrukcje sterujące wykonaniem programu,
- instrukcje inicjujące i sterujące procesem wnioskowania,
- instrukcje operujące na bazie wiedzy,
- instrukcje podsystemu komunikacji z użytkownikiem,
- instrukcje symulatora sieci neuronowej,
- instrukcje związane z parametryzacją baz wiedzy,
- instrukcje obsługujące dostęp do plików,
- instrukcje obsługujące dostęp do baz danych,
- instrukcje dotyczące dynamicznej wymiany danych.

Przykładowy blok sterujący

```
control
  run;
  createAppWindow;
  vignette( "NET", "Demonstracyjna baza wiedzy" );
  setAppWinTitle("Konfiguracja sieci komputerowych");
  menu "Konfiguracja"
    1. "Dobór rodzaju sieci"
    2. "Dobór konfiguracji serwera (NOVELL)"
    3. "Dobór konfiguracji stanowisk roboczych"
    4. "Nowa konsultacja"
    5. "Wyjście"
  case 1;
    solve( nos, "typ_systemu_operacyjnego=Jaki_system" );
  case 2;
    solve( server,"konfiguracja_serwera=Jaki_serwer" );
  case 3;
    solve( pc, "konfiguracja_stacji_roboczych=Jakie_stacje");
  case 4;
    delNewFacts;
  case 5;
    exit;
  end;
 end;
end;
```