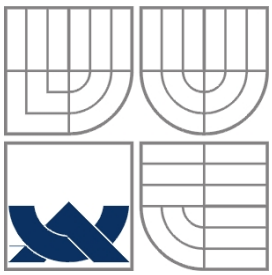


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

EXPERTNÍ SYSTÉMY
EXPERT SYSTEMS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE **MICHAL VESELOVSKÝ**
AUTHOR

VEDOUCÍ PRÁCE **RNDR. JIŘÍ DVOŘÁK, CSC.**
SUPERVISOR

BRNO 2011

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2010/2011

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Michal Veselovský

který/která studuje v **bakalářském studijním programu**

obor: **Strojní inženýrství (2301R016)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Expertní systémy

v anglickém jazyce:

Expert systems

Stručná charakteristika problematiky úkolu:
Expertní resp. znalostní systémy představují významnou oblast umělé inteligence. Tyto systémy se používají při řešení složitých problémů, kde běžné metody selhávají a kde lidští experti nejsou k dispozici nebo jsou omezeně dostupní.

Cíle bakalářské práce:
1. Popsat principy činnosti expertních systémů.
2. Na základě průzkumu webu charakterizovat vybrané dostupné prostředky pro tvorbu expertních systémů a provést jejich srovnání.

Seznam odborné literatury:

GOSMAN, S. a kol. Umělá inteligence a expertní systémy. Praha, Kancelářské stroje 1990.
POPPER, M.; KELEMEN, J. Expertné systémy. Bratislava, ALFA 1989.

Vedoucí bakalářské práce: RNDr. Jiří Dvořák, CSc.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2010/11.

V Brně, dne 15.11.2010

L.S.

Ing. Jan Roupec, Ph.D.	prof. RNDr. Miroslav Doupovec, CSc.
Ředitel ústavu	Děkan

ABSTRAKT

Expertní systémy (ES) jsou komerčně jedno z nejúspěšnějších využití umělé inteligence (UI neboli AI – Artificial Inteligence) a to již od 80. let 20. století. Hojně se využívají ve zdravotnictví, průmyslu, vědě, obchodu, bankovnictví atd.

Expertní systém je software využívající znalostí expertů pro řešení velice složitých úloh a problémů, které by jinak vyžadovaly účast či konzultaci jednoho nebo více odborníků na danou problematiku. Tento software simuluje způsob rozhodování člověka-experta při řešení složitých úkolů, a snaží se dosáhnout nejpravděpodobnějšího výsledku, ideálně shodného s názorem experta.

Typickým rysem expertních systémů, odlišujícím je od obyčejných programů, je oddělení samotného rozhodovacího mechanismu systému (inferenčního mechanismu) a báze znalostí - stejný expertní systém pouze s jinou bází znalostí tak může sloužit k různým účelům. Toho se využívá při tvorbě prázdných expertních systémů – shellů.

Dalšími rysy, které však nemusí platit pro všechny ES, jsou schopnost rozhodování při neurčitosti a schopnost vysvětlit dané rozhodnutí.

Cílem této práce je popsat základní principy funkce ES s využitím volně dostupných informačních zdrojů, a následně popsat a rozebrat prostředky pro tvorbu těchto systémů pomocí informací především z oficiálních webových stránek.

ABSTRACT

Expert systems (ES) are commercially one of the most succesfull use of artificial intelligence (AI) - since eighties of the 20th century. They are often used in medicine, industry, science, trade, banking etc.

Expert system is a software using knowledge of human experts for solving very complicated tasks and problems, which would otherwise require participating or consultation of one or more specialists on these issues. This software simulates decision-making of human expert in solving complicated tasks, and tries to reach the most probable result, ideally same as the expert´s opinion.

Typical feature of expert systems, differencing it from usual software, is separating of decision making engine (inference engine) and knowledge base – same expert system with different knowledge base may serve for different purposes. This feature is used for creating empty expert systems – shells.

Other features, which may not occur at all ES, are ability to make decision with uncertainty, and ability to explain the decision.

Goal of this work is to describe basic principles of ES, using freely available information sources, and then describe and analyze resources for creating of these systems, using mainly information from official web sites.

KLÍČOVÁ SLOVA

Expertní systémy, báze znalostí, inferenční mechanismus, prázdný expertní systém, shell, tvorba expertních systémů, umělá inteligence.

KEYWORDS

Expert systems, knowledge base, inference engine, empty expert system, shell, development of expert systems, artificial inteligence.

Bibliografická citace

VESELOVSKÝ, M. Expertní systémy. Brno, 2011. 40 s. Bakalářská práce na Fakultě strojního inženýrství VUT v Brně na Ústavu automatizace a informatiky. Vedoucí bakalářské práce RNDr. Jiří Dvořák CSc.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Expertní systémy“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny uvedeny v seznamu literatury na konci práce.

V Brně dne ....
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce RNDr. Jiřímu Dvořákovi, Csc. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne ....
podpis autora

Obsah:

1. ÚVOD

V této práci pokud možno jednoduše a pochopitelně nastíním problematiku expertních systémů, jejich dělení dle různých kritérií, naznačím strukturu a způsob rozhodování ES, dále popsat a stručně charakterizovat prostředky pro tvorbu ES – především speciální programová prostředí a prázdné expertní systémy. Vysvětlím, kdy je použití ES vhodné a kdy ne, kdy se vyplatí ES vytvářet nebo koupit, a kdy je výhodnější využít lidského experta. Text není zaměřen na získaní znalostí potřebných k tvorbě ES, cílem práce je pouze seznámit čtenáře s problematikou tak, aby měl po přečtení základní znalosti o funkčnosti a využití ES, a dobrý základ pro další širší studium.

Expertní systémy jako pojem jsou známy již téměř od poloviny 60. let dvacátého století. První pokusy začaly vznikat s rozvojem umělé inteligence, pod kterou ES svojí charakteristikou spadají. Komerčně úspěšné systémy se začaly poprvé objevovat na počátku 80. let. V této době již existoval software, který dokázal skutečně nahradit lidského experta s vysokou úspěšností (až v 90% se systém shodoval s lidským expertem), například ve zdravotnictví při diagnóze infekčních onemocnění (MYCIN), plicních onemocnění (PUFF – CENTAUR), hledáni pravděpodobných ložisek rud, kovů a cenných materiálů pod povrchem (PROSPECTOR), kompletní konfigurování počítačů a výpočetních sestav (R1/XCON) a mnoho dalších.

Tyto systémy, jinak nazývané jako expertní systémy první generace, měly několik nedostatků, jejichž odstranění přinesla až druhá generace. Mezi tyto nedostatky patřila schopnost pouze jednoho způsobu reprezentace znalostí, omezené schopnosti vysvětlit své rozhodování a využití pouze znalostí, které systému předal expert při jeho tvorbě.

Od té doby se ES neustále vyvíjí. Jejich tvorbou a vývojem se zabývají stovky firem po celém světě a byly vytvořeny již tisíce různých systému. Velká většina z nich se sice komerčního využití nedočkala, jednalo se především o výzkumné projekty, pokusy a vědecké experimenty s limitovaným praktickým využitím, několik se jich však přesto uchytilo a tyto systémy výrazně urychlily či zjednodušily výrobní/diagnostické práce a nebo jen snížily náklady mnoha uživatelům po celém světě. Současné systémy druhé generace již využívají víceurovňové báze znalostí, jsou schopny mnohem lépe vysvětlovat své rozhodování a jsou doplněny prostředky pro automatické získávání znalostí.

Hlavní výhodou ES ve srovnání s lidským expertem je především nižší dlouhodobá nákladnost a lepší časová efektivita. Místo zaměstnávání jednoho či více specialisty, které musíte platit, musí spát, jíst a tudíž nejsou vždy dostupní, můžete mít software běžící 24 hodin denně bez přestávek a v podstatě bez dalších větších nákladů. Díky tomu jsou expertní systémy velice efektivní a jejich rošíření se pravděpodobně bude ještě dále zvětšovat.

Důležité je především dobře zhodnotit, zda se vám do ES vyplatí investovat, nebo je pro vás lidský expert výhodnější. Pokud máte jednorázový problém, který potřebujete vyřešit a nepočítáte s podobnými problémy do budoucna, je výhodnější si pozvat experta, aby problém vyřešil. Pokud však počítáte s řešením podobného typu úkolů pravidelně a často, investice do vývoje ES a nebo jednoduše zaplacení licence za již existující ES se vám pravděpodobně vyplatí.

Podmínkou výhodného využití ES je dostatečná složitost úkolu nebo nutnost práce s neurčitostmi, kvůli čemuž na to jiný výpočetní software nestačí nebo není schopen zpracovat výsledky v přijatelném čase. V jiném případě nám dostatečně poslouží standartní výpočtové programy či pracovník s nižší kvalifikací.

Velkou výhodou expertních systémů je rychlost vývoje ve srovnání s výchovou, vzděláváním a získáváním zkušeností člověka-experta. Vývoj ES sice může trvat i několik měsíců (až let) a vyžaduje jednoho či více zkušených expertů při vývoji báze znalostí, kteří musí předat své znalosti a zkušenosti, nicméně vzdělání a získání zkušeností pro skutečného lidského experta trvá vždy řadu let. Expertní systém pak lze jednoduše rozkopírovat a dál šířit. Vhodnou alternativou mimo tvorby vlastního systému je zakoupení nebo stažení prázdného expertního systému (tzv. shellu), do kterého za pomoci znalostního inženýrství pouze dodáme vlastní bázi znalostí.

Další výhody zahrnují opakovatelnost a trvanlivost získaných závěrů, schopnost uchování znalostí do budoucna, možnosti využití ES pro potřeby vzdělávání studentů nebo nových zaměstnanců atp.

Nevýhody pak vyplývají z principu ES. Mechanismy a báze znalostí jsou vetšinou úzce zaměřeny a pokrývají pouze malou část celkové problematiky, např. lékařské diagnostické systémy jsou určeny k rozpoznávání pouze určitých druhů onemocnění (pouze plicní, či pouze onemocnění krve atd). Nedokáží určit libovolnou diagnózu pacienta, protože existuje přiliš velké množství proměnných a naprogramovat takový ES je i v současné době velice obtížné a prakticky nereálné.

Proto se také ES nepoužívají pro celkové řízení výroby či podniku, kdy by zastupovaly manažera nebo vedoucího. Takový člověk musí mít spíše povrchnější a všeobecnější informace (o výrobě, managementu, plánování, právních a finančních záležitosti), zatímco ES dosahuje nejlepších výsledků a je nejsnazší jej vytvořit, pokud se specializuje na menší odbornou oblast.

Další nevýhodou je neschopnost samotného systému poznat, kdy na zadaný úkol nestačí. V takovém případě se ES pokusí problém vyřešit, ale bez dostatečných znalostí většinou podá nesprávný výsledek. Důležitý nedostatkem se pak stává velice úzká specializace systému ve chvíli, kdy se jej pokusíme použít pro řešení problému za změněných podmínek, pro něž nebyl určen. Riziko špatného výsledku je opět velice vysoké a program vás na toto riziko často nemusí vůbec upozornit. Následky využití takového špatného výsledku pak mohou být velice závažné.

Z těchto nevýhod plyne nejdůležitější závěr – na žádný, byť sebedokonalejší software, se nedá na 100% spolehnout, a použití lidského rozumu a kontroly by mělo být samozřejmostí.

2. CHARAKTERISTIKA ES

2.1 Zařazení ES z pohledu umělé inteligence (UI)

Obecně se dají úlohy určené pro počítačové řešení rozdělit na 3 druhy [1, s. 36].

a) Úlohy, pro které známe algoritmus a máme dostatečnou výpočetní kapacitu pro jejich výpočet. Příklady – výpočty ozubení, výpočty daní, výpočty elektrických obvodů…

b) Úlohy, pro které známe algoritmus řešení, ale nemáme dostatečnou výpočetní kapacitu. S obrovským rozvojem výpočetních technologií je sice takových úloh čím dál méně, ale vypočítat například pohyby hvězd a všech viditelných těles ve vesmíru je pro nás stále nemožné.

c) Úlohy, pro které neznáme algoritmus řešení. Pokud neznáme algoritmus, standartní výpočetní programy jsou pro řešení těchto úloh nevyužitelné, navíc bez algoritmu řešení ani nevíme, zda-li k řešení máme dostatečnou výpočetní kapacitu. Tyto úlohy jsou obecně nejsložitější a svou podstatou ideální pro řešení s využitím UI. Vycházíme totiž z principu, že člověk je schopen tyto úlohy metodou úsudku na základě znalostí a zkušeností řešit, a právě tento princip se UI snaží napodobit.

Expertní systémy spadají právě do kategorie umělé inteligence. To nám přináší řadu výhod, přičemž jednou z těch hlavních je schopnost pracovat s neurčitostmi. Lidé totiž narozdíl od programů dokáží pracovat s fakty, které nejsou potvrzené, nebo jsou to pouze podložené domněnky. K těmto neurčitostem se řadí i neúplná, nepřesná data, příliš obecné či prázdné pojmy. ES se pokouší tento nedostatek odstranit a využívají k práci s takovými daty číselnou hodnotu „jistoty“ v intervalu <0;1>. Tato hodnota vyjadřuje míru určitosti vstupních dat. S takovými daty je ES schopen dále pracovat, vybrat nejpravděpodobnější řešení problému a k tomuto řešení opět dodat hodnotu z daného intervalu vyjadřující, jak moc si je svým závěrem jistý.

Další výhoda, plynoucí z uplatnění principu UI, je schopnost podat vysvětlení svých rozhodnutí pomocí vysvětlovacího modulu, buďto automaticky v průběhu rozhodování, nebo na požádání. Systém se pak vrací ke svým rozhodnutím a z průběhu vyhodnocování vám podá odůvodnění, proč tak učinil a jak dospěl k danému rozhodnutí. Další využití spočívá ve zdůvodnění, proč se vás na danou informaci ptá. Díky této schopnosti si uživatel může snáze ověřit pravdivost výsledků, lépe jim porozumět, nebo takto může systém fungovat jako prostředek pro vzdělávání.

2.2 Charakteristické vlastnosti ES

Expertní systémy se snaží co nejvěrněji simulovat rozhodovací činnost experta. Člověk expert řeší odborné úlohy díky svým znalostem, které se dají rozdělit na formální (faktickou) část a na heuristickou část (zkušenostní). Formální část je snadno získatelná, jsou to v podstatě jednotlivá fakta logicky pospojovaná dohromady. Při srovnání dvou znalců se stejným formálním základem (vystudovali stejnou školu atp.), bude ten s lepšími heuristikami (s lepšími zkušenostmi získanými řešením úloh podobného typu) lepším expertem.

Myšlenka expertních systému se tedy zakládá na převzetí všech znalostí od lidského experta, včetně heuristických, a jejich uspořádání a kódování do báze znalostí tak, aby je software dokázal dostatečně dobře využít. Konečný cíl je dosáhnout kvality rozhodnutí odpovídající úrovni experta, na jehož znalostech je ES postaven. Hlavní síla ES se tedy nachází v dobře zpracované bázi znalostí, mnohem méně důležitý je pak samotný program (neboli „prázdný expertní systém“). Expertní systémy druhé generace mají navíc schopnost se pomocí různých modulů pro získávání informaci samostatně učit, doplňovat svoji bázi znalostí a teoreticky se i ponaučit ze svých dřívejších rozhodnutí a chyb.

Ve snaze přiblížit se způsobu rozhodování člověka by měly být schopny dát radu i v případě, kdy

nemají dostatek dat. Využívá se k tomu již zmíněný princip neurčitosti, kdy ES musí pracovat s nejistými údaji, subjektivními názory, velmi přibližnými výsledky vyhodnocovacích metod, a dává pak výsledky s mírou určitosti daného rozhodnutí. Proto by měly být v bázi znalostí obsaženy i alternativní způsoby odvozování, které by měly umožňovat dojít ke správnému výsledku s využitím jiných dat.

Hlavním charakteristickým rysem ES odlišujícím tento typ softwaru od normálních vypočtových programů je tedy oddělení samotného řídicího programu a báze znalostí. U běžného softwaru jsou všechny znalosti většinou obsaženy přímo v kódu programu, je tedy mnohem složitější je později modifikovat, doplňovat, dokonce i pouze dohledat požadované informace. U ES jsou všechny tyto úkoly, pokud je samozřejmě báze znalostí dobře navržená, mnohem snazší. Oddělení báze znalostí od řídicího mechanismu má navíc další výhodu; stejný expertní systém s jinou bází znalostí tak může být využit v různých praktických oblastech pro různé činnosti [1, s. 40].

2.3 Struktura ES

Expertní systémy se skládají z několika složek:

a) Báze znalostí – nejdůležitější součást expertního systému

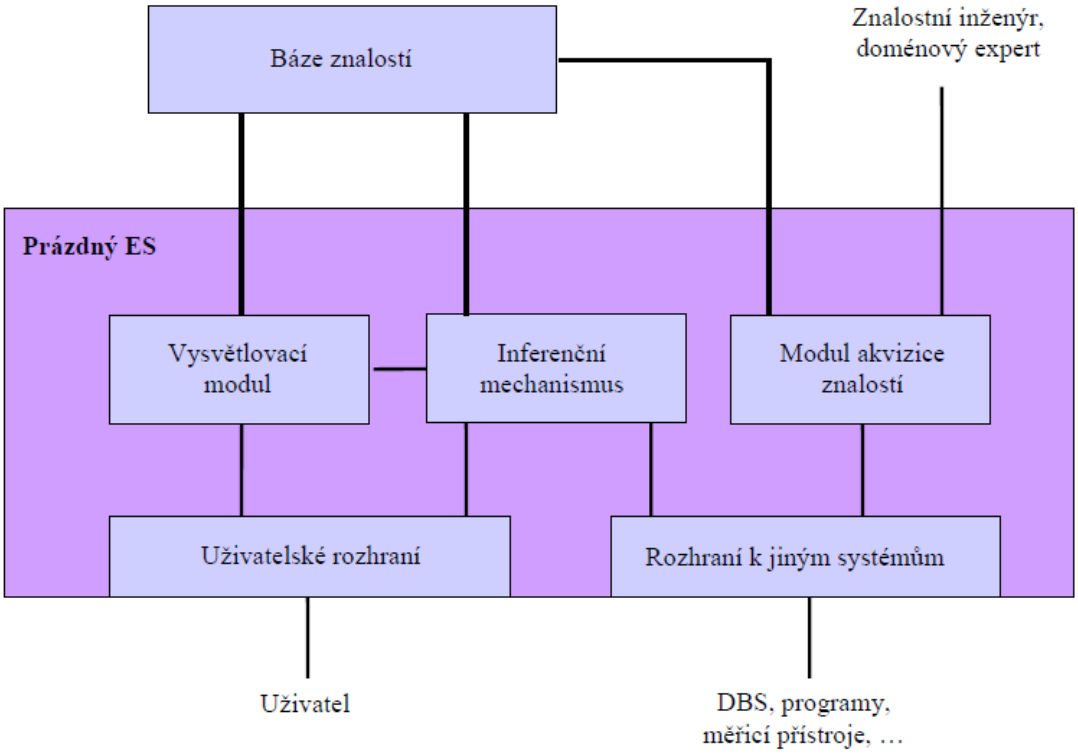
b) Báze faktů – pro ukládání průběžných výsledků

c) Inferenční mechanismus – stará se o interpretaci pravidel z báze znalostí

d) Komunikační modul – zprostředkovává komunikaci s uživatelem

e) Vysvětlovací modul – umožňuje vysvětlit uživateli postup svého rozhodování

Základní struktura systému je zobrazena na obrázku:



Struktura expertních systémů [3]

2.3.1 Báze znalostí (BZ)

Báze znalostí představuje obecný systém pravidel a heuristik, využitelných pro řešení daných problémů. V podstatě je to tedy speciální typ databáze, tvořený pasivními údajovými strukturami – to znamená, že tyto struktury netvoří vykonávatelné instrukce programu. (Výjimku tvoří procedurální poznatky ve tvaru vykonávatelných instrukcí nějakého programovacího jazyka, IF-THEN atp.)

BZ je tvořena všemi znalostmi, které ES potřebuje pro svou činnost. Řadí se mezi ně vše od nejzákladnějších pouček, přes odborné učebnicové znalosti, vysoce specializované znalosti až po soukromé, praxí získané znalosti expertů (i neoveřené, či nepublikovatelné heuristiky) a metaznalosti (znalosti o jiných znalostech). Tyto údaje se během řešení problému, na rozdíl od báze faktů, nemění. To však nevylučuje změnu údajů jiným způsobem, např. modifikací údajů expertem, nebo pomocí modulu pro získávání informací.

Jak bylo naznačeno výše, právě různé heuristiky, nepotvrzené znalosti a metaznalosti , odlišují jak experta od obyčejného pracovníka, tak skutečně výkonný a spolehlivý expertní systém od nekvalitního softwaru vhodného nanejvýš k pobavení nudících se zaměstnanců.

Ke každému inferenčnímu mechanismu lze vytvořit mnoho různých bázi znalostí s různými daty a pro různé účely. Jeden prázdný expertní systém (shell) tak s různými bázemi může fungovat v různých vzájemně vzdálených problémových oblastech [1, s. 40].

Při tvorbě BZ je velice důležité myslet na reprezentaci údajů. Při výběru reprezentačních prostředků je třeba uvažovat o třech hlediscích:

a) vyjadřovací účinnosti (co a jak jimi lze reprezentovat)

b) odvozovací účinnosti (co a jak umožňují odvozovat)

c) výpočtové účinnosti (efektivitě daných prostředků)

Volba způsobu reprezentace je hlavním úkolem tvůrců ES. Reprezentační prostředky vybíráme dále podle předpokládáných vlastností inferenčního mechanismu a dalších modulů systému.

Při tvorbě báze znalostí pro již vytvořený prázdný expertní systém jsou však již reprezentační prostředky dané systémem. Tvůrce báze znalostí tedy pouze musí pouze rozumět vyjadřovací, odvozovací a výpočtové účinnosti tak aby mohl co nejlépe převést své znalosti do BZ [2, s. 86].

Základní způsoby reprezentace znalosti jsou:

a) Pravidly (if-then) – nejpoužívanější

b) Objekty – umožňuje hierarchické uspořádání pravidel pro větší přehlednost a rychlost prohledávání báze.

c) Logickými formulemi – nevhodné pro zpracování neurčitosti

d) Rámci – datové struktury s informacemi o objektech, třídách atd.

Více informací o reprezentaci znalostí najdete v kapitole „2.4.1 Členění dle reprezentace znalostí.“

Při větším množství pravidel může aplikace dat na všechna pravidla trvat velice dlouho, což snižuje celkový výkon systému. Proto se především u objektové reprezentace využívá rozdělení znalostí dle jisté hierarchie, což výrazně zkracuje dobu prohledávání.

Blackboard

Expertní systémy pro řešení velice složitých problémů často využívají několik různých bází znalostí. Na počátku řešení se vybere hlavní báze znalostí, která zpracuje data do báze faktů a započne řešení problému. Důležité hodnoty a fakta pak zapisuje do virtuální tabule (tzv. blackboard). Tyto

údaje pak mohou zaktivovat činnost jiné báze znalostí, která tato fakta přebere a dále vyhodnotí, jakékoliv relevantní výsledky potom opět vloží do tabule pro další řešení. Proces se opakuje až do úplného vyřešení problému. Principu blackboardu využívá například systém GBBopen [zdroj].

Tento postup vychází z představy několika odborníků, často z různých oborů (zde zastoupených bázemi znalostí), kteří společně řeší zadaný problém a zapisují si své častečné výsledky na společnou tabuli, až nakonec dojdou ke konečnému řešení [1, s. 43].

2.3.2 Báze faktů (BF)

Vedle báze znalostí si ES vytváří bázi faktů, vznikající během řešení konkrétního problému. Do této báze si systém ukládá zadané vstupní údaje, dílčí výsledky, odvozená data a nejistá či nepřesná data. Systém údaje v BF v průběhu rozhodování doplňuje, modifikuje, případně i ruší. Způsob organizace dat je pak určen organizací údajů v bázi znalostí a funkcemi inferenčního mechanismu. Údaje z BF jsou přístupné i ostatním modulům systému, jež je dokáží využít k různým účelům.

BF je dynamická struktura – mění se v průběhu řešení problému, nejčastěji přídáváním nových položek, často však i modifikací a rušením. Část údajů vzniká při zadávání problému, tzv. počáteční fakta, část vzniká v průběhu řešení - odvozená fakta a částečné výsledky. Většina BF také k datům přídává čas kdy byl údaj do BF zařazen.

V bázi faktů existují také údaje s podmíněnou platností. Ty zahrnují údaje, které nemůžeme získat přímo (dotazem, měřením atd), ani odvozením z jiných. Systém je tedy vytváří odhadem, jsou to tedy zkušební očekáváná data, jejichž platnost končí pokud jsou vyloučeny jinými údaji nebo podmínkami. Do té doby s nimi systém pracuje jako s normálními údaji. Všechny další údaje odvozené z podmíněně platných dat mají také podmíněnou platnost, tzn když vyloučíme první údaj, všechny údaje odvozené z toho prvního jsou taktéž neplatné [2, s. 91].

2.3.3 Inferenční mechanismus (IM)

Inferenční mechanismus řídí proces aplikací produkčních pravidel na obsah báze údajů. Vybírá takové pravidlo, které připadá do úvahy, tj. takové, které má splněny všechny předpoklady, a postará se o vykonání procedur, spojených s aplikací pravidla a uloží získané výsledky do báze údajů.

Základní hlediska pro zkoumání inferenčních mechanismů [2, s. 72]:

a) Vnější hledisko – podstatné jsou ty funkce, které využíváme při práci s ES. IM by se měl chovat logicky, měla by se projevovat jeho dedukční, usuzovací, asociační schopnost atd.

b) Vnitřní hledisko – podstatné jsou ty funkce, které spojujeme s teoretickými principy jeho konstrukce.

c) Implementační hledisko – zde nás zajímají funkce z hlediska programovacích jazyků (OS, knihovny, podpůrný software atd)

Pro nás je nejdůležitější vnější hledisko, z jehož perspektivy je IM prostředek k napodobování celkové schopnosti uvažování lidského experta. IM z tohoto pohledu je základní mechanismus uvažování a procesů, zakládající se na znalostech uložených v BZ, díky kterým je schopen jednotlivé problémy řešit.

IM tak pomocí různých metod aplikuje pravidla a znalosti z báze znalostí na obsah báze faktů. Vybírá vhodné pravidlo, které splňuje všechny podmínky a požadavky, postará se o procedury spojené s aplikací daných pravidel a výsledky uloží do BF. Toto je základní funkce IM, mimoto má ještě mnoho dalších funkcí, kterým se ale nyní věnovat nebudeme.

Tato základní funkce je především u pravidlových systémů realizována pomocí 2 odlišných metod.

a) Zpětný chod – zpětné řetězení – odvozování řízené cílem

b) Přímý chod – dopředné řetězení – odvozování řízené fakty

Zpětný chod řeší problém nacházením vhodného řešení pro předem stanovený cíl. Využívá se pro ověřování hypotéz (známe cíl a vstupní údaje a hledáme způsob jak tohoto cíle dosáhnout za daných podmínek, resp. jak ověřit jeho platnost).

Neznáme algoritmus řešení, hledáme tedy co nejefektivnější postup. V případě této metody tak, že od daného cíle zpětně odvíjíme podmínky a požadavky potřebné k jeho dosažení. O splnitelnosti všech podmínek většinou bohužel není možné rozhodnout okamžitě, proto vznikají další podcíle řešení, u nichž opět ověřujeme jejich splnitelnost. Tento cyklus se opakuje dokud není možné ověřit splnitelnost všech podcílů i hlavního cíle ze zjistitelných faktů. Tato fakta získáme ze zadání na počátku úlohy, z dotazů kladených uživateli nebo z jiných zdrojů informací a dat.

U **přímého chodu** získáváme řešení interpretací a odvozením vstupních dat, tzn. snažíme se zjistit co z těchto údajů vyplývá a k jakému řešení je tedy možné dospět.

Vycházíme tedy z počátečních údajů, které se během procesu odvozování dále doplňují dle potřeby ES, jelikož počáteční údaje jsou jen velice zřídka dostačující k určení řešení problému. Aplikujeme tyto údaje na pravidla z báze znalostí. Pravidla, která jsou použita vygenerují nová fakta, která jsou opět aplikována na relevantní pravidla. Proces pokračuje dokud nejsou generována žádná nová fakta, nebo jsou získána cílová fakta. Tento způsob nejlépe funguje pokud máme dostatek počátečních dat a existuje mnoho možných řešení vycházejících z těchto dat.

Počáteční fakta většinou umožňují získat velké množství vyplývajících dat. Proto je pro tuto metodu typické testování velkého počtu závěrů a mnoho potřebných údajů, z nichž jen malá část je pro nás skutečně relevantní – bohužel není známo která. Nevýhodou tedy je o mnoho vyšší složitost oproti metodě zpětného chodu. Naopak výhodou je schopnost získat všechny možné výsledky vyplývající ze zadaných dat – u zpětného chodu toto neplatí, protože řešení je zaměřeno pouze na jediný cíl.

Pozorováním lidí při řešení problémů bylo zjištěno, že člověk obě tyto metody většinou kombinuje. Snaží se využít široké možnosti řešení přímým chodem a postupně je vytřídit metodou zpětného chodu. Tento princip se tedy snažíme pro zvýšení účinnosti využít i při tvorbě inferenčních mechanismů [2, s. 73].

2.3.4 Vstupní rozhraní – komunikační modul (KM)

Jakkoli vyspělý expertní systém se pravděpodobně nikdy nestane skutečně využívaným, pokud není schopen jednoduše, srozumitelně a přirozeně komunikovat. Právě k tomuto účelu u ES slouží komunikační modul. KM zajišťuje několik různých funkcí, přičemž k těm hlavním patří především tyto:

a) Zahájení a ukončení činnosti ES, nebo započetí činnosti různých dalších modulů.

b) Zprostředkování dialogu v průběhu činnosti. KM předkládá dotazy uživateli v dialogovém režimu, uživatel se je snaží co nejpřesněji zodpovědět, a KM následně vyhodnocuje jeho odpovědi, žádá dodatečné informace a upozorňuje na nesprávnou/nesmyslnou odpověď.

c) Vyřizování požadavků uživatele, zpracování povelů.

Mnoho ES dovoluje po zahájení činnosti upřesnit jak formu a rozsah dialogu dle konkrétního

uživatele, tak způsob a rozsah odvozovací funkce v závislosti na schopnostech systému. Podstatný je v tomto případě přesný popis problému, který se má řešit – tzn. zadání počátečních faktů a určení cíle řešení, včetně jeho podmínek.

Forma a rozsah dialogu mohou být určeny několika různými režimy. KM by měl používat různou formu dialogu např. podle kvalifikace uživatele; zda-li je to naprostý laik, vystudovaný člověk, nebo expert s dlouholetými zkušenostmi. V případě laika lze očekávat dialog výrazně delší, s mnohem více potřebnými informacemi pro dostatečné upřesnění řešení z daných odpovědí, zatímco u experta bude pravděpodobně postačovat dialog stručný, heslovitý. Dalším příkladem může být rozdíl mezi člověkem nemajícím zkušenosti s prací s ES (takovému je třeba podávat dostatek instrukcí jak dál postupovat) a člověkem zvyklým pracovat s ES (pro toho mohou být takové dodatečné informace zbytečné a rušivé).

KM může režimy měnit především dvěma způsoby:

a) adaptivně – podle způsobu odpovědí uživatele vyhodnotí jeho zkušenosti a odbornost, a následně upraví režim dialogu.

b) dialogem – KM zahájí dialog otázkami, kterými se ptá na odbornost a znalosti uživatele, na jeho zkušenosti s ES apod. Následně vybere nejvhodnější režim dialogu.

c) kombinací – prvně předběžně určí režim pomocí otázek, a podle odpovědí uživatele při samotném průběhu zadávání faktů režim adaptivně upraví.

Pro usnadnění práce by měl KM být rovněž schopen zpracovávat i grafické soubory, jako např. různé grafy. Tato vlastnost je velice výhodná, jedním grafem lze často nahradit až několik různých otázek a jejich složitých odpovědí. Využití moderních technologií jako dotykové obrazovky atd. je již také poměrně běžné.

Samotný dialog může mít mnoho forem. Systém na své otázky může nabízet předdefinované odpovědi, z nichž uživatel vybírá tu správnou nebo nejvhodnější. Může se jednat o pouhé „Ano“, „Ne“, „Nevím“, nebo také mnohem komplikovanější odpovědi. Existuje i možnost vybrat více odpovědí, často je lze uspořádat od nejvhodnější nebo nejpravděpodobnější. Další variantou je odpověď psaná samotným uživatelem, nejčastěji se jedná o jednoslovné odpovědi, případně numerické hodnoty. Neustálý pokrok ve vývoji nám již však omezeně umožňuje psát odpovědi přirozeným jazykem, a KM je schopen jej přijmout a interpretovat. Tato metoda však nemusí být vždy dostatečně spolehlivá a přesná, případně musí být doplňována dalšími otázkami, proto je často lepší používat některou z předchozích možností.

Důležitou vlastností KM je zpracovávání povelů a příkazů. Pro tuto schopnost existuje celý příkazový jazyk; seznam předdefinovaných slov/symbolů, každé s určitým významem, a často s možností přidat parametr, jenž upřesňuje požadavek. Nejčastěji příkazy dávají uživateli možnost zasahovat do činnosti systému. Pokud tedy místo odpovědi na otázku zadáte příkaz, systém jej vykoná. Moderní systémy samozřejmě nahrazují příkazy jednoduššími grafickými ovládacími prvky.

Nejčastější druhy povelů:

a) Spuštění vysvětlovacího modulu - když chceme získat podrobnější informace o rozhodování, průběhu řešení, nebo chceme vědět proč se systém ptá zrovna na tuto konkrétní otázku

b) Zadávání údajů o kterých si myslíme že jsou důležité, ale systém se o ně nezajímá, nebo mazání či úprava údajů z báze faktů.

c) Zasahování do činnosti IM, nastavování priorit, blokování či odblokováni určitých funkcí, změny parametrů funkcí.

d) Přerušení činnosti IM a získání dosud odvozených výsledků.

Existují také zvláštní případy, kdy ES dialog s uživatelem kompletně nebo částečně vynechá. Ke kompletnímu vynechání dialogu dochází pokud je ES propojen například s nějakým měřicím zařízením nebo strojem. Částečné vynechání je například situace kdy uživatel zadá pouze vstupní údaje, a tím ukončí interakci se systémem. Od něj se pak vyžaduje, aby zpracoval výsledek pouze ze zadaných faktů, a došel k nejpravděpodobnějším výsledkům bez dalšího získávání dat [2, s. 97].

2.3.5 Vysvětlovací modul – VM

Pokud vyžadujete radu od experta v oboru, většinou očekáváte že dokáze svůj názor či radu náležitě vysvětlit a zdůvodnit na základně svých znalostí a způsobu, jakým došel k výsledku. Je tedy vhodné požadovat tuto schopnost i expertního systému, čímž se výrazně zvýší jeho atraktivita a užitečnost pro uživatele. Pokud je systém schopen dostatečně zdůvodnit důvod a způsob svého rozhodnutí, pro uživatele je snazší si výsledek ověřit, samostatně si jej zanalyzovat a uvěřit mu – případně jej zamítnout. Právě k tomuto účelu slouží „vysvětlovací modul“, zkráceně VM.

Schopnost vysvětlovat svou činnost je jedna z charakteristik většiny ES a jedna ze zásadních schopností odlišující ES od jiných druhů software. Přesto to však není nutná součást každého systému. Existují situace, kdy je činnost vysvětlovacího modulu, stejně jako v předchozím případě komunikačního modulu, nežádoucí. To však platí pro mizivé procento aplikací, naopak v drtivé většině je možnost vysvětlení činnosti doporučena a vyžadována.

Vysvětlovací moduly jsou přítomny v mnoha variantách - od velice jednoduchých, poskytujících značně zjednodušené vysvětlení až po velice komplikované a důmyslné moduly, které jsou schopny nám podat podrobnou vyčerpávající odpověď.

Nejjednoduší typ VM v podstatě odpovídá na 2 nejčastější typy otázek:

a) Proč? (Z jakého důvodu?)

- Proč se ES ptá na tento fakt?

- Proč ES odvozuje tento fakt?

- Proč ES zvolil tento způsob rozhodování

b) Jak? (Jakým způsobem?)

- Jak ES odvodil tento fakt?

- Jak ES došel k tomuto výsledku?

- Jak ES dospěl k tomuto způsobu rozhodování?

Tyto otázky máme většinou možnost zadat ve chvíli kdy je aktivní komunikační modul a systém se nás ptá na nějaký fakt nebo data. Svou otázku vložíme místo očekávané odpovědi, čímž se zaktivuje vysvětlovací modul. Ten se naši otázku snaží zodpovědět co nejlépe dle svých technických možností – čím důmyslnější VM, tím přesnější, srozumitelnější a podrobnější odpověď.

Obdobně jako v případě KM, ani zde nejsou ty nejpodrobnější odpovědi vždy žádané. V případě kdy ES komunikuje s odborníkem, může být taková odpověď nadbytečná a zdržující. Naopak v případě kdy komunikujeme s naprostým laikem může být jednoduchá odpověď příliš stručná či odborná a uživatel jí nemusí porozumět. Ideální pak tedy samozřejmě je, když VM dokáže přizpůsobit odbornost a rozsah svých odpovědí odbornosti a potřebě uživatele, např. na základě dat získaných z komunikačního modulu [2, s. 100].

2.4 Členění ES

Rozlišujeme 3 základní druhy členění expertních systémů:

a) Dle reprezentace znalostí

b) Dle typu řešených úloh

c) Dle obecnosti a uzavřenosti

2.4.1 Členění dle reprezentace znalostí

Expertní systémy je možné rozdělit podle reprezentace znalostí v bázi znalostí [1, s. 41].

Nejčastější způsob je reprezentace postavená na pravidlech – tyto ES nazýváme „pravidlové expertní systémy“. Báze znalostí je tak tvořena množinami pravidel typu:

podmínka/situace → z ní vyplývající akce/činnost

Výklad tohoto pravidla je jednoduchý; objeví-li se v bázi faktů předdefinovaná situace, systém provede pravidlem danou akci. Při využívání těchto pravidel se využívají dva principy popsané již v podkapitole „2.3.3 Inferenční mechanismy“; zpětný chod, využívající pravidel způsobem odvozování od cílů k datům, a dopředný chod, využívající odvozování od dat k cílům. O využití těchto postupů rozhoduje inferenční mechanismus.

Velmi častým speciálním případem jsou systémy založené na úpravě této základní reprezentace. Jsou to především diagnostické systémy, využívající pravidla typu:

předpoklad → z něj vyvozený závěr

Pokud máme pravidla daná tímto způsobem, znalosti je možné vyjádřit jako „inferenční síť“. Tato síť je reprezentována orientovaným grafem. Tvrzení v této sítí tvoří uzly grafu, a pravidla jsou reprezentována orientovanými čárami. Tato síť tak vytváří pevnou strukturu odvozovacího procesu díky jasně dané návaznosti jednotlivých pravidel i množin pravidel. Pravidla tedy nevystupují samostatně, ale jako celá síť ve tvaru logického stromu, v případě více stromů pak do tvaru lesa. Z toho vyplývá že dopad jednoho pravidla se může stát předpokladem pro další pravidla. Tento princip je základ pro vznik cílových hypotéz, z nichž se již nic dalšího neodvozuje. Typickými příklady pravidlových systémů jsou například Ruleworks [11] nebo EXSYS [13].

Další způsob reprezentace znalostí je založen na „logickém programování“. Tyto systémy se zakládají na vyjádření znalostí v podobě logických formulí. Systémy pak mají za úkol ověřit pravdivost těchto formulí. Systémy založené na tomto způsobu reprezentace ale nejsou příliš vhodné k využívání neurčitých znalostí, a navíc reprezentace znalostí v takovéto bázi není příliš přehledná. Systémy založené na logickém programování jsou nejčastěji systémy naprogramované v PROLOGu [7]

Poslední způsob je reprezentace znalostí v takzvaných „rámcích“. Rámce tvoří datové struktury, v nichž jsou obsaženy všechny informace o objektech a jejich třídách, o různých situacích a důsledcích atd.

2.4.2 Členění dle typu řešených úloh

Dalším způsobem členění expertních systému je dělení podle typu úloh pro které ES využíváme. Podle toho dělení rozdělujeme ES na systémy „plánovací“ a „diagnostické“ [1, s. 43].

Diagnostické expertní systémy mají za úkol interpretovat vstupní data a vyhodnotit která z daného počtu navrhovaných cílových hypotéz nejvíce odpovídá skutečné situaci. Při řešení úkolu tedy dochází k prověřování a zkoušení cílových i dílčích hypotéz (podhypotéz) podle přesně daného postupu, daného strojovým modelem takového řešení. Tento model řešení bývá vytvořen expertem již při samotné tvorbě expertního systému – expert tedy již při navrhování ES vytvoří postupy jak řešit jednotlivé problémy dle počátečních dat. Tyto postupy vychází z expertových vlastních zkušeností při řešení podobných problémů, a ES se je jen snaží co nejvěrněji napodobit. Tyto systémy se také často nazývají „klasifikační“.

Plánovací expertní systémy nejčastěji řeší případy u kterých známe požadovaný cíl a vstupní data. Hlavní úkol ES tedy je najít způsob jak z těchto počátečních dat dojít k požadovanému cíli, za využití všech dostupných informací o řešeném případu. Tyto systémy se zakládají na principu vygenerování možného řešení a jeho následném ověření. Z toho důvodu je podstatnou součástí systému generátor řešení, který automaticky vytváří možné kombinace, jak dojít k cíli. U rozsáhlejších úloh, ve kterých figuruje větší množství proměnných často dochází ke „kombinatorické explozi“ - se zvyšujícím se množstvím proměnných velice rychle roste počet možností při generování řešení. A čím více máme řešení, tím více jich je nutno ověřit, což neúměrně zvyšuje potřebnou výpočetní kapacitu a celkový čas hledání řešení. Protože kombinatorická exploze je poměrně nežádaný efekt, snažíme se ji co nejvíce omezit využitím znalostí expertů i všech dostupných dat o případu. Takto profiltrovaná řešení se následně testují pomocí dat z báze faktů.

Řešení získané plánovacím systémem zahrnuje seznam možných řešení, ideálně s číslem nebo popisem vyjadřujícím přesnost či optimálnost daného řešení. Alternativní název pro tyto systémy je „generativní“ ES.

2.4.3 Členění dle obecnosti a uzavřenosti

Poslední základní dělení ES se zakládá na obecnosti. Takto rozdělujeme ES na „problémově orientované“, „prázdné systémy“ a „hotové systémy“ [1, s. 44].

Problémově orientované byly již první vyvinuté ES. jsou určené použitou reprezentací znalostí a inferenčním mechanismem a je možné je využívat pouze v oblasti pro kterou byly navrženy.

Prázdné systémy jsou v podstatě ES bez své hlavní součástí – bez báze znalostí. Takové systémy mají obecnější využití podle báze znalostí, kterou následně dodáme. Tyto ES vznikly zobecněním problémově orientovaných systémů.

Hotové systémy jsou kompletní aplikace vhodné k okamžitému použití. Jsou využívány pro často se opakující úlohy.

Zvláštní kategorii pak tvoří speciální prostředí pro tvorbu ES. Zjednodušeně jsou to programovací jazyky navržené především pro tvorbu expertních systémů, umožňující navrhnout si prázdný expertní systém přesně dle svých požadavků, případně dokonce navrhnout a dodat bázi znalostí.

2.5 Tvorba ES

Tvorbou expertních systémů se zabývá obor nazývaný „znalostní inženýrství“. Od softwarového inženýrství se liší především tím, že pracuje s nepřesnými, neúplnými znalostmi, jejichž množství není známo, a nevyužívá předem známých postupů řešení (algoritmů).

2.5.1 Hlediska pro tvorbu ES

Při vytváření ES je zapotřebí dbát na několik základních hledisek pro jejich tvorbu a důkladně posoudit zda-li se jeho vytváření vůbec vyplatí, nebo je lepší využít jiných dostupnějších prostředků. Těchto hledisek je velké množství a ne všechna musí být vždy dodržena; záleží především na požadavcích a možnostech zadavatele.

Vhodnost použití – ES je vhodné využívat pro řešení problémů vyžadujících logické uvažování, u kterých je nutno použít heuristické postupy atd. ES dosahuje lepších výsledků než obyčejné programy za určitých podmínek [3]:

- problém je těžko specifikovatelný nebo strukturovatelný

- řešení není založeno na určených výpočtových postupech a algoritmech

- princip řešení je nutno zakládat na neurčitých nebo špatně vyjádřitelných znalostech

- využívané údaje jsou nepřesné, neúplné nebo nedostupné

V takových případech je využití obyčejného softwaru nedostatečné a tvorba ES je skutečně opodstatněná.

Využitelnost ES - je důležité si správně zhodnotit zda-li vůbec ES využijeme dostatečně na to aby se investice vložená do jeho vývoje vrátila. Takovými případy mohou být např. nedostatek lidských expertů či jejich těžká dostupnost, nebo potřeba řešit podobné problémy velice často, hromadně a dlouhodobě. V takové situaci by lidský expert přišel výhledově dráž než vývoj expertního systému, a investice do ES je tedy pro firmu výhodná. Dalším možností je potřeba uchování znalostí, například ve chvíli kdy expert odchází z firmy, a je třeba uchovat jeho znalosti a zkušenosti. Expertní systém tak stává velice vhodnou náhradou.

2.5.2 Tvorba báze znalostí

Efektivita a kvalita ES závisí největší měrou na kvalitě a obsahu báze znalostí – proto je její tvorba jedním z nejdůležitějších procesů při vytváření ES.

Tvorba báze znalostí je dlouhodobý proces sestávající z jednotlivých fází. Základem je získávání znalostí od experta a jejich kódování do vhodného tvaru, využitelného expertním systémem. K získávání znalostí dochází interakcí příslušných lidských expertů a specialisty pro tvorbu báze znalostí – znalostního inženýra (tato profese vznikla zároveň se vznikem prvních expertních systémů). Znalostní inženýr má za úkol získat od experta potřebné informace, formulovat je způsobem vhodným pro počítačovou prezentaci a následně je zakódovat.

Základní fáze tvorby BZ jsou [1, s. 44]:

– Identifikace problému – závisí na expertech popisovaného oboru. Experti musí definovat klíčové pojmy, specifikovat vztahy mezi nimi a sestavit schémata modelu realného objektu.

– Návrh koncepce báze znalostí – koncepce BZ je dána vyžadovanou formou znalostí (to většinou závisí na inferenčním mechanismu). V této fázi volíme základní principy – definujeme uzly, vztahy mezi nimi, určíme principy vkládání dat na tabuli atd.

- Volba reprezentace znalostí – určuje způsob formálního zápisu znalostí, tzn. jak zapsat získané znalosti, aby odpovídaly požadavkům ES. Příkladem může být tvar pravidel IF->THEN, nebo zapsání údajů do připravených ramců. Tato fáze je nejdůležitější a musí být provedena velice důkladně, protože nedostatky z této fáze se později již velice obtížně opravují.

- Realizace a ladění – nejpracnější fáze. V této fázi se naplno projeví nedostatky jako neúplné a chybné znalosti, je tedy stále potřeba konzultací s experty. Ladění báze se většinou provádí velkým množstvím cvičných případů, např. ve zkušebním provozu.

Samotný proces získávání znalostí může zabrat až několik týdnů či měsíců, dle rozsáhlosti projektu. Je to práce pro dva lidi (jeden expert a jeden znalostní inženýr) nebo pro více lidí (více expertů, a/nebo více znalostních inženýrů). V současné době se stávájí velice užitečnými také různé metody strojového učení [3]. Celému procesu získávání znalostí se často věnují celé publikace a zasahuje vysoko nad rámec této práce.

Vytvoření kompletní báze znalostí může trvat od čtvrt do půl roku v případě malé báze znalostí (50-350 produkčních pravidel) ale také až 5 let v případě velmi velké báze znalostí (10000 produkčních pravidel) [2, s. 120].

3. SROVNÁNÍ PROSTŘEDKŮ PRO TVORBU ES

3.1 Základní rozdělení

Prostředky pro tvorbu expertních systémů lze rozdělit do 3 skupin [4]:

a) Obecné programovací jazyky (Pascal, Delphi, C++Builder...). Expertní systémy mohou být vytvořeny téměř v jakémkoliv programovacím jazyce, tyto jazyky vám však jejich tvorbu nijak neusnadní. Přesto v nich vzniká poměrně velké množství komerčních i nekomerčních systémů.

b) Speciální programovací prostředí (CLIPS, Lisp, Prolog...). Prostředí vhodná především pro tvorbu umělé inteligence, nebo přímo pro tvorbu expertních systému a bází znalostí. V těchto prostředích vzniká velké množství expertních systémů (prázdných i kompletních), jsou uzpůsobené pro řešení problémů vznikajících při tvorbě umělé inteligence (a tedy i ES).

c) Prázdné expertní systémy - shells (EXSYS, M.4, HUGIN, RuleWorks...). Jsou to expertní systémy bez báze znalostí. Tyto systémy často mají prostředky, které uživateli pomáhají vytvořit si vlastní bázi znalostí pokud možno rychle a jednoduše. Shells jsou vhodné především pro malé a střední systémy. Jsou velice výhodné pro většinu firem, protože jsou finančně i časově méně náročné (cena těch jednoduších se pohybuje v řádu desetitisíců korun – komplexnější zahraniční systémy však dosahují ceny i několika set tisíc korun). Nevýhodou je že kvůli jejich obecnosti nedokáží vždy optimálně vyjádřit všechny znalosti a poznatky experta.

V další části práce se budeme zabývat pouze speciálními prostředími a shelly, protože standartní jazyky jsou pro potřeby ES příliš obecné a tudíž málo vhodné a používané. Bohužel z důvodu obrovského množství různých prostředků není možné popsat všechny dostupné, proto zde bude rozebrán pouze výběr těch nejznámějších a nejzajímavějších produktů.

Pro získání informací byly nejčastěji využívány oficiální webové stránky zde uvedených produktů, méně často pak části článků a recenzí volně dostupných na webu. Jelikož však každý výrobce prezentuje svůj produkt v co nejlepším světle, a každý výrobce poskytuje jiný typ informací, je toto srovnání pouze orientační a poněkud nekonzistentní. Skutečnou účinnost, přesnost, rychlost, jednoduchost, praktičnost a další popisované charakteristiky by bylo možné ověřit pouze intenzivnim testováním, které by vyžadovalo obrovské množství času, usílí a ve většině případů i peněz. A některé informace, které neposkytne přímo výrobce, nebylo možné jiným způsobem získat.

Proto jsou u každého prostředku popsány především jeho klíčové charakteristiky, principy činnosti, zajimavé vlastnosti. Když je to možné, je uvedena doba prvního vydání, poslední aktualizace, cena produktu atd. Pokud nějaká vedlejší informace chybí, pak ji pravděpodobně není možné dohledat, protože ji výrobce nezmiňuje. Pokud chybí informace o důležité vlastnosti expertního systému, buď systém tuto vlastnost nemá, nebo ji výrobce nezmiňuje.

3.2 Speciální programovací prostředí

CLIPS [5]

CLIPS je jeden z nejčastěji využívaných prostředků určených pro tvorbu ES, především díky své rychlosti, jednoduchosti a účinnosti.

CLIPS byl vytvořen v roce 1985 v NASA, v současnosti je již vyvíjen jako freeware – kdokoliv jej může používat a modifikovat zdarma. Díky jeho vlastnostem jej v současnosti využívají vlády, podniky i školy k mnoha různým účelům, a stal se základem pro mnoho prázdných systémů a jiných prostředků pro tvorbu ES.

Jeho zkratka znamená „C Language Integrated Production System“. CLIPS je napsán v jazyce C, díky čemuž je s ním výrazně propojen – přídavky a rozšíření jsou napsána v C, lze jej i zavolat jako proceduru v C. Jeho uživatelské prostředí je nejvíce podobné jazyku LISP.

Hlavními vlastnostmi CLIPSu jsou:

a) 3 způsoby manipulace se znalostmi – pravidlově řízené, objektově orientované a procedurální (reprezentace funkcemi)

b) Inferenční mechanismus – CLIPS využívá dopředného řetězení, porovnávání se vzorem a algoritmus RETE („Rete je účinný porovnávací algoritmus redukující dobu porovnávání na základě síťové struktury, ve které jsou uloženy informace o ztotožnění podmínek s fakty v bázi faktů.“ [3]).

c) Možnosti vývoje a uživatelský interface - standardní verze obsahuje interaktivní, textově orientované vývojové prostředí, včetně prostředků k ladění, on-line nápovědy, a integrovaného editoru. Dále obsahuje funkce jako je rolovací menu, integrovaný editor, či podporu více oken.

d) Začlenitelnost a rozšiřitelnost – CLIPS může být vložen do kódu, zavolán jako podprogram, či včleněn do jazyků C, Java, ADA atd. Také umožňuje své rozšiřování pomocí různých protokolů.

e) Přenosnost – díky univerzalitě jazyka C může CLIPS fungovat téměř na jakémkoliv operačním systému beze změny kódu – Windows, MacOS, Linux atd. Navíc se dodává s přístupným kompletním zdrojovým kódem, takže si jej může koncový uživatel upravit přesně dle svých potřeb a požadavků.

f) Ověřování a schvalování – CLIPS obsahuje prostředky pro kontrolu chyb a sémantické správnosti pravidel – odhaluje nekonzistentní data a zabraňuje jim neoprávněně využívat pravidla nebo generovat chyby.

g) Dokumentace – ke CLIPSu existuje velice detailní a obsáhlá dokumentace i uživatelská příručka.

CLIPS je vhodný jak pro výukové účely, tak pro firmy různých velikostí, především díky nizkým pořizovacím a udržovacím nákladům a relativní jednoduchosti vycházející z rozšířeného jazyka C.

LISP [6]

LISP byl poprvé specifikován v roce 1958, což jej činí jedním z nejstarších dosud používaných programovacích jazyků. Původně měl poměrně široké využití, v současnosti se využívá především pro potřeby umělé inteligence. Zkratka je odvozena od sousloví „LISt Processing“, protože samotný jazyk je v podstatě prostředek pro zápis algoritmů pracujících se seznamy. LISP je jedním z hlavních představitelů funkcionálního programování a v současné době má mnoho různých verzí a odnoží.

LISP dal základ pro mnoho dnes běžných programovacích technik, např. stromové struktury, automatická správa paměti atd. Lisp nerozlišuje kód a data, čímž se výrazně zjednodušuje syntaxe. Celý program je tak složen z s-výrazů nebo ozávorkovaných seznamů ve tvaru (f a b c), kde na prvním místě je operátor/funkce a na dalších argumenty funkce. Všechny další funkce jazyka mají identickou syntaxi. Jistou nevýhodou je že některým lidem se zdá syntaxe LISPu, založená především na závorkách, velice nepřehledná a neestetická. To je však pouze subjektivní nevýhoda, záleží na individualitě. Z LISPu je odvozeno mnoho dalších jazyků, většina z nich jsou různě ořezané a upravené varianty, v jiných jsou přidány různé další funkce a vlastnosti. Pravděpodobně nejpoužívanější je v současnosti Common Lisp. (V Common Lispu jsou napsány například systémy TMYCIN [9] nebo blackboardový systém GBB).

LISP je velice flexibilní jazyk, je schopen vytvořit efektivní programy s jasně definovanými významy, a je v něm tedy možné vytvořit od základu téměř jakýkoliv expertní systém pokud máme dostatečné znalosti a schopnosti. Jeho hlavní nevýhodou proti CLIPSu nebo PROLOGu je absence implementovaného inferenčního mechanismu – ten je nutno naprogramovat od počátku. Na rozdíl od PROLOGu je využíván především v USA.

PROLOG [7]

PROLOG vytvořen přibližně v roce 1972 na základě tzv. „Q-systémů“. Název vznikl z anglického výrazu PROgramming in LOGic, a jak již z názvu vyplývá, PROLOG je založen na logickém programování (na rozdíl od LISPu, jenž je založen na funkcionálnim programování).

„Prolog je jedním z mnoha programovacích jazyků používaných k neprocedurálnímu programování. Jedná se o jazyk, který je díky své jednoduché syntaxi ideální pro první seznámení s filozofií logického programování.“[7] V Evropě je v současnosti také nejpoužívanější jazyk pro tvorbu umělé inteligence, tedy i expertních systémů, protože je možno ho skvěle využít k řešení problémů, popsatelných ve formě objektů a vztahů mezi těmito objekty.

Jednou z největších výhod PROLOGu je tedy velice jednoduchá syntaxe. Díky tomu je snadno přístupný pro velký okruh lidí, kteří znají alespoň základy programování. Je také vhodný pro výukové učely, díky vysoké názornosti příkladů a principů. To však nebrání tomu, aby byl v hojné míře používán většími firmami k tvorbě velice komplexních expertních systémů a jejich doplňků. Vyhodnocovací mechanismus PROLOGu totiž funguje jako inferenční mechanismus se zpětným řetězením, na rozdíl od LISPu tedy není nutné jej od programovat. Typickým systémem vytvořeným v jazyce PROLOG je například FLEX [zdroj].

Jeho rozšířenosti napomáhá fakt, že většina jeho variant je naprosto zdarma, tudíž dostupná volně pro každého.

Shrnutí:

LISP a PROLOG jsou poměrně rozšířené programovací jazyky, vhodné právě k řešení problémů za využití UI, a tedy vhodné i pro tvorbu expertních systémů, ale i jiných aplikací. CLIPS je na rozdíl od nich zaměřený právě na tvorbu především ES, a je to spíše vývojové prostředí než skutečný programovací jazyk. Velkou výhodou všech 3 prostředků je nízká pořizovací cena, všechny jsou v nějaké podobě volně ke stažení zdarma (ačkoliv některé specialní varianty vyvíjené komerčními firmami mohou být placené).

3.2 Vývojová prostředí a prázdné expertní systémy – shells

3.2.1 Nekomerční prostředky

eXpertise2GO [8]

eXpertise2GO je zajímavý projekt vytvořený především pro propagaci expertních systémů, experimenty, výukové účely a rozšíření povědomí běžných uživatelů o expertních systémech.

Aplikace E2GSwing(dříve E2GLite) je prázdný expertní systém, zdarma použitelný přímo přes webový prohlížeč jako Java aplikace, a je doplněn kompletním návodem k vytvoření své vlastní báze znalostí, takže si tvorbu a využití ES může vyzkoušet téměř každý. Na oficiálním webu projektu jsou také články doplněné o animace a demonstrace funkcí, a články pojednávající o principech funkce ES, o způsobech reprezentace znalostí, funkce inferenčních mechanismů, zprácování nejistoty atd. Projekt tedy může fungovat jako komplexní vzdělávací pomůcka v oblasti expertních systémů. Projekt je vyvíjen od roku 2001, poslední aktualizace proběhla v roce 2009.

Charakteristické vlastnosti:

a) Znalosti reprezentuje pravidly

b) Inferenční mechanismus využívá dopředného i zpětného řetězení.

c) Je schopný zpracovávat nejistoty

d) Umí vysvětlit svá rozhodnutí

e) K provozu potřebujete pouze webový prohlížec s Java podporou

f) Podporuje mnoho jazykových verzí, včetně cizích znakových sad (azbuka, činština atd.)

Projekt samozřejmě není úplně bez chyb, webové stránky jsou poměrně nepřehledné a graficky zastaralé, aplikace také není vhodná pro komerční a ryze praktická využití. Největší uplatnění tak tedy najde ve vzdělávání, ať už na školách nebo jako materiál pro samouky.

TMYCIN [9]

Neboli „Tiny EMYCIN“, je velice jednoduchý expertní systém, který má svůj vzor v systému EMYCIN. TMYCIN se nesnaží převzít všechny vlastnosti vzorového systému, spíše přebírá ty nejdůležitější a nejvíce využívané schopnosti a prezentuje je v úsporné a jednoduché formě. Vnitřní implementace je však od EMYCINu odlišná, protože byla napsána kompletně od začátku. TMYCIN je napsán v Common LISPu a skládá se pouze z jedenácti stránek kódu.

TMYCIN je naprosto zdarma, je volně dostupný i modifikovatelný, a jeho vývoj je již ukončen. Navzdory své jednoduchosti zvládá vysvětlování svých rozhodovacích postupů a dokáže zpracovat neurčitosti. Inferenční mechanismus využívá zpětného řetězení a znalosti jsou reprezentovány pravidly. Celý systém je ovládán jednoduchými textovými příkazy.

TMYCIN je pro komerční využití příliš jednoduchý, ačkoliv si své praktické využití také našel. Nejvhodnější však je pro studijní a vzdělávací účely, jako ideální demonstrační prostředek při výuce expertních systémů.

JESS [10]

JESS je pravidlový systém napsaný kompletně v jazyce Java, vytvořený v roce 1995. JESS je malý, jednoduchý a velice rychlý systém, využívající všech možností které mu Java poskytuje. Využívá pokročilého RETE algoritmu.

JESS je zdarma pro nekomerční a vzdělávací účely, pro jeho komerční použití je nutno zakoupit licenci.

Charakteristické vlastnosti:

a) Znalosti vyjádřeny pravidly

b) Využívá Rete algoritmus

c) Inferenční mechanismus podporuje zpětné řetězení

d) Umožňuje vytvářet Java skripty, aniž by bylo nutné kompilovat jakýkoliv Java kód. (Lze vytvářet objekty, volat metody a funkce atd.)

e) Vysoká rychlost, malá velikost, příjemné ovládání

RuleWorks [11]

Je to prostředek pro tvorbu vysoce výkoných, objektově orientovaných, na pravidlech založených aplikací. Používá se při rapid-prototypingu, plánování, diagnostice, získávání dat, konfigurování komponent, kontrolování a monitorování procesů nebo jako podpora při rozhodování především v různých výrobních závodech a továrnách, ale i v bankách, pojišťovnictví, dopravě, vzdělávání a v medicíně. Ačkoliv je Ruleworks naprosto zdarma, kvalitou dosahuje některých komerčních produktů.

Charakteristické vlastnosti

a) Znalosti reprezentuje pravidly (IF-THEN/WHEN-DO)

b) Podpora objektově orientovaných datových modelů

c) Inferenční mechanismus podporuje dopředné řetězení

d) Programové prostředí mu umožňuje přímé vytváření, mazání a úpravy objektů v paměti

e) Díky jazyku C (C++) kompatibilní s většinou platforem.

EZ-Xpert 3.0 [12]

EZ-Xpert představuje revoluční způsob tvorby pravidlových systémů. Nabízí vysokou rychlost testování pravidel, přesnost a rychlost tvorby. Nevyužívá samostatný inferenční mechanismus – vše je zahrnuto v jednom kódu

EZ-Xpert je díky svým přehledným menu, jednoduchému způsobu tvorby kódu (vyplňováním tabulek a kolonek) a rozsáhlému průvodci velice snadné používat. Uživatel nemusí psát žádný kód, protože je generován automaticky, a hned připraven ke kompilaci v C++ nebo Visual Basicu. Uživateli dále poskytuje seřazený seznam činností, které je ještě nutno dokončit než bude program funčkní. Zároveň algoritmicky testuje, zjednodušuje a řeší konflikty v kódu namísto uživatele. Díky tomu je možné vytvořit standartní expertní systém s 500 pravidly přibližně za den, bez pomoci znalostního inženýra, a tedy mnohem levněji.

EZ-Xpert 3.0 je zdarma včetně jakéhokoliv dalšího šíření, práv a licencí. Výrobci pouze nabízejí volitelnou konverzi již existujících systémů s méně než 100 pravidly do EZ-Xpert za poměrně vysokou cenu $25 000.

Charakteristické vlastnosti:

a) Znalosti jsou reprezentovány pravidly. Pravidla jsou zapsána přímo v kódu celé aplikace

b) Nevyužívá samostatný inferenční mechanismus – celou aplikaci tvoří jediný kód.

c) Vysoká rychlost tvorby funkčních programů

d) Snadné používání

e) Rychlé zpracovávání požadavků – je schopen otestovat az 20 000 pravidel za sekundu.

f) Automatická kontrola – systém automaticky ověřuje kompletnost, bezkonfliktnost atd. podle 23 kritérií přímo při tvorbě systému – zkracuje čas potřebný pro testování

g) Jelikož má pouze jeden kód nahrazující inferenční mechanismus i bázi znalostí, jakékoliv další úpravy po kompilaci jsou velice obtížné a složité – v podstatě popírá základní princip expertních systémů

GBBopen [13]

GBBopen je moderní, opensourcový projekt založený na technologii blackboardu (viz. kapitola 2.3.1 Báze znalostí). Tento projekt využívá poznatků získaných z UMass Generic Blackboard systému a komerčního GBB systému k vytvoření blackboardového systému nové generace, volně přístupného širokému publiku. Prostředí je navrženo tak, aby bylo vytváření aplikací snadné a efektivní, programy byly flexibilní a účinné. Systém je naprogramován jako plynulé rozšíření Common Lispu.

GBB je hojně využíván především v armádě (Kanadský „Innovative Naval Combat Management and Decision Support system“ - INCOMMANDS, CIFAR - real-timový, několikaúrovňový systém vyhodnocování bojových situací pro US Army, radarové navádění střel atd.)

Charakteristické vlastnosti:

a) Využívá několika bází znalostí spojených přes tzv. „blackboard“

b) Znalosti reprezentovány objekty

c) Hierarchická struktura bází znalostí

d) Velice modulární struktura

e) Prostředky pro vyhledávání objektů [3]

f) Prostředky pro inteligentní řízení a integraci

g) Grafické rozhraní pro tvorbu, ladění a používání komponent

h) Moduly mohou být psány v libovolném programovacím jazyce

i) Otevřené a libovolně rozšiřitelné prostředí díky opensource licenci

3.2.2 Komerční prostředky

EXSYS [14]

EXSYS je velice oblíbený a často využívaný komerční shell, provozují jej velké i menší firmy, vlády i univerzity. Vyvíjený je přibližně od roku 1983 až dodnes. EXSYS se vždy snažil vytvořit co nejpřehlednější a nejjednodušší systém, který může používat i člověk bez větších zkušeností s programováním.

Zatím poslední verze – EXSYS Corvid – byla poprvé představena roku 2001. Shell je vytvořen v jazyce Java, lze jej tedy provozovat přes webový prohlížeč, což značně zvyšuje možnosti jeho využití.

Tento systém stojí přibližně $12000 za jednu vývojářskou licenci a jednu licenci pro provoz produkčního serveru. Firma také nabízí za $200 na hodinu služby znalostního inženýrství, je tedy možné si objednat expertní systém i s vytvořením báze znalostí přímo na míru zákazníka – a cena se bude odvíjet podle velikosti a složitosti báze znalostí. Také nabízí školicí kurzy pro studenty a zaměstnance a mnoho dalších služeb, jež dohromady tvoří velice slušné zázemí.

Firma má na svém webu mnoho informací o systému a především zkušební verze již hotových systémů, které si případný zájemce může zdarma vyzkoušet, je možné si stáhnout i omezenou 30ti denní verzi. Nechybí také on-line tutoriály a návody, které mohou pomoci nezkušenému zákazníkovi s používáním systému.

Charakteristické vlastnosti:

a) Znalosti vyjádřené jednoduchými a přehlednými pravidly ve tvaru IF-THEN-ELSE – je snadné je číst, pochopit a upravovat

b) Vyznačuje se jednoduchou a rychlou tvorbou báze znalostí

c) Inferenční mechaismus využívá kombinaci dopředného i zpětného řetězení.

d) Schopný zpracovávat nejistoty a vysvětlovat rozhodnutí

e) Podporuje mnoho různých platforem

f) Umí využívat principu tabule – blackboardu

g) Velice profesionální zázemí firmy s dlouholetými zkušenostmi na poli expertních systémů

HUGIN [15]

HUGIN je expertní systém využívaný především ve finančnictví k odhalování pojistných a finančních podvodů, ale také v lékařství, při odhadování rizika, k řešení oborově specifických problémů a stanovování bezpečnosti. Jeho hlavním úkolem je podpořit rozhodování člověka, obvykle se nepoužívá k samostnému řešení problémů.

HUGIN je jeden z hlavních průkopníků využití Bayesovských sítí při zpracování neurčitosti a jejich využití v praxi. („*Bayesovská síť* je orientovaný acyklický graf, jehož uzlům odpovídají náhodné proměnné a vazby reprezentují kauzální závislosti mezi těmito proměnnými. Hrana $X \rightarrow Y$ znamená, že X kauzálně ovlivňuje Y (pozorování X poskytuje kauzální podporu Y, pozorování Y poskytuje diagnostickou podporu pro X). Bayesovská síť umožňuje provádět prediktivní i diagnostické inference.“ [3 s.44])

Systém je vyvíjen přibližně od roku 1989 a stále se rozvíjí. Firma od svého založení vyvíjí aplikace pro různá použití. Pro komerční účely dodává software, které umožňuje vytváření vlastních

aplikací, nebo ořezanou verzi která slouží pouze k využívání již vytvořených aplikací atd. Obdobné rozdělení existuje pro vzdělávací software, jehož využití spočívá především v seznámení se s Bayesovskými sítěmi a dalšími principy.

HUGIN je komerční/vzdělávací produkt, nejvyšší komerční verze HUGIN Developer v síťové licenci (více než 10 PC) stojí přibližně 30 000$, cena vzdělávací verze s víceméně stejnými možnostmi se pohybuje kolem 7 500$. Firma nabízí zdarma k vyzkoušení i omezenou testovací verzi HUGIN Lite.

Charakteristické vlastnosti:

a) Využívá objektově orientovaných Bayesovských sítí

b) Automatické získávání bází znalostí z databází

c) Dodává se s kompletní knihovnou všech dostupných bází znalostí a s kompletními návody a příručkami k použití

d) Velice vhodný pro zpracovávání neurčitosti

e) Propracovaný grafický interface pro 5 různých programovací prostředí: C, C++, .NET, Java, Active-X

f) Podpora naprosté většiny moderních platforem a systémů

XpertRule [16]

XpertRule je velice vyspělý systém, vyvíjený postupně od založení firmy v roce 1985 až dodnes, využívaný především v poradenství a řešení problémů (call centra, zákaznický servis, webový techsupport atd.). Využívá pravidlově řízených bází znalostí a rozhodovacích stromů, podle nichž řídí své rozhodovací procesy.

Jednou z nejvýraznějších vlastností je velice pokročilý systém pro rozeznávání psaného textu, který dokáže zpracovat (například mail od zákazníka s popisem problému), vyextrahovat z textu odpovědi na otázky obsažené v jeho rozhodovacím stromu a následně automaticky odpovědět s popisem možného řešení. Pokud mu některé informace chybí, je schopen si e-maily s dotazy (nebo jinou formou) vyžádat potřebné informace. Pokud ani poté není schopen vyřešit problém sám, pak zkontaktuje operátora/technika a přiloží všechna dosud získaná data. Tento postup velice zjednodušuje práci například zákaznického servisu, kdy dokáže část problémů vyřešit bez cizí pomoci a u ostatních případů alespoň velice pomůže lidskému technikovi radami a předáním relevantních informací.

Toto je samozřejmě jen příklad jeho funkce, jeho využití je mnohem širší – například vypočítávání přesných cen produktů s různými příplatky a všemi poplatky (pojištění, prodej strojů...), pomoc s výběrem zboží nebo s konfigurací výrobku dle přání zákazníka, odhadování rizik při plánování procesů a mnoho dalších.

Přesná cena pro komerční použití není známa, pro vzdělávací účely však jedna licence stojí 795£.

Charakteristické vlastnosti:

a) Znalosti jsou reprezentované pravidly

b) Využívá pravidla graficky zpracované do rozhodovacích stromů a tabulky příkladů – přehlednější a snadnější udržovatelné při větším počtu pravidel než standartní IF-THEN-ELSE reprezentace.

c) Schopen zpracovávat neurčitosti

d) Pokročilá schopnost zpracovávání psaného textu

e) Výborný grafický interface, jenž umožňuje předávání poznatků, dat, pravidel atp. bez znalosti programování – výrobce uvádí až 20-ti násobnou časovou úsporu proti programování v .NET nebo jazyce Java -> velice snadná správa báze znalostí.

f) Systém je možné plně provozovat přes webové servery

g) Schopen se sám učit ze svých předchozích rozhodnutí a jejich důsledků

M.4 [17]

Výborný nástroj pro tvorbu expertních systémů, naprogramovaný v jazyce C. Nabízí možnost použití metod umělé inteligence a inteligentního zpracování dat v programech třetích stran. M.4 má mnoho typických využití (výběr, analýza, plánování, konfigurace, diagnostika, vzdělávání) v různých odvětvích (továrny, finance, obchod, správa, zpracování dat).

Důležitou součástí M.4 je Kernel knihovna, balík podprogramů nepříliš využívaný u konkurenčních nástrojů, přesto velice užitečný. Díky jeho flexibilitě je možno z M.4 během okamžiku odstranit nepotřebné části (interface, různé moduly atd), a vložit jej do jakéhokoliv programu napsaného v jazyce C. Poté lze pouze jednoduchým zavoláním funkce využít inteligentní expertízu poskytovanou tímto nástrojem. Díky modularitě nástroje je také možné kdykoliv měnit, upravovat, či odstraňovat jeho moduly, či dokonce upravovat samotný kód, a užívat tak jen součásti, které jsou pro uživatele skutečně potřebné.

M.4 byl prodáván za cenu $1200 za jednu licenci, nabízí však i levnější síťové licence (přesná cena však není uvedena). Vývoj softwaru již byl ukončen.

Charakteristické vlastnosti:

a) Využívá objektově orientovanou reprezentaci znalostí – třídy, instance, metody dědičnosti, předávání dat

b) Inferenční mechanismus využívá dopředné i zpětné řetězení a porovnávání se vzorem

c) Jeho funkce zahrnují zpětné stopování činností, rekurzi a iterace, zpracování symbolických seznamů, kontrolování hodnot a další.

d) Systém má schopnost pracovat s neurčitostmi

e) Systém umí vysvětlit svá rozhodnutí a postupy

f) Snadná včlenitelnost do dalších programů v C

g) Díky jazyku C je snadno přenosný mezi většinou platforem

G2 [18]

je komplexní prvotřídní vývojové prostředí od firmy GENSYM pro tvorbu, implementaci a údržbu aplikací běžících v realném čase a založených na principu umělé inteligence.

G2 má obrovské pole působnosti, uplatňuje se v diagnostice a monitorování, optimalizaci, rozvržení výroby, řízení, simulování, modelování atd. Schopnost pracovat v reálném čase jej činí velice efektivním a užitečným nástrojem, uplatňuje se například při řízení bezpilotních letadel.

Schopnostem a možnostem tohoto řešení odpovídá i cena, pohybující se mezi $6 000-$40 000 za licenci, záleží na konfiguraci nástroje, přídavných modulech, nastavení, extra požadavcích atd.

Charakteristické vlastnosti:

a) Objektově orientované prostředí – umožňuje vyjádřit pravidla, metody, procedury, vše za využití přirozeného jazyka – umožňuje vývojářům snadno pochopit, testovat a upravovat jednotlivé modely

b) Podpora neuronových sítí („Umělá neuronová síť je distribuovaný výpočetní systém sestávající z dílčích podsystémů (neuronů), který je inspirován neurofyziologickými poznatky o struktuře a činnosti neuronů a nervových systémů živých organizmů, a který je ve větší či menší míře realizuje.“ [19])

c) Díky architektuře klient-server umožňuje pracovat několika uživatelům zároveň

d) Velice vysoká rychlost zprácování dat – naplno využívá potenciálu hardware na kterém funguje

e) Schopnost pracovat kompletně v realném časem včetně úprav aplikace za chodu

f) Multitasking – umí zpracovat několik různých souběžně běžících událostí nebo datových toků zároveň

g) Výborně zpracované grafické prostředí

h) Na rozdíl od většiny ostatních nástrojů nabízí kompletní podporu při tvorbě aplikací – od návrhu, přes vývoj, simulace a testování, uvedení do provozu až po následnou údržbu

i) Bohatá podpora platforem a propojitelnost s mnoha dalšími systémy (databáze, řídící systémy)

FLEX [20]

FLEX je hojně využívané vývojové prostředí pro tvorbu pravidlových expertních systémů implementované v PROLOGu (je založen na logickém programování). FLEX poskytuje mnoho funkcí umožňujících programátorům i nezkušeným uživatelům vytvářet a udržovat velice solidní a dobře udržovatelné aplikace. FLEX využívá k definování pravidel, rámců a procedur vlastní speciálně navržený jazyk Knowledge Specification Language (KSL), jenž podobný přirozenému anglickému jazyku. Díky tomu je užívání FLEXu velice intuitivní a příjemné i pro IT laiky.)

FLEX je vyvíjen společností Logic Programming Associates Ltd, která si za jednu vývojářskou licenci účtuje 1495£ + další částky za různé podpůrné aplikace.

Charakteristické vlastnosti:

a) Znalosti reprezentovány pravidly a rámci

b) Rámce jsou uspořádaný hierarchicky s mnoha stupni dědičnosti.

c) Inferenční mechanismus využívá dopředné i zpětné řetězení

d) Logické programování

e) Nevýhodu logického programování při zpracování neurčitosti kompenzuje využíváním Fuzzy logiky, Bayesovských aktualizací a faktorů určitosti

f) Přehledné grafické prostředí s jasně zobrazenými stromovými strukturami mezi rámci, s možností upravovat prvky přímo v kódu

g) Prostředí je kompatibilní s jazyky C#, .NET, Java. Aplikace vytvořené ve FLEXu také mohou být nahrány přímo na web

4. ZÁVĚR

Toto téma jsem si vybral z části kvůli zájmu o studium na zastřešujícím ústavu (Ústav automatizace a informatiky), z částí kvůli osobním sympatiím k danému tématu. Jako dlouholetý hráč PC her jsem částečně seznámen s problematikou umělé inteligence v zábavním průmyslu, proto jsem se začal zajímat i o využití UI v průmyslu, lékařství a dalších oborech. Expertní systémy se tedy zdály být ideálním spojovacím článkem obou bodů mého zájmu.

V první části této práce jsem se pokusil vysvětlit naprosto základní principy expertních systémů, tak aby byly srozumitelné i čtenáři, který se s tématem dosud nesetkal (což jsem byl na počátku práce i já sám). Využíval jsem k tomu z velké části literaturu poskytnutou vedoucím práce, z menší části různé další zdroje, především publikace volně dostupné na internetové síti. Struktura práce se částečně odvíjí ze základní struktury poskytnutých knih, doplněné mými vlastními poznatky a informacemi, získanými poměrně intenzivním studiem dané problematiky. Vše je přehledně označeno bibliografickými citacemi (dle platné normy), často včetně stránek ze kterých byly čerpány stěžejní informace.

Ve druhé části se věnuji srovnání, charakteristice a popisu nejznámějších, nejzajímavějších nebo nejpoužívanějších prostředků pro tvorbu expertních systémů. Kapitolu jsem rozdělil na prázdné expertní systémy, dále rozlišené na komerční a nekomerční, a na speciální programová prostředí (ta zde popisovaná jsou všechna zdarma, tudíž nebylo nutné je dále rozdělovat). Pro získávání potřebných informací jsem využil především oficiální stránky produktů, občas doplněné kusými informacemi z jiných zdrojů. Každý prostředek je popsán několika krátkými odstavci se základními informacemi pokud byly dostupné (cena, vývojář nebo firma, stáří, poslední aktualizace, nejzajímavější vlastnosti kvůli kterým jsem si je vybral do své práce). Dále jsou pomocí textových odrážek vypsány další charakteristické vlastnosti systémů v organizované posloupnosti: první reprezentace znalostí a vlastnosti báze znalostí, následně vlastností inferenčních mechanismů, schopnost systému řešit neurčitosti, schopnost systému vysvětlovat svou činnost, informace o grafickém prostředí, přenosnost mezi platformami, detaily o jazyku v němž je systém napsán a nakonec další zajímavé vlastnosti. Toto pořadí je pouze informativní, ale mělo by přibližně odpovídat skutečnosti. Pokud nějaké informace nejsou uvedeny, výrobce je buďto nezmiňuje a nebo jimi systém nedisponuje. Popsány jsou především vlastnosti a charakteristiky navazující na první, teoretickou části této práce.

Mým původním záměrem bylo, kromě popisu základních charakteristik těchto prostředků, také jejich kompletní srovnání, porovnání podle vlastností, reprezentace znalostí, inferenčních mechanismů, stavu vývoje, kým byl vytvořen, v jakém jazyce byl naprogramován a podobně. Ideálně formou tabulky nebo jiným přehledným způsobem. Při této snaze jsem však narazil na závažný problém – nekonzistenci údajů. Téměř u každého nástroje byly uvedeny jiné druhy informací, někde jsem nenašel žádné údaje o datu poslední aktualizace nebo prvního vydání, často chyběly informace o autorech nebo v jakém programovacím jazyce je nástroj vytvořen. Jinde výrobce především nezmiňoval žádné informace o způsobu funkce inferenčního mechanismu, často nebyla dostupná data o způsobu reprezentace znalostí. U některých jsem dokonce velice těžko zjišťoval zda-li se jedná o prázdný systém nebo spíše vývojové prostředí, proto tyto kategorie poněkud splývají a jsou spojeny do jedné kapitoly. Snažil jsem se tyto vlastnosti dohledat i pomocí dalších zdrojů, má snaha však byla neúspěšná. Taková srovnávací tabulka by tak byla z velké části nekompletní a prakticky nevyužitelná.

Proto jsem se rozhodl vytvořit srovnávací tabulku, vytvořenou dle mých subjektivních znalostí o problematice, získaných během řešení práce. Při vytváření této tabulky jsem vybral několik, pro potenciálního uživatele důležitých, charakteristik a ohodnotil jsem je zvlášť pro každý systém. Za tyto charakteristiky považuji: cenu (pro 10 licencí nebo síťovou licenci), uživatelskou přívětivost (tedy jak snadno se systém ovládá), vhodnost pro začátečníky (jak hluboké znalosti programování a ES jsou potřeba pro dostatečné využití potenciálu nástroje), univerzalitu (rozsah úkolů pro které je systém vhodný), pro koho je systém vhodný (malé nebo velké firmy, školy) a celkovou výbavu (přídavné moduly, zvláštní funkce, výbava navíc atd.). Tyto charakteristiky jsou ohodnoceny pouze subjektivně a orientačně, na stupnici 1-3 (1 nejnižší, 3 nejvyšší). Bližší info pak lze dohledat v samotné práci.

Srovnávací tabulka:

	Cena 10 licencí	*Uživatelská jednoduchost*	*Vhodnost pro začátečníky*	*Univerzálnost řešených úloh*	*Vhodný pro*	*Softwarová vybavenost*
CLIPS	-	2	2	2	Vše	3
LISP	-	1	1	3	Vše	1
PROLOG	-	2	2	2	Vše	2
E2GSwing	-	3	3	1	Vzdělávání	2
TMYCIN	-	2	2	1	Vzdělávání, malé firmy	1
JESS	-	2	2	2	Malé firmy	2
RuleWorks	-	2	1	2	Malé firmy	2
GBBopen	-	1	1	3	Vše	3
EZ-Xpert	-	3	3	1	Malé i velké firmy	2
EXSYS	$12 000	2	2	3	Vše	3
HUGIN	$30 000	2	1	2	Vzdělávání, velké firmy	3
XpertRule	8 000£+	3	3	2	Vše	3
M.4	$12 000	2	1	3	Malé firmy	2
G2	$38 000+	1	1	3	Velké firmy	3
FLEX	15 000£	3	2	2	Vzdělávání, velké firmy	3

Celkovým záměrem mé práce bylo zhotovit dokument, po jehož přečtení by čtenář získal základní informace o tématu, a případný zájemce o využití expertního systému by měl mít představu o výhodnosti a vhodnosti různých prostředků pro jejich tvorbu, včetně jejich vlastností a klíčových charakteristik.

Jako vhodné rozšíření práce bych doporučoval zvětšit množství popsaných prostředků a alespoň ty neplacené z nich objektivním způsobem otestovat, aby se potvrdily vlastnosti udávané jejich výrobci.

SEZNAM POUŽITÝCH ZDROJŮ

[1] GOSMAN, Svatoslav. Umělá inteligence a expertní systémy. Praha: Kancelářské stroje, 1990. 168 s. ISBN 80-7018-004-8.

[2] POPPER, Mikuláš; KELEMEN, Jozef. Expertné systémy. 1. vydaníe. Bratislava: Alfa, vydavateľstvo technickej a ekonomickej literatúry, 1989. 360 s. ISBN 80-05-00051-0.

[3] DVOŘÁK, Jiří. *Expertní systémy* [online]. 2004 [cit. 2011-04-11]. Dostupné z: <http://www.uai.fme.vutbr.cz/~jdvorak/Opory/ExpertniSystemy.pdf>.

[4] ŠTÝBNAROVÁ, Libuše. *Hypertextová příručka k předmětu Znalostní a expertní systémy* [online]. [cit. 2011-04-10]. Softwarové prostředky. Dostupné z WWW: <http://ui.fpf.slu.cz/diplomky/znalostni_a_expertni_systemy/def/Soft.html>.

[5]RILEY, Gary. *CLIPS : A Tool for Building Expert Systems* [online].Sourceforge.net, March 13, 2011 [cit. 2011-05-20]. Dostupné z WWW: <http://clipsrules.sourceforge.net/>.

[6]KUKLÍNEK, Jan. *Fakulta informatiky Masarykovy univerzity* [online]. [cit. 2011-05-21]. LISP. Dostupné z WWW: <http://www.fi.muni.cz/usr/jkucera/pv109/2005/xkuklin.htm>.

[7]RUBÁČEK, Filip. *Prolog: Programování v Prologu* [online]. 1999, 2005 [cit. 2011-05-21]. Dostupné z WWW: <http://iris.uhk.cz/logpro/iteorie.html>.

[8]*Expertise2GO : Web-enabled expert systems* [online]. 2001, 2009 [cit. 2011-05-22]. Dostupné z WWW: <http://www.expertise2go.com/webesie/e2gdoc/>.

[9]NOVAK JR., Gordon. *TMYCIN: expert system shell* [online]. 7.10.2003 [cit. 2011-05-23]. Dostupné z WWW: <http://www.cs.utexas.edu/~novak/tmycinb.html>.

[10]FRIEDMAN-HILL, Ernest. *JESS: the Rule Engine for the Java Platform* [online]. 10.9.1997, 8.11.2008 [cit. 2011-05-23]. Dostupné z WWW: <http://www.jessrules.com/>.

[11]*RuleWorks: Programming by rules* [online]. 2011 [cit. 2011-05-23]. Dostupné z WWW: <http://www.ruleworks.co.uk/index.html>.

[12]*EZ-Xpert 3.0: Inference Without Engine!* [online]. 12.8.2005 [cit. 2011-05-23]. Dostupné z WWW: <http://www.ez-xpert.com/index.html>.

[13]The GBBopen project [online]. 2011 [cit. 2011-05-25]. Dostupné z WWW: <http://gbbopen.org/index.html>

[14]*EXSYS: Knowledge Automation Expert System Technology* [online]. 2011 [cit. 2011-05-22]. Dostupné z WWW: <http://www.exsys.com/index.html>.

[15] *HUGINEXPERT: The leading decision support tool* [online]. 2011 [cit. 2011-05-22]. Dostupné z WWW: <http://www.hugin.com/>.

[16] *XpertRule* [online]. 2011 [cit. 2011-05-23]. Dostupné z WWW: <http://www.xpertrule.com/>.

[17]*Hallogram Publishing* [online]. 1998 [cit. 2011-05-24]. M.4. Dostupné z WWW: <http://www.hallogram.com/m4/index.html>.

[18]*California State University Northridge* [online]. [cit. 2011-05-24]. Gensym G2. Dostupné z WWW: <http://www.csun.edu/~hceen007/gensym_g2.htm>.

[19] JIŘINA, Marcel. *Neuronové sítě* [online]. 29.6.2002, 9.5.2006 [cit. 2011-05-24]. Dostupné z: <http://gerstner.felk.cvut.cz/biolab/33KP/prednasky_ann/prezentace_ns.ppt>.

[*20*]*Logic Programming Associates Ltd* [online]. [cit. 2011-05-25]. Flex Toolkit. Dostupné z WWW: <http://www.lpa.co.uk/flx.htm>.