UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY

A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2009 Tomáš Málek

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Návrh expertního systému

Tomáš Málek

Bakalářská práce

2009

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Katedra informačních technologií
Akademický rok: **2008/2009**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Tomáš MÁLEK**

Studijní program: **B2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Návrh expertního systému**

Zásady pro vypracování:

Cílem práce bude navrhnout a vytvořit expertní systém pro diagnostiku potíží s IT technikou (např. potíže s připojením k internetu, hardware apod.). První část bakalářské práce bude zaměřena na teorii kolem expertních systémů a problematiku, pro kterou bude systém navrhován. Expertní systém bude vytvořený jako internetová aplikace s využitím jazyka PHP a databáze My SQL.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

MAŘÍK V., ŠTĚPÁNKOVÁ O., LAŽANSKÝ J., Umělá inteligence (1), 1. vydání - dotisk, Praha : Academia, 2004. s. 264. ISBN 80-200-0496-3. MAŘÍK V., ŠTĚPÁNKOVÁ O., LAŽANSKÝ J., Umělá inteligence (2), 1. vydání, Praha : Academia, 1997. s. 15 ? 68. ISBN 80-200-0504-8. Další zdroje na internetu.

Vedoucí bakalářské práce: **Ing. Pavel Škrabánek**
Katedra řízení procesů

Datum zadání bakalářské práce: **15. ledna 2009**

Termín odevzdání bakalářské práce: **15. května 2009**

Univerzita Pardubice Česká republika

L.S.

doc. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan
vedoucí katedry

V Pardubicích dne 31. března 2009

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární zdroje a informace, které jsem při jejím zpracování využil, jsou uvedeny na konci v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.
Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

Ve Ždírci nad Doubravou 12. 8. 2009

Tomáš Málek

Anotace

V teoretické části jsou popsány expertní systémy, jejich historie, vlastnosti a možnosti jejich využití. Dále jsou zde podrobněji rozebrány expertní systémy diagnostického charakteru. Tyto znalosti jsou poté použity při realizaci praktické části.

Klíčová slova

expertní systém; diagnostický expertní systém; PHP; MySQL

Title

Proposal of expert system

Annotation

The theoretical section describes expert systems, their history, characteristics and how to use them. There are also further analyzed the nature of diagnostic expert systems. These skills are then used in the implementation of the practical part.

Keywords

expert system; diagnostic expert system; PHP; MySQL

Obsah

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

CD	Compact Disc
CSS	Cascade Style Sheets
E-R diagram	Entity Relationship Diagram
GNU	GNU's Not Unix
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HW	Hardware
JVM	Java Virtual Machine
LGPL	Lesser General Public License
OS	Operating System
PC	Personal Computer
PHP	PHP: Hypertext Preprocesor
SQL	Structured Query Language
URL	Uniform Resource Locator

1. Úvod

V dnešní době, kdy informační technologie jsou čím dál rozšířenější, ve všech možných odvětvích lidské činnosti, je pro drtivou většinu uživatelů výpočetní techniky velmi důležité, aby je jejich hardware nezklamal (a to nejen v osudnou chvíli). Když už ale takováto situace nastane, mnohdy nemusí být žádný technik, který by závadu odstranil, právě na blízku. Právě proto je velmi důležité, aby i průměrně zdatní uživatelé výpočetní techniky věděli, jak se s menšími závadami na počítači svépomocí vypořádat a kam sáhnout v případě poruchy svého stroje.

Tato práce si klade za cíl navrhnout a poté zrealizovat jednoduchý expertní systém, pro diagnózu a návrh řešení problémů s hardwarovým vybavením počítače. Takový systém by měl být co možná nejpružnější, protože některé závady hardwaru spolu velmi úzce souvisí. Při řešení problémů se tudíž může velmi lehce stát, že se od jedné diagnózy, na kterou ukazovala valná většina symptomů, které chování systému vykazovalo, dostaneme k diagnóze úplně jiné.

V první části této práce jsou popsány expertní systémy, je zmíněna jejich historie, přiblíženy jejich vlastnosti a také možnosti využití. Detailněji jsou rozebrány diagnostické expertní systémy, protože jejich vlastnosti, jsou pro tuto práci stěžejní.

V další části jsou uvedeny základní prostředky pro realizaci aplikace. Konkrétně relační databázový systém MySQL a skriptovací jazyk PHP, jejich základní popis a vlastnosti. Jsou zde také uvedeny důvody, proč bylo vybráno právě toto řešení, dále pak jeho výhody a nevýhody.

V části následující je popsána praktická část této práce, a sice návrh systému sloužícího k diagnostikování závady na hardwaru a její následné řešení. Je zde uveden postup samotného návrhu aplikace, zvažovaná kritéria a přiblížení postupu její realizace.

V závěrečné části práce se nachází zhodnocení a shrnutí realizované aplikace, včetně jejích nedostatků a doporučení, pro jejich nápravu, tudíž i k dalšímu zdokonalení této aplikace.

2. Expertní systémy

2.1. Vymezení pojmu expertní systém

Hned na začátku by bylo vhodné odpovědět na základní otázku. Co jsou to vlastně expertní systémy? Tato úloha však není nijak jednoduchá, obzvlášť je-li požadováno uvést krátkou a výstižnou definici. Oficiální definice pojmu expertní systém, totiž snad ani neexistuje. Z hlediska teoretického i implementačního není velký rozdíl mezi systémy expertními a znalostními. Jediným opravdovým rozdílem by snad mohla být kvalita poskytovaných znalostí. Neexistuje však kritérium, které by znalostní systémy od expertních systémů jednoznačně odlišilo a oddělilo. Zde jsou uvedeny některé možné definice expertního systému.

Expertní systém je:

- počítačový systém založený na reprezentaci poznatků, získaných od špičkových expertů, které potom využívá pro řešení zadaných problémů.
- počítačový program simulující chování a uvažování experta při řešení obtížných úloh ze specifických oblastí.
- počítačový systém, jenž hledá řešení problému v rozsahu určitého souboru tvrzení nebo seskupení znalostí, které byly formulovány experty pro specifickou oblast.
- počítačový systém, vybavený znalostmi experta (z určité oblasti), v jejichž rozsahu je schopný určit rozhodnutí (řešení).
- počítačový program, který má za úkol poskytovat expertní rady, rozhodnutí nebo doporučit řešení v konkrétní situaci. [3]

Jedním z prvních a nejdůležitějších aspektů při tvorbě expertního systému, který je nutné, ještě před začátkem jeho vývoje, potřeba promyslet je, zda se pro tuto oblast vůbec vyplatí systém realizovat. Je třeba si odpovědět na otázku, zda je vhodné řešit právě tento druh problémů ze specifické oblasti pomocí expertního systému.

2.2. Vlastnosti expertních systémů

Ačkoliv se může zdát, že realizací expertního systému se tvůrce snaží co nejlépe napodobit chování, myšlení a uvažování odborníka na daný typ problému, opak je pravdou. Cílem realizace expertního systému totiž není vyvinout systém uvažující jako člověk, včetně lidských chyb a omylů, ale systém, který dosáhne co nejlepších reakcí na reálná data, tj. co nejlepšího způsobu vyhodnocování a rozhodování.

Architektura expertních systémů se skládá z tzv. *báze znalostí*, kde jsou uloženy explicitně vyjádřené znalosti experta a *řídícího mechanismu*, pomocí něhož je určena strategie pro využití znalostí z báze. Díky této vlastnosti systému je zajištěná vysoká znovupoužitelnost systému.

Báze znalostí obsahuje odborné znalosti získané od experta v daném oboru, potřebné k řešení daného problému. Jsou zde znalosti, které jsou běžně dostupné například z učebnic či návodů, ale také znalosti soukromé, které expert svými zkušenostmi z oboru sám, za dobu po kterou se problémem zabývá, získal a které mu v některých případech při řešení problémů pomohli. Těmto (exaktně nedokázaným) znalostem pramenícím ze zkušeností se říká heuristické. Přitom právě rozsah a kvalita těchto speciálních heuristických znalostí činí z pracovníka v oboru experta. Podobné to je i s bází expertního systému, čím více takových informací obsahuje, tím je lepší.

Podobně jako expert za svoji praxi prochází jistým vývojem z hlediska nasbíraných zkušeností, tak i expertní systém se postupně vyvíjí a zdokonaluje. Požadavkem na bázi znalostí proto také je, aby se nechala lehce modifikovat a doplňovat o nové znalosti. Zároveň musí být přehledná, aby se v ní expert, zadávající nové poznatky, snadno zorientoval.

Komunikace uživatele s expertním systémem je obdobná komunikaci s expertem. Probíhá tedy v dialogovém režimu, přičemž uživatelem může být laik nebo méně zkušený odborník a expertní systém zde zastupuje špičkového experta na danou oblast. Expertní systém pokládá uživateli otázky, na které mu tento odpovídá. Dle získaných odpovědí od uživatele a svých poznatků si upřesňuje „představu“ a buď určí závěr a řešení nebo formuluje další otázku.

Podobně jako expert by i expertní systém měl být schopný pracovat s neurčitými znalostmi (z báze znalostí) a zároveň s nejistotou uživatele při zodpovídání dotazů položených systémem, tedy s nejistotou v tzv. *bázi dat*. Báze dat je výraz užívaný pro označení množiny všech údajů k danému případu (získaných od uživatele). Ke zpracování neurčitosti bohužel neexistuje žádný univerzální postup a bohužel asi ani nikdy nebude, protože ošetření neurčitosti bude vždycky do jisté míry závislé na způsobu implementace.

Některé expertní systémy mají dialog s uživatelem nahrazen jiným způsobem získání dat, například měřením údajů na reálných objektech pomocí měřící techniky nebo hledáním v databankách.

Dalším možným požadavkem na expertní systém je to, aby k závěrům, které určil po dialogu s uživatelem, přidal rozumné vysvětlení a zdůvodnění proč právě takováto diagnóza je ta pravá a popsal postup, jak postupovat při řešení problému.

Některé z uvedených vlastností nemusí být u vyvíjeného expertního systému vůbec vyžadovány. Například existují expertní systémy, které vůbec nepracují s neurčitostí.

2.3. Historie expertních systémů

Pojem expertní systém se poprvé objevil na přelomu sedmdesátých a osmdesátých let minulého století jako důsledek poznání faktu, že kvalita systémů s umělou inteligencí závisí daleko více na kvalitě obsažených informací než na kvalitě programových mechanismů pro jejich využití. Expertní systémy se od začátku vyznačují tím, že jsou založeny na odborných znalostech špičkových expertů ze specifických oblastí.

Díky úspěchům prvních expertních systémů se o nich rychle začalo hodně mluvit a to nejen v odborných kruzích. V odborných časopisech často vycházely velmi nadsazené články, které zapříčinili velká, mnohdy až nereálná, očekávání. Tak došlo k tomu, že se pojem expertní systém stal částečně zamlženým. A i díky tomu výše zmíněná přesná definice pojmu expertní systém zatím neexistuje.

V prvních letech vývoje expertních systémů se očekávalo vytváření několika problémově nezávislých a široce využitelných systémů s všeobecně platnou standardizací. S postupem času však většinou začal být preferován spíše charakter

speciálních, problémově orientovaných a ad hoc vytvořených subsystémů v rámci rozsáhlých programových celků. V porovnání s minulostí, kdy byly preferovány samonosné expertní systémy, jsou v současnosti naopak upřednostňovány vnořené aplikace. [1]

2.4. Rozvoj a využití expertních systémů

V poslední době se objevuje obrovské množství expertních systémů v nejrůznějších aplikačních oblastech a to jak diagnostických, plánovacích tak i hybridních. Nejméně polovina z nich je pak svým určením nasazena v oblasti medicíny, snad právě proto, že v této oblasti jsou znalosti nejlépe strukturálně rozčleněny. Každoročně je do přehledu existujících systémů zaváděno opravdu veliké množství systémů nově vzniklých. Někdy jsou dokonce referovány pouze nové báze znalostí, které jsou spojovány s řídícím mechanismem již vzniklým a opakovaně používaným.

Z pohledu historie se ukazuje, že několik prvních systémů zásadně ovlivnilo a dále také ovlivňuje některé zásadní směry ve vývoji dalších expertních systémů. Skoro každý nově vzniklý systém má přímou nebo nepřímou vazbu na některý systém vzniklý již v minulosti. Tyto dědičné rysy jsou na některých expertních systémech z hlediska jejich architektury, reprezentace nebo využívání znalostí dosti znatelné.

Ve vývoji expertních systémů lze odlišit následující čtyři důležité etapy:

Počáteční fáze (1965-1970):

Expertní systém DENDRAL byl vyvinut v letech 1965-1969 na univerzitě ve Stanfordu. Jde o typický plánovací expertní systém, určený k odvozování struktur chemických sloučenin. Jeho existence je datována dříve, než se vůbec objevil pojem expertní systém. Byl tedy absolutně prvním expertním systémem. Dodnes se využívá a ve své úzce specifické problémové oblasti dosahuje lepších výsledků, než špičkoví experti.

Úkolem systému MACSYMA poskytuje soubor nástrojů pro manipulaci s matematickými vzorci a výrazy. Využíván je hlavně v oblasti nukleární fyziky, při řešení soustav velkého počtu rovnic.

Etapa výzkumných prototypů (1970-1975):

Systém MYCIN patří k nejznámějším expertním systémům. Vyvinut byl na Stanford Research Institute. Jedná se o systém sloužící k určení bakteriální infekce ze základních dostupných dat a k určení vhodné medikace pro stabilizování stavu pacienta. Tento systém není ani tak významný z pohledu medicínského, jako z pohledu umělé inteligence. Hlavně jde o metody zpracování neurčitosti.

Expertní systém PROSPECTOR, který byl také vyvinut na Stanford Research Institute, je určen pro vyhodnocení dostupných geologických dat a rozhodnutí, zda se v dané lokalitě vyplatí provést hloubkové vrty.

Na systému HEARSAY-II se poprvé ukázalo, že počítač může spolehlivě „rozumět“ přirozené řeči v rozsahu úzce specifikované oblasti. Jeho hlavní zásluhou je rozvoj výzkumu v oblasti porozumění a porovnávání řeči.

I když systémy DENDRAL, MYCIN a HEARSAY disponují zcela odlišnými architekturami, prokázala se díky nim, životaschopnost expertních a znalostních systémů. Postupně se začali nasazovat i v jiných oblastech, než pro které byly původně vytvořeny s cílem otestovat opětovnou použitelnost a rozšíření oblastí použití. Ze systémů MYCIN a PROSPECTOR byly poté vyčleněny problémově nezávislé části. Ty byly potom používány jako jádro nově vznikajících systémů. Byly tak vytvořeny tzv. *prázdné expertní systémy* neboli *pouzdra*. Ze systému PROSPECTOR vznikl prázdný systém jménem KAS. Ze systému MYCIN systém EMYCIN.

Etapa experimentálního nasazování (1975-81):

Na základě prázdných systémů pak byly vytvářeny další poměrně úspěšné problémově orientované expertní systémy. Zde je několik příkladů.

Systém PUFF byl založen na základě systému EMYCIN a jedná se o systém poskytující konzultace v oblasti potíží dýchacích cest.

Úkolem systému SACON, jenž využívá architekturu systému MYCIN, je poskytování konzultací k obsáhlému softwarovému balíku MARC, sloužícího k analýze metodou konečných prvků.

Dalšími systémy postavenými na architektuře MYCIN jsou např. ONCOCYN, sloužící k řízení léčby pacientů na onkologii, HEADMED, jenž se využívá v klinické psychofarmakologii, CLOT, určený pro diagnostiku srážlivosti krve, nebo DART, který byl vyvinut pro diagnostiku závad počítačů.

Nejznámějším systémem, založeným na architektuře PROSPECTOR, je systém AL/X, který diagnostikuje závady na technologickém zařízení platformy pro těžbu ropy v Severním moři.

Ukázkou systému založeného na architektuře HEARSAY-II může být pak systém HASP, který byl určen pro práci s daty, získanými ze sonaru. Za cíl měl určit pozici lodí, jejich typ a orientaci.

V tomto období experimentování ale nevznikali systémy založené pouze na jádrech systémů již existujících. Vznikali také nové typy expertních systémů se svým vlastním (nepřevzatým) typem architektur. Za zmínku stojí především systém INTERNIST, později modifikovaný a přejmenovaný na CADUCEUS, jenž svojí bází znalostí pojal více jak tři čtvrtiny veškerých znalostí z oblasti interní medicíny.

Etapa komerčně dostupných systémů (od roku 1981):

K prvním komerčně nasazovaným expertním systémům, které se objevily, beze sporu patří XCON a XSEL. Tyto systémy vycházejí ze svého předchůdce R1, což byl systém zpracující objednávky počítačů VAX používaný firmou DEC.

Systém DIPMETER ADVISOR byl vyvinut pro několik naftařských společností, se zaměřením na poskytování rad ohledně obsluhy vrtných zařízení. Zároveň byl zaměřen na postupy v případě problémů s hloubkovými vrty. Zajímavostí může být, že náklady na vývoj přesáhly 21 milionů dolarů, ale uživatelé (těžební společnosti) udávali návratnost v řádech měsíců až týdnů.

Opakovatelná využitelnost expertních systémů není taková, jak se z počátku očekávalo. Důvodem je, že obecná použitelnost systémů je obvykle limitována na určitou problémovou oblast nebo omezený počet typů úloh. U plánovacích expertních systémů je dosažitelná opakovaná použitelnost téměř nulová, protože většina aplikací vyžaduje od počátku vývoje specifickou reprezentaci znalostí. [1]

2.5. Úlohy vhodné pro řešení

Protože i expertní systémy mají pouze omezené možnosti, je ještě před zahájením vývoje takového systému potřeba rozmyslet si, zda je právě expertní systém nejvhodnějším řešením, pro řešení zadaného problému. To potom záleží hlavně na charakteristických vlastnostech úlohy. Expertní systém prakticky modeluje způsob uvažování, jakým řeší daný úkol expert. Ten se při řešení opírá o znalosti, jak obecně známé, tak i o tzv. heuristické znalosti, které získal ze svých zkušeností v problematické oblasti během svého působení v oboru. Pokud tedy záleží řešení zadaného typu úloh právě na heuristických znalostech a tyto znalosti jsou řešiteli realizujícímu systém známé či přístupné, pak je vhodné expertní systém realizovat.

Při výběru způsobu řešení je jedním z nejdůležitějších kritérií *složitost úlohy*. Jestliže se jedná o úlohu, která není složitá (tedy takovou, kterou je běžně možné vyřešit bez asistence experta), tak lze jen prozkoumat všechna možná řešení a následně z nich určit to, které bude nejvhodnější. V tom případě není potřeba ani expertní systém. Složitost se nechá odhadnout podle toho, jak rychle roste počet kroků řídícího algoritmu vzhledem k rostoucímu rozsahu úlohy. Opravdu složité úlohy ani řešit klasickými sériovými algoritmy nelze. Protože nejsou zatím známé žádné vhodné optimalizační algoritmy, tak jsou místo těchto složitých úloh řešeny úlohy takové, které jsou uvedeným úlohám blízké, a tudíž mohou sloužit jako kvalitní náhrada. Řešení určené pomocí heuristických znalostí samo o sobě neposkytuje, nýbrž nahrazuje optimální řešení. To platí i pro expertní systémy, protože tyto znalosti používají. Významně to souvisí s faktem, že ani expertní systém není neomylný. Dokonce se může stát, že systém řešení zadaného problému vůbec nenalezne. To však není důsledkem selháním systému, jeho báze znalostí nebo řídícího mechanismu, ale právě použitím heuristických znalostí. Tudíž pokud lze zadanou úlohu definovat tak, aby jí zvládal nějaký jednoduchý algoritmus, budou jeho výsledky vždy přesnější, než výsledky takovéhoto expertního systému.

2.6. Architektury expertních systémů

Podle určitosti zaměření lze dnes existující expertní systémy rozdělit na prázdné a dedikované. Dále pak podle typu řešených úloh je možné je rozčlenit do třech kategorií, a to na systémy plánovací, diagnostické a hybridní.

2.6.1. Prázdné

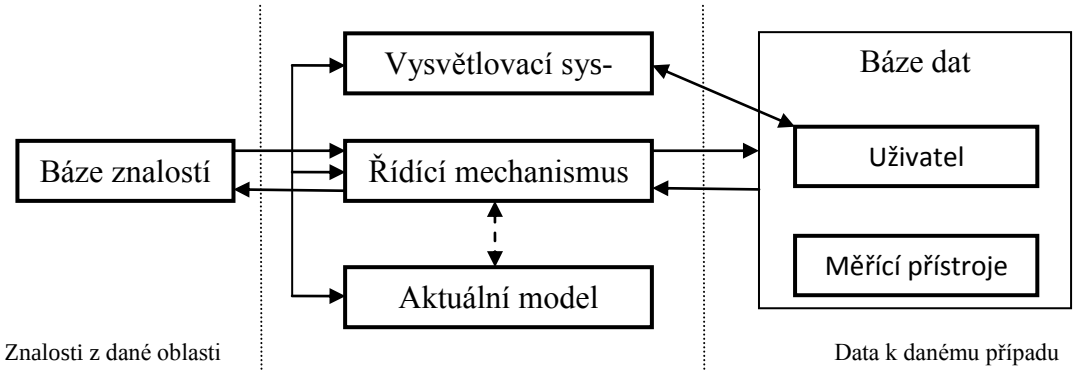
Ještě než budou popsány jednotlivé typy, bylo by vhodné zmínit se o prázdných expertních systémech. Prázdnými expertními systémy jsou obvykle označovány takové systémy (a to jak diagnostického, tak i plánovacího charakteru), které jsou bez problémově závislých částí. To znamená, že se jedná o systémy, které nemají bázi znalostí ani bázi dat. Teprve doplněním báze znalostí se prázdný expertní systém orientuje a stává se tak z něj problémově orientovaný expertní systém, to znamená systém určený, pro řešení určité problematiky. Dodáním další části, a sice báze dat, je poté řešen konkrétní případ. Vývoj prázdných expertních systémů byl preferován hlavně v počátku jejich rozvoje. Tehdy byla patrná snaha o vytváření opakovatelně použitelných systémů, nebo alespoň jejich problémově nezávislých částí, tedy řídících a vysvětlovacích mechanismů. Později, kdy již bylo jasné, že každá aplikační oblast a každá kategorie úloh vyžaduje poněkud specifičtější řídící mechanismy a odlišnou reprezentaci znalostí, bylo od vývoje opakovatelně využitelných expertních systémů postupně upouštěno. Na závěr zmínky o tomto typu, by bylo vhodné poznamenat, že prázdné expertní systémy se podařilo vyvinout pouze pro řešení diagnostických úloh. Zbylé dva typy mají totiž svoji efektivitu značně závislou na použité reprezentaci znalostí, jejíž problémová závislost je většinou hodně vysoká, tudíž jsou pro vývoj prázdných expertních systémů nevhodné a v praxi nerealizovatelné. [1]

2.6.2. Dedikované

Pro řešení úloh z určité oblasti jsou určeny *dedikované* expertní systémy. U tohoto druhu systémů je jejich vlastní architektura, řídící mechanismus i samotný typ reprezentace znalostí specifickým způsobem svázány s oblastí problémů, pro jejichž řešení je systém vyvíjen. Tyto dedikované systémy mají pak sice charakter prázdných systémů, ale použitelné jsou pouze v rámci určité oblasti. Přeorientování na jiný druh problémů by bylo velice složité a v mnohých případech dokonce až nemožné.

2.6.3. Diagnostické

Úkolem diagnostických expertních systémů je provádět efektivní interpretaci dat s cílem určit, která z předem stanovených hypotéz nejlépe souhlasí s reálnými daty týkajícími se konkrétního daného případu. Systém si z dialogu s uživatelem nashromáždí odpovědi na své otázky a podle nich pak určí diagnózu. Následně může dle určené diagnózy navrhnout uživateli postup při řešení problému, přičemž může i vypsat zdůvodnění, proč právě tato určitá diagnóza je ta hledaná a proč právě navrhované řešení je to pravé, které problém nebo závadu odstraní. Více podrobností o tomto typu je uvedeno v posledním oddíle této kapitoly (2.8). Na následujícím obrázku je zobrazeno blokové schéma architektury typického diagnostického expertního systému.



Obrázek 1. Blokové schéma architektury diagnostického expertního systému

2.6.4. Plánovací

Pomocí plánovacích expertních systémů jsou obvykle řešeny takové úlohy, kde je předem určen počáteční stav a cíl, kterého je třeba dosáhnout. Úkolem tohoto systému je pak pomocí známých dat o řešeném problému navrhnout co možná nejoptimálnější postup (posloupnost dílčích kroků), jak tohoto cíle dosáhnout. Částí tohoto typu systémů je generátor operátorů, který automaticky kombinuje posloupnosti operátorů. V tomto typu systémů se využívá znalostí experta a dat o reálném případu k omezení kombinatorické exploze, která je důsledkem vytváření posloupnosti kroků, při jejich rostoucím počtu. Činností plánovacího expertního systému je vytvořen seznam navrhovaných postupů (řešení), kde každý

postup je označen jistou mírou optimálnosti. Seznam je uložen do zásobníku vhodných řešení, který se při řešení úkolu mění. Řídící mechanismus pak s pomocí obou bází (dat a znalostí) ovlivňuje výběr přípustných operátorů, omezuje generativní schopnost generátoru s použitím báze znalostí a řídí testování shody generovaných řešení s daty z báze dat. Tím také vytváří zásobník řešení a nastavuje jejich optimálnost. [1]

2.6.5. Hybridní

Hybridní expertní systémy jsou charakteristické tím, že mají svoji architekturu kombinovanou. Z části využívají diagnostických postupů a z části plánovacích. Typickým představitelem těchto systémů jsou systémy, které nejdříve testují diagnózu sledovaného subjektu a při zjištění určitého stavu, je spuštěn proces plánování dalšího postupu.

2.7. Základní kameny expertních systémů

Expertní systémy se v oblasti základů umělé inteligence těší výsadnímu postavení. To právě proto, že jejich předmětem je návrh a tvorba samostatně existujících, v praxi využitelných systémů na bázi umělé inteligence. Nejvíce je sledována problematika tří zásadních oblastí, a sice:

- reprezentace znalostí
- řešení řídících mechanismů
- práce s neurčitou informací

2.7.1. Reprezentace znalostí

Základním problémem pro tvorbu expertního systému je beze sporu efektivita při reprezentaci znalostí. Reprezentace znalostí musí být provedena tak, aby:

- byla pro danou oblast dostatečně přirozená i expresivní
- umožnila aplikování deduktivních prostředků
- umožnila rychlý přístup k bázi znalostí i bázi dat

Zároveň je nutné při návrhu reprezentace znalostí dbát na to, aby byla *modulární* a lehce *modifikovatelná*. Tento požadavek je důležitý hlavně proto, aby se báze znalostí nechala udržovat v aktuálním stavu a obohacovat o nové znalosti z dané oblasti. Díky tomu je zároveň umožněno, aby báze byla použitelná jak pro expert-

ní systém, pro řešení úloh, tak pro člověka k učení. Báze znalostí je totiž zdrojem koncentrovaných znalostí, vhodných i k výuce teorie v dané oblasti. Modularita reprezentace znalostí v bázi zajistí snadné lokální změny znalostí. Zbytek báze se tak nezmění. Toto se však také může stát nevýhodou při používání báze. Při vyhledávání v bázi totiž musí řídící mechanismus mnohdy projít bázi celou, protože není sémanticky uspořádaná. Je-li báze rozsáhlá, znamená to značné zatížení a časovou náročnost. Dále je možné, že se některé poznatky mohou v bázi vyskytovat víckrát.

Z toho plyne další důležitý požadavek na reprezentaci, a sice na její *sémantické sdružování*. Tento požadavek je dán potřebou rychlého vybavení znalostí a vytvoření hierarchie pojmů. Tato hierarchie pak dovoluje postupovat směrem od pojmu obecného ke speciálnímu a naopak, nebo odvození na základě analogie.

Znalosti bývají, z hlediska problematiky reprezentace v systémech umělé inteligence, velmi často děleny do dvou skupin. A to na:

- znalosti **reprezentované deklarativně**
- znalosti **reprezentované procedurálně**

Znalosti reprezentované deklarativně (jindy též nazývané „poznatky“) vyjadřují, co je, či teprve má být poznáno, nebo dokázáno. Naopak znalosti reprezentované procedurálně (označovány též „pravidla“) popisují, jak poznávat či odvozovat. Reálné systémy většinou obsahují oba dva typy reprezentací znalostí a není jednoduché je od sebe odlišit. Při tvorbě expertních systémů je prakticky povolena jakákoliv reprezentace znalostí, která je náležitě efektivní (např. i kombinovaná).

2.7.2. Konceptualizace a ontologie

Řešením reálných úloh se většinou omezujeme jen na určitou oblast, některé její objekty uvažujeme, jiné nepodstatné schválně ignorujeme. Soubor všech objektů potom tvoří tzv. *universum* dané úlohy. Určení tohoto universa definováním všech jeho objektů a vymezením vztahů mezi nimi se pak označuje *konceptualizace*. Explicitní popis konceptualizace se potom nazývá *ontologie*. Důležité je, aby tato ontologie byla reprezentována nezávisle na způsobu jejího budoucího použití. Ontologie doplněné o znalosti o způsobu řešení tvoří vlastní bázi znalostí. Tato báze je pak již problémově zaměřená na určitý typ úlohy a nějaké určité řešení.

2.7.3. Řídící mechanismy

Řídící mechanismus zajišťuje využívání znalostí zavedených v systému. Nějaká kompaktní teorie řídících nebo odvozovacích mechanismů zatím nespatřila světlo světa. I přes všechny rozdíly v jednotlivých realizacích systémů je ve většině případů použito strategie *prohledávání stavového prostoru*. Většinou však v reálné aplikaci tento přístup nestačí. Právě proto se používají některé další způsoby řízení. Zde jsou uvedeny některé další známé způsoby:

- **dekompozice úlohy**
- **hledání v hierarchicky uspořádaném stavovém prostoru**
- **vyjádření znalostí formou metapravidel**

K realizaci řídících mechanismů se často používají také techniky, které s řešením úloh ve stavovém prostoru bezprostředně až tak nesouvisí. Následující výčet uvádí alespoň tři nejznámější z nich:

Agenda

Podstatou **Agendy** je, že v průběhu běžícího procesu řešení úlohy se jako vedlejší produkt vytváří a postupně upravuje zásobník, do kterého se průběžně přidávají další úkoly, které by měly být prioritně řešeny, jakmile skončí stávající dílčí úkol.

Démoni

Metoda využití **Démonů** vychází z představy o existenci jakýchsi démonů, tiše sledujících inferenční proces a čekajících na určitou událost. Nastane-li tato situace, pak zasáhnou do procesu. Proto se při každém kroku řešení testuje, zda nejsou splněny podmínky pro vyvolání démonů.

Tabule

Použití **Tabule** předpokládá rozdělení báze znalostí na několik bází menších. Komunikace mezi dílčími bázemi pak probíhá přes tzv. tabule, což jsou sdílené datové a řídící struktury.

2.7.4. Neurčitost

Na závěr by bylo vhodné se zmínit alespoň okrajově o zpracování neurčitých informací. Neurčitost při práci se znalostmi a daty je v expertních systémech dosti typickým jevem, i když s ní každý expertní systém nutně pracovat nemusí.

Nejistota nebo neurčitost bývá v expertních systémech obvykle vyjadřována pomocí číselných parametrů nejčastěji desetinného charakteru. Tyto parametry se v různých systémech mohou nazývat různě, většinou však jde o míru důvěry, stupně důvěry, faktor jistoty apod. Zmíněné číselné parametry jsou pak přiřazovány k jednotlivým elementům reprezentace znalostí, např. k pravidlům, rámcům apod. Typickým intervalem pro vyjadřování neurčitosti bývá rozmezí od nuly do jedné.

2.8. Přiblížení diagnostických systémů bez neurčitosti

V úvodu návrhu je třeba si teoreticky stanovit ontologii problémové oblasti. Tato ontologie musí obsahovat množinu prvků, podstatných pro danou úlohu, dále pak množinu možných symptomů a množinu možných závad. Dále je nutné definovat relace mezi jednotlivými objekty universa podstatné pro danou úlohu. Ontologii součástí si lze představit jako určitý seznam, nezávislý na typu dané úlohy. Ontologie závad je již vázaná na samotnou diagnostiku a lze ji použít i pro jiné účely (např. pro posuzování kvality, nebo spolehlivosti). Spojením jednotlivých částí pak dojde ke vzniku diagnosticky zaměřené ontologie. Je-li sestavená ontologie problémově nezávislá, potom vyžaduje doplnění dalších znalostí. Je-li tomu naopak a ontologie je problémově orientovaná, tedy spojuje jednotlivé pojmy vztahy závislými na typu úlohy, je reprezentuje takováto ontologie již bázi znalostí.

2.8.1. Diagnostické uvažování

Na začátku je třeba si vytvořit předběžný model diagnostického procesu, který je nutné řídícím mechanismem vytvořit. Z charakteristiky tohoto modelu vyplynou dílčí požadavky na jednotlivé odvozovací a řídící procesy. Spolu se začátkem diagnostického procesu je nutné vložit počáteční informace o symptomech pozorovaných při poruše. Z nich jsou potom stanoveny hypotézy o možných diagnózách.

2.8.2. Výpočetní model

Práce systému je vlastně určena výpočetním modelem. V tomto modelu jsou heuristické znalosti reprezentovány pomocí produkčních pravidel, jež mají jak deklarativní tak procedurální sémantiku. Pravidla se definují ve tvaru podmíněných příkazů. Levá strana (podmínka) je označována jako *antecedent*, pravá strana (výraz) potom *konsekvent*. Samotný inferenční (odvozovací) mechanismus může zpracovat pravidla dvojím způsobem.

- **Dopředné řetězení (inference řízená daty)** - při splnění antecedentu se vykoná akce konsekventu, při tom dojde k odvození nových dat a jejich uložení do pracovní paměti (odvozená data jsou používána v dalších krocích). Nové odvozování je spuštěno při novém vložení dat do pracovní paměti a ukončeno, když už není další pravidlo pro odvození dalších dat.
- **Zpětné řetězení (inference řízená cílem)** - mechanismus se pokouší splnit zadaný cíl. Jestliže je cíl určen konsekventem, stačí nalézt antecedent.

Které pravidlo bude použito, pokud pro daný soubor dat platí antecedent více pravidel, určuje mechanismus *řešení konfliktů*. V dopředném řetězení se pro řešení konfliktů používají tři dílčí kritéria.

- **Specifičnost pravidla** - je-li antecedent konjunkcí více podmínek je vybráno pravidlo, které má antecedent tvořen větším počtem podmínek.
- **Neopakování pravidla** - pokud bylo pravidlo na data použito, znovu se již na ta samá data neaplikuje. Zamezí se tím nekonečným cyklům.
- **Odvozování z novějších dat** - je-li v daném okamžiku možno aplikovat více pravidel, použije seto, které využívá novějších dat.

Negace je vhodné vyjádřit prefixem „not“. Lze k ní přistupovat ve stylu předpokladu uzavřenosti světa, čímž je myšleno, že negace výroku plyne z jeho nepřítomnosti v pracovní paměti.

Je vhodné používat *strukturovanou reprezentaci* znalostí i dat a tak definovat, že některé údaje patří k sobě. Proto je vhodné používat reprezentaci pomocí rámců a např. používat i odvozování v jejich hierarchii.

2.8.3. Jazyk pro reprezentaci úlohy

Podle uvedených vlastností diagnostického procesu je vhodné si určit jazyk pro reprezentaci dat a znalostí.

- **Pravidla** - mají svůj identifikátor, je typu FORWARD nebo BACK-WARD, jeho tělo je definováno podobně jako tělo metody objektu, jsou v něm použity podmíněné výrazy s použitím antecedentu a konsekventu.
- **Rámce** - jsou to vyjádření objektů, se kterými budou pravidla pracovat. Častými vztahy mezi objekty jsou vztahy instance třídy a podtřídy. Ty jsou dvou základních typů a to PART_OF a IS_A. Rámec, jenž je instancí nebo podtřídou jiného rámce automaticky dědí všechny jeho položky, hodnoty však nemusí. Strategie přidání hodnoty vyjadřuje zdědění hodnoty položky z nižšího stupně hierarchie do všech stejnojmenných položek získaných děděním.
- **Jednoduchá data** - jednoduché výroky jako start apod.

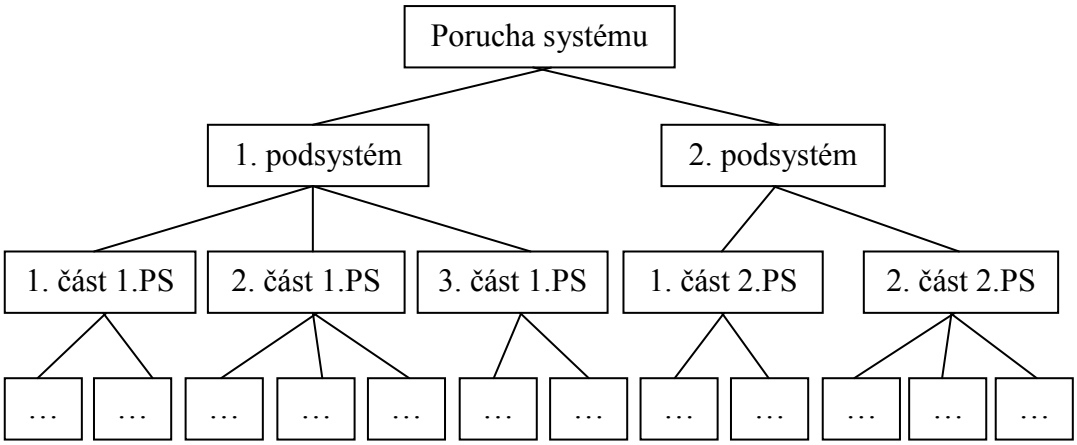
2.8.4. Postup řešení diagnostické úlohy

- Určení základní architektury
- Vytvoření řídících pravidel
- Vytvoření diagnostických pravidel
- Upřesňovací pravidla zpětného zřetězení
- Výběr diagnózy, která je řešením úlohy

2.8.5. Reprezentace diagnóz pomocí rámců

- Vytvoření hierarchie diagnóz
- Reprezentace diagnóz prostřednictvím jazyka rámců
- Definování řídících pravidel
- Vytvoření diagnostických pravidel
- Zavedení specializujících pravidel zpětného zřetězení
- Provedení dalšího upřesnění diagnózy pomocí zpětného zřetězení

Následující obrázek vykresluje typický příklad stromové hierarchie diagnóz.



Obrázek 2. Příklad stromové hierarchie diagnóz

2.8.6. Výhody reprezentace diagnóz pomocí rámců

Hlavní výhodou reprezentace diagnóz pomocí rámců, s využitím relací PART_OF a IS_A, je to, že je díky tomu lze přehledně a přirozeně strukturovat. Navíc je dosaženo daleko úspornějšího vyjádření znalostí. Celé soubory pravidel jsou nahrazeny jen malým počtem pravidel, jejichž počet nezávisí na velikosti báze znalostí a použité hierarchii rámců.

3. Základní popis prostředků k vytvoření aplikace

3.1. Výběr prostředků pro realizaci aplikace

Jelikož se má dle zadání jednat o aplikaci internetovou, je zapotřebí uvažovat tak, aby vytvořená aplikace byla spustitelná pod různými platformami. Možným řešením by mohlo být využití programovacího jazyka Java k vytvoření klasické okenní aplikace s přístupem k databázi. Problém by ovšem mohl nastat v případě absence JVM (Java Virtual Machine) na uživatelském PC. Ten samý problém by nastal v případě, že by byl řídící částí internetové aplikace javovský aplet. Nabízí se proto klasické řešení pomocí skriptovacího jazyka PHP a databázového systému MySQL. Rozhraním aplikace tedy bude klasická internetová stránka, zobrazovaná ve webovém prohlížeči. Chování expertního systému bude řízeno skriptem v PHP a používané znalosti budou uloženy v databázi.

3.1.1. Relační databázový systém MySQL

MySQL je označení nejoblíbenějšího a nejvyužívanějšího systému pro správu relační databáze v oblasti internetových aplikací. V porovnání s konkurencí je to jeden z nejrychlejších databázových systémů vůbec. Navíc je to Open Source projekt, což u databázových systémů není zvykem. Snad právě proto je tak oblíbený. Jedná se o systém typu klient/server. V tomto případě bude klientem aplikace v PHP, která bude MySQL serveru zasílat SQL dotazy a ten na ně bude vracet odpovědi. Ty budou posléze zpracovány PHP skriptem. Samotnou databázi tvoří tabulky. Každá tabulka může mít sloupce různých datových typů, do kterých se nechají zadat různé hodnoty. Tyto zadané hodnoty mohou být omezeny a to jak rozsahem, tak například proti vynechání. To znamená, že lze u povinných údajů vynutit, aby po vložení řádku tabulky (záznamu) nebyl údaj vynechán a aby v něm nebyla tzv. neurčitá hodnota (NULL). Zároveň lze v tabulkách definovat tzv. primární a cizí klíče, pomocí nichž lze potom tabulky spojovat a zajistit tak vztahy mezi tabulkami (relace). Díky nim je zajištěna tzv. referenční integrita.

Z historie MySQL se sluší alespoň uvést, že vznik MySQL je datován v roce 1994. Dalšími dvěma historickými milníky jsou rok 2008, kdy firmu MySQL AB koupila firma Sun Microsystems a další rok 2009, kdy tuto firmu i s MySQL koupil gigant na poli relačních databázových systémů, a sice Oracle Corporation.

3.1.2. Skriptovací jazyk PHP

Jedná se o interpretovaný programovací jazyk, jehož syntaxe je velmi podobná jazyku C. Zkratka PHP znamená Hypertext preprocesor. Samotný název jazyka prošel několika změnami, to však není až tak podstatné. Slouží ke generování dynamických internetových stránek. Všechny výpočetní operace se provádějí na serveru (serverový jazyk) a ten pak klientovi pošle výsledek ve formě HTML stránky. Z toho plyne i multiplatformnost aplikací v něm vytvořených. Snadno také komunikuje s MySQL i jiným relačními databázemi. V současné době podporuje PHP i objektové programování. Vývoj PHP jde stále kupředu a přibývají další a další technologie a funkce. I PHP je Open Source projekt, a i proto se těší velké rozšířenosti a oblíbenosti v řadách vývojářů. K velkým výhodám patří také široká podpora komunity a dostupnost již hotových zdrojových kódů, které jsou šířeny nejčastěji pod volnou licencí a tudíž je možné je ihned použít ve vytvářené aplikaci.

3.2. Kritéria výběru technologií

Z hlediska databáze je zapotřebí takový systém, který bude umožňovat vytvoření stromové architektury, pro vytvoření báze znalostí.

Z hlediska klientské aplikace je třeba vytvořit editační rozhraní, kde expert bude vkládat své znalosti. Zároveň tento přístup musí být chráněn, proti zneužití.

Potom bude také potřeba vytvořit rozhraní uživatelské. Tedy rozhraní pro běžné uživatele, kteří budou potřebovat radu od expertního systému a nebudou v něm mít právo cokoliv měnit.

3.3. Výhody a nevýhody

Hlavní výhodou internetové aplikace v PHP je, že je přístupná všude, kde je nějaká možnost se k internetu připojit. Na internetu se k ní může dostat kdokoliv, kdo bude potřebovat. Dalším plusem může být, že není nutná žádná instalace (ani této aplikace ani žádných dalších podpůrných prostředků).

Naopak nevýhoda internetové aplikace je celkem jasná. Uživatel z nějakého důvodu přijde o možnost se k aplikaci dostat - např. porucha připojení, modemu, nebo PC a pak mu nezbývá, než to zkusit odjinud.

4. Popis vytvářené aplikace

4.1. Úvodní popis aplikace

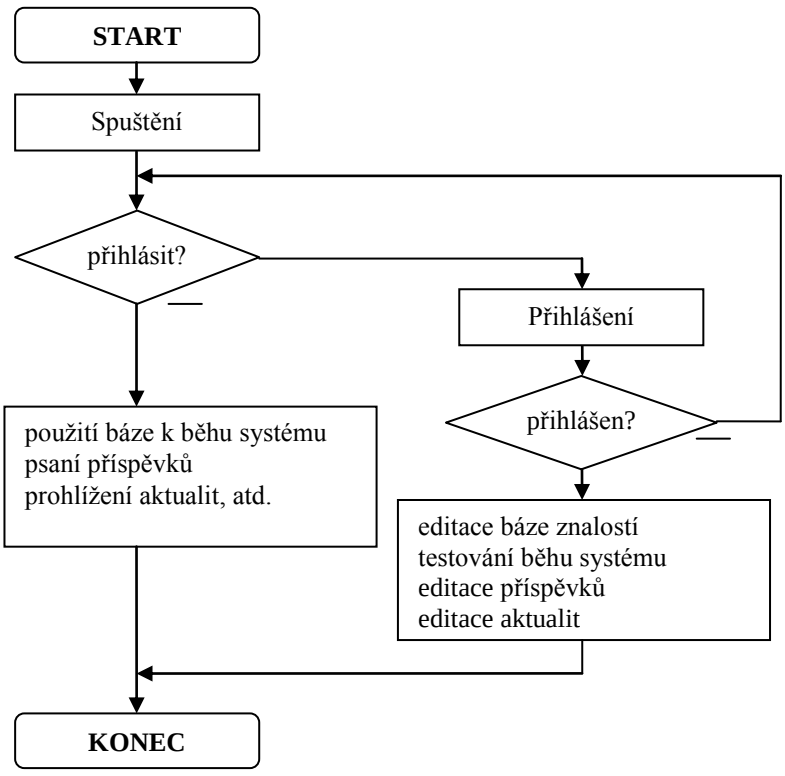
Aplikace má sloužit jako diagnostický expertní systém pro řešení technických potíží s výpočetní technikou.

Je nutné sestavit vhodnou databázovou strukturu, tedy strom. Na tomto stromu bude následně sestavena báze znalostí. Zároveň bude nutné uchovat přihlašovací údaje, pro možnost ověření identity před editací báze znalostí.

Klientská část bude sloužit k obsluze uživatelů. Pohyb ve stromu bude realizován pomocí tlačítek (eventuálně odkazů). Editační část bude sloužit pro zadávání, úpravě nebo mazání expertových znalostí ve stromu (bázi znalostí) s možností vytváření, na sebe vzájemně navazujících, větví stromu. Zároveň bude chráněná proti přístupu neoprávněných uživatelů.

4.2. UML Activity Diagram

Následující obrázek znázorňuje způsob běhu aplikace z hlediska uživatelských oprávnění a možností funkcí, které tyto práva uživateli umožňují používat.



Obrázek 3. UML Activity diagram

4.3. Adresářová struktura

Následující tabulka popisuje adresářovou strukturu vytvářené aplikace včetně umístění jednotlivých souborů.

adresář:	obsah:
css	soubory kaskádových stylů pro formátování rozložení stránky
include	PHP skripty, obsahy stránky a objekty, které jsou do aplikace vkládány příkazem include()
menu	menu vkládaná do těla stránky
nbproject	adresář vytvořený implicitně vývojovým prostředím NetBeans
scripts	adresář určený k ukládání skriptů v jazyce javascript
secure	v tomto adresáři jsou uloženy informace, pro připojení k databázi
tela	uložená předdefinovaná těla k vložení příkazem include()

Tabulka 1. Adresářová struktura a umístění souborů aplikace

4.4. Požadavky pro spuštění aplikace

Aplikace ke svému běhu vyžaduje:

- webový server s PHP (Apache)
- databázový server MySQL

Obě dvě uvedené služby jsou zdarma dostupné pro použití na více operačních systémech. Pro spuštění aplikace na localhostu je případně možné použít balík XAMPP. Tento balík totiž obsahuje obě uvedené služby.

Za použití tohoto balíku (konkrétně ve verzi XAMPP 1.7.0) byla také aplikace vytvořena. Vyvíjena a laděna byla ve vývojovém prostředí NetBeans IDE (ve verzi 6.5.1) pod OS Windows Vista Home Premium v 32bitové verzi. Testována pak byla ve webových prohlížečích Mozilla Firefox (verze 3.5.2), Opera (verze 9.64) a MS Internet Explorer (verze 8.0).

4.5. Technologie použité k implementaci

K již zmíněným prostředkům, zvoleným pro vytvoření aplikace je vhodné dodat, které konkrétní technologie byly využity při jejím vývoji.

Bázi znalostí, tvoří pouze jedna databázová tabulka, na které je postavena datová struktura strom. Je to důležité vzhledem k záměru vytvořit hierarchické uspořádání znalostí. Proto bylo nutné použít tabulku (resp. zpracovatele tabulek) typu InnoDB, která umožní využití cizích klíčů, potřebných pro vytvoření stromové struktury.

Základní stavební kámen aplikace tvoří objektová část, naprogramovaná v PHP, ve které je implementován řídící mechanismus. Ta je podrobně popsána, včetně ukázek kódu, v samostatném oddíle (viz kapitola 4.7.1). Využití objektového programování je velkou výhodou, protože metody objektu lze napsat pouze jednou a pak je používat na více místech v aplikaci a to velmi jednoduše. Stačí si vytvořit instanci objektu a zavolat její metodu.

Neméně důležitou technologií použitou pro kontrolu identity aktuálního uživatele je mechanismus Sessions. Využívá se tak, že na začátku se zapne (odstartuje), poté při přihlášení uživatele jsou do něj (resp. do jeho asociativního globálního pole *$_SESSION*) zadány údaje o přihlášeném uživateli, ve formě jednotlivých proměnných (login, password, atp.). Díky tomu je, při vstupu do administrátorského rozhraní, možné tyto údaje (resp. samotnou jejich „nastavenost“) kontrolovat pomocí funkce *isset()*, nebo pomocí porovnání obsahu proměnných uvedeného globálního pole s přihlašovacím jménem uživatele uloženého v databázi, či v nějaké konstantě.

Další neméně důležitá technologie je CSS, která se stará hlavně o vizuální vzhled a rozvržení stránky. Využito je externího souboru (*.css), ve kterém je pro jednotlivé elementy HTML kódu (obalovací tagy div, tlačítka atd.) vytvořeno speciální nastavení určující, jak a kde se má každý objekt zobrazovat a jak se má chovat (např. při pohybu kurzoru přes jeho plochu). Tento soubor je pak importován do hlavního souboru (index.php) v oblasti hlavičky HTML kódu.

4.6. Databázová část

4.6.1. Tabulky

Při návrhu tabulek bylo nutné uvažovat tak, aby se na struktuře hlavní tabulky, sloužící k uchování znalostí, nechala implementovat datová struktura zvaná *strom*. V tabulce nad níž tento strom budu realizovat, bude tedy nutné uchovávat kromě id záznamu, ještě id předchozího záznamu, tedy jeho předka ve stromové struktuře. Bylo tedy nutné, z hlediska potřeby využití cizích klíčů, jak již bylo uvedeno, využít zpracovatele tabulek InnoDB, jenž práci s cizími klíči podporuje. Nastavit typ zpracovatele, ze standardního MyISAM na InnoDB, lze již při samotném vytváření tabulky, nebo i dodatečně použitím v příkazu alter.

Zároveň bylo nutné zohlednit možnost editování záznamů v databázi, kvůli údržbě a aktuálnosti dat v bázi znalostí expertního systému. Následující E-R diagram ukazuje strukturu tabulek, využitých pro uchování dat, včetně struktury relací. Za povšimnutí stojí především relace položek id a id_predka, která je typu 1:N. Pomocí tohoto vztahu je zajištěno hierarchické větvení stromu dotazů a odpovědí.

Obrázek 4. E-R diagram

4.6.2. Indexy

Z hlediska možností databáze MySQL je vhodné využít indexů, sloužících pro urychlení vyhledávání a tudíž zlepšení celkové odezvy SQL dotazů. Naštěstí v tabulkách jsou tyto indexy již zavedeny díky tomu, že jako zpracovatel těchto tabulek byl nastaven typ InnoDB, který podporuje jak primární, tak cizí klíče. Oba typy klíčů jsou totiž sami o sobě brány jako index. Primární i cizí klíč jsou zvoleny typu *UNSIGNED INT*, což je pro indexy nejvýhodnější z hlediska efektivity vyhledávání a zároveň výhodné z hlediska rozsahu klíčů, protože se tím rozsah typu *INT* omezí pouze na nezáporná čísla a zároveň se zdvojnásobí.

4.6.3. Pohledy

Pohledy jsou možností jak si předdefinovat v databázi často používané SQL dotazy. Zároveň lze pomocí nich omezit některé položky, které nechceme vypisovat. Další výhodou je, že je pomocí nich možné zabezpečit dotazy proti SQL Injection (podvodnému upravení SQL dotazu a získání dat, pro uživatele neurčených). Použité pohledy jsou použité pro výpis příspěvků (PrisView) a aktualit (AktuView) z databáze. Zde uvádím kód použitý pro vytvoření zmiňovaných pohledů.

```
CREATE OR REPLACE VIEW PrisView AS (
SELECT ID, Mail, Datum, Text
FROM Prispevky
WHERE Typ = 'PRI'
ORDER BY Datum
);
```

Pro zavolání pohledu a tedy získání záznamů (které má pohled určené vrátit, svým zdrojovým kódem) je potom nutno použít následující příkaz.

```
SELECT * FROM PrisView;
```

Obdobné příkazy byly použity i pro realizování pohledu, vypisujícího aktuality z téže tabulky, jen s tím rozdílem, že místo řetězce PŘI, je v podmínce uveden výraz AKT. Tímto je docíleno, aby v jedné tabulce mohli být záznamy dvou různých určení, ale se stejnými položkami, a nechalo se z nich vybrat jen ten určitý druh, který je potřeba.

4.7. PHP aplikace

4.7.1. Objektová část

Samotnou podstatu aplikace tvoří objekt nazvaný Operator. Pomocí instancí tohoto objektu jsou řešeny zásadní úlohy v uživatelském i administrátorském rozhraní.

Zde uvádím na ukázku zdrojové kódy některých metod tohoto stěžejního objektu:

Metoda insert:

Tato metoda slouží k přidávání záznamů do stromové hierarchie báze znalostí.

```
function insert($odpoved, $otazka, $id_predek = null)
    {
        $otazka = db_esc($otazka);
        $odpoved = db_esc($odpoved);
        $sql_predek = '';
        if(!is_null($id_predek)){
            $sql_predek = ", predek_id = '$id_predek'";
        }
        $err = mysql_query("INSERT INTO dotazy SET odpoved='$odpoved', otazka='$otazka' $sql_predek");
        pr(mysql_error());
        return $err;
    }
```

Obrázek 5. Ukázka kódu metody insert

Metody *update* a *delete* mají podobnou stavbu jako metoda *insert*, *update* slouží k úpravě záznamů a *delete* k jejich vymazání.

Metoda get:

Sloužící k načítání znalostí z databáze podle id.

```
function get($id = null)
    {
        if(!is_null($id)) {
            $vysledek = mysql_fetch_all(mysql_query("SELECT * FROM dotazy WHERE id = '$id'"));
            $vysledek = $vysledek[0];
        } else {
            $vysledek = mysql_fetch_all(mysql_query("SELECT * FROM dotazy WHERE predek_id IS NULL"));
        }

        pr(mysql_error());
        return $vysledek;
    }
```

Obrázek 6. Ukázka kódu metody get

Metoda get_all:

Tato metoda slouží k postupnému načtení všech záznamů z databáze znalostí.

```
function get_all($id = null)
    {
        if(!is_null($id)){
            $pole = array();
            while(true){
                $aktualni = $this->get($id);
                $pole[] = $aktualni;
                if($aktualni['predek_id']=='') break;
                $id = $aktualni['predek_id'];
            }
            $pole = array_reverse($pole);
            return $pole;
        } else {
            return array();
        }
    }
```

Obrázek 7. Ukázka kódu metody get_all

Metoda ***get_next*** slouží k načtení následujícího záznamu, vyhledávaného podle položky predek_id. Metoda ***check*** slouží k otestování stavu proměnné $id. Poslední zmíněná metoda ***je-posledni*** je obdobou a vrací hodnotu true, pokud je v $id neurčitá hodnota null.

Více než vhodné by bylo ještě popsat, jak jednotlivá rozhraní tyto metody využívají. Obě rozhraní (jak administrátorské, tak uživatelské) mají vytvořenu instanci objektu Operator pomocí následujícího příkazu.

```
$operator = new Operator();
```

K jednotlivým metodám je potom přistupováno pomocí příkazů v následujícím formátu.

```
$historie = $operator->get_all($id);

nebo

$odpovedi = $operator->get_next($id);
```

Podstatou je tedy funkce operátoru „šipka“, jenž zajistí vyvolání členských metod objektu se zadaným parametrem. Dále by bylo vhodné popsat, jakým způsobem se vlastně informace dostávají do systému a jakým způsobem se pak ze systému (resp. z databáze) dostávají na obrazovku. K tomu jsou použity dvě základní metody, kterými PHP disponuje z hlediska zpracování formulářů. Jsou to metody

GET a POST. Základní rozdíl mezi nimi je v tom, metoda GET posílá data z formuláře jako součást URL. Tyto data se tak objeví jako součást adresy v adresním řádku prohlížeče. Vhodné to je hlavně v případě, kdy jsou data krátká a takového charakteru, že nevadí, když je uvidí neoprávněná osoba. Posílat touto metodou je tedy vhodné pouze data, jejichž změnou a následným odesláním ke zpracování nikdo nezíská soukromá data sloužící k přihlášení apod. Tato metoda je použita pro předávání id, při procházení stromovou hierarchií znalostí. Metoda POST to naopak nedělá tímto způsobem, ale data odesílá jako samostatný HTTP objekt. Z toho plyne, že je o něco bezpečnější, z hlediska možné změny dat v adrese a zároveň, že se pomocí ní nechá odeslat více dat. V PHP navíc existují asociativní pole, do nichž se odesílaná data z formulářů ukládají. Jsou to $_REQUEST, $_GET a $_POST. Výhodou oddělené existence posledních dvou polí je hlavně případ, kdy je potřeba pracovat s oběma metodami a zároveň mezi nimi rozlišovat. Následující obrázky ukazují, jak jsou použity popsané metody pro zpracování údajů z formulářů.

```
<?
    if(isset($_GET['akce'])) {
        if($_GET['akce']=='pridat') {
            $operator->insert($_POST['odpoved'], $_POST['otazka'], $id);
        }
        if($_GET['akce']=='upravit') {
            $operator->update($id, $_POST['odpoved'], $_POST['otazka']);
        }
        if($_GET['akce']=='smazat') {
            $aktualni = $operator->get($id);
            $operator->delete($id);
            ob_clean();
            header('Location: '.$_SERVER['PHP_SELF'].'?id='.$aktualni['predek_id']);
            exit;
        }
    }
?>
```

Obrázek 8. Ukázka kódu - použití metod GET a POST

```
<? if(!is_null($id)): ?>
    <? $aktualni = $operator->get($id); ?>
    <form method="POST" action="?id=<?=$id;?>&akce=upravit">
        <label for="odpoved">Odpověď</label>
            <input name="odpoved" type="text" value="<?=$aktualni['odpoved'];?>">
        <label for="odpoved">Následující otázka</label>
            <input name="otazka" type="text" value="<?=$aktualni['otazka'];?>">
        <button type="submit">upravit</button>
    </form>
    <? endif; ?>

    <form method="POST" action="?id=<?=$id;?>&akce=pridat">
        <label for="odpoved">Odpověď</label><input name="odpoved" type="text">
        <label for="odpoved">Následující otázka</label><input name="otazka" type="text">
        <button type="submit">přidat</button>
    </form>

    <?if($operator->je_posledni($id)):?>
        <a href="?id=<?=$id;?>&akce=smazat">Smazat</a>
<? endif; ?>
```

Obrázek 9. Ukázka kódu - metody GET a POST použité ve formuláři

4.7.2. Layout stránky

Layout webové stránky byl zvolen dvousloupcový s hlavou a patou, kde v levém sloupci je umístěno menu aplikace, v pravém samotný obsah stránky. Tento layout byl zrealizován pomocí kaskádových stylů, jak bylo popsáno výše. Screenshot rozvržení stránky nemá valnou cenu na tomto místě vkládat, protože náhledy oken aplikace přijdou v popisu následujících částí.

4.7.3. Administrátorské rozhraní

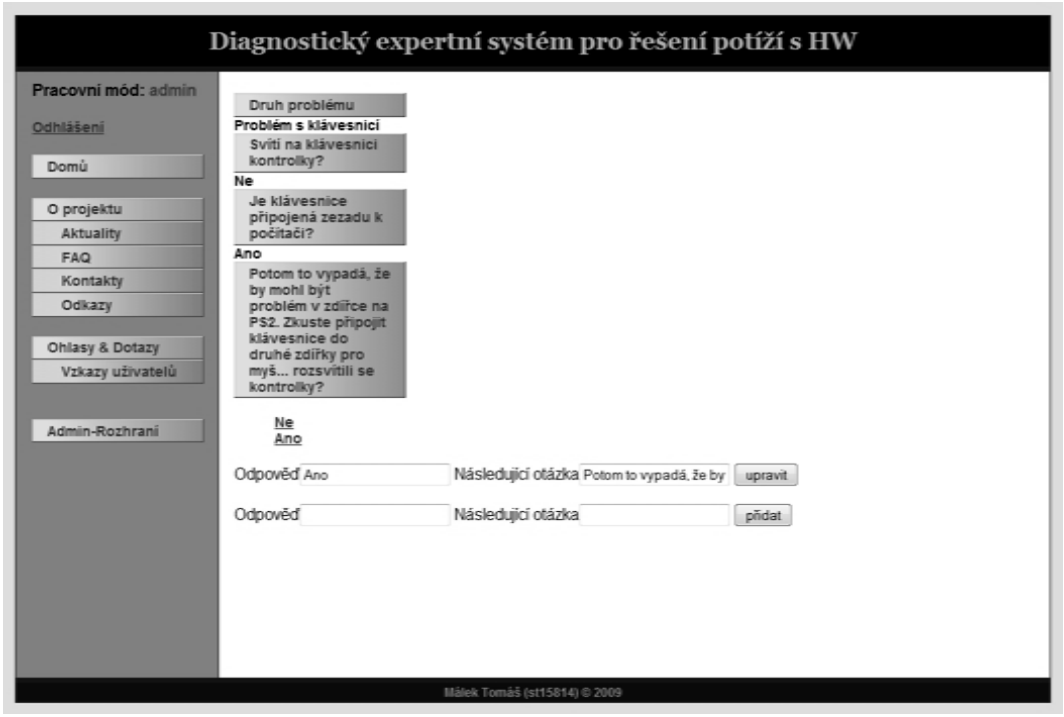
Administrátorské rozhraní je tvořeno za pomoci již zmíněného objektu Operator. Je zde vytvořena jeho instance a pomocí ní jsou pak volány funkce uvedeného objektu. Vstupem do administrátorského rozhraní se zapnou některé volby, které běžný uživatel nemá, například možnost vkládání, editace, či mazání znalostí. Dále pak volby mazání příspěvků z databáze a to obou typů, jak aktualit, tak příspěvků.

4.7.4. Ověření přihlášení

Ověřování přihlášení je realizované pomocí sessions. Při vstupu do administrátorského rozhraní je pomocí podmínky zajištěna kontrola, zda je spuštěná session nastavená (pomocí funkce *isset()*). Tak lze kontrolovat obsah proměnných globálního pole *$_SESSION*, ve kterém jsou uloženy údaje získané při přihlášení uživatele. Pokud session nastavená není, přesměruje aplikace uživatele na přihlašovací formulář, po jehož vyplnění, dojde k přihlášení do administrátorského rozhraní. Po přihlášení se také v levém menu zobrazí možnost odhlášení. Po kliknutí na tuto volbu je právě běžící Session ukončena příkazem *session_unset()* a následně dojde k přesměrování na základní stránku uživatelského rozhraní. Díky příhodné vlastnosti globálního pole *$_SESSION* lze mechanismus session využít také k vypisování chybových nebo potvrzujících hlášení. Stačí si přidat do tohoto pole jednu proměnnou (třeba s názvem hlaska) a do ní podle stavu přihlášení ukládat textové řetězce, které budou poté (příkazem echo) zobrazeny. Podobně, jako lze testovat nastavenost, nebo obsah proměnné login v tomto poli, lze tedy také testovat stav nebo obsah jiných proměnných. Pomocí nich lze pak uchovávat a testovat i jiné údaje v relacích s odstartovanou session.

4.7.5. Editační část

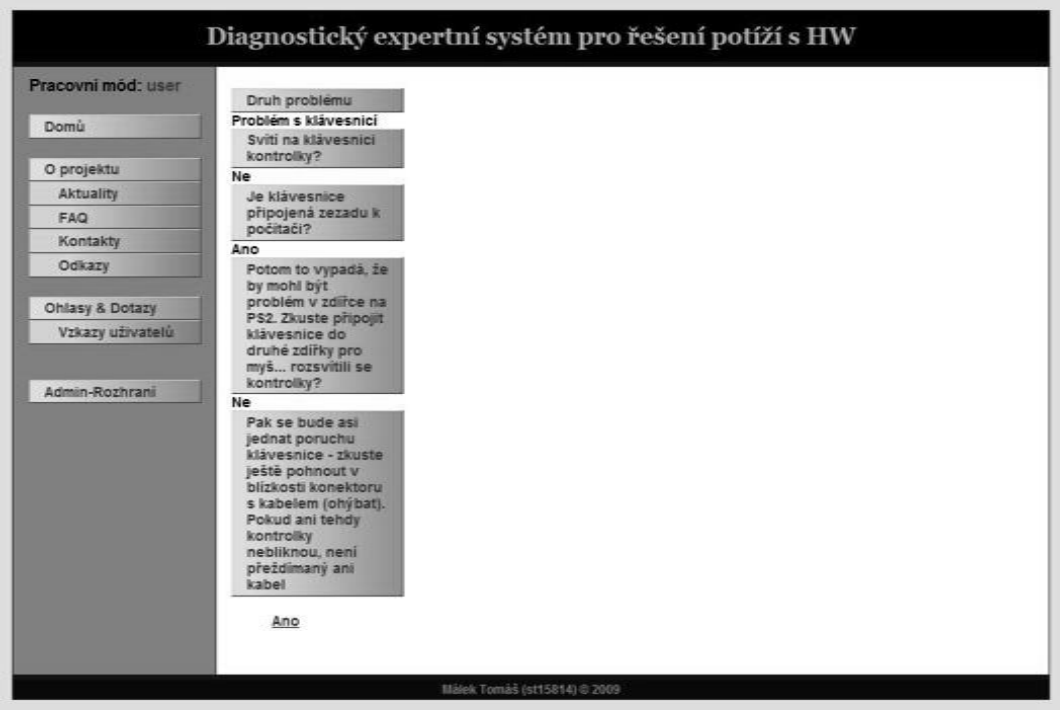
Editační rozhraní je hlavní součástí rozhraní administrátorského. Jiný uživatel než administrátor měnit bázi znalostí vytvořenou v databázi nemůže. Je zde vytvořena formulářová struktura, která umožňuje znalosti v bázi měnit, přidávat nebo mazat. Ve stromu postaveném nad hlavní tabulkou s názvem dotazy, je pak možné libovolně se pohybovat, klikáním na tlačítka a odkazy. U zvolených problémových oblastí je zde pak možno přidávat nebo upravovat zadané znalosti včetně editace v různých úrovních („patrech“) stromu. Přidávání je realizováno tak, že se nejprve zadá odpověď na otázku a posléze do druhého editačního textového pole možná následující otázka. Je-li potřeba vytvořit novou tématickou oblast problémů k řešení, je nutné se vrátit do základního zobrazení oblastí, kliknutím na první tlačítko. Poté vložit jako odpověď název této nové oblasti. Pod tuto novou oblast bude následně vkládána nová hierarchie dotazů a možných odpovědí na ně. Následující obrázek ukazuje vzhled editačního administrátorského rozhraní.



Obrázek 10. Obrazovka aplikace - administrátorské rozhraní

4.7.6. Uživatelské rozhraní

Podobně jako administrátorské (editační) rozhraní je i uživatelské tvořeno za pomoci objektu Operator. Liší se tím, že uživatel nemá práva cokoli z báze znalostí měnit a tudíž jsou mu k dispozici pouze odkazy a tlačítka, pomocí kterých se stromovou hierarchií pohybuje odpovídáním na dotazy. V uživatelském rozhraní je možné prohlížet všechny sekce (hlavní sekci s běhovým prostředím systému, aktuality, časté otázky, kontakty, odkazy) a v sekci vkládání příspěvků je možno i zanechat po sobě stopu. Jak uvádí detailní popis v další části. Následující obrázek ukazuje vzhled popisovaného uživatelského rozhraní s naznačeným postupem stromovou hierarchií znalostí v bázi.



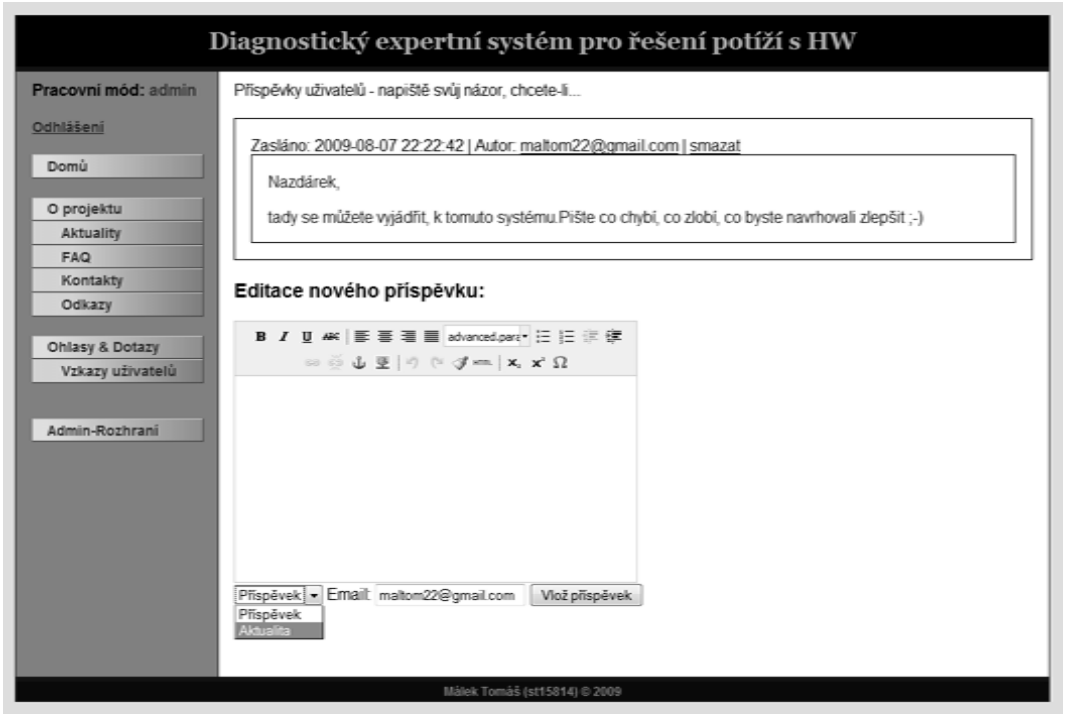
Obrázek 11. Obrazovka aplikace - uživatelské rozhraní

4.7.7. Vkládání příspěvků

Jelikož je, pro správný směr dalšího vývoje aplikace, velmi důležitá zpětná vazba od uživatelů systému, je v aplikaci také implementována možnost psát příspěvky. Uživatel může zanechat svůj názor na systém, zkušenosti s jeho ovládáním, nahlásit chybu, na kterou v průběhu používání narazil, nebo může napsat, která eventuální oblast mu v systému chybí. Samozřejmostí je také možnost vznést dotaz na svůj problém v případě, že by v systému požadované informace nebyly.

Aby byla práce uživatelů při zadávání příspěvků co nejjednodušší a nejpříjemnější, je v aplikaci zanesen wysiwyg editor *TinyMCE* vytvořený v jazyce javascript, je volně stažitelný z internetu a poskytován pod licencí GNU LGPL. Tento editor nabízí mnoho možností na zformátování textu, který je pak včetně formátovacích tagů zanesen do databáze. Při následném načtení textu z databáze je již tento plně formátován, protože prohlížeči je k zobrazení předán včetně formátovacích tagů.

Následující obrázek ukazuje obrazovku aplikace, konkrétně sekci příspěvků s implementovaným wysiwyg editorem. Znázorněno je popsané rozhraní pro administrátora, s viditelnými prvky, které obyčejný uživatel nemá právo použít.

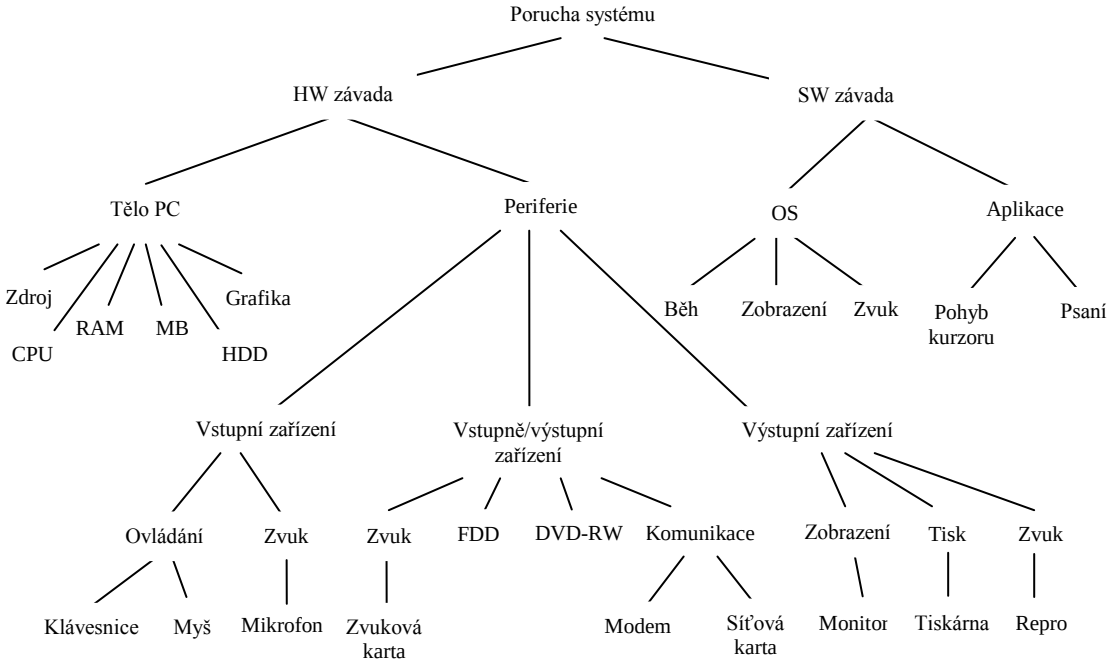


Obrázek 12. Obrazovka aplikace - rozhraní pro přidávání příspěvků

5. Navržená struktura znalostí

5.1. Stromová hierarchie oblastí možných závad

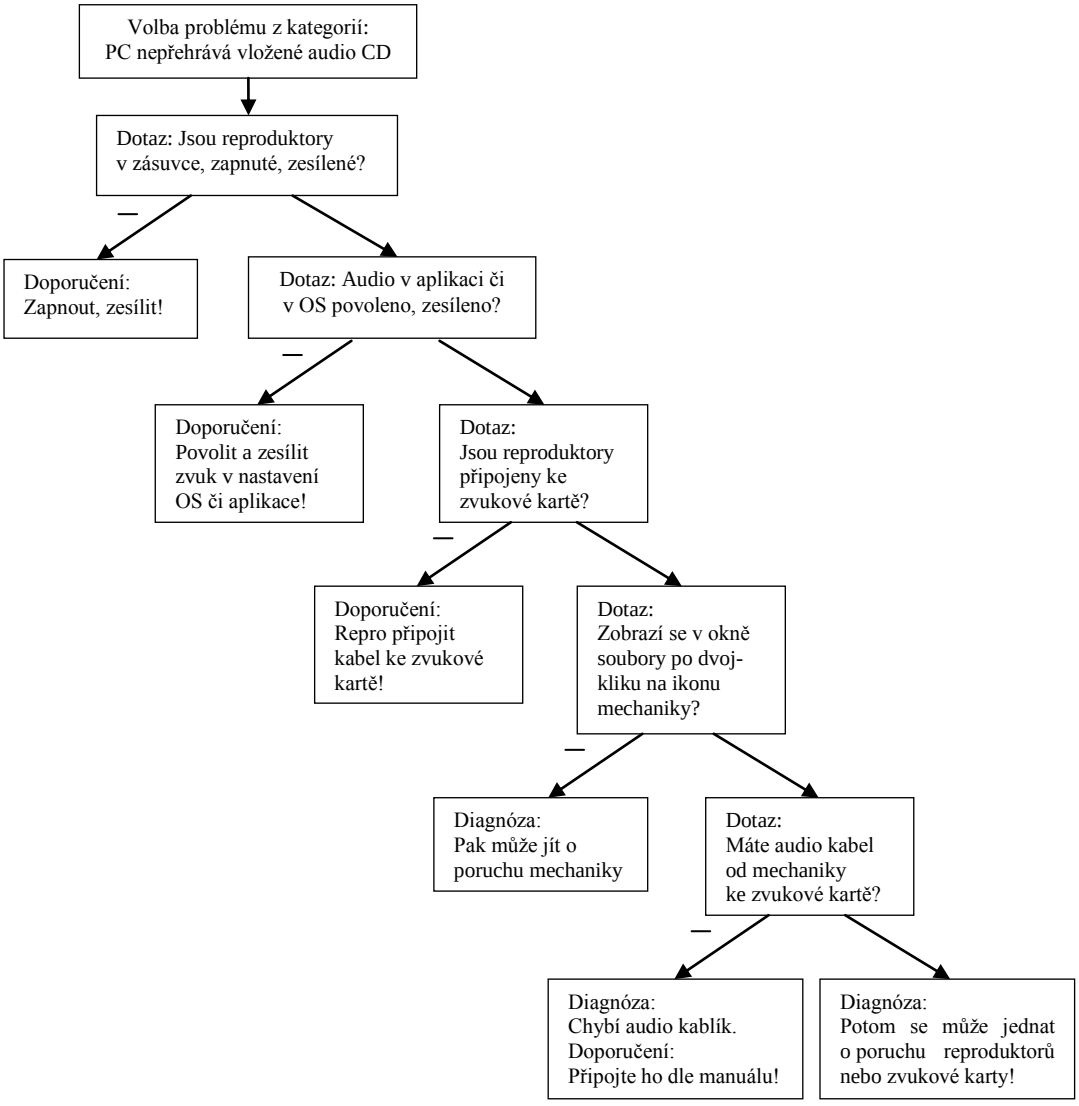
Následující obrázek nastiňuje možnou strukturu a uspořádání jednotlivých oblastí možných závad na počítačovém systému.



Obrázek 13. Stromová struktura oblastí závad

5.2. Příklad postupu řešení problému

Pro ukázku postupu uvažování, při řešení jednoduchého problému, jsem zvolil případ, kdy uživateli počítač nepřehrává vložené audio CD. Konkrétně demonstruji na případu, kdy uživatel buď nemá vůbec, nebo má odpojen audio kabel propojující optickou mechaniku a zvukovou kartu. Uživatel tedy zvolí příslušnou položku, načež se ho systém bude ptát na určitou posloupnost otázek, aby dokázal vyhodnotit diagnózu a určil její řešení. Systém se snaží nejdříve vyhodnotit zda nejde o jeden z častých případů (vypnuté reproduktory či zakázaný zvuk v OS), poté přistupuje k ověření možných závad na HW (optická mechanika atd.). Následující blokové schéma znázorňuje postup při diagnostickém procesu.



Obrázek 14. Příklad postupu uvažování při řešení jednoduššího problému

První otázkou, na kterou se systém uživatele zeptá je, zda má zapnuté reproduktory. Podle odpovědi buď doporučí (v případě Ne) reproduktory zapnout, nebo položí další otázku, zda je povolený zvuk v aplikaci nebo nastavení operačního systému. Pokud ano, následuje dotaz, zda jsou reproduktory připojeny k výstupu zvukové karty. Pokud ano, položí další dotaz, zda se v otevřeném okně jednotky zobrazí ikony souborů. Pokud ne, usoudí, že by se mohlo jednat o problém s mechanikou a stanoví takovou diagnózu. Pokud ano, zeptá se, jestli je připojen audio kabel od mechaniky ke zvukové kartě. Pokud ano, pak usoudí na závadu reproduktorů či zvukové karty. Pokud ne, tak doporučí kabel připojit.

6. Závěr

Byl vytvořen systém, který nejen že zvládá ze stromově uspořádané hierarchie znalosti načítat a poté zobrazovat, ale i další znalosti ve vhodné formě přijímat a do databáze ukládat. K tomuto účelu bylo implementováno vhodné a na ovládání příjemné editační rozhraní. Stěžejní částí této aplikace je objekt, jenž tyto hlavní úkony obstarává. Z důvodu získání ohlasů na funkčnost aplikace, byla do aplikace zahrnuta také možnost vyjádření uživatelů formou příspěvků. Z hlediska budoucího zdokonalení, by bylo vhodné přidat možnost registrace dalších uživatelů, kteří by mohli přidávat své znalosti a postupy, které jim pomohli při řešení problémů v této oblasti, o které by se chtěli podělit. Tím by se systém stal použitelným pro více expertů a určitě také uživatelů.

6.1. Použité zdroje

[1] MAŘÍK, Vladimír, ŠTĚPÁNKOVÁ, Olga, LAŽANSKÝ, Jiří *Umělá intelligence (2)* . 1. vyd. Praha : Academia, 1997. 15-68 s. ISBN 80-200-0504-8.

[2] *Expertní systém : Wikipedie, otevřená encyklopedie* [online]. 2009 [cit. 2009-07-29]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Expertní_systém>.

[3] *MySQL : Wikipedie, otevřená encyklopedie* [online]. 2009 [cit. 2009-07-29]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/MySQL>.

[4] *MySQL* : ZAJÍC, Petr. *Seriál o populární databázi MySQL* [online]. 2005-2006 , [cit. 2009-07-29].
Dostupný z WWW: <http://www.linuxsoft.cz/mysql/>. ISSN 1801-3805.

[5] *PHP : Wikipedie, otevřená encyklopedie* [online]. 2009 [cit. 2009-07-29]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/PHP>.

[6] *PHP :* ZAJÍC, Petr. *Seriál o PHP* [online]. 2004-2006 , [cit. 2009-07-29]. Dostupný z WWW: <http://www.linuxsoft.cz/php/>. ISSN 1801-3805.