ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA

Provozně ekonomická fakulta

Katedra informačního inženýrství

MODELOVÁNÍ ZNALOSTÍ V INTELIGENTNÍCH SYSTÉMECH

Disertační práce

Autor: **Ing. David BUCHTELA**

Školitel: Doc. Ing. Arnošt Veselý, CSc.

Obor: Informační management

Praha 2010

Touto cestou bych rád poděkoval svému školiteli doc. Ing. Arnoštu Veselému, CSc. za poskytnutí cenných rad a názorů a pomoc při tvorbě mé disertační práce. Chtěl bych rovněž poděkovat katedře informačního inženýrství Provozně ekonomické fakulty ČZU v Praze za zázemí poskytnuté v průběhu psaní disertační práce.

David Buchtela

Obsah

1 Úvod

V současné době je úspěch v řadě oblastí založen na kreativitě a schopnostech, expertíze a dovednostech, zlepšování a inovacích. Proto jsou v prostředí, kdy jedinou jistotou je nejistota, jistým zdrojem přetrvávající konkurenceschopnosti znalosti. Jakýkoliv rozhodovací proces prováděný člověkem nezbytně vyžaduje přítomnost dat a znalostí, neboť kvalifikovaně rozhodovat bez znalostí v dané problémové oblasti a konkrétních údajů o konkrétní situaci nelze. Zatímco data jsou získávána pozorováním reálných situací spojeným s jejich měřením, znalosti získává člověk vzděláním a zkušenostmi.

Každé získávání znalostí je vázáno na jazyk, kterým vypovídáme o té části světa, kterou poznáme. V běžném životě obvykle používáme k vyjádření svých znalostí přirozeného jazyka. Pro popis úzce vymezeného oboru poznávání skutečností se používají umělé jazyky (matematické formule, chemické značky, programovací jazyky, schémata elektrických obvodů atd.), které jsou jednoznačnější a úspornější.

Ze samotné podstaty znalostí vyplývá jejich výjimečnost a složitost. Reprezentace a modelování znalostí je proto dnes aktuálním tématem nejen ve výzkumu věnovaném umělé inteligenci, ale i v oblasti podpory rozhodování v různých oblastech lidské činnosti. Tam, kde nejde o řešení úkolů čistě formálních (dokazování teorémů apod.), ale o problémy mající empirický podklad (od her, jako jsou šachy, přes řízení výrobních a jiných složitých procesů až po soustavy automatického zodpovídání dotazů v přirozeném jazyce, ovládání robota apod.), je nutné vypracovat speciální soustavu reprezentace znalostí, na kterých mohou operovat systémy při řešení problémů, vyhledávání informací a podpoře rozhodování.

Tato disertační práce se proto zabývá metodami získávání znalostí a jejich modelování (reprezentací) v inteligentních (znalostních) systémech. Hlavní důraz disertační práce klade autor na metodu modelování znalostí ve formě oborových doporučení. Dále se práce zaměřuje na problematiku podpory rozhodování na podkladě uložených znalostí.

1.1 Motivace autora

Autor práce se podílí na výzkumné činnosti v Centru biomedicínské informatiky při Ústavu informatiky akademie věd ČR v.v.i., které se mimo jiné zabývá výzkumem a návrhem systémů pro podporu rozhodování vycházejících z lékařských doporučení. Lékařská doporučení (v podobě volného textu) představují dokument, který provádí lékaře (uživatele) rozhodnutími a kritériemi v různých oblastech zdravotní péče definovanými expertním zkoumáním dostupných důkazů a zásad současné medicíny. Jen v české republice je sepsáno více než 700 (cca 10 000 stran A4 textu) doporučených postupů diagnostiky a léčby onemocnění z různých oborů medicíny.

Vzhledem k uvedenému počtu lékařských doporučení chce autor svou disertační prací přispět ke zlepšení přístupu uživatelů (lékařů) ke znalostem uloženým v doporučeních a umožnit lepší podporu jejich rozhodování. Proto je součástí práce ucelená metoda reprezentace znalostí nejen v lékařských doporučeních a to od fáze návrhu modelu znalostí přes jeho implementaci (vhodné zakódování) až po napojení modelu na reálná data a ověření jeho vhodnosti.

1.2 Cíle disertační práce

Předkládaná disertační práce je zaměřena na oblast modelování a reprezentace znalostí a jejich použití ve znalostních (inteligentních) systémech. Cílem práce není úplný rozbor existujících metod reprezentace znalostí, ale vyzdvižení metod zaměřených na užší oblast modelování oborových doporučení. Vzhledem k motivaci autora se jedná především o oblast lékařských doporučených postupů diagnostiky a léčby.

Cíle předkládané práce jsou

- Zmapovat oblast získávání a reprezentace znalostí
- Představit a popsat nejvýznamnější existující metody reprezentace znalostí v oborových doporučeních

- Na základě získaných poznatků a praktických zkušenostní navrhnout vlastní zobecněný model reprezentace znalostí obsažených v oborových doporučeních
- Popsat vhodný způsob kódování (implementace) znalostního modelu pro použití ve znalostních systémech
- Navrhnout datové rozhraní mezi modelem znalostí a reálnými daty uloženými v databázi či informačním systému
- Ověřit použitelnost navrženého modelu a jeho implementace v praktických aplikacích především v oblasti medicíny

1.3 Přínosy disertační práce

Disertační práce obsahuje původní myšlenky a postupy představující autorův přínos do problematiky modelování znalostí v inteligentních (znalostních) systémech. Hlavní přínosy navrženého modelu reprezentace znalostí jsou následující:

- Upřesnění modelu reprezentace znalostí v oborových doporučeních a doplnění vlastního způsobu modelování opakovaných činností
- Rozšíření o původní metodu reprezentace časových transakcí a způsobu rozhodování na základě parametrů získaných v různých časových úsecích (transakcích)
- Přesnější definice rozhodovacích kritérií a jejich vyhodnocování v tří-hodnotové logice
- Formální reprezentace grafického modelu v XML. Výsledkem reprezentace je návrh XSD (*XML Schema Definition*)
- Návrh paramodelu (v XML), který je použit jako rozhraní mezi znalostním modelem a reálnými daty pacientů v klinických informačních systémech
- Rozšíření modelu o klíčové atributy a algoritmus výběru relevantního modelu - toto rozšíření je dále využito v systému reprezentace lékařských znalostí

1.4 Metodika dosažení cílů práce

1.4.1 Použitá metodika

Metodika této disertační práce je založena především na studiu odborné literatury a zobecnění vlastních praktických zkušeností. Navržený aparát čerpá dále z matematické logiky, teorie grafů a formálních nástrojů informatiky, zejména algoritmizace a datových modelů.

Práce autora se skládala z následujících kroků:

- studium odborné literatury
- analýza existujících metod získávání a reprezentace znalostí
- zobecnění získaných i vlastních poznatků a zkušeností
- návrh konceptu nové (zobecněné) metody reprezentace znalostí obsažených v oborových doporučeních
- užití formálních modelů pro implementaci navržené metody
- ověření navrženého modelu na praktických aplikacích převážně v oblasti medicíny

Vlastní práce je členěna do čtyř částí, první dvě části jsou převážně teoretické, další dvě části se týkají vlastního návrhu autora. V první části (kapitola 2) jsou popsány základní teoretické poznatky z oblasti modelování znalostí a to pojmy znalost, znalostní systém a metody získávání znalostí. Dále jsou zde popsány vybrané metody reprezentace znalostí. Druhá část (kapitola 3) se zaměřuje na způsoby reprezentace lékařských doporučení a jejich zhodnocení. Třetí část (kapitola 4) obsahuje vlastní návrh reprezentace znalostí v oborových doporučeních, a to rozdělený na část konstrukce modelu (kapitola 4.2) a část implementace (kapitola 4.3). Čtvrtá část (kapitola 5) se zabývá přehledem praktických aplikací, ve kterých je použit navržený znalostní model. Práce obsahuje i dvě přílohy, v první příloze (kapitola 8.1) je detailní XML schéma navrženého modelu a v druhé příloze (kapitola 8.2) je popis uplatnění modelu mimo

oblast medicíny, konkrétně v oblasti modelování průchodu studenta výukovým systémem na České zemědělské univerzitě.

1.4.2 Terminologie

Předložená práce je vypracována v českém jazyce a snaží se držet české odborné terminologie, pro lepší porozumění odborným výrazům jsou anglické termíny zapsány kurzívou v závorkách.

Odborné termíny jsou vysvětleny přímo v textu práce nebo ve formě poznámky pod čarou. Pokud není vysvětlení některého termínu v textu obsaženo nebo mohou vzniknout pochybnosti o smyslu využití daného termínu, je tento termín použit ve smyslu uvedeném v [31].

2 Znalosti a znalostní systémy

2.1 Data, informace a znalost

Po obsahové stránce je možné data, znalosti i informace definovat stejným způsobem jako odraz (model, reprezentaci) reálného světa. Výsledkem tohoto odrazu jevů, procesů a vlastností, které existují a probíhají v té části reality, kterou odrážejí, jsou pak jakékoli znalosti, vědomosti, poznatky, zkušenosti nebo výsledky pozorování procesů, projevů, činností a prvků reality (reálného světa, skutečnosti). Odlišnost mezi daty, znalostmi a informacemi se projeví, začneme-li uvažovat nad jejich účelem.

Data[1] obvykle chápeme jako údaje, tj. číselné hodnoty, znaky, texty a další fakta zaznamenaná (a uložená v databázi) ve formě uspořádané posloupnosti znaků zvolené abecedy. Jedná se tedy o jakékoli vyjádření (reprezentaci) skutečnosti, schopné přenosu, uchování, interpretace či zpracování [26].

Pojem **informace**[2] patří k nejobecnějším kategoriím současné vědy i filozofie. Podle toho, v kterém vědním oboru nebo v jaké oblasti lidské činnosti se používá, jsou aplikovány specifické přístupy ke zkoumání informace a jsou k dispozici různé způsoby jejího definování. V nejobecnějším slova smyslu se informací chápe údaj o reálném prostředí, o jeho stavu a procesech v něm probíhajících. Informace snižuje nebo odstraňuje neurčitost systému (např. příjemce informace). Množství informace je dáno rozdílem mezi stavem neurčitosti systému (entropie[3]), kterou měl systém před přijetím

[1] Slovo **data** pochází z latinského slovesa *dato,* tj. *dávat*, a podstatná jména z něj odvozená znamenají *dané, danost, údaj*. V češtině se výraz *data* používá pro množné číslo, jednotné číslo je *údaj*. Synonyma: *údaje, skutečnosti, fakta*.

[2] V latině, která dala světu termín **informace**, se sloveso *informo,* používalo k vyjádření následujících činností: *formovat, utvářet, vzdělávat, upravovat, podávat představu (pojem) něčeho*. Podstatné jméno *informatio*,pak označovalo *představu, obrys, výklad, poučení*.

[3] **Entropie** je neurčitost, nejistota, neuspořádanost; střední hodnota míry informace potřebné k odstranění neurčitosti, která je dána konečným počtem vzájemně se vylučujících jevů.

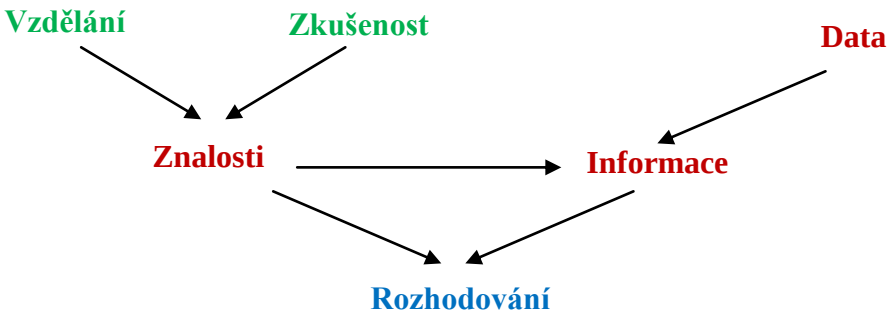
informace a stavem neurčitosti, která se přijetím informace odstranila. V tomto smyslu může být informace považována jak za vlastnost organizované hmoty vyjadřující její hloubkovou strukturu (varietu), tak za produkt poznání fixovaný ve znakové podobě v informačních nosičích.

V informační vědě a knihovnictví se **informací** rozumí především sdělení, komunikovatelný poznatek, který má význam pro příjemce nebo údaj usnadňující volbu mezi alternativními rozhodovacími možnostmi [26]. Významné pro informační vědu je také pojetí informace jako psychofyziologického jevu a procesu, tedy jako součásti lidského vědomí (např. N. Wiener[4] definuje informaci jako "obsah toho, co se vymění s vnějším světem, když se mu přizpůsobujeme a působíme na něj svým přizpůsobováním"). V exaktní vědě se např. za informaci považuje sdělení, které vyhovuje přísným kriteriím logiky či příslušné vědy. V ekonomické vědě se informací rozumí sdělení, jehož výsledkem může být zisk nebo užitek.

Znalost je to, co jednotlivec vlastní (ví) po osvojení dat a informací a po jejich začlenění do souvislostí. Znalost je tedy výsledek poznávacího procesu a předpoklad uvědomělé činnosti.

Všimněme si rozdílu vycházejícího z rozboru rozhodovacího procesu, v němž expert vybavený znalostmi zvažuje data a informace relevantní pro daný problém (viz obrázek 1). Znalosti expert získal vzděláním a zkušenostmi a vybraná data a informace se týkají konkrétní situace či případu. Komplexní poznání je pak možno považovat za množinu znalostí, informací a dat vztahujících se k určité problematice.

[4] **Norbert Wiener** (26.11.1894 - 18.3.1964) - americký matematik, zakladatel kybernetiky. Je spoluautorem teorie podobnosti činnosti nervové soustavy a počítače, základu neurokybernetiky.



obrázek 1 Vztah data-informace-znalost

2.2 Typy znalostí

Systémy, které se chovají inteligentně, jsou schopné předvídat projevy okolí a důsledky svých zásahů do tohoto okolí. Jsou vybaveny jistým modelem okolí i sebe sama a podle těchto modelů se snaží dosáhnout určitých cílů. Znalosti jsou údaje, které slouží ke konstrukci uvedených modelů. Podle způsobu vyjádření znalostí je můžeme rozdělit následovně (podle [7]).

2.2.1 Implicitně vyjádřené znalosti

Představme si program na řešení soustavy *n* lineárních rovnic o *n* neznámých. Znalosti použité při řešení jsou v podobě kódu roztroušeny po celém programu, tj. jsou vyjádřeny implicitně. Je dosti obtížné odvodit metodu řešení soustavy rovnic z předloženého kódu, tj. extrahovat z něj znalosti. Stejně tak je složitá modifikace znalostí, vyžaduje totiž změnu programu. Znalosti jsou popsány procedurou samou a získáme je tím, že proceduru provedeme. Procedurálně vyjádřené znalosti budeme nazývat **pravidla.**

2.2.2 Explicitně vyjádřené znalosti

Uvažujme znalosti uložené v databázi, kde je program oddělen od vlastních dat. Program pouze popisuje, jak databázi využívat, sám však znalosti o řešené úloze neobsahuje. Znalosti jsou tedy uloženy explicitně. Extrakce i modifikace znalostí je

potom poměrně jednoduchá. Díky takovému vyjádření mohou znalosti sloužit jednak k nalezení výsledku, jednak k vysvětlení postupu jeho hledání.

2.2.3 Deklarativně vyjádřené znalosti

Explicitní vyjádření znalostí umožňuje i odvození nových skutečností a tak znalosti rozšířit, např. z toho, že „kočky rády mléko“ a „Micka je kočka“, můžeme usoudit, že „Micka ráda mléko“. Popsaný způsob vyjádření znalostí se nazývá deklarativní. Deklarativně vyjádřené znalosti budeme nazývat **poznatky.**

2.3 Inteligentní systém

Inteligentním systémem lze chápat člověka, skupinu lidí, stroj nebo systém tvořený lidmi a stroji. Inteligentní systém získává a zpracovává poznatky o objektech okolního světa ve tvaru $(p)\delta(x)$, kde

- x je prvek (objekt) reálného světa (univerza)
- δ nějaká vlastnost prvků univerza
- $p \in \langle 0,1 \rangle$ udává míru jistoty, že prvek x má vlastnost δ

Typické jednoduché systémy, které bývají součástí složitějších inteligentních systémů, lze popsat následovně:

- **Inteligentní čidlo** - může realizovat člověk, kolektiv lidí (komise), technické zařízení nebo lidé za pomocí techniky. Podstata inteligence čidla spočívá ve schopnosti zjištěnou (naměřenou) hodnotu spojit s objektem, kterého se údaj týká.
- **Inteligentní výkonný prvek (realizátor)** – systém (člověk či stroj), který realizuje vyhledání konkrétního objektu na základě poznatků ve tvaru $(p)\delta(x)$, které jsou o tomto objektu známy.
- **Datový sklad** – činnost tohoto systému spočívá ve vedení souborů, kartoték a bází dat a dokumentů, představujících poznatky o objektech a zpřístupnění údajů

o těchto poznatcích uživatelům na základě jejich požadavků. Údaje ve tvaru $(p)\delta(x)$ jsou ve skladu vždy spojeny s příslušným objektem x.

- **Transformátor poznatků** - systém, který na základě poznatků o objektech množiny X vytváří poznatky o objektech množiny Y, tj. převádí poznatky typu $(p)\delta(x)$ pro $x \in X$ na poznatky typu $(q)\beta(y)$ pro $y \in Y$.

Složitějšími inteligentními systémy, složenými z výše uvedených, jsou také expertní a znalostní systémy.

2.4 Znalostní a expertní systém

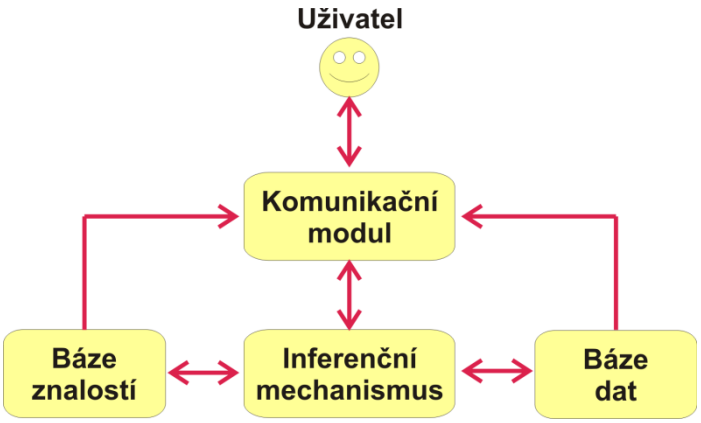
Znalostní a expertní systém jsou realizovány poměrně rozsáhlou soustavou kooperujících programů. Jednotlivé programové celky této soustavy tvoří prvky funkčně vymezených a svou strukturou i posláním odlišných modulů. Vezmeme-li do úvahy i jejich vzájemné vazby, tvoří tyto moduly architekturu znalostního resp. expertního systému [8].

V každém tradičně koncipovaném znalostním systému je možné rozlišit čtyři základní složky (moduly), a to *bázi znalostí, bázi dat, inferenční mechanismus* a *komunikační modul* (viz obrázek 2). V expertním systému najdeme navíc *vysvětlovací modul, generátor výsledků* a *modul externích zdrojů* (viz obrázek 3).

2.4.1 Báze znalostí

Báze znalostí obsahuje znalosti experta(ů) potřebné k řešení zvoleného problému. Tyto znalosti jsou vyjádřeny naprosto explicitně a předem je dána pouze strategie využívání znalostí z báze znalostí (inferenční mechanismus). Znalosti experta nemají statický charakter, proto je nutné, aby bylo možné do báze znalostí jednoduše zahrnovat přírůstky nových znalostí. Báze znalostí musí být též transparentní, tj. čitelná pro experta i pro další odborné pracovníky, a bezesporná (udržování bezespornosti je netriviální problém). Existuje více způsobů, jak údaje do báze znalostí zapisovat. Jedním z nejrozšířenějších způsobů je zapisovat bázi znalostí pomocí produkčních

systémů, ale existují i expertní systémy pracující s rámcovými či asociativními reprezentacemi.



obrázek 2 Schéma znalostního systému

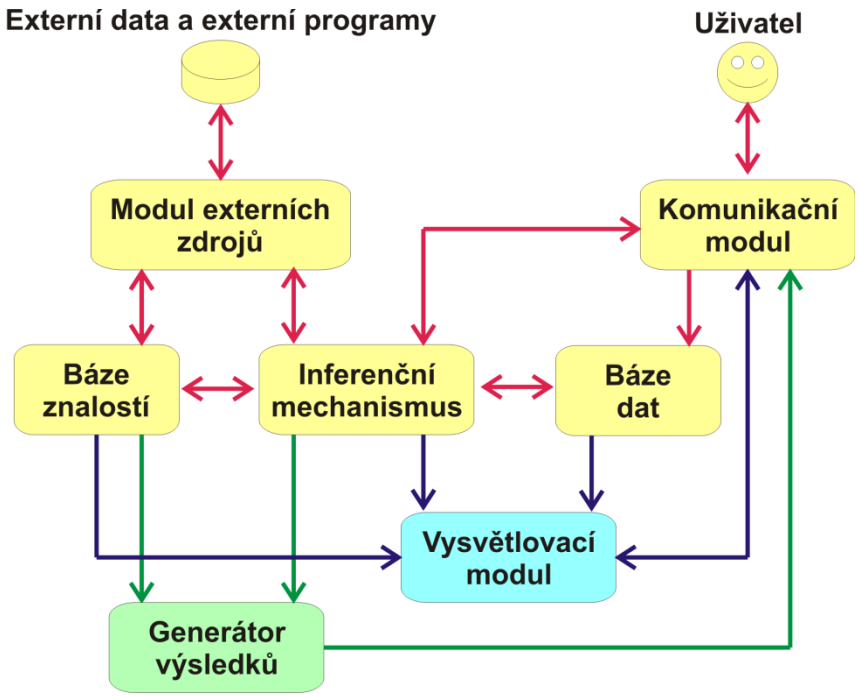
2.4.2 Báze dat

Báze dat je, stejně jako báze znalostí, pasivní datová struktura. Stejně jako ona obsahuje seznam dat bez toho, že by v ní byl obsažen návod (procedura, algoritmus), který by říkal, jak s těmito pravidly nakládat. Báze dat je nositelkou konkrétně zadaných nebo odvozených faktů, popřípadě předpokládaných (až odhadnutých) údajů o nějakém specifickém problému. Jsou to informace, které vychází z konkrétního prostředí, konkrétních zařízení a provozů. Báze dat tvoří tu část expertního systému, která nejvíce odráží údaje experta a jeho praktické zkušenosti. Do báze dat se také na počátku vkládají vstupní údaje, během procesu se zde ukládají odvozená fakta, mezivýsledky a výsledky.

2.4.3 Inferenční mechanismus

Inferenční mechanismus je složen ze souboru kooperujících programů zabezpečujících procedurální složku činnosti expertního systému. Tím inferenční modul umožňuje v určitém rozsahu napodobovat expertovu schopnost uvažovat. Modul simuluje především ty schopnosti, které souvisí s efektivním využíváním poznatků a zkušeností, získaných na základě asociací, hierarchií, příčinně-důsledkových vazeb, kontextů

a spojování poznatků do vhodně souvisejících celků a posloupností. Takto zavedený inferenční mechanismus tedy odpovídá mechanismům všeobecného uvažování, opírajícího se o bázi znalostí, na jejímž základě je možno konkrétní problémy řešit.



obrázek 3 Schéma expertního systému

2.4.4 Komunikační modul

Znalostní systém většinou není izolovaný od ostatního (softwarového nebo fyzického) prostředí a musí si tedy s ním vyměňovat informace. Například může být pro nějaký expertní systém důležité znát momentální teplotu uvnitř tavící pece nebo aktuální počet nezaměstnaných v daném okrese. Mminimálně musí uživatel zadat otázku nebo problém, který se má znalostním systémem zpracovávat.

Komunikační modul bude zabezpečovat především komunikaci s uživatelem. Právě způsob komunikace je jedním z podstatných rozdílů mezi znalostními a expertními systémy. Komunikační modul znalostních systémů je určen především pro sběr

informací, které sám potřebuje k řešení problémů. Expertní systémy více dbají na to, aby uživatel dostal informace nejen o tom, jaké řešení systém navrhuje, ale i o tom, proč je právě vybrané řešení optimální.

S komunikačním modulem, především v expertních systémech, úzce souvisí i další důležité moduly:

- Generátor výsledků - modul, který slouží ke generování výsledku v komunikovatelné podobě, vhodné pro uživatele.
- Vysvětlovací modul - modul sloužící ke generování vysvětlení výsledku (rozhodnutí) s použitím pravidel obsažených v bázi znalostí. Vysvětlovací modul se uplatní především v expertních systémech. Typické možnosti vysvětlování, které expertní systémy nabízejí, je vysvětlení *why* (zdůvodnění, proč expertní systém klade určitý dotaz uživateli), *how* (jak systém odvodil své doporučení) a *what-if* (jaké by bylo doporučení, kdyby uživatel na určitý dotaz systému odpověděl jinak) [A8].
- Modul externích zdrojů - modul, který zajišťuje zadávání a zpracovávání údajů z vnějšího okolí. Modul externích zdrojů bude, na rozdíl od komunikačního modulu, zajišťovat komunikaci s daty, programy a fyzickým prostředím.

2.5 Získávání znalostí

Tvorba znalostního (expertního) systému je vždy netriviální záležitost. Ze samotné podstaty expertních znalostí vyplývá jejich výjimečnost a složitost, která se promítá i do procesu jejich získávání. Navíc se stále obohacuje spektrum technik, které lze při získávání znalostí použít (viz [2] a [3]). Nelze tedy stanovit postup, který je za všech okolností optimální. Existují však ověřené všeobecné zásady a heuristiky, které mohou být při získávání znalostí užitečné. Velmi častý je případ, kdy nejlepších výsledků dosáhneme vhodnou kombinací různých metod.

Proces objevování znalostí, pomocí získávání dříve neznámých (skrytých) informací z velmi rozsáhlých databází se nazývá data mining. Při provádění data miningu

nalézáme skryté vzory a trendy v rozsáhlých databázích, které pak slouží jako podklady pro rozhodování [2].

2.5.1 Indukce

Indukci lze chápat jako úsudek, kdy z jednotlivých speciálních případů odvodíme obecně platný závěr. Platí-li pro předmět A_1 určitá skutečnost B, pro předmět A_2 také skutečnost B, pro *n*-tý předmět A_n také skutečnost B, pak můžeme obecně říci, že pro všechny předměty A platí skutečnost B. Na indukci (generalizaci z příkladů) je založena většina metod strojového učení. Tyto metody lze použít pro automatizované získávání znalostí z dat.

Hovoříme-li o indukci jako metodě usuzování, rozlišujeme:

- **úplnou indukci** - o úplné indukci hovoříme tehdy, když máme uzavřený systém, v němž známe všechna fakta tohoto systému, z nichž můžeme usuzovat na obecnější závěry. Například jako předměty A_i vezmeme dny v týdnu (pondělí, úterý, středa, čtvrtek, pátek, sobota a neděle), pro které platí skutečnost B, že mají každý 24 hodin. Lze tudíž říci, že všechny dny týdne mají 24 hodin. Tento typ úsudku je velmi často používán k dotazování platnosti nějaké (matematické) věty.
- **neúplnou indukci** - o neúplné indukci hovoříme tehdy, neznáme-li všechny prvky nebo fakta, přičemž z nich vyvozujeme na obecné závěry. Neúplná indukce může mít vždy jen pravděpodobnostní charakter. Všem postupům neúplné indukce je společné to, že závěr neplyne z předpokladů nutně, je vždy pouze v jistém stupni pravděpodobný. Jako příklad můžeme vzít průzkum sledovanosti televizních stanic, kde v určitém vzorku populace sledují všichni lidé nad 60 let nejraději ČT1. Jelikož se nenašla (v našem vzorku) jediná výjimka, dospějeme k závěru, že všichni lidé ve věku nad 60 let sledují nejraději ČT1. I když jsou zde všechny případy pravdivé, nedobrali jsme se ve výsledku jistoty, ale jen určité míry pravděpodobnosti. Vyskytne-li se pak jediná výjimka neslučitelná s vyvozeným závěrem, tento závěr

přestává být platným. Neúplná indukce je často využívána například ve statistice a ve vědách, které statistiku používají jako metodu (sociologie, ekonomie…).

2.5.2 Dedukce

Výroková logika by jako model lidského vnímání světa a myšlení o něm neměla valnou cenu, kdyby se v ní nedalo vyjádřit, co z čeho vyplývá. Jde o schopnost, která se v případě skutečného lidského myšlení nazývá schopnost dedukce.

Dedukcí se obvykle rozumí usuzování od obecného k zvláštnímu a jednotlivému, avšak mnohem přesněji je dedukce vyvozováním nových tvrzení při dodržování pravidel logiky.

Problém rozhodování, zdali určitá formule *A* vyplývá z množiny formulí *U*, se nazývá problém dedukce a je jedním ze základních problémů logiky. Ve výrokové logice hovoříme o formuli *A* vyplývající z množiny formulí *U* jako (tauto)logickém důsledku *U*. Množina formulí *U* je v tomto pojetí speciální množinou předpokladů (speciálních axiomů), na níž je postavena určitá teorie. Formule *A* je (tauto)logickým důsledkem množiny formulí *U*, platí-li pro všechny modely množiny formulí *U*, že formule *A* je v nich splněna (*true*).

2.5.3 Redukce

Předložená úloha je dekomponována na řadu dílčích podúloh (podcílů), které se dále mohou strukturovat v ještě jednodušší úlohy. Celý proces je rekurzivně opakován až do okamžiku, kdy je celá úloha rozdělena na řešení triviálních elementárních úloh, na jejichž vyřešení má výpočetní systém k dispozici prostředky.

Cílem redukce je předložit k řešení takové elementární úlohy, jejichž řešení je zřejmé. Takovými úlohami mohou být úlohy odpovídající jednomu kroku ve stavovém prostoru, případně i úlohy složitější, jejichž řešení je ovšem známo.

Tyto metody je možné použít na řadu praktických úloh, jako je například dokazování vět, symbolické integrování jako formálních manipulací s výrazy, redukci hledání výhry u řady her na hledání dalšího vhodného tahu, např. u šachů, dámy apod.

2.5.4 Analogie

Analogie je odvození závěru na základě podobnosti s jinou situací. Analogie se používá např. při případovém usuzování. Místo aby znalosti měly podobu (obecných) pravidel získaných od experta, jsou tvořeny souborem dříve vyřešených (typických) případů. Případové usuzování lze přirovnat k americkému právu založenému na precedentech, usuzování na základě pravidel je analogické evropskému (kontinentálnímu) pojetí práva.

Výhodou je, že na rozdíl od klasických expertních systémů, není třeba pracně získávat znalosti (pravidla) od experta, stačí "jen" získat dostatek reprezentativních případů.

Příklad: Pacient, který přijde k lékaři, mu sděluje své potíže. Lékař zjišťuje, že pacientův problém vykazuje určité symptomy. Přitom disponuje znalostí, že lidé s onemocněním X vykazují tytéž či podobné symptomy. Na základě analogie mezi pacientem a lidmi s onemocněním X lékař usoudí, že pacient trpí také touto chorobou. Tento závěr potom obvykle slouží jako počáteční hypotéza, která může být později lékařskými testy potvrzena nebo vyvrácena.

2.5.5 Generování a testování

Generování a testování je metoda pokusů a omylů. Při generování a testování se opakovaně generuje možné řešení a testuje se, zda vyhovuje všem požadavkům. V případě, že nalezneme vyhovující řešení, cyklus končí. Znalosti jsou v těchto systémech reprezentovány pravidly

IF nastane_situace THEN proveď_akci

V daném okamžiku běhu systému mohou být splněny podmínky více pravidel. Běh generativního systému spočívá v opakování tří základních fází:

- **fáze porovnání** - vytvoření rozhodovací množiny obsahující všechna v dané chvíli aplikovatelná pravidla, např.:

  1) IF *diastolický tlak je vysoký* THEN *rizikový pacient*

  2) IF *cholesterol je vysoký* THEN *rizikový pacient*

  3) IF *diastolický tlak je nízký* AND *cholesterol je nízký* THEN *nerizikový pacient*

- **fáze rozhodnutí sporu** - výběr právě jednoho pravidla z rozhodovací množiny, např. pravidla 2)

- **fáze úkonu** - provedení příslušné akce uvedené na pravé straně pravidla (za THEN), tj. akce týkající se rizikového pacienta

2.5.6 Abdukce

Abdukce je usuzování z pravdivého závěru na předpoklady, které mohly tento závěr způsobit. Při abdukci vycházíme z platnosti implikace a závěru. Z tabulky pravdivostních hodnot pro implikaci je vidět, že předpoklad může být pravdivý nebo nepravdivý. Lze se tedy jen domnívat, že předpoklad může platit. Někdy je abdukce označována za odvozování nejlepšího vysvětlení pro pozorovaná fakta. Abdukce zachovává nepravdu, tj. když budeme předpokládat, že platí implikace a neplatí závěr, lze jednoznačně říci, že neplatí předpoklad.

2.5.7 Heuristiky

Heuristiky jsou pravidla "vycucaná z prstu" založená na zkušenosti. Heuristiky (heuristická pravidla) je označení pro znalosti používané v expertních systémech. Jde o znalosti založené na zkušenostech experta, na zobecnění situací, ve kterých se expert rozhodoval.

Jde vlastně o nalezení náhradního řešení v případě, že přímé řešení není známo, ale pravidla pro hledání a nalezení jsou v zásobě zkušeností a je možnost nalézt podobnosti a vyhodnotit je. Je zřejmé, že právě v prvcích zkušeností, podobností a vyhodnocení

podobností se mohou uplatnit nejširší tvůrčí schopnosti člověka, jeho schopnost kombinovat, vážit a vyhledávat. Pro heuristické postupy je možno vytvořit i do značné míry standardní schéma.

Pro tyto úlohy však není možné sestrojit efektivní algoritmus (složitost algoritmů řešení je obvykle exponenciálního řádu) vedoucí vždy k optimálnímu řešení. Heuristiku pak můžeme chápat, jako postup získání řešení problému, které však není přesné (optimální) a nemusí být nalezeno v krátkém čase. Slouží však nejčastěji jako metoda rychle (s podstatně nižší složitostí) poskytující dostatečné a dosti přesné řešení, které však nelze obecně dokázat. Nejčastější použití heuristického algoritmu nalezneme v případech, kde není možné použít jiného lepšího algoritmu, poskytujícího přesné řešení s obecným důkazem.

2.5.8 Získávání znalostí z volných textů

V lidské společnosti je text asi vůbec nejčastějším prostředkem pro výměnu informací. Hlavní zvláštností volného textu je jeho nestrukturovanost a tím i obtížná uchopitelnost. V souvislosti se zpracováním textů uvažujeme dva základní pojmy dokument (*document*) a termín (*term*).

Podle [9] chápeme dokument jako jakýkoliv souvislý úsek textu, který může být považován za samostatnou jednotku (kniha, článek, kapitola, odstavec, webová stránka, e-mail). Termínem je nejčastěji slovo, dvojice slov nebo ustálené víceslovné spojení.

Dokument lze reprezentovat vektorem termínů, kde každému termínu t_j ($1 \le j \le T$) je ve vektoru vyhrazena pevná j-tá pozice d_{ij}, která udává informaci o frekvenci výskytu daného termínu v dokumentu D_i ($1 \le i \le N$):

$$D_i = (d_{i1}, d_{i2}, \ldots, d_{iT})$$

Vzniká obdélníková matice, jejíž řádky odpovídají dokumentům a sloupce termínům.

Tato reprezentace je velmi častá, skrývá v sobě však několik omezení:

- Nepostihuje kontext, ve kterém se slovo objevilo

- Je obtížné vyjádřit sémantickou podobnost termínů
- Otázkou je, jak volit množinu reprezentativních termínů – velký počet termínů vede na extrémně velké a řídké matice, malý počet naopak způsobuje ztrátu informace
- Další otázkou je složitost jednotlivých termínů

Alternativou k uvedené reprezentaci mohou být statistické postupy, které se snaží minimalizovat původní množinu termínů a tím redukovat dimenzi vektoru termínů. Jako příklad lze uvést metodu indexace latentní sémantiky (viz [5]). Vedle statistických metod se rozvíjejí i metody lingvistické, které modelují dokumenty strukturovaně na základě poznatků zpracování přirozeného jazyka (viz [2]).

2.6 Neurčitost ve znalostních systémech

Valná většina údajů a znalostí, které užíváme při rozhodování či řízení, je do určité míry nejistá. Použijeme-li Pythagorovu větu pro výpočet délky úhlopříčky obdélníku, není výsledek ani obsažená znalost zatížena nejistotou. Použijeme-li ale tentýž vztah pro výpočet vzdálenosti protilehlých rohů obdélníkového pole (pole zpravidla není přesný obdélník), nemusí se již výsledek rovnat skutečné vzdálenosti. Použili jsme totiž nejistý předpoklad, že pole je obdélník.

Dalším typem neurčitosti je fakt, že ani jednoduchou znalost vlastně nelze formulovat přesně deterministicky. Uvažujme například jednoduchou znalost „Pustíme-li předmět, který držíme v ruce, pak padá dolů“. Toto platí vždy, pokud předmět není lehčí než vzduch. Znalost můžeme upřesnit na „Pustíme-li předmět těžší vzduchu, který držíme v ruce, pak padá dolů“. Tuto znalost můžeme aplikovat vždy, pokud ale nejsme potápěčem pracujícím pod vodou. Takto můžeme postupovat téměř bez omezení.

Trochu jiný typ nejistoty je obsažen ve znalosti lékaře „Pacient se zápalem plic má zvýšenou teplotu“. Každý zkušený lékař ví, že zápal plic je skutečně provázen teplotami, přesto se vzácně vyskytne případ zápalu plic bez zvýšené teploty.

Další typ neurčitosti lze ilustrovat výrokem „Všichni vynikající hráči košíkové jsou vysocí“. Kromě neurčitosti popsané v předchozím odstavci (čas od času se vyskytne vynikající hráč, který není vysoký) je neurčitost skryta i ve významu slov „vynikající“ a „vysoký“.

Velká variabilita typů neurčitosti se odrazila také ve velkém počtu matematických postupů, které však nejsou bez nevýhod. Jedná se často o metody teorie pravděpodobnosti (např. Bayesovské metody) či statistické metody (viz [A8]). Jde obvykle o metody poměrně komplikované, které vedou k řešení algoritmicky složitých úloh. Jiným velice rozšířeným přístupem jsou metody založené na teorii fuzzy[5] množin a fuzzy logiky (viz [11]).

2.6.1 Základy fuzzy logiky

Fuzzy logika se poprvé objevila v roce 1965 v článku, jehož autorem byl profesor L.A. Zadeh. Tehdy byl definován základní pojem fuzzy logiky a to fuzzy množina. Fuzzy teorie se snaží pokrýt realitu v její nepřesnosti a neurčitosti.

Fuzzy logika přináší nové, v jistém smyslu přesnější chápání pojmů pravda/nepravda. V klasické logice (aristotelské, booleovské) rozeznáváme pouze dvě pravdivostní hodnoty výroku. ANO/NE přístup používáme podvědomě i tam, kde to není zapotřebí. Například na otázku "Je den?" se nedá jednoznačně odpovědět v časných ranních či pozdních večerních hodinách. Ba co víc, odpověď v tomto případě je individuální, někomu den začíná dříve, někomu později. Vymezení den/noc má tedy neostré (fuzzy) hranice, výrok "Je den" může být pravdivý jen zčásti, např. z 60%. V klasické logice platí, že výrok je buď pravdivý, nebo je nepravdivý a nikdy nemůže být zároveň pravdivý a přitom i nepravdivý. Toto omezení ve fuzzy logice neplatí. Fuzzy logika připouští paradoxy, namísto černobílého (0/1) vidění přináší škálu možností, od zcela nepravdivého, přes téměř nepravdivé, paradoxní, téměř pravdivé až ke zcela pravdivému.

[5] **Fuzzy logika** znamená roztřepaná, nejasná, nepřesná, chlupatá, neurčitá, vágní logika.

2.6.1.1 Zavedení fuzzy množin

U klasické množiny prvek *u* do množiny *A* patří nebo nepatří, což vyjadřuje charakteristická funkce:

$$\mu_A(u)=1 \qquad \text{jestliže} \qquad u \in A$$

$$\mu_A(u)=0 \qquad\qquad\qquad u \notin A$$

U fuzzy množiny mohou prvky mohou množiny *A* patřit částečně, stupeň příslušnosti prvku k množině definujeme prostřednictvím funkce příslušnosti:

$$\mu_A(u): A \rightarrow \langle 0,1 \rangle$$

- **Komplement** fuzzy množiny A: $\mu_{A'}(u) = 1 - \mu_A(u)$
- **Průnik** fuzzy množin A a B: $\min\{\mu_A(u), \mu_B(u)\}$
- **Sjednocení** fuzzy množin A a B: $\max\{\mu_A(u), \mu_B(u)\}$
- **Kartézský součin** fuzzy množin A a B: $\min\{\mu_A(u), \mu_B(u)\}$

Příklad: Každé hodnotě rychlosti přiřadíme číslo z intervalu <0,1>, který vyjadřuje míru našeho přesvědčení, že daná rychlost je nízká. Čím vyšší (nižší) je stupeň příslušnosti, tím více (méně) platí, že příslušná rychlost je nízká.

Problém při vyhledávání: Při vyhledávání dat klasickým způsobem sice uživatel získá nějakou množinu, ale již neví, jak hodně odpovídá výsledek jeho požadavku. Pokud se na výstup podíváme jako na fuzzy množinu, potom stupeň příslušnosti by měl odrážet míru relevantnosti.

Úlohou fuzzy teorie je zachytit vágně specifikované požadavky uživatele v dotazu a adekvátně k tomu vypočítat stupeň příslušnosti. Uživatel tedy musí mít možnost používat vágní pojmy buďto přímo nebo je jednoduchým způsobem reprezentovat.

Jeden z hlavních problémů je určení funkce příslušnosti. V případě, že prvky universa jsou reálné čísla, existuje více možností matematického popisu průběhu růstu respektive klesání hodnot stupně příslušnosti. Pro prvky universa v okolí hraničních bodů by mělo platit, že čím víc se blíží prvky universa k hraničním bodům, tím pomaleji roste (klesá) hodnota stupně příslušnosti.

2.7 Reprezentace znalostí

Efektivní reprezentace znalostí v počítači je právem považována za podstatný problém. Znalosti musí být reprezentovány tak, aby reprezentace:

- byla pro danou oblast dostatečně přirozenou a přitom expresivní
- umožnila aplikaci efektivních deduktivních prostředků
- zabezpečovala rychlý přístup k položkám v bázi znalostí i bázi dat

Na reprezentaci znalostí jsou kladeny často protichůdné požadavky. Při vytváření báze znalostí je velmi důležité, aby použitá reprezentace dovolovala jednoduše upřesňovat znalosti. Zejména je důležité, aby bylo možné inkrementálně rozšiřovat bázi znalostí. Jde tedy o **požadavek modularity reprezentace**. Je zřejmé, že báze znalostí bude modulární, bude-li složena z nezávislých, jednoduchých, uniformních částí. Přidáním dalších takovýchto částí zjemníme bázi znalostí, vzhledem k vzájemné nezávislosti nezasáhne úprava jedné části zbytek báze atd.

Na druhé straně však existují důvody proti takovémuto uspořádání znalostí. Jednotlivé znalosti vyjadřují rozmanité závislosti řešené problematiky, které mohou být od sebe svojí formou značně odlišné. Proto je zcela přirozený požadavek na různé, neuniformní reprezentace, postihující co nejlépe podstatu znalostí. Aby bylo možné jednotlivé znalosti rychle z báze vyvolávat, je důležité, aby příbuzné znalosti byly volně

sdružovány a nebyly reprezentovány nezávisle. Jedná se o **požadavek sémantického sdružování znalostí.**

Příkladem modulární reprezentace znalostí jsou produkční pravidla a jazyk predikátové logiky 1. řádu, myšlenku sdružování podobných znalostí vystihují rámce a scénáře a sémantické sítě (viz [7]).

2.7.1 Produkční pravidla

Systém produkčních pravidel je jedním z nejčastěji užívaných aparátů pro reprezentaci znalostí v expertních systémech. Obecně je produkční systém definován třemi složkami:

- souborem produkčních pravidel
- bází dat
- interpretem pravidel (řídící algoritmus rozhodující, kdy má být které pravidlo uplatněno).

Produkční pravidla mají tvar IF *situace S* THEN *akce A*. Aplikování takovéhoto pravidla znamená: "Nastala-li v bázi dat situace S, vykonej akci A" Produkční systém tedy pracuje v cyklu *rozpoznání situace* → *vykonání akce*. Podle omezení na tvar pravidel a podle sémantiky přisouzené produkčnímu systému, dostaneme konkrétní tvar odvozování. Budou-li např. pravidla i údaje v bázi dat (podmíněně) pravdivými formulemi výrokové logiky, dostaneme odvozování pravidlem *modus ponens*. Budeme-li chápat pravidla jako přepisovací pravidla nějaké gramatiky, lze jimi generovat příp. rozpoznávat slova příslušného jazyka. Nejčastěji používaným pro implementaci produkčních systémů je programovací jazyk LISP[6] (*List processing*).

[6] **LISP** je funkcionální programovací jazyk s dlouhou historií, jehož teoretické základy byly položeny v roce 1955 v *Proposal for Dartmouth Summer Research Project on Artificial Inteligence* (McCarthy, Minsky, Orchester, Shannon). Dnes se stále používá v oboru umělá inteligence.

Vhodnými aplikacemi reprezentace znalostí ve formě produkčních pravidel jsou takové oblasti, kde se zásoba znalostí skládá z mnoha disparátních faktů, na sobě nezávislých a kde lze řídící mechanizmus oddělit od reprezentace znalostí. Příkladem je expertní systém MYCIN [22], jehož cílem je pomoci lékaři při diagnostice a při výběru terapie. Tento systém obsahuje několik set produkčních pravidel, která reprezentují znalost lékaře-experta v dané oblasti, a umožňuje vyvozovat důsledky i na základě neúplné znalosti.

K výhodám produkčních systémů patří především jejich modulárnost. Jednotlivá produkční pravidla lze dodávat, rušit nebo pozměňovat, aniž se tím naruší některá jiná pravidla. Změna jednoho produkčního pravidla může způsobit změnu v chování systému jako celku, ale nemá přímý vliv na ostatní pravidla, protože aplikace těchto pravidel závisí pouze na obsahu aktivovaného ohniska. Na druhé straně tato modularita systému může vést k některým nepříznivým důsledkům, protože u rozsáhlých systémů přísné zachování principů modularity vede k neefektivnímu provádění programu. Při formulaci produkčních pravidel je třeba dodržovat přísné podmínky pro jejich tvar, což vede k jisté uniformitě struktury báze znalostí. Tato uniformita usnadňuje porozumění celému systému, i když algoritmy interpretu jsou pak méně průhledné, než by tomu bylo při použití programovacího jazyka.

Pravidla jsou nejpoužívanější prostředek pro reprezentaci znalostí. Jejich výhodou je jednoduchost a srozumitelnost (expert se často vyjadřuje způsobem, který lze přímo kódovat jako pravidla) a modularita (snadno se aktualizuje báze znalostí). Nevýhodou je, že pravidla neumožňují vyjadřovat strukturální znalosti v dané aplikační oblasti. Typickým příkladem takovýchto znalostí je taxonomie.

2.7.2 Predikátová logika

Predikátová logika je jedním z nejlépe prozkoumaných systémů pro reprezentaci a zpracování znalostí. Ačkoli původně navržená pro studium relace logického důsledku, stala se predikátová logika nebo v jiné terminologii teorie prvního řádu prototypem schémat pro reprezentaci znalostí. Původně byla predikátová logika zkoumána pro

potřeby matematické, ale brzy se ukázalo zejména v souvislosti s rozvojem informatiky, že jde o velmi univerzální prostředek umožňující analyzovat jazykové výrazy nezbytné právě pro reprezentaci znalostí v mnoha, na první pohled odlišných, tématických oblastech. Navíc se tento prostředek reprezentace ukázal jako výhodný pro studium odvoditelnosti (inference) mezi tvrzeními, což je patrně ten nejvýznamnější příspěvek predikátové logiky pro porozumění výpočtovým problémům soudobé informatiky.

Na rozdíl od elementárního pohledu výrokové logiky (která stojí v pozadí expertních systémů založených na deklarativních pravidlech), zkoumáme v teoriích prvního řádu samotnou vnitřní strukturu výroků, a proto musíme začít specifikací jazyka.

Jazyk predikátové logiky obsahuje:

- individuové proměnné
- predikátové symboly
- funkční symboly a konstanty
- kvantifikátory
- logické spojky (známé z výrokové logiky)

Předpokládáme, že konkrétní jazyk vždy obsahuje alespoň jeden predikátový symbol (aby bylo o čem hovořit), konstanty a funkční symboly už povinné nejsou, neboť nerozšiřují podstatně vyjadřovací sílu jazyků prvního řádu - jsou ovšem dobrou pomůckou jak pro zlepšení srozumitelnosti, tak pro vytvoření "technického" zázemí pro efektivní odvozování.

Po specifikaci jazyka můžeme přistoupit k obvyklému pojmu termu. Termy reprezentují všechna univerzální i speciální jména objektů, o kterých může být při použití daného jazyka řeč - vytvářejí se z individuových proměnných, konstant a funkčních symbolů. Právě funkční symboly slouží k vytváření jmen složitě strukturovaných objektů. Predikátové symboly a kvantifikátory pak slouží k vytváření formulí, což jsou právě ty jazykové výrazy, které reprezentují znalosti. Mezi nimi zastávají důležitou roli sentence

- formule, které mají tu vlastnost, že jim lze přiřadit pravdivostní hodnotu a přitom se chovají jako výroky.

Ze sémantického hlediska má nejvýznamnější roli pojem splnitelnost interpretace či modelu, což je interpretace, která je pravdivá. Systémy pro logickou dedukci by nebyly užitečné, pokud by umožňovaly odvodit z pravdivých předpokladů nepravdivé důsledky. Dedukční systém, který zaručuje, že každý logický důsledek je i sémantickým důsledkem (tj. co je dokazatelné, je i pravdivé), se nazývá korektní dedukční systém. Dedukční systém, ve kterém lze odvodit vše, co je sémantickým důsledkem (tj. co je pravdivé, je i dokazatelné), se nazývá úplný dedukční systém. Pro predikátovou logiku je však důkaz úplnosti dedukčního systému dost obtížný [26].

2.7.2.1 Resoluční princip

V predikátové logice je logické odvozování, spolu s výsledkem o korektnosti a úplnosti, jedinou cestou, jak zjišťovat splnitelnost množiny formulí. Zde nelze použít tabulkovou metodu oblíbenou ve výrokové logice pro svoji exponenciální časovou náročnost vzhledem k počtu proměnných. Množina všech možných interpretací predikátových formulí je totiž potenciálně nekonečná.

Jednou z možností, jak odvozování alespoň částečně automatizovat, je **resoluční princip**. Resoluční princip (pro výrokový počet) lze zjednodušeně popsat následujícím postupem (podle [26]):

- Všechny formule převedeme do tautologicky ekvivalentních formulí v konjunktivní normální formě (klausule)
- Pokud lze v některé dvojici klausulí nalést komplementární literály (dvojici literálů, kde jeden je negací druhého), vytvoříme jejich resolventu, tj. novou klausuli, kde vynecháme komplementární literály. Tento postup opakujeme tak dlouho, dokud lze najít dvojici klausulí s komplementárními literály nebo pokud nezískáme prázdnou klasuli (resolventu)

- Pokud proces končí získáním prázdné klausule, je daná množina klausulí nesplnitelná. Pokud se proces vytváření zastaví bez získání prázdné klausule, množina klasulí je splnitelná.

V predikátové logice je idea resolučního principu totožná s postupem ve výrokové logice. Komplementarita literálů pro úspěšnou tvorbu resolventy dvou klausulí bude v predikátové logice ale podmíněna hledáním speciální substituce (maximálního unifikátoru). Postup hledání vhodné substituce se označuje jako unifikační algoritmus [26].

2.7.3 Sémantické sítě

Sémantické sítě (počáteční formulace v [19]) jsou výsledkem úsilí najít reprezentaci znalostí, která by jednak zajistila, aby v ní každý fakt byl snadno nalezen, tj. aby byl mimo jiné reprezentován jenom na jednom místě, jednak aby všechna fakta týkající se některého předmětu byla i v reprezentaci dána do jasné souvislosti a aby každou novou informaci bylo možno "začlenit" do kontextu dosavadních znalostí. Znalost musí být organizována pomocí pojmů, s nimiž jsou sdruženy nejen popisy, ale i způsoby, jak s nimi zacházet tak, aby bylo možno vyjadřovat pouze částečnou znalost a ta aby mohla být postupně doplňována tak, aby umožnila srovnání daného pojmu s prostorem, který mu v síti dat odpovídá a tím umožnila pochopit kontext, v němž se pojem vyskytuje.

U odborníků v oblasti umělé inteligence vzbuzují sémantické sítě zájem především pro svůj vyjadřovací potenciál, o němž se věří, že může zakódovat každý fakt či koncept kódovatelný v jiných formálních systémech a že tedy může síť sloužit jako společné médium reprezentace různých druhů znalostí. Znalosti uložené v síti jsou deklarativní a jsou použitelné pro potřebu některé aktivity jenom tehdy, existuje-li nějaká vnější procedura k jejich vybavování a aplikaci. Ta by měla ztělesňovat znalosti o manipulaci informace v síti a může mít i přístup k jiným informačním zdrojům. Z hlediska použití informace obsažené v síti jsou sémantické sítě i tyto procedury vzájemně závislými partnery. Proto je vždy u sítí důležité uvažovat současně procedury pro manipulaci s nimi.

2.7.4 Rámce a scénáře

Projevem snahy o vytvoření obecné reprezentace, použitelné pro široký okruh problémů, je tzv. teorie rámců. Její základní myšlenkou je, že vstoupíme-li do nové situace, vybereme z paměti strukturu, která jí odpovídá (podle předchozí zkušenosti) a srovnáme ji s aktuálně vnímaným světem. Rámce jsou tedy datové struktury, reprezentující znalosti pomocí typických příkladů.

V původní formulaci Minského [10] se rámec považuje za základní, prototypovou strukturu uloženou v paměti a vytvářející jakousi kostru, kterou člověk změnou jednotlivosti přizpůsobuje skutečnosti, když se setká s novou situací. Rámec je tedy charakterizován jako datová struktura určená k reprezentaci stereotypní situace (jako je např. pobyt v jistém druhu obývacího pokoje či návštěva večírku k oslavě dětských narozenin). Ke každému rámci je připojeno několik druhů informace: některé se týkají toho, jak rámce použít, jiné toho, co se dá očekávat jako příští událost, a další zase informují, co dělat, nejsou-li očekávání splněna.

Rámec je možno si představit jako strukturu sestávající z uzlů a hran. "Horní" uzly jsou pevné, reprezentují fakta, jež jsou vždy platná pro danou situaci, uzly na nižší úrovni představují volná místa (*slots*), tedy rubriky, které musí být naplněny specifickými údaji, přičemž mohou být specifikována omezení na to, které údaje je možné doplnit.

Vyplňování hodnoty položky probíhá buď jejím vyhledáním přímo v databázi (kde jsou zachyceny údaje o aktuálním světě) nebo zprostředkovaně pomocí procedury (sdružené s příslušnou položkou) odkazující eventuálně na jiný podrámec. Předpokládá se, že rubriky jsou nejprve doplněny prozatímními objekty, které slouží jako bezpříznakové (primární) údaje a jsou nahrazeny objekty sekundárními (se specifickým příznakem), ukáže-li se to jako vhodné pro určitou situaci. Rámce se kombinují do struktur, které se v literatuře nazývají systémy rámců, epizody, scénáře, plány apod.

Rámce byly navrženy pro reprezentování stereotypních znalostí. Představují např. vhodný prostředek pro zachycení inherentní struktury nějakých konceptů. Jsou rovněž vhodné pro aplikace, kde se provádí porovnávání mezi daty a hypotézami.

Rámce mají ale i své nevýhody. Vzhledem k provázanosti rámců může přítomnost nebo nepřítomnost jiných rámců ovlivnit položky v daném rámci (znalosti reprezentované pomocí rámců jsou tedy méně modulární). Jiným problémem je, že systémy používající rámce obvykle neumožňují rozlišit mezi základními vlastnostmi (hodnotami položek), které musí mít každá instance daného rámce a doplňkovými vlastnosti, které příslušné instance mohou, ale nemusí mít. To, že hodnota položky může být změněna kdekoliv v hierarchii rámců znamená, že nelze zaručit, že některé vlastnosti nebudou nekonzistentní.

2.7.5 Znalostní ontologie

Znalostní ontologie, jako pojem znalostního modelování, je v současnosti chápána jako označení domluvené terminologie pro určitou aplikační oblast, která umožňuje sdílení znalostí z této oblasti. Nejčastější jsou tzv. doménové ontologie, jejichž předmětem je určitá specifická věcná oblast (např. medicína) [A8].

Hlavními přínosy využívání ontologií jsou:

- usnadnění komunikace mezi lidmi a organizacemi díky jasnému vymezení používaných pojmů
- usnadnění spolupráce počítačových systémů - ontologie zde hraje roli výměnného formátu znalostí
- usnadnění vývoje znalostní aplikace - ontologie zde tvoří základní, konceptuální vrstvu báze znalostí

Příkladem doménové ontologie v oblasti medicíny je ontologie UMLS (*Unified Medical Language System*) [28]. Ontologie UMLS je tvořena třemi základními částmi:

- Metatezaurus - víceúčelová vícejazyčná slovníková databáze obsahující informace o biomedicínských a zdravotnických konceptech, jejich názvech (synonymech) a relacích mezi nimi.

- Sémantická síť - tvořena sémantickými typy, které umožňují konzistentní kategorizaci konceptů reprezentovaných v metatezauru, a množinou relací mezi sémantickými typy.
- Lexikon pojmů - obecný anglický lexikon zahrnující mnoho biomedicínských pojmů ve formě jak běžných anglických slov, tak biomedicínských termínů. Lexikon se využívá v systémech pro porozumění přirozenému jazyku.

3 Reprezentace znalostí v oborových doporučeních

V řadě oblastí lidské činnosti je možné definovat doporučené postupy řešení problémů nebo typických případů. Jedním z oborů, kde se setkáme s doporučenými postupy, je medicína. Vzhledem k poměrně obsáhlému a odborně propracovanému souboru lékařských doporučení je pozornost autora zaměřena především na modelování znalostí (doporučených postupů) v medicíně, získané poznatky lze ale aplikovat i v jiných oborech.

Lékařská doporučení jsou potřebná pro podporu rozhodování v klinické praxi. Hlavním záměrem je zvýšení kvality péče o pacienty a snížení nákladů. Doporučení jsou obvykle vydávána expertními skupinami v příslušné oblasti medicíny (např. kardiologie). Doporučení sice nejsou právně závazným dokumentem, jejich důležitost a vážnost je ale potvrzena záštitou českých, evropských nebo světových lékařských institucí (např. České kardiologické společnosti).

Oborová (lékařská) doporučení jsou vydávána jako volný text, ve kterém jsou zaznamenány doporučené postupy řešení určitých problémů (např. diagnostika a léčba nemocí pacientů). Jedná se tedy o způsob zaznamenání znalostí experta, resp. expertní skupiny, do textové podoby. Znalosti získává expert převážně metodou indukce (viz kapitola 2.5.1), kdy z opakovaného pozorování (vyšetření pacientů) vyvozuje obecné závěry. Dalšími metodami požitými při získávání znalostí v doporučeních jsou dedukce (viz kapitola 2.5.2) a analogie (viz kapitola 2.5.4), kdy expert (lékař) na základě dříve vyřešených případů (výsledků léčby) a obecných předpokladů odvozuje závěr (např. způsob léčby) na podobných případech. Především v oblasti medicíny se ve velké míře uplatní i heuristiky (viz kapitola 2.5.7), tj. formalizace znalostí založených na zkušenostech experta (lékaře).

Přístup k informacím obsaženým v konvenční podobě doporučení (ve formě volného textu) může být bohužel obtížný. Předpokladem pro vývoj systémů podpory rozhodování je tedy vytvoření počítačově zpracovatelné reprezentace znalostí

obsažených v lékařských doporučeních. Proto se vývojem takové reprezentace doporučení zabývá řada výzkumných skupin.

3.1 Způsoby reprezentace lékařských doporučení

Pravděpodobně nejznámějším jazykem pro reprezentaci lékařských doporučení v systémech podpory rozhodování je Ardenská syntaxe [17]. Jedná se o formalismus založený na pravidlech pro reprezentaci a vyhodnocování medicínských podmínek a doporučení ve formě medicínských logických modulů (MLM - *Medical Logic Modules*). Řada přístupů sdílí hierarchickou dekompozici doporučení ve formě sítí dílčích úkolů, které se rozkládají v čase [18]. Všechny dále uvedené metody modelování mohou kombinovat jednotlivé kroky doporučení v orientovaných cyklických grafech.

Asbru je vyvíjen ve spolupráci Univerzity Bena Guriona a Vídeňské technologické univerzity [21]. Jedná se časově orientovaný jazyk založený na specifikaci záměrů a základní kostry plánu, který je používán pro reprezentaci klinických protokolů.

EON byl vyvinut na Stanfordské univerzitě a poskytuje sadu modelů a programových komponent pro vytváření aplikací postavených na doporučeních [24]. EON používá úkolově orientovaný přístup k definici služeb podpory rozhodování, které mohou být implementovány různými technikami [25]. V EON architektuře se používá prostředí Protége2000 ke konstrukci informačního modelu pacientových dat, model medicínské specializace a model doporučení, který formalizuje znalosti potřebné k realizaci doporučení s ohledem na klinická rozhodnutí a aktivity.

GUIDE je část modelovacího a výkonného rámce, který je vyvíjen na Univerzitě v Pavii [20]. Podporuje integraci modelů doporučení do pracovních procesů v organizaci použitím analyticko rozhodovacích modelů jako jsou rozhodovací stromy a diagramy vlivu, a simulaci implementace doporučení v prostředí Petriho sítí.

PRODIGY byl vyvinut na Univerzitě v Newcastlu ve Velké Británii [16]. Cílem projektu PRODIGY je vytvoření nejjednoduššího, rychleji pochopitelného modelu

reprezentace třídy doporučení. Týmy klinických lékařů zde používají vývojové prostředí Protégé [6] pro kódování tří komplexních doporučení řízení chronických nemocí.

PROforma byla vyvinuta v Moderní výpočetní laboratoři výzkumu rakoviny ve Velké Británii [4]. Kombinuje logické programování a objektově orientované modelování, formálně zakotveném v jazyce R^2L. PROforma podporuje čtyři typy činností: akce, složené plány, rozhodování a dotazy. Všechny činnosti sdílejí atributy popisující cíle, kontrolní toky, předpoklady a následné podmínky.

3.2 GLIF model lékařských doporučení

GLIF (*Guideline Interchange Format*) verze 3 byl vyvinut ve spolupráci Univerzity Columbia, Harvardské, McGillovy a Stanfordské univerzity (InterMed). Poslední verze GLIF modelu byla publikována ve specifikaci GLIF3.5 [14][29][15]. GLIF model je reprezentován orientovaným grafem, jehož uzly mohou být typu *akce, rozhodování, větvení, synchronizace* a *stav*. Použitý výrazový jazyk byl původně založen na Ardenské syntaxi a pro popis kritérií a výrazů byl používán jazyk GEL (*Guideline Expression Language*). V současnosti je používán jazyk GELLO, tj. rozšířený objektově orientovaný jazyk, který také podporuje množinu funkcí z jazyka GEL [23].

GLIF model byl také použit jako model reprezentace počítačových klinických doporučení v projektu SAPHIRE [27]. Systém SAPHIRE průběžně monitoruje pacienty prostřednictvím jednoúčelových agentů a pomocí inteligentního systému podpory rozhodování pomáhá profesionálům v zajištění zdravotní péče.

3.2.1 Model GLIF v UML

UML (*Unified Modelling Languauge*) je grafická notace pro objektové modelování. Přeloženo do češtiny to znamená způsob, jak pomocí obrázků přehledně a srozumitelně dokumentovat objektový model informačního nebo znalostního systému. UML je primárně určen k dokumentování systému ve všech fázích jeho vývoje od úvodních

myšlenek o vzhledu systému až po hotový program. Výrazně usnadňuje komunikaci mezi znalostními inženýry a experty, případně mezi programátory navzájem. Pomocí UML lze vytvářet přehledné diagramy doplňující textovou dokumentaci.

UML definuje osm základních druhů diagramů pro modelování veškerých shromážděných informací o systému. Jsou to (podle [1]):

- **Diagramy tříd** – zachycují statickou strukturu systému, znázorňují třídy a jejich vazby. Třída je kategorie nebo skupina věcí, které mají podobné vlastnosti a stejné nebo podobné chování. Chování věcí v této třídě popisují specifické operace – metody.
- **Diagramy případů užití** – představují popis chování systému z pohledu uživatele.
- **Diagramy činností (aktivit)** – popisují průběh procesu či činnosti. Činnosti se běžně vyskytují v podobě sekvence, uživatel může ale specifikovat i body rozhodování, paralelně prováděné akce (včetně synchronizačních bodů) a zasílání signálů.
- **Diagramy spolupráce** – zachycují průběh činnosti v systému a komunikaci spolupracujících objektů zasíláním zpráv. Oproti diagramům sekvencí jsou více zaměřeny na strukturu.
- **Stavové diagramy** – popisují chování objektu nebo algoritmu. V každém okamžiku je objekt v určitém stavu, který může být změněn. Stavové diagramy znázorňují přechody objektů z jednoho stavu do druhého.
- **Diagramy sekvencí** – zachycují časovou dynamiku interakcí mezi objekty. Uživatel tak může snadno sledovat průběh událostí včetně bodů synchronizace a zasílání zpráv.
- **Diagramy komponent** a **diagramy nasazení** – popisují rozdělení výsledného systému na funkční celky (komponenty) a jejich umístění na výpočetních uzlech celého systému.

Pro reprezentaci modelu GLIF je možné v UML využít kombinaci diagramů činností (aktivit) a stavových diagramů. Výhodou je využití všeobecně podporované metodologie UML s řadou, i volně dostupných, nástrojů pro konstrukci diagramů a jejich následnou implementaci ve formálních (programovacích) jazycích. Nevýhodou je obtížné zachycení všech rozhodovacích kritérií v rozhodovacích bodech diagramu činností.

3.2.2 Model GLIF v systému Protégé

Protégé[7] [4] je vývojové prostředí pro návrh znalostních systémů, které umožňuje především konstrukci doménových ontologií, uživatelskou úpravu vstupních formulářů a vkládání dat.

Jedná se o platformu, která je snadno rozšiřitelná o grafické komponenty (grafy, tabulky), multimediální prvky (zvuky, obrázky, videa) a různé formáty uložení a zobrazení v relačních databázích, RDF[8], XML nebo HTML[9]. Jedním ze standardních rozšíření systému je i podpora GLIF modelu. Architekturu celého systému Protégé znázorňuje obrázek 4.

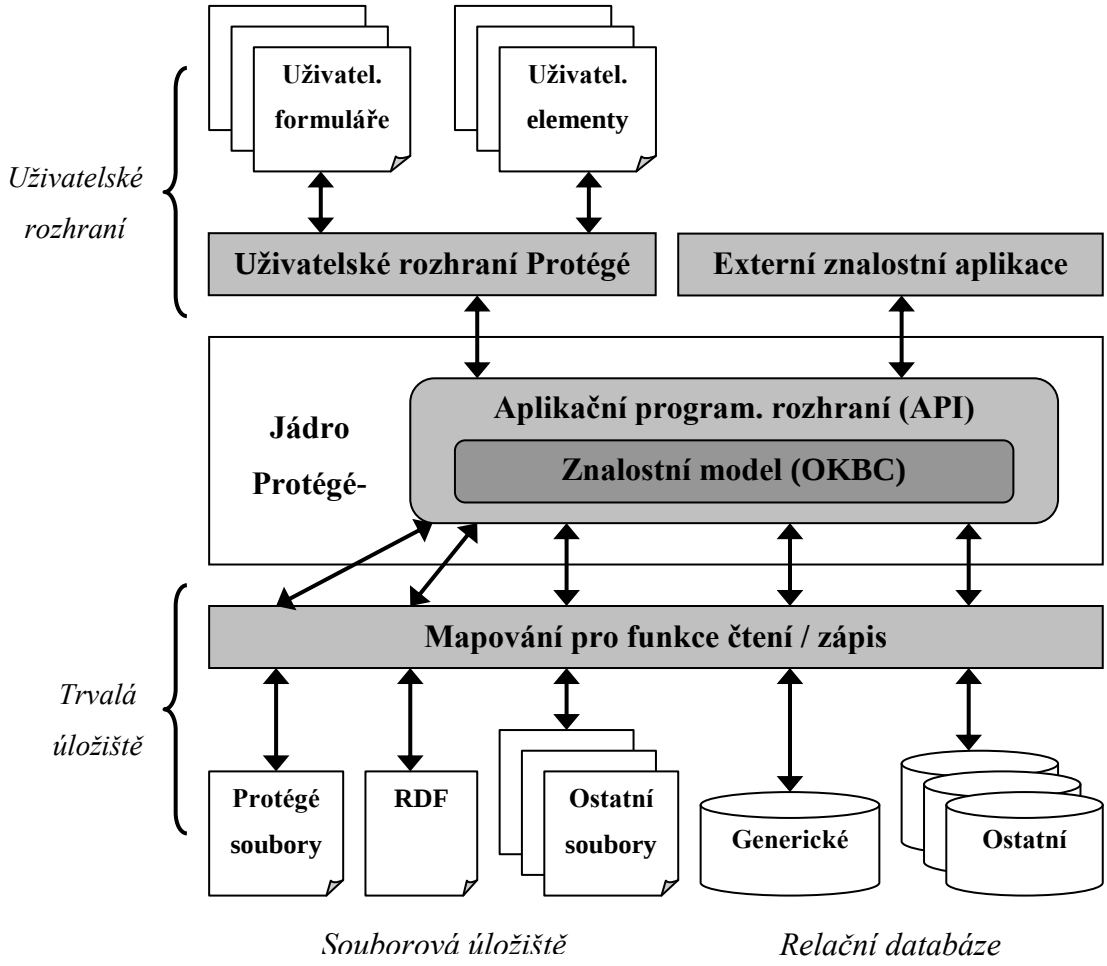
3.2.2.1 Znalostní model Protégé

Jádrem systému Protégé je znalostní model. Jakákoliv akce s objekty (instance, třídy,…) uloženými ve znalostní bázi je možná pouze prostřednictvím aplikačního programátorského rozhraní (API – *application programmers interface*).

[7] **Protégé** je nástrojem vyvinutým týmem M. Musena v institutu Stanford Medical Informatics. Vývoj prošel od verze Protégé-I (1989) přes Protégé-II (1992), Protégé/Win (1996) až po současnou verzi Protégé-2000 (1999).

[8] **RDF** (*Resource description framework*) – vestavěný formát uložení znalostní báze v systému Protégé

[9] **HTML** (*HyperText Markup Language*) - jazyk, který je základem pro vytváření webových stránek. Základem tohoto jazyka jsou značky (*tag*), což je text ve špičatých závorkách < >.



obrázek 4 Architektura systému Protégé-2000

V původní verzi systému Protégé byl znalostní model založen na objektové implementaci produkčních pravidel v jazyce C (CLIPS – *C Language Integrated Production System*). Současná verze znalostního modelu je založena na protokolu otevřené připojitelnosti ke znalostní bázi (OKBC – *Open Knowledge Base Connectivity*) (viz [7]). OKBC protokol se snaží stanovit sadu obecných pravidel, které umožňují lepší spolupráci různých znalostních systémů.

Zavedením tzv. **meta-tříd**, jejichž instancemi jsou třídy, je možné vytvářet nebo měnit třídy stejně jednoduše jako instance. Tím je podpořena idea, kdy ontologie jako celek je vytvářena znalostním inženýrem a vlastní naplnění znalostní báze provádí doménový

expert. Díky meta-třídám není expert limitován a má k dispozici rozšířený sortiment objektů a tříd, které může použít.

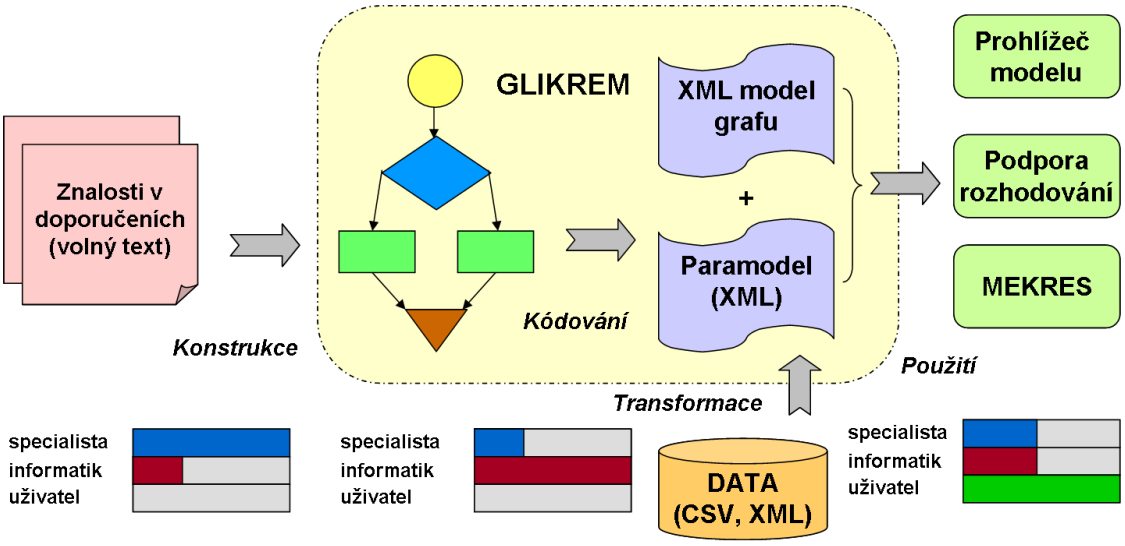
Přesto, že je možné v prostředí Protégé modelovat doporučení pomocí GLIF modelu, je toto prostředí určeno především pro návrh znalostních ontologií a znalostních bází. GLIF model slouží především k reprezentaci rozhodovacího algoritmu (procedurálních znalostí) obsaženého v doporučeních, který představuje odlišné konstrukce a vztahy oproti konstrukcím používaným ve znalostních ontologiích. Tím je dáno určité omezení v modelování GLIFu v prostředí Protégé.

4 Návrh znalostního modelu GLIKREM

Návrh modelu reprezentace znalostí obsažených v oborových doporučeních GLIKREM (*GuideLInes Knowledge REpresentation Model*) vychází GLIF (*Guidelines Interchange Format*) modelu publikovaného ve specifikaci GLIF 3.5 [27]. Oproti původnímu modelu obsahuje GLIKREM vlastní rozšíření modelu o nové prvky, upřesnění rozhodování v rozhodovacích krocích, následnou implementaci v jazyce XML (*eXtensible Markup Language*) a návrh modelu parametrů jako datového rozhraní mezi modelem a reálnými daty.

4.1 Proces konstrukce a implementace GLIKREM

Celý proces konstrukce modelu znalostí (GLIKREM) z volného textu oborových doporučení, jeho následnou reprezentaci v XML a použití výsledného modelu znázorňuje obrázek 5.



obrázek 5 Proces konstrukce, kódování a použití GLIKREM

Ve fázi konstrukce modelu z textových doporučení je důležité najít procesní strukturu doporučení, všechny podstatné parametry modelu a jejich vzájemné vztahy. K tomu lze využít některou z metod dolování znalostí z textu [2]. Při automatickém hledání

parametrů je třeba brát v úvahu i víceslovná spojení, čímž se proces hledání stává podstatně složitější a náchylnější na chyby (viz kapitola 2.5.8).

Efektivnější řešení spočívá ve spolupráci informatika a experta z daného oboru, nejlépe autora příslušných doporučení. Výsledkem jejich spolupráce je grafický model (GLIKREM), který nejlépe odpovídá znalostem obsaženým v textových doporučeních. Grafický model je možné (vhodné) vytvořit přímo expertem v grafickém editoru (viz kapitola 5.1).

Ve fázi implementace (kódování) modelu je grafický model doporučení zakódován v XML. Mimoto je vytvořen i seznam základních a odvozených parametrů modelu. Základní parametry představují přímo měřitelné (nebo jinak získatelné) hodnoty, odvozené parametry se získají aritmetickou, logickou či logicko-aritmetickou operací nad základními parametry.

Při sestavení seznamu parametrů a jejich rozdělení na základní a odvozené hraje opět významnou roli spolupráce s expertem příslušného oboru. Výsledkem je datový model, který pak slouží jako rozhraní mezi GLIKREM a skutečnými vstupními daty uloženými v databázi či informačním systému.

Použití GLIKREM zakódovaného v XML je možné v různých typech aplikací pro různá oborová doporučení nebo reprezentace procedurálních znalostí. Jedná se například o dále uvedené typy aplikací (podrobněji o vybraných aplikacích v kapitole 5).

- **Obecný prohlížeč** umožňující přehledné zobrazení libovolného modelu doporučení ve formě grafu a umožňuje snadnou prezentaci prostřednictvím moderních technologií (internet, mobilní telefony, …). Tento typ je vhodný především pro studijní účely, např. pro praktické lékaře (viz kapitola 5.2).
- **Obecný prohlížeč řízený daty** umožňuje procházení a zobrazení grafického modelu na základě uložených reálných dat. Vyhodnocování rozhodovacích podmínek grafu je prováděno automaticky a v případě, že se systém nedokáže

rozhodnout, je vyžadováno vložení chybějících dat nebo ruční rozhodnutí (viz kapitola 5.3).

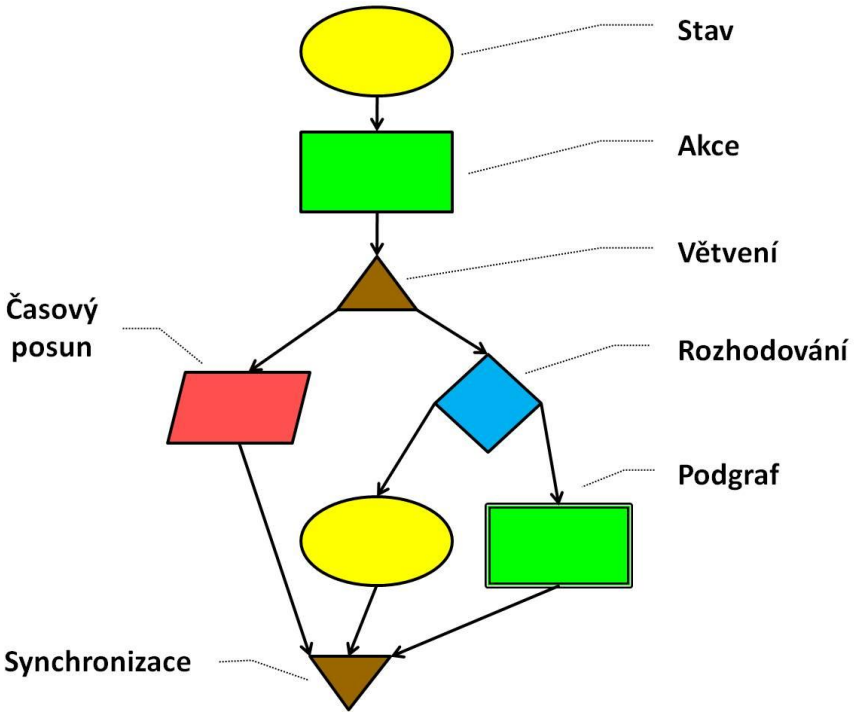
- **Připomínkový systém** je prohlížeč modelu řízeného daty, spuštěný souběžně se zadáváním dat do databáze. Systém kontroluje, zda jsou zadávaná data v souladu s oborovými doporučeními. V případě nesouladu je uživatel na tuto skutečnost upozorněn.
- **Systém reprezentace lékařských znalostí MEKRES** (MEdical *Knowledge REpresentation System*) je systém založený na reprezentaci znalostí pomocí GLIKREM a výběru relevantního modelu na základě vložených dat o pacientovi (viz kapitola 5.4).

4.2 Konstrukce GLIKREM

Výsledkem fáze konstrukce reprezentace znalostí obsažených v textových doporučeních je uspořádaná pětice: $\boldsymbol{GLIKREM} = (\boldsymbol{V}, \boldsymbol{v_0}, \boldsymbol{H}, \boldsymbol{\tau}, \boldsymbol{P})$, kde:

- $\boldsymbol{V}$ je neprázdná konečná množina vrcholů prostého orientovaného grafu zvaných kroky, pro kterou platí $\boldsymbol{V} = \boldsymbol{V_s} \cup \boldsymbol{V_a} \cup \boldsymbol{V_c} \cup \boldsymbol{V_b} \cup \boldsymbol{V_y} \cup \boldsymbol{V_t}$ a zároveň $V_i \cap V_j = \emptyset$ pro $i \neq j$, kde:
    - V_s je konečná množina vrcholů typu *stav*
    - V_a je konečná množina vrcholů typu *akce*
    - V_c je konečná množina vrcholů typu *rozhodování*
    - V_b je konečná množina vrcholů typu *větvení*
    - V_y je konečná množina vrcholů typu *synchronizace*
    - V_t je konečná množina vrcholů typu *časový posun*
- $\boldsymbol{H} \subseteq \boldsymbol{V} \times \boldsymbol{V}$ je neprázdná konečná množina orientovaných hran $(v_i, v_j) \in H$, kde $v_i, v_j \in V$ a $v_i \neq v_j$. Hranou pak rozumíme spojnici z vrcholu v_i do vrcholu v_j.

- $\boldsymbol{v_0} \in \boldsymbol{V_s}$ je počáteční vrchol typu *stav*. Jedná se o vrchol, který nemá žádné vstupy, tj. platí pro něj $\forall v \in V: \{(v, v_0): (v, v_0) \in H\} = \emptyset$.
- $\boldsymbol{K} \subseteq \boldsymbol{V_s}$ je neprázdná konečná množina koncových vrcholů typu *stav*. Každý koncový vrchol $v_k \in K$ nemá žádné výstupy (hrany vedoucích z koncového vrcholu), tj. pro každý vrchol $v_k \in K$ platí $\forall v \in V: \{(v_k, v): (v_k, v) \in H\} = \emptyset$.
- $\boldsymbol{\tau}$ je aktuální čas v modelu, který může nabývat diskrétních hodnot (časových transakcí) $\tau_0, \tau_1, \ldots, \tau_{max}$, kde τ_{max} představuje maximální časový úsek sledovaný v modelu. Pro každý model je definován počáteční čas τ_0 a jednotkový časový posun $\Delta\tau$.
- $\boldsymbol{P} = \{\boldsymbol{p_i^t}\}$ je posloupnost parametrů modelu (paramodel), které jsou stanoveny pro časy $t \in \{t_0, t_1, \ldots, t_{max}\}$.



obrázek 6 Všechny typy vrcholů v GLIKREM

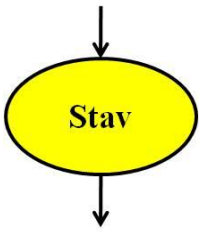
4.2.1 Typy vrcholů

Každý vrchol (krok) $v_i \in V$ může mít žádný nebo konečný počet (n) vstupů (hran vedoucích do vrcholu v_i) a žádný nebo konečný počet (m) výstupů (hran vedoucích z vrcholu v_i). Vrchol v_i může být pouze jeden z následujících možných typů (viz obrázek 6):

4.2.1.1 Vrchol typu *stav*

Vrchol $\boldsymbol{v_i \in V_s}$ typu *stav* (*state*) značí stav, ve kterém se zkoumaný objekt nachází při vstupu do modelu nebo po provedení některého předchozího kroku.

Stav, značený oválem, má žádný nebo jeden vstup (vstupní hranu) a žádný nebo jeden výstup.



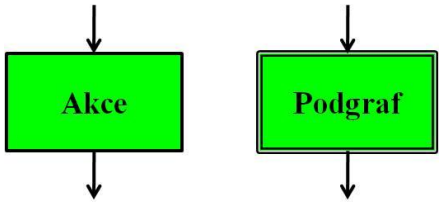
obrázek 7 Grafický symbol vrcholu typu *stav*

4.2.1.2 Vrchol typu *akce*

Vrchol $\boldsymbol{v_i \in V_a}$ typu *akce* (*action*) může představovat jednoduchou akci, tj. specifickou činnost nebo událost. Jednoduchou akci značíme obdélníkem s jednoduchým okrajem.

Každý podgraf, který je grafem ve smyslu definice GLIKREM, je (v grafu vyšší úrovně) také vrcholem typu *akce*. Takovou akci (podgraf) značíme obdélníkem s dvojitým okrajem.

Každá akce (jednoduchá akce i podgraf) má právě jeden vstup (vstupní hranu) a právě jeden výstup (výstupní hranu).

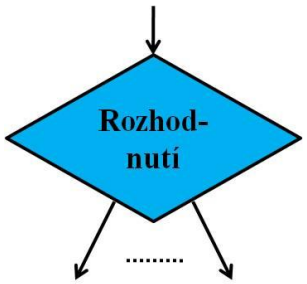


obrázek 8 Grafický symbol vrcholu typu *akce*

4.2.1.3 Vrchol typu *rozhodování*

Vrchol $\boldsymbol{v_i} \in \boldsymbol{V_c}$ typu *rozhodování* (*case*) představuje větvení (výběr následného kroku) na základě splnění logického kritéria (kritérií), kdy další postup grafem je dán výsledkem aritmetického nebo logického výrazu nad konkrétními daty anebo rozhodnutí uživatele, kterou částí grafu bude dále pokračovat.

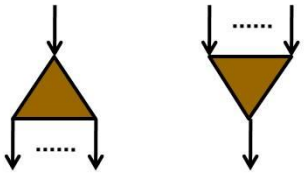
Rozhodování se značí kosočtvercem, který má právě jeden vstup (vstupní hranu) a dva nebo více výstupů (výstupní hrany) dále označovaných jako následné větve či volby.



obrázek 9 Grafický symbol vrcholu typu *rozhodování*

4.2.1.4 Vrchol typu *větvení* a *synchronizace*

Vrchol $\boldsymbol{v_i} \in \boldsymbol{V_b}$ typu *větvení* (*branch*) se používá při modelování nezávislých větví grafu, které mohou probíhat souběžně. Větvení se značí trojúhelníkem s vrcholem nahoru a má právě jeden vstup (vstupní hranu) a dva nebo více výstupů (výstupních hran).



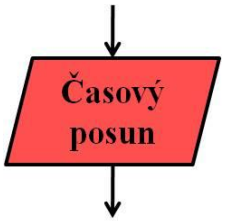
obrázek 10 Grafický symbol vrcholu typu *větvení* a *synchronizace*

Vrchol $\boldsymbol{v_i} \in \boldsymbol{V_y}$ typu *synchronizace* (*synchronization*) slouží pro souběžně probíhající větve garfu jako slučovací bod po splnění synchronizační podmínky (viz kapitola 4.2.7). Synchronizace, značená trojúhelníkem s vrcholem dolu, má dva nebo více vstupů (vstupních hran) a právě jeden výstup (výstupní hranu).

4.2.1.5 Vrchol typu *časový posun*

Vrchol $\boldsymbol{v_i} \in \boldsymbol{V_s}$ typu *časový posun* (*time shift*) značí krok, ve kterém dochází k posunu času modelu τ o zadaný počet (s) časových jednotek ($\Delta\tau$), tj. $\boldsymbol{\tau = \tau + s \cdot \Delta\tau}$ (viz kapitola 4.2.9)

Časový posun, značený kosodélníkem, má právě jeden vstup (vstupní hranu) a právě jeden výstup (výstupní hranu).

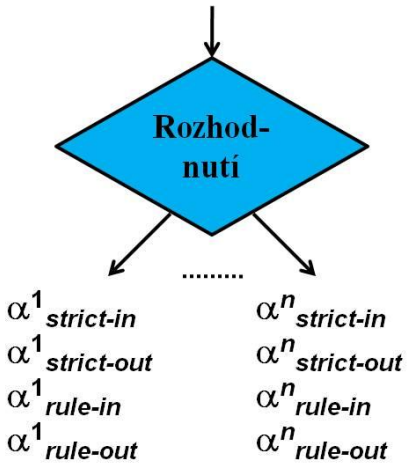


obrázek 11 Grafický symbol vrcholu typu *časový posun*

4.2.2 Rozhodovací kritéria

V každém rozhodovacím kroku jsou pro každou následnou volbu (hranu) $\alpha^1 \ldots \alpha^n \in \boldsymbol{H}$ definovány čtyři rozhodovací kritéria (viz obrázek 12). Na základě jejich vyhodnocení je vybrán, automaticky nebo ručně, následný postup modelem. Pro každou následnou větev α_i bude provedeno následující vyhodnocení rozhodovacích kritérií:

- *strict-in* – jestliže je splněna podmínka (např. logický výraz) nezávisle na uživateli, určitě se bude pokračovat příslušnou větví α_i
- *strict-out* – jestliže je splněna podmínka, příslušnou větví α_i se určitě nebude pokračovat – tato větev je zakázána
- *rule-in* – při splnění této podmínky je pouze doporučeno pokračovat následnou větví α_i – je vyžadován zásah uživatele, uživatel by si měl vybrat pouze z větví s pravdivou podmínkou *rule-in*.
- *rule-out* – při splnění této podmínky není doporučeno pokračovat následnou větví α_i – opět vyžadován zásah uživatele



obrázek 12 Rozhodovací kritéria v rozhodovacím kroku

4.2.3 Parametry a paramodel

Všechny typy rozhodovacích kritérií v rozhodovacím nebo synchronizačním vrcholu mají obvykle tvar logické formule. Logickými proměnnými jsou parametry z posloupnosti (paramodelu) $P = \{p_i^t\}$, kde index $i = 1 \ldots n$ slouží k rozlišení jednotlivých parametrů modelu a index t pak rozlišuje hodnotu každého parametru p_i v čase $t_0 \ldots t_{max}$.

Při sestavení seznamu parametrů a jejich rozdělení na základní a odvozené hraje opět významnou roli spolupráce s expertem příslušného oboru. Výsledkem je datový model, který pak slouží jako rozhraní mezi GLIKREM a skutečnými daty uloženými v databázi či informačním systému.

Parametry p_i^t paramodelu *P* mohou být trojího typu a to:

- **base** – základní parametr, tj. přímo měřitelná nebo získatelná číselná či logická hodnota
- **derived** – parametr odvozený ze základních provedením logické, aritmetické či logicko-aritmetické operace. Výsledkem může být číselná nebo logická hodnota
- **onfly** - pomocný parametr (obsahující pouze logickou hodnotu) pro definici synchronizačních podmínek.

4.2.4 Rozhodování ve tří-hodnotové logice

Při vyhodnocování logických podmínek (*strict-in, strict-out, rule-in, rule-out*) se velice často setkáme se situací, kdy nejsou známy jednoznačné hodnoty všech vyhodnocovaných proměnných (vstupních parametrů). Neznámá hodnota proměnné nemusí být přitom chybou a je třeba takový stav brát v úvahu. Logické formule, složené z proměnných (parametrů modelu) a logických či relačních operandů, mohou nabývat následující stavy:

- **pravda** – formule je splněná – označíme hodnotou 1
- **nepravda** – formule není splněna – označíme hodnotou 0
- **neznámo** – stav formule není jednoznačně určený – označíme hodnotou $\frac{1}{2}$

Z matematické logiky plyne, že jakoukoliv logickou formuli lze převést do disjunktivního (resp. konjunktivního) normálního tvaru, složeného pouze z logických operací negace (doplněk), konjunkce (logický součin) a disjunkce (logický součet). Výsledný tvar formule je ekvivalentní s původní formulí.

Definici operací doplněk, logický součin a logický součet v tří-hodnotové logice uvádějí následující vztahy Booleovy algebry:

- doplněk: $\bar{p} = 1 - p$
- logický součin: $(p_1 \cdot p_2 \cdot \ldots \cdot p_n) = \min\{p_1, p_2, \ldots, p_n\}$
- logický součet: $(p_1 + p_2 + \cdots + p_n) = \max\{p_1, p_2, \ldots, p_n\}$

4.2.5 Pravidla rozhodování v rozhodovacích krocích

Každý rozhodovací krok v modelu GLIKREM obsahuje minimálně dvě a maximálně *n* následných větví (hran, které vedou z daného rozhodovacího kroku). Pro každou následnou větev jsou definována rozhodovací kritéria *strict-in*, *strict-out*, *rule-in* a *rule-out* (viz obrázek 12).

Při vyhodnocování rozhodovacích kritérií (v tří-hodnotové logice) a výběru následné větvě (volby) ze všech možných větví (voleb) vedoucích z rozhodovacího kroku definujeme následující postup:

Fáze načtení vstupních parametrů:

- Z modelu parametrů (paramodelu) jsou načteny všechny dostupné hodnoty parametrů, které vstupují do rozhodovacích kritérií všech následných větví

Fáze vyhodnocení striktních kritérií:

- Nejprve se vyhodnotí kritérium *strict-out* příslušné větve. Při jejím splnění (hodnota pravda – 1) se další kritéria již nevyhodnocují, z pohledu možného průchodu grafem je příslušná větev zakázána.
- V případě, že nelze kritérium *strict-out* vyhodnotit jednoznačně, tj. výsledkem je hodnota neznámo ($\frac{1}{2}$), je nutné dodatečně doplnit hodnoty vstupních parametrů a opakovat vyhodnocení.

- V případě, že je kritérium *strict-out* vyhodnoceno jako nepravdivé (hodnota 0), vyhodnotí se kritérium *strict-in*. V případě, že je kritérium *strict-in* pravdivé (hodnota 1), bude se pokračovat touto větví a další kritéria se již vyhodnocovat nebudou.
- U kritéria *strict-in* je opět požadován jednoznačný výsledek. Pokud je výsledná hodnota neznámo ($\frac{1}{2}$), je nutné dodatečně doplnit hodnoty vstupních parametrů a opakovat vyhodnocení.
- V případě, že kritérium *strict-in* příslušné větve je vyhodnoceno jako nepravdivé (hodnota 0), pokračuje se vyhodnocováním striktních kritérií *(strict-out* a *strict-in*) další následné větve v pořadí.

Fáze vyhodnocení doporučujících kritérií:

- Pokud není v předchozí fázi vyhodnoceno žádné kritérium *strict-in* jako pravdivé, budou se vyhodnocovat kritéria *rule-in* a *rule-out* u všech následných větví.
- Jestliže je vyhodnoceno kritérium *rule-in* jako pravdivé, je doporučeno pokračovat příslušnou větví.
- Je-li jako pravdivé vyhodnoceno kritérium *rule-out*, není doporučeno (ale ne zakázáno) příslušnou větví pokračovat.
- Kritéria *rule-in* a *rule-out* se vyhodnocují na základě známých hodnot vstupních parametrů. Pokud je výsledkem některé kritéria hodnota neznámo ($\frac{1}{2}$), není vyžadováno doplnění chybějících hodnot parametrů.
- Na základě rozdělení následných větví na doporučené a nedoporučené je vyžadován ruční výběr následné větve uživatelem. Nevylučuje se stav, že větev je jak doporučena (na základě nějakých parametrů) tak nedoporučena (na základě jiných parametrů).

Rozhodovací kritéria mohou být závislá na několika různých parametrech. Pro korektní rozhodnutí v každém rozhodovacím kroku je nutné, aby platila následující pravidla, a to pro všechny možné kombinace hodnot vstupních parametrů:

- Nesmí nastat situace, kdy jsou všechna kritéria *strict-out* (na všech následných větvích) pravdivá. Musí tedy vždy existovat alespoň jedna následná větev, která není zakázána.
- Ve všech povolených větvích (kde kritérium *strict-out* je nepravdivé) smí být nejvýše jedno kritérium *strict-in* pravdivé.

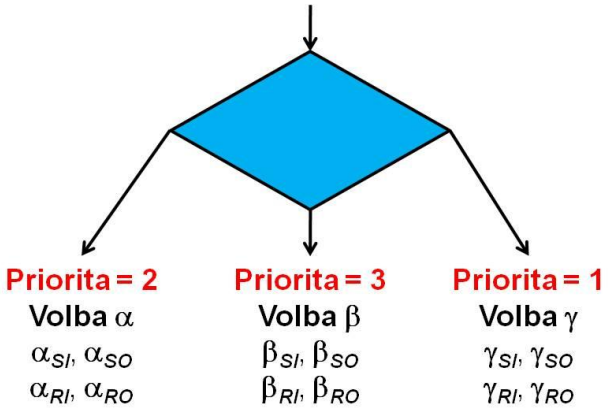
Ověření korektnosti modelu není triviální záležitostí. Při použití *n* vstupních parametrů pro příslušný rozhodovací krok je třeba zkoumat 2^n kombinací hodnot vstupních parametrů. Obecně exponenciální složitost lze snížit např. vhodným použitím rezolučního pravidla (viz kapitola 2.7.2.1).

4.2.6 Priorita rozhodovacích větví

Při důsledném uplatňování předchozích pravidel může nastat situace, ve které bude po uživateli vyžadováno upřesnění hodnot vstupních parametrů i v případě, kdy to pro další průchod modelem není nezbytně nutné. Upřesnění vstupních parametrů může však znamenat provedení nějakého, třeba i ekonomicky náročného, měření nebo speciálního úkonu.

Počet potřebných upřesnění (dodatečných doplnění chybějících parametrů) je závislé na pořadí vyhodnocování jednotlivých větví. Z uvedených důvodů je výhodné stanovit pořadí jejich vyhodnocování, tj. určit prioritu. Priorita je přiřazena jednotlivým větvím rozhodování při návrhu modelu po expertní analýze modelovaného problému (viz obrázek 13).

V uvedeném příkladu se tedy nejprve bude vyhodnocovat větev γ (priorita = 1), poté větev α (priorita = 2) a nakonec větev β (priorita = 3). Jestliže bude u větve γ splněna podmínka γ_{SI}, nebudou se větve α a β vůbec vyhodnocovat.



obrázek 13 Priorita větví v rozhodovacím kroku

Uvažujme následující situaci:

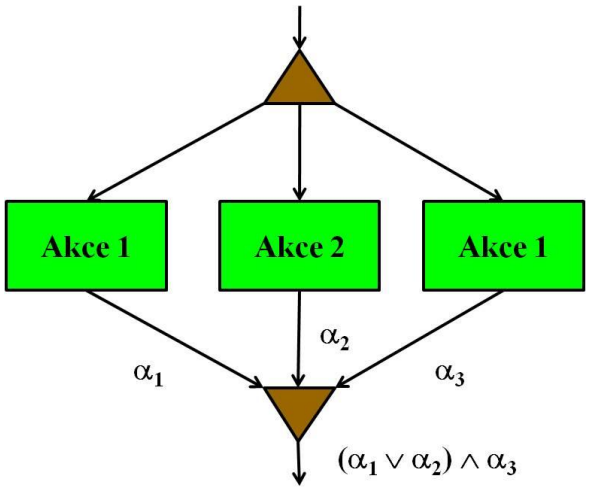
- Pacient má velmi vysokou úroveň rizika, jestliže je u něj diagnostikována cukrovka (na jiných parametrech již nezáleží). Diagnózu cukrovky bude představovat parametr P_1 a bude se o ní rozhodovat ve *strict-in* kritériu větve γ.
- Pacient má vysokou úroveň rizika, pokud má alespoň dvě ze tří možných orgánových poškození. Každé orgánové poškození bude představováno parametry P_2, P_3 a P_4. Rozhodnutí o uvedené míře rizika bude obsahovat *strict-in* kritérium větve α.
- Pacient má střední úroveň rizika, pokud má alespoň jedno orgánové poškození, tj. je splněn alespoň jeden z parametrů P_2, P_3 nebo P_4. Rozhodnutí o uvedené míře rizika bude obsahovat *strict-in* kritérium větve β.

Při stanových prioritách jednotlivých větví (jak uvádí obrázek 13) se nejprve správně vyhodnotí přítomnost (případně vyloučí přítomnost) cukrovky ve větvi γ, následně se stanoví, zda má pacient alespoň dvě orgánová poškození (větev α) a nakonec (pokud nenastane některá z předchozích situací) se vyhodnotí větev β, tj. zda má pacient alespoň jedno orgánové poškození.

V případě, že by nebyla stanovena priorita jednotlivých větví, bylo by pro rozhodnutí zřejmě nutné striktně zjistit hodnoty všech parametrů P_1 až P_4.

4.2.7 Synchronizační podmínky

Při modelování paralelně probíhajících větví pomocí prvků větvení a synchronizace je nutné upřesnit, které z větví musí uživatel určitě projít a které jsou pouze volitelné. Uvedené případy lze realizovat pomocí synchronizační podmínky v synchronizačním kroku (viz obrázek 14). Synchronizační podmínku je možné vyjádřit v disjunktivním (konjunktivním) normálním tvaru, kde vstupními proměnnými jsou plovoucí (onfly) parametry reprezentující průchod jednotlivými větvemi. Hodnota onfly parametru je nastavena pomocí skryté operace (viz kapitola 4.3.1) v posledním kroku příslušné větve.



obrázek 14 Synchronizační podmínka

Ohodnocení parametrů je pak následující:

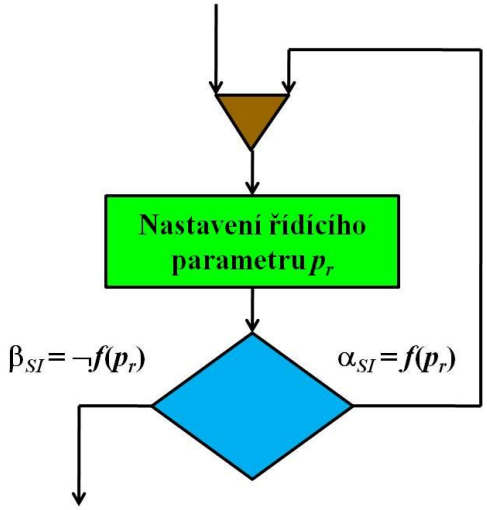
- **hodnota 1** – touto větví uživatel prošel
- **hodnota 0** – touto větví uživatel neprošel

V ukázkové situaci se bude pokračovat v průchodu grafem (na vrchol následujícím za synchronizačním), bude-li splněna synchronizační podmínka $(\alpha_1 \vee \alpha_2) \wedge \alpha_3$, tj. jestliže uživatel povinně projde větví α_3 a zároveň volitelně alespoň jednou z větví α_1 nebo α_2.

Pokud by v synchronizačním vrcholu nebyla definována žádná synchronizační podmínka, implicitně se předpokládá podmínka ve tvaru $\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n$.

4.2.8 Opakovaná činnost – cyklus

Pro modelování opakovaných činností (cyklu) se použije konstrukce z vrcholů synchronizace, akce a rozhodnutí (viz obrázek 15).



obrázek 15 Opakovaná činnost v GLIKREM

Opakování činností (cyklus) je závislé na hodnotě řídícího parametru p_r resp. na vyhodnocení logické formule $f(p_r)$. Jestliže bude formule $f(p_r)$ nabývat hodnoty 1 (pravda), bude pravdivé i rozhodovací kritérium $\alpha_{SI} = f(p_r)$ a provede se další krok cyklu. Jestliže bude formule $f(p_r)$ nabývat hodnoty 0 (nepravda), bude pravdivé rozhodovací kritérium $\beta_{SI} = \neg f(p_r)$ a cyklus se ukončí (bude se pokračovat větví β).

4.2.9 Čas v modelu

V GLIKREM je definován aktuální čas modelu τ a časový posun Δτ, kde čas τ představuje časové transakce, tj. celočíselné (*s*) násobky časového posunu Δτ od počátečního času τ_0:

$$\tau = \tau_0 + s \cdot \Delta\tau$$

Dále je definován paramodel, tj. model parametrů p_i^t, ve kterém jsou uloženy hodnoty parametrů p_i v čase *t*. Předpokládejme, že časy uložení parametrů jsou vyšší než počáteční čas modelu, tj.:

$$\forall t: t \geq \tau_0$$

Pro každý parametr p_i je definována časová tolerance ε_i jako násobek Δτ, která představuje platnost parametru p_i v časovém intervalu $t \pm \varepsilon_i \cdot \Delta\tau$.

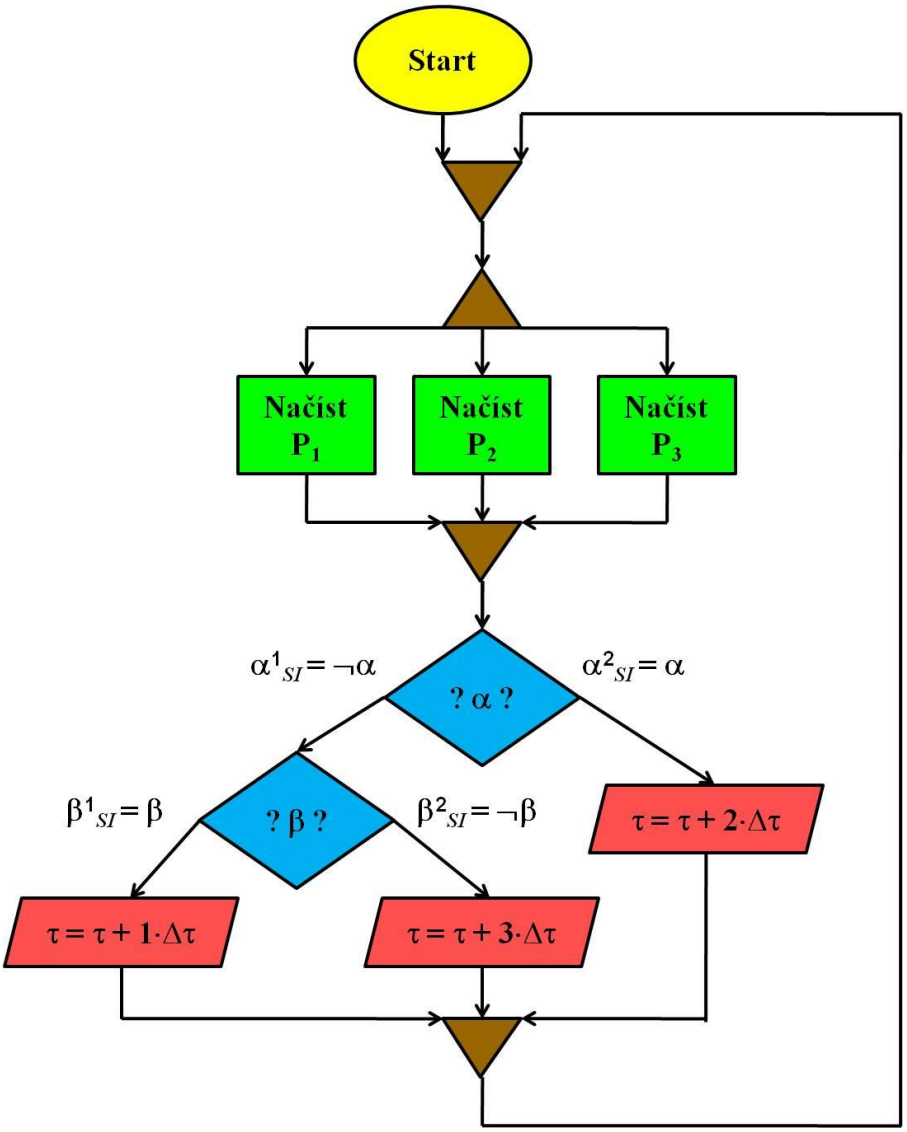
Při vyhodnocování kritérií *strict-in*, *strict-out*, *rule-in* a *rule-out* v rozhodovacích krocích modelu GLIKREM se použijí pouze parametry z paramodelu, které jsou platné (jsou k dispozici) v čase modelu τ. Parametry p_i nemusí však být k dispozici přesně ve stejných časových úsecích (transakcích) odpovídajících času τ, pro vyhodnocování logických formulí se proto použijí hodnoty těch parametrů p_i^t (resp. hodnoty parametrů p_i v čase *t*), pro které platí:

$$\boldsymbol{p_i^t: t = \max\{t_{ij}\}}$$

$$\boldsymbol{t_{ij}: (\tau - \varepsilon_i \cdot \Delta\tau) \leq t_{ij} \leq (\tau + \varepsilon_i \cdot \Delta\tau)},$$

kde τ je čas v modelu, ε_i je časová tolerance (relevantnost) *i*-tého parametru a t_{ij} jsou všechny časové úseky, které jsou (pro *i*-tý parametr) platné v čase modelu τ.

Použití času v modelu GLIKREM lze ukázat na následující modelové situaci (viz obrázek 16). Pro vyhodnocení rozhodovacích kroků α a β jsou použita (pro zjednodušení) pouze kritéria *strict-in*, která závisí na hodnotách všech parametrů P_1, P_2 a P_3 načtených z paramodelu.



obrázek 16 Modelování času v GLIKREM

Na začátku GLIKREM modelové situace jsou nastaveny počáteční časové parametry:

- počáteční čas modelu $\boldsymbol{\tau_0 = 0}$
- čas modelu $\boldsymbol{\tau = \tau_0 + 1 \cdot \Delta\tau}$
- časový posun $\boldsymbol{\Delta\tau = 1}$ (časová jednotka)

Je definován model parametrů (paramodel), který obsahuje parametry P_1, P_2 a P_3 a jejich tolerance ε_1, ε_2 a ε_3. Dostupnost hodnot parametrů v časech t_{ij} uvádí tabulka 1.

tabulka 1 Dostupnost hodnot parametrů v časových úsecích

Parametr P_1	**Časová tolerance $\varepsilon_1 = 0{,}5$**
Čas záznamu hodnoty parametru – označení / skutečný čas	
t_{11}	1,1
t_{12}	2
t_{13}	3,2
t_{14}	4,1
t_{15}	5,3
Parametr P_2	**Časová tolerance $\varepsilon_2 = 1{,}75$**
Čas záznamu hodnoty parametru – označení / skutečný čas	
t_{21}	2,1
t_{22}	3,5
Parametr P_3	**Časová tolerance $\varepsilon_3 = 0{,}75$**
Čas záznamu hodnoty parametru – označení / skutečný čas	
t_{31}	1,6
t_{32}	3,9
t_{33}	5,2

V rozhodovacích krocích je pro výběr následujícího kroku použito vyhodnocování pouze striktních rozhodovacích kritérií *strict-in*. Kritéria *strict-out* předpokládejme v podobě negace příslušného kritéria *strict-in*. Doporučovací kritéria *rule-in* a *rule-out* nejsou v modelové situaci použity vůbec. Dalším předpokladem je vyhodnocování rozhodovacích kritérií ve dvou-hodnotové logice.

tabulka 2 Použité hodnoty parametrů v časových úsecích

Scénář		**Čas modelu τ**	**Časový interval platnosti a platné časy t_{1j} parametru P_1**	**Časový interval platnosti a platné časy t_{2j} parametru P_2**	**Časový interval platnosti a platné časy t_{3j} parametru P_3**
	Start	$s = 1$ τ_1	$\langle 0.5 \cdot \Delta\tau, 1.5 \cdot \Delta\tau \rangle$ t_{11}	$\langle -0.75 \cdot \Delta\tau, 2.75 \cdot \Delta\tau \rangle$ t_{21}	$\langle 0.25 \cdot \Delta\tau, 1.75 \cdot \Delta\tau \rangle$ t_{31}
1	α	$s = 3$ τ_3	$\langle 2.5 \cdot \Delta\tau, 3.5 \cdot \Delta\tau \rangle$ t_{13}	$\langle 1.25 \cdot \Delta\tau, 4.75 \cdot \Delta\tau \rangle$ t_{21}, t_{22}	$\langle 2.25 \cdot \Delta\tau, 3.75 \cdot \Delta\tau \rangle$???
2	$\neg\alpha \wedge \beta$	$s = 2$ τ_2	$\langle 1.5 \cdot \Delta\tau, 2.5 \cdot \Delta\tau \rangle$ t_{12}	$\langle 0.25 \cdot \Delta\tau, 3.75 \cdot \Delta\tau \rangle$ t_{21}, t_{22}	$\langle 1.25 \cdot \Delta\tau, 2.75 \cdot \Delta\tau \rangle$???
3	$\neg\alpha \wedge \neg\beta$	$s = 4$ τ_4	$\langle 3.5 \cdot \Delta\tau, 4.5 \cdot \Delta\tau \rangle$ t_{14}	$\langle 2.25 \cdot \Delta\tau, 5.75 \cdot \Delta\tau \rangle$ t_{22}	$\langle 3.25 \cdot \Delta\tau, 4.75 \cdot \Delta\tau \rangle$ t_{32}

Průchod modelem (resp. první krok opakované činnosti) je, na základě splnění či nesplnění rozhodovacích kritérií, možný podle jednoho z následujících scénářů:

- *scénář* 1: v prvním rozhodovacím kroku je splněna (pravdivá) podmínka α
- *scénář* 2: v prvním rozhodovacím kroku není splněna (nepravdivá) podmínka α a zároveň v druhém rozhodovacím kroku je splněna podmínka β, tj. platí $\neg\alpha \wedge \beta$
- *scénář* 3: v prvním rozhodovacím kroku není splněna (nepravdivá) podmínka α a zároveň v druhém rozhodovacím kroku není splněna podmínka β, tj. platí $\neg\alpha \wedge \neg\beta$

Na základě použitého scénáře průchodu prvním krokem opakované činnosti v modelu je vykonán příslušný posun času modelu (τ) a v následujícím kroku opakování jsou načteny hodnoty parametrů P_1, P_2 a P_3 platných v novém čase τ. Platné (použité) hodnoty parametrů v časových úsecích uvádí tabulka 2.

Z výsledné tabulky je patrné, že pro některé hodnoty času modelu (τ) je možné použít hodnoty parametru ve více časových úsecích. V takovém případě se použije „novější“ hodnota (nejvyšší t_j).

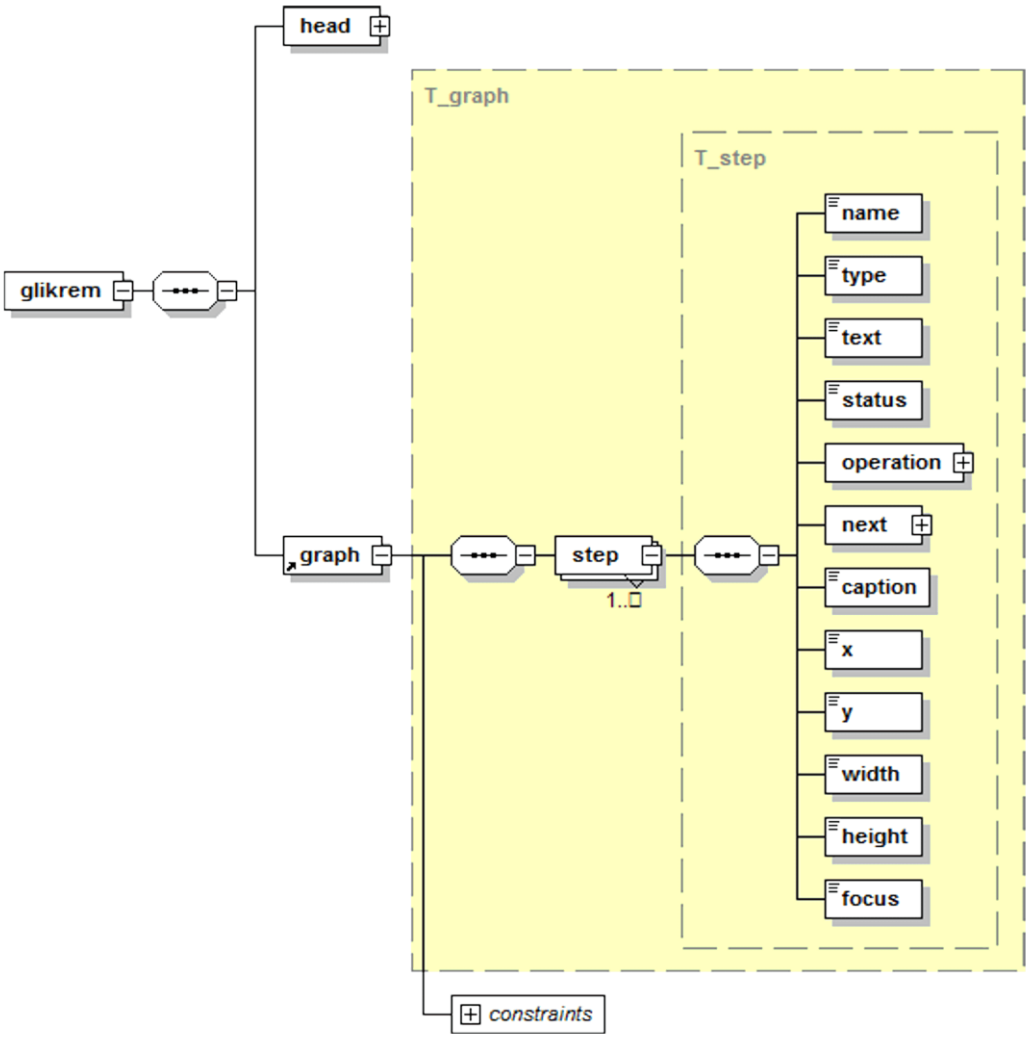
V některých případech (označeno **???**) není však pro daný čas modelu (τ) dostupná hodnota parametru v žádném časovém úseku. V takovém případě je nutné, pro další pokračování průchodu modelem, chybějící hodnotu doplnit uživatelem.

Pokud uvedený model měl reprezentovat nějakou reálnou situaci, např. časový posun τ by znamenal naplánování dalšího vyšetření pacienta, předpokládá se, že by pacient navržený termín skutečně dodržel (max. s malým rozpětím vzhledem k $\Delta\tau$). V opačném případě by se muselo vyhodnocování modelu řešit jako by se jednalo o první průchod, tj. nastavit počáteční čas modelu τ_0 na skutečný čas příchodu pacienta.

4.3 Implementace GLIKREM

4.3.1 Implementace modelu

Pro reprezentaci grafického modelu bylo navrženo vlastní XML schéma (viz kapitola 8.1). Celý model se skládá z jednotlivých kroků reprezentujících příslušné vrcholy grafického modelu. Kromě atributů vyplývajících přímo z modelu obsahuje navržená syntaxe XML i atributy vhodné pro další zpracování, například pro grafické zobrazení nebo řízení průchodu modelem a načtení či uložení parametrů (viz obrázek 17).



obrázek 17 XML schéma reprezentace grafického modelu

Význam jednotlivých elementů je následující:

<glikrem> - kořenový element celého modelu

<head> - definice hlavičky modelu – viz kapitola 5.4

<graph> - kořenový element grafického modelu, skládá se z kroků <**step**>

<step> - parametry jednoho kroku, obsahuje elementy:

<name> - jednoznačná identifikace (název) kroku

<type> - typ vrcholu (kroku), je možný pouze výběr z typů:

- *action* – vrchol typu „Akce“
- *subgraph* – vrchol typu „Podgraf“
- *case* – vrchol typu „Rozhodnutí“
- *branch* – vrchol typu „Větvení“
- *synchronization* – vrchol typu „Synchronizace“
- *state* – vrchol typu „Stav“
- *time shift* – vrchol typu „Časový posun“

<caption> - text v záhlaví grafického symbolu kroku

<text> - vlastní text popisující krok

<x> - *x*-ová (horizontální) souřadnice grafického symbolu

<y> - *y*-ová (vertikální) souřadnice grafického symbolu

<width> - šířka (v pixelech) grafického symbolu

<height> - výška (v pixelech) grafického symbolu

<focus> - zvýraznění kroku pro zobrazení cesty v grafu, možné hodnoty jsou:

- *not* = nezvýrazněn
- *auto* = zvýrazněn (automaticky)
- *user* = zvýrazněn (uživatelská simulace)

<status> - postavení kroku v rámci modelu, možné hodnoty jsou:

- *start* = počáteční krok
- *end* = koncový krok
- *in* = vnitřní krok (všechny ostatní)

<operation> - seznam operací probíhajících „na pozadí“ kroku (viz obrázek 18)

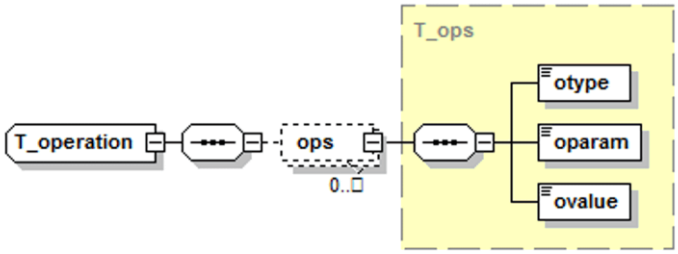
<ops> - ohraničení jedné skryté operace

<otype> - typ operace, je možný výběr z:

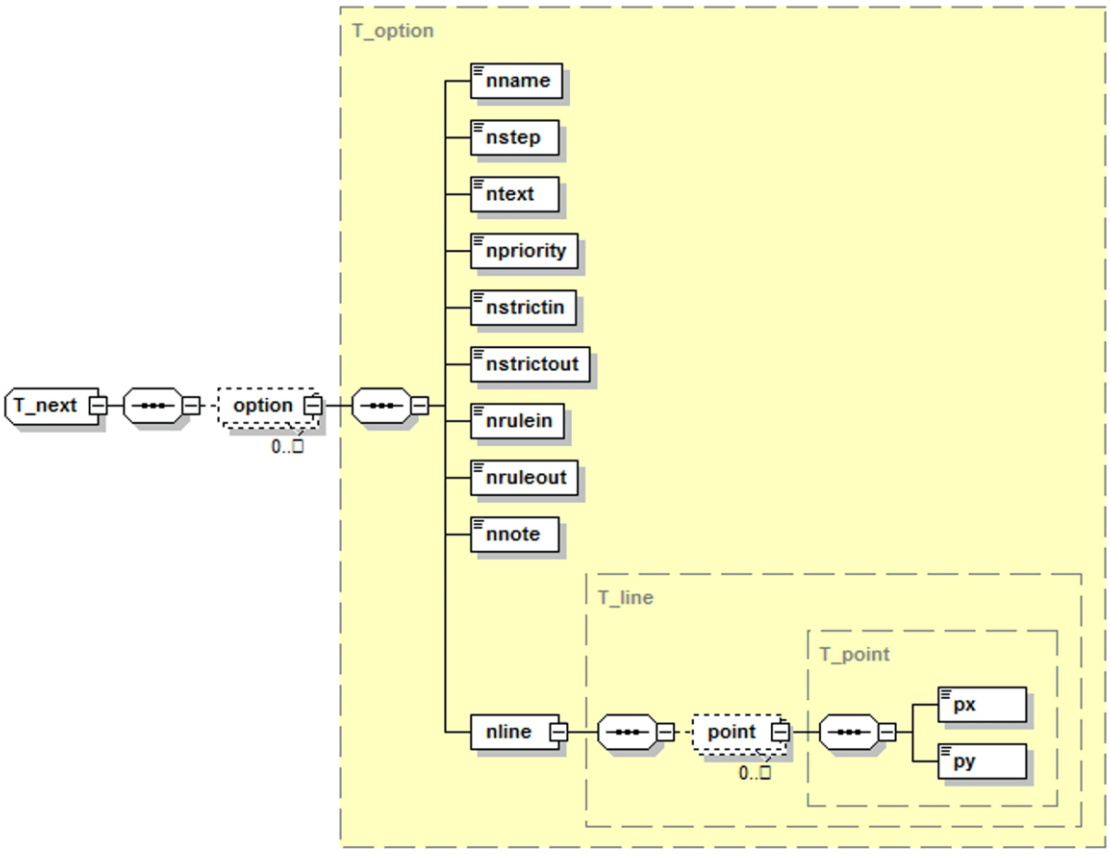
- *insert* = vložení (zadání) hodnot parametrů
- *get* = načtení hodnot parametrů (zjištění hodnoty)
- *put* = nastavení (uložení) hodnoty parametru
- *open* = otevření souboru (např. podgrafu)
- *shift* = časový posun o definovaný násobek jednotkového časového úseku

<oparam> - název parametru nebo název souboru

<ovalue> - pro operaci *put* obsahuje hodnotu, na kterou má být parametr nastaven, pro operaci *shift* pak násobek jednotkového časového úseku



obrázek 18 XML schéma operací na pozadí kroku



obrázek 19 XML schéma následných větví

<next> - seznam následných větví (voleb) kroku (viz obrázek 19)

<option> - ohraničení jedné volby

<nname> - název, identifikace následné větve

<nstep> - identifikátor (název) kroku, kterým se má pokračovat

<ncaption> - text zobrazený u příslušné volby

<npriority> - priorita vyhodnocení větve udaná číslem, defaultně hodnota 1

<nstrictin> - *strict-in* rozhodovací kritérium

<nstrictout> - *strict-out* rozhodovací kritérium

<nrulein> - *rule-in* rozhodovací kritérium

<nruleout> - *rule-out* rozhodovací kritérium

<nnote> - poznámka popisující příslušnou větev

<nline> - popis spojnice vedoucí z tohoto kroku do následného

<point> - sekvence bodů spojnice

<px> - x-ová souřadnice bodu

<py> - y-ová souřadnice bodu

4.3.2 Implementace parametrů

Parametry podstatné pro modelování oborových doporučení pomocí GLIKREM nemusí a často ani nebývají uloženy způsobem vhodným pro přímé použití modelem. Výhodnějším řešením je navržené rozhraní mezi GLIKREM a konkrétními uloženými daty ve formě paramodelu reprezentovaném v XML. Skutečné hodnoty se pak získají vyhodnocením dotazu do konkrétní databáze nebo informačního systému.

Pro reprezentaci paramodelu (modelu parametrů) je, stejně jako u grafického modelu, využito jazyku XML. Celý model se skládá z elementů XML reprezentující příslušný parametr (viz obrázek 20).

Význam jednotlivých XML elementů paramodelu je následující:

<paramodel> – kořenový element celého paramodelu

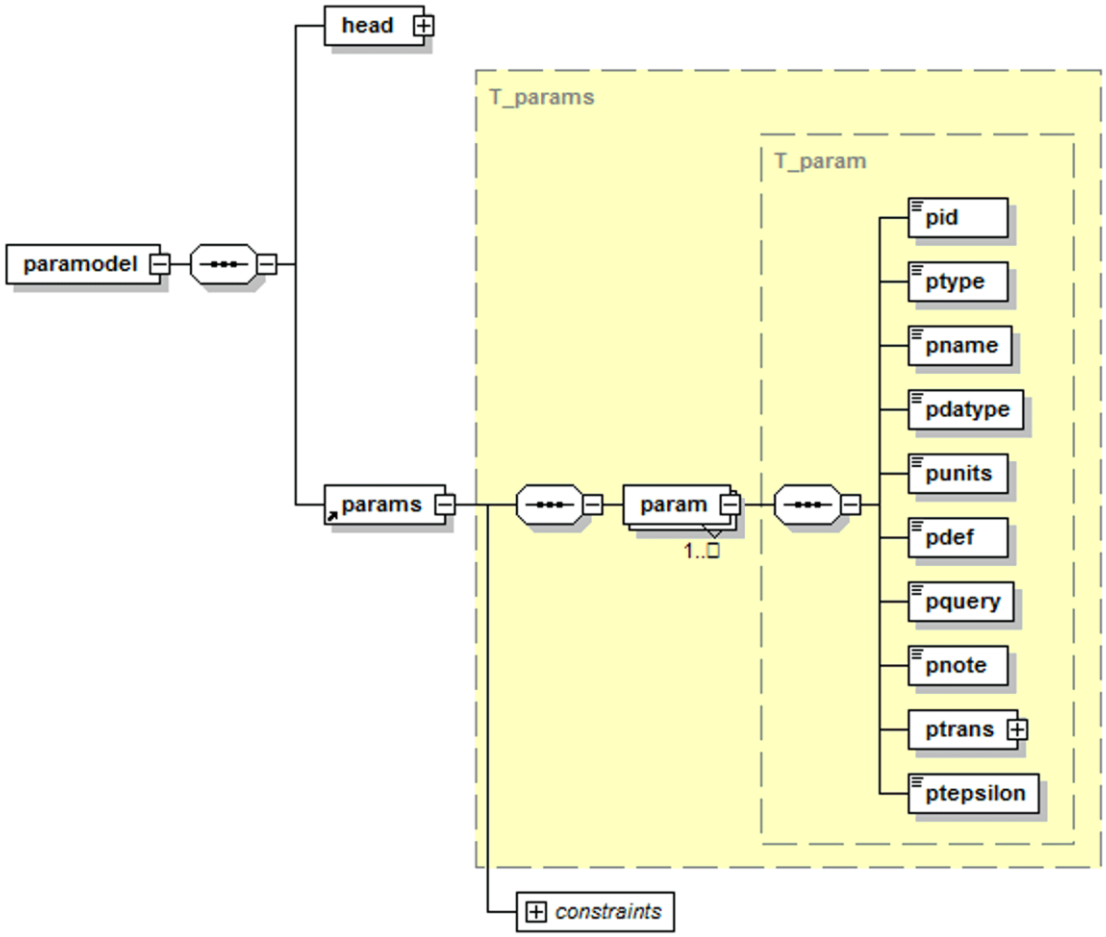
<params> – kořenový element parametrů, obsahuje elementy <**param**>

<param> - popis jednoho parametru

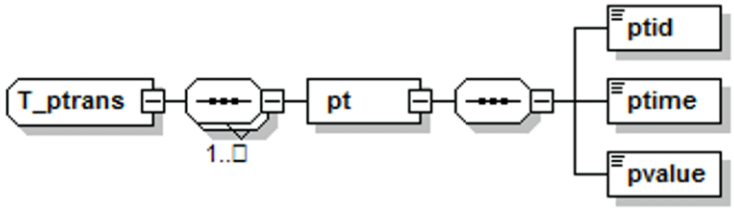
<pid> - jedinečná identifikace (název) parametru

<ptype> - typ parametru, možné hodnoty jsou:

- *base* – základní parametr
- *derived* – odvozený parametr
- *onfly* – plovoucí parametr
- **<pname>** - text úplného názvu parametru
- **<pdatype>** - datový typ parametru odpovídající některému z XML datových typů
- **<pvalue>** - vlastní hodnota parametru
- **<punits>** - jednotky parametru
- **<pdef>** - definice operace nad základními parametry pro odvozené parametry
- **<pquery**>- definice dotazu do databáze na hodnotu parametru (např. SQL)
- **<pnote>** - text poznámky k parametru



obrázek 20 XML schéma paramodelu



obrázek 21 XML schéma časové transakce parametru

Časové atributy jednotlivých parametrů jsou definovány jako sekvence časových transakcí (viz obrázek 21).

<ptrans> - kořenový element časových transakcí, obsahuje elementy <**pt**>

<pt> - element popisující hodnotu parametru v jedné transakci

<ptid> - identifikátor (číslo) transakce

<ptime> - čas platnosti hodnoty parametru

<pvalue> - hodnota parametru

<ptepsilon> - tolerance (časový interval) platnosti hodnoty parametru

Základní identifikace modelu parametrů (paramodelu) a globální časové atributy jsou uvedeny v hlavičce modelu (viz obrázek 22).

<head> - hlavička paramodelu

<gname> - název modelu parametrů (paramodelu)

<author> - autor(ři) paramodelu

<date> - datum poslední aktualizace paramodelu

<status> - označení platnosti modelu, možné hodnoty jsou

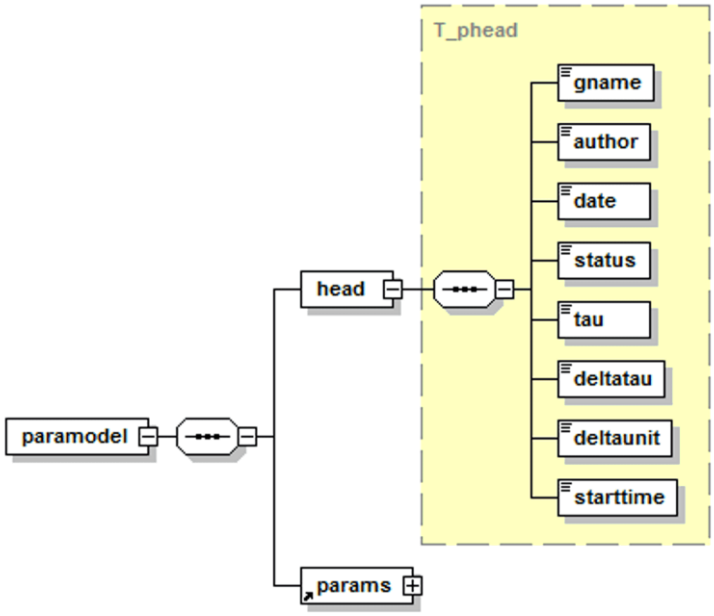
- *valid* – platný model
- *draft* – jen návrh modelu
- *expired* – model, jehož platnost vypršela

<tau> - aktuální čas modelu (hodnota τ)

<deltatau> - jednotkový posun aktuálního času použitý v časovém kroku ($\Delta\tau$)

<deltaunit> - jednotka časového posunu vyjádřená v počtu sekund

<starttime> - počáteční čas modelu (τ_0)



obrázek 22 XML schéma hlavičky paramodelu

4.3.3 Rozhodovací a synchronizační podmínky

Zápis rozhodovacích podmínek (*strict-in*, *strict-out*, *rule-in*, *rule-out*) dodržuje syntaxi jazyka **XPath**[10]. Rozhodovací podmínky pracují s parametry uloženými v odpovídajícím paramodelu (modelu parametrů).

Uveďme příklad definice podmínky testující pravdivost obou parametrů PRIZNAKY a NALEZY:

/params/param[pid="PRIZNAKY"]/pvalue **and** params/param[pid="NALEZY"]/pvalue

Zápis synchronizačních podmínek je obdobný (opět v jazyku **XPath**), předpokládá ale definici onfly parametrů definujících průchod příslušným krokem.

[10] **XPath** slouží k reprezentaci XML dokumentu v podobě stromu uzlů. Uzlem může být element, atribut nebo text. K popisu se používá podobná syntaxe jako k popisu cesty v souborovém systému. Navíc lze použít standardní predikáty a výrazy. Bližší specifikaci XPath lze nalézt na http://www.w3.org/TR/xpath.

Příklad definice synchronizační podmínky, kdy je vyžadován průchod větví S3 nebo větví S4:

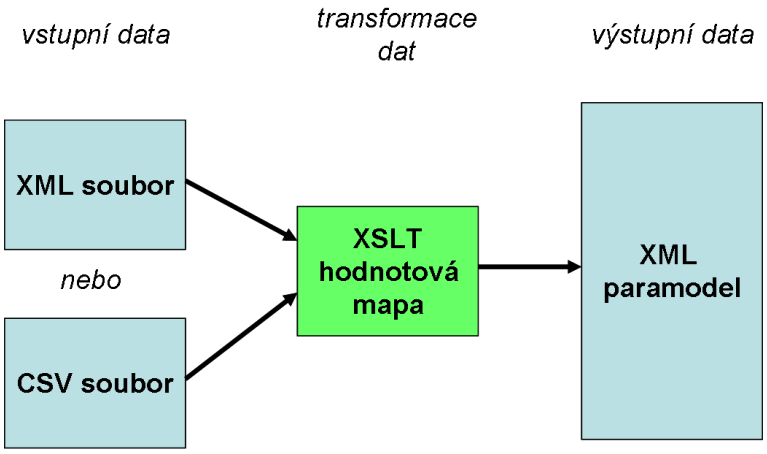
/params/param[pid="DG_S3"]/pvalue **or** /params/param[pid="DG_S4"]/pvalue

Definice odvozených parametrů (element <*pdef*> v paramodelu) využívá opět zápis v jazyce XPath.

4.3.4 Transformace vstupních dat

Reálná data mohou být uložena v informačních systémech a databázích různých typů. Není však velký problém data z těchto systémů exportovat do formátů XML nebo CSV (*Comma-Separated Values*). Před použitím těchto dat v GLIKREM je třeba je vhodným způsobem transformovat do paramodelu (v XML). Zjednodušený proces transformace je znázorňuje obrázek 23.

Pro každý parametr v paramodelu je třeba, s ohledem na uložení reálných vstupních hodnot v datovém souboru (XML nebo CSV), definovat specifickou hodnotovou mapu (value-map). Definice hodnotové mapy je ve formě XSLT (*eXtensible Stylesheet Language Transformations*) souboru.



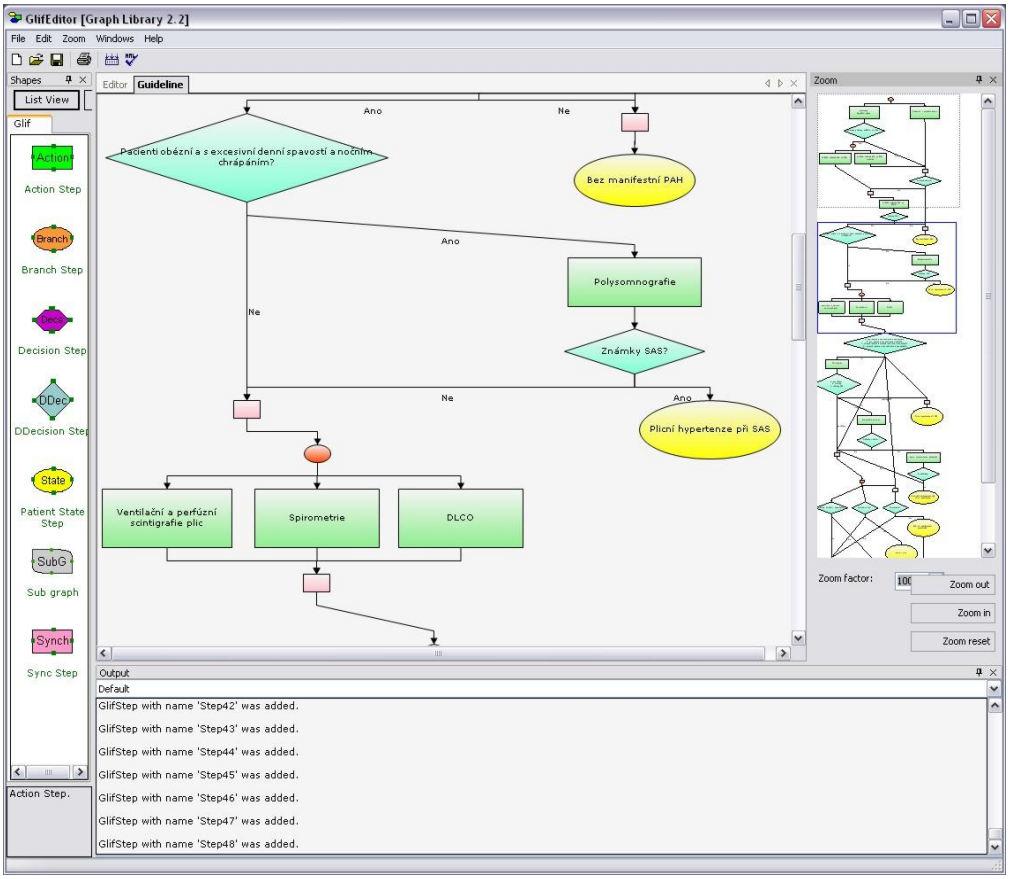
obrázek 23 Proces transformace dat do paramodelu

5 Praktické aplikace GLIKREM

Navržený model reprezentace znalostí v oborových doporučeních byl autorem disertační práce vyvíjen během několika let a stal se součástí několika praktických aplikací, na jejichž vývoji se autor také (částečně či zcela) podílel.

5.1 Editor GLIKREM

Pro vytvoření grafického modelu, nejlépe přímo autory textových lékařských doporučení, je výhodné použít programový nástroj s uživatelsky příjemným prostředím. GLIKREMeditor (viz obrázek 24) umožňuje vizuálním způsobem konstrukci grafického modelu procesní struktury lékařských doporučení a jeho uložení ve formátu XML.

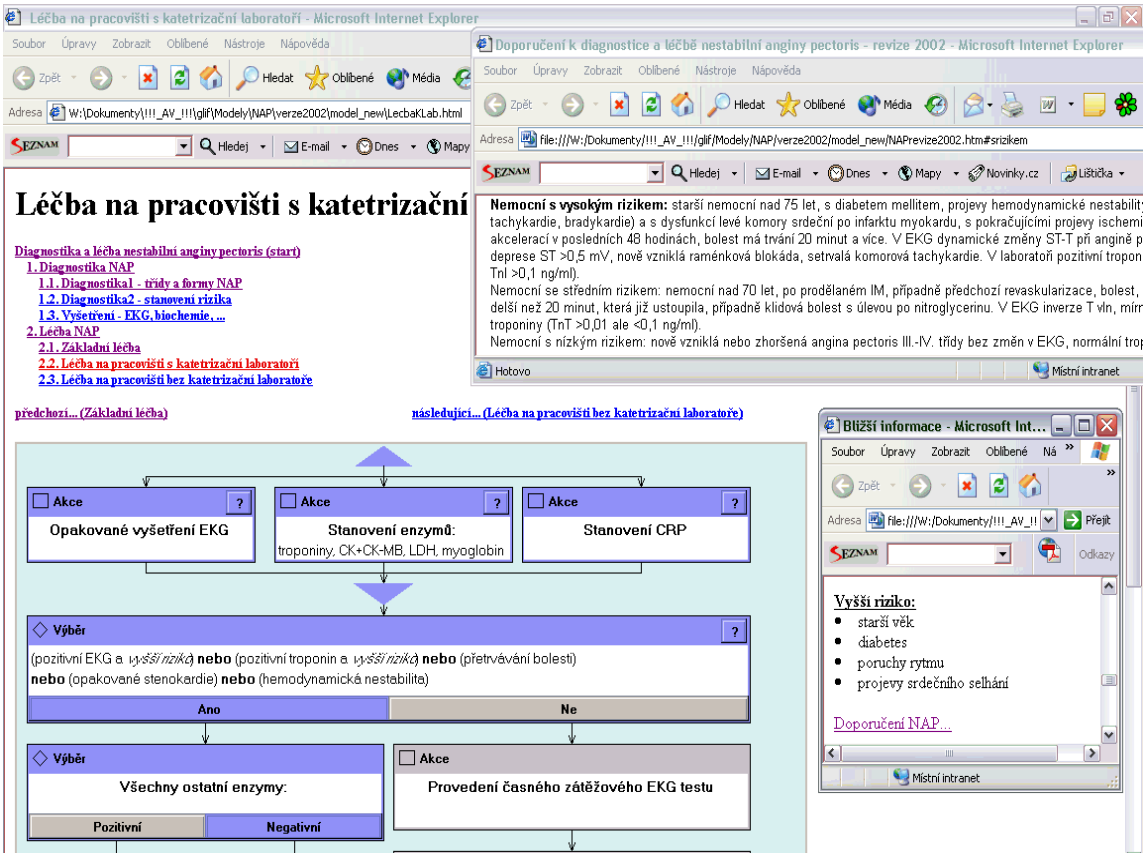


obrázek 24 GLIKREMeditor

Aplikace GLIKREMeditor je vytvořena pomocí volně šiřitelných knihoven Netron Graph Library a Netron Neon Library. Tyto velmi rozsáhlé knihovny vytvořil Francois M. Vanderseypen, jako součást projektu The Netron Project1[30]. Celý Netron Project je striktně objektově orientovaný a napsaný v jazyce C#, pomocí Microsoft Visual Studio .NET.

5.2 Obecný prohlížeč

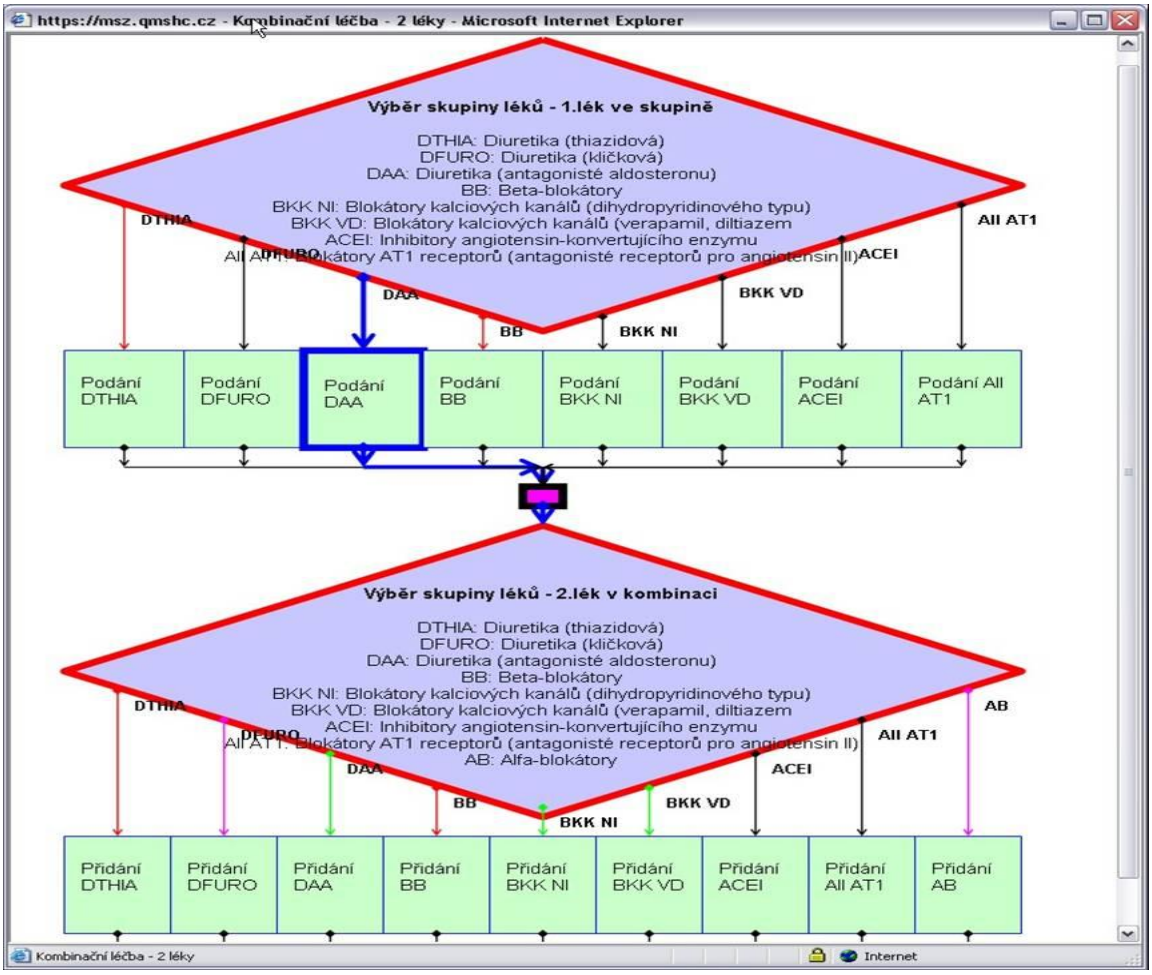
První aplikací obecného prohlížeče (GLIF-VIEW) je zobrazení modelu jako Java applet pomocí internetového prohlížeče. Aplikace umožňuje snadnou prezentaci na webu i offline a je velice vhodná pro studijní účely. Systém umožňuje interaktivní procházení modelem a zvýraznění uživatelem vybraných větví grafu (viz obrázek 25).



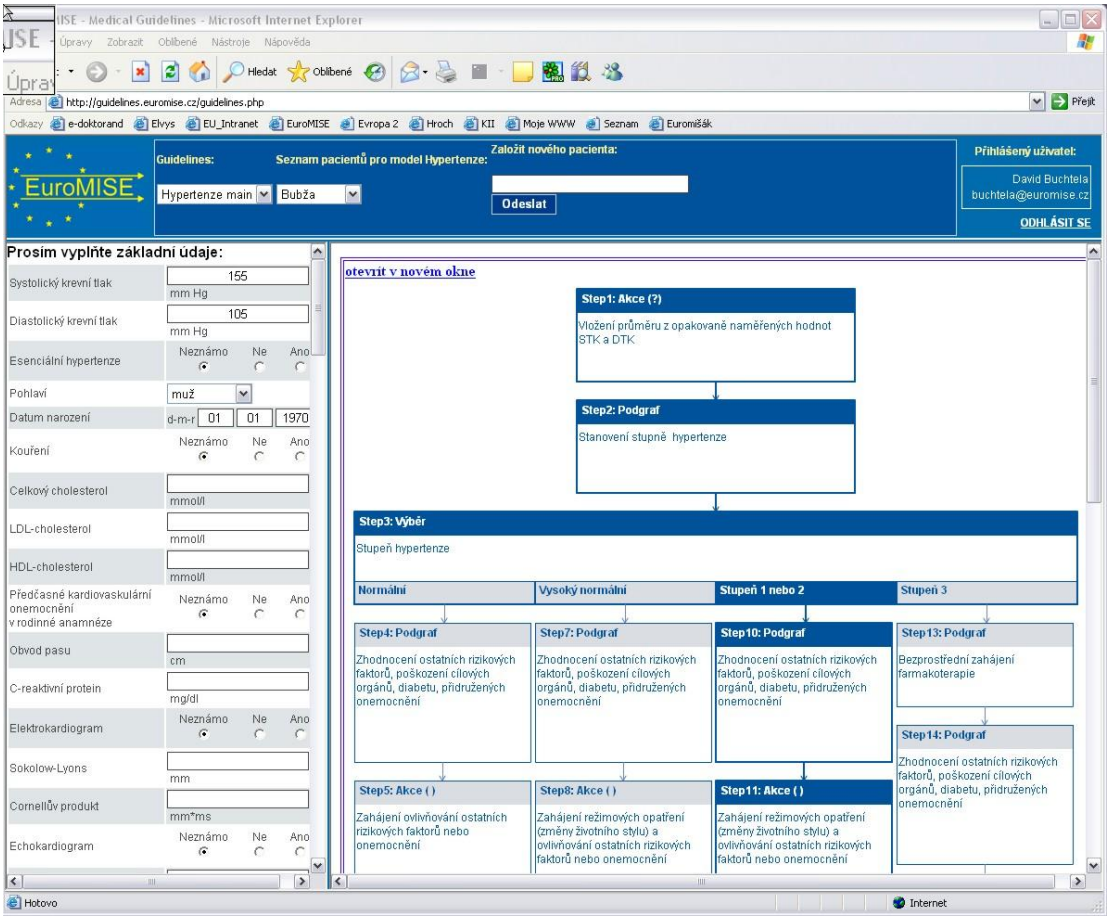
obrázek 25 Obecný prohlížeč GLIF-VIEW - model NAP 2002

5.3 Obecný prohlížeč řízený daty

Aplikace obecných prohlížečů řízených daty GL-ONLINE (viz obrázek 26) a PROCESSING GUIDELINES (viz obrázek 27) jsou určeny k zobrazování dat v textové i grafické podobě a analýzám dat. Základem je třívrstvá architektura webové aplikace. Data pacientů jsou uložena na databázovém serveru. Uživatel aplikaci provozuje pomocí klasického internetového prohlížeče a zprostředkování dotazů zajišťuje webový server. Aplikace umožňuje pouze autorizovaný přístup do aplikace, neboť se pracuje s reálnými daty pacientů. Pro lepší orientaci jsou při procházení modelem doporučení, na základě vyhodnocení rozhodovacích kritérií, barevně odlišeny doporučené (zeleně), nedoporučené (fialově) a zakázané (červeně) větve.



obrázek 26 Aplikace GL-ONLINE - model Hypertenze 2003



obrázek 27 Aplikace PROCESSING GUIDELINES – model Hypertenze 2003

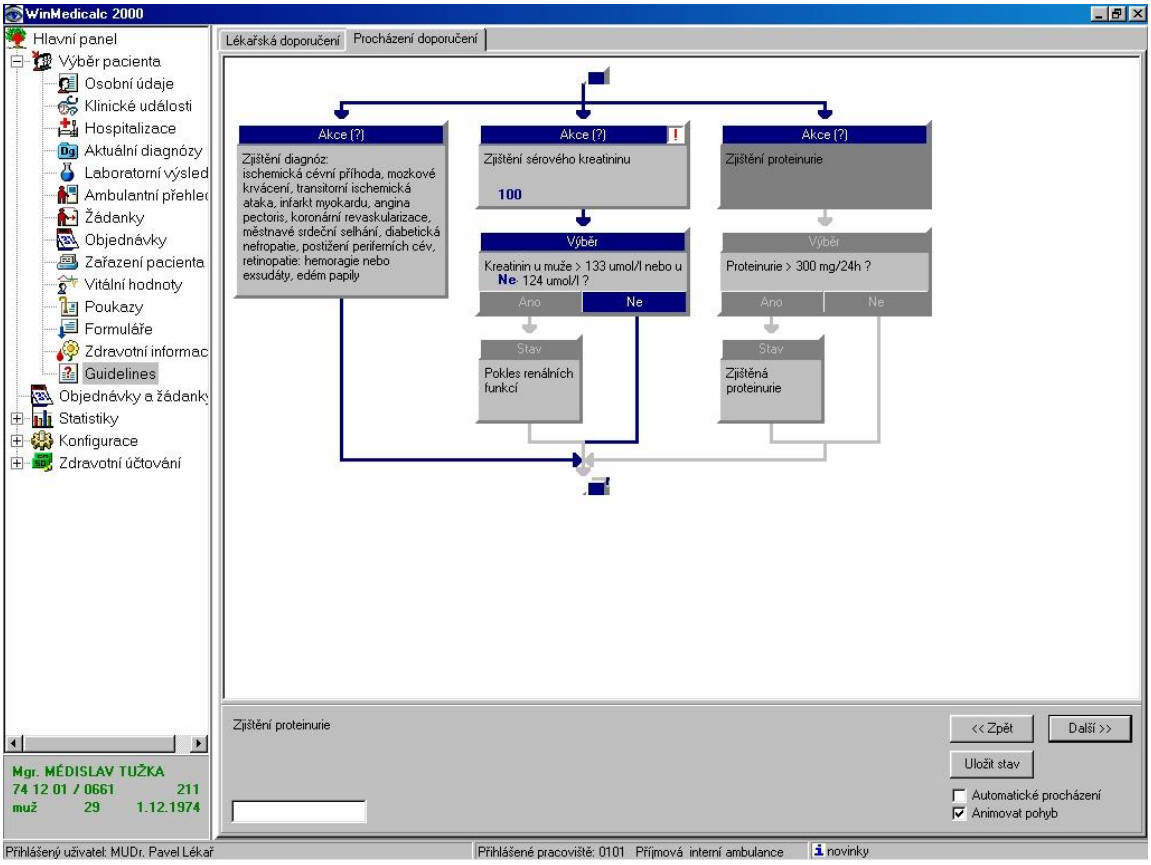
Aplikace GLIF-KIS (viz obrázek 28) umožňuje zabudování modelu do klinického informačního systému[11]. Toto řešení využívá přímého napojení na data o pacientech. Samozřejmostí je opět pouze autorizovaný přístup.

5.4 Systém reprezentace lékařských znalostí

Pro použití modelů doporučení v systému MEKRES (*MEdical Knowledge REpresentation System*) je znalostní model (GLIKREM) rozšířen o klíčové atributy.

[11] V uvedeném případě se jedná o zabudování modelu GLIKREM do klinického informačního systému WinMedicalc 2000 vyvinutého společností Medicalc software s.r.o.

Klíčové atributy se uplatní v algoritmu výběru relevantních formalizovaných znalostí účastníkům systému. Klíčové atributy jsou zakódovány opět v jazyce XML a uloženy v hlavičce (element head) společně s grafickým modelem (viz obrázek 29). Formulace klíčových atributů pro daný znalostní model je v souladu s termíny a pojmy znalostní ontologie příslušné předmětné oblasti, pro oblast medicíny např. s ontologií UMLS (viz kapitola 2.7.5).



obrázek 28 Aplikace GLIF-KIS – model Hypertenze 2003

Význam jednotlivých XML elementů hlavičky (elementu <**head**>) je následující:

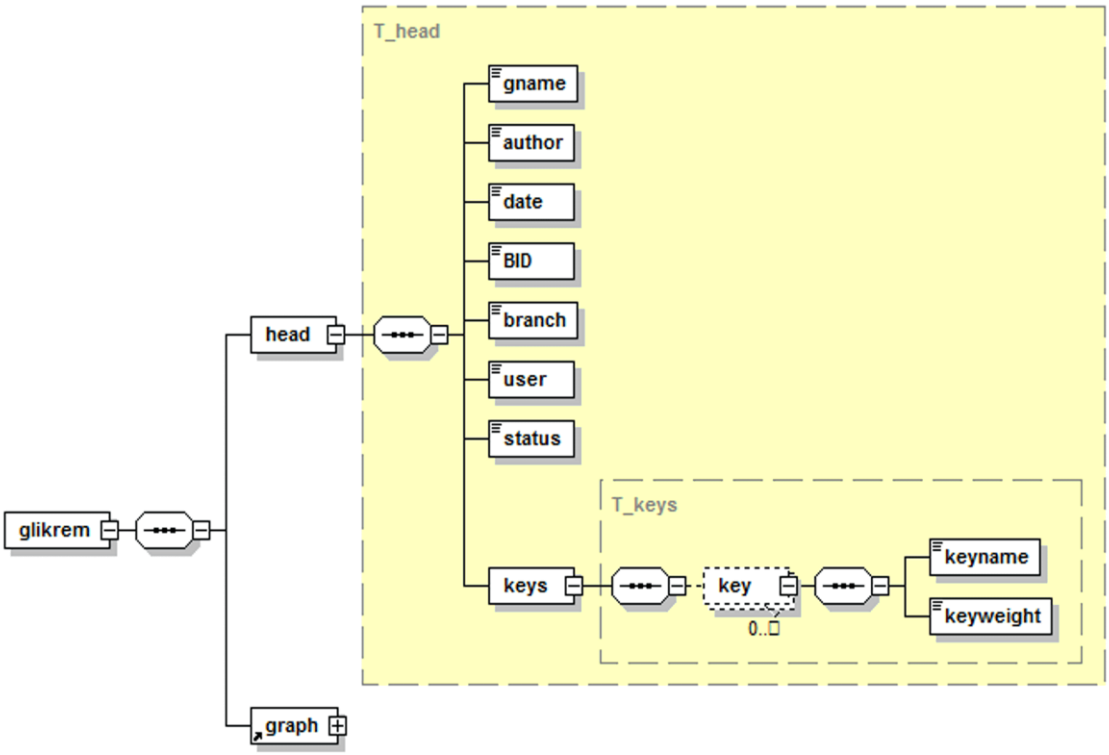
<gname> – název znalostního modelu

<author> – autor(ři) modelu

<date> – datum poslední aktualizace

<BID> – identifikátor oblasti – např. podle mezinárodního číselníku nemocí

<branch> – oblast (obor), které se model týká



obrázek 29 XML schéma klíčových atributů (hlavička modelu)

<user> – označení účastníka systému, kterému je model primárně určen:

- *patient* - pacient
- *GP* – praktický lékař
- *specialist* – lékař specialista
- *operator* – operátor záchranné služby
- *everybody* – jakýkoliv uživatel

<status> – označení platnosti modelu, možné hodnoty jsou

- *valid* – platný model
- *draft* – jen návrh modelu
- *expired* – model, jehož platnost vypršela

<keys> – seznam klíčů (klíčových atributů) vyskytujících se v modelu

<key> – ohraničení jednoho klíčového atributu

<keyname> – označení klíče

<keyweight> – váha klíče udává, do jaké míry je klíč popsán modelem

Celý systém reprezentace znalostí MEKRES a algoritmus výběru relevantního znalostního modelu je znázorněn na obrázek 30. Algoritmus výběru relevantního modelu lze popsat následovně:

Pro uživatele systému a jeho atributy (data o uživateli) je vybrána množina oborů (<branch>) a klíčů (<keys>), které odpovídají stavu uživatele (datům).

- např. atributům pacienta „STK“, „DTK“ a „glycaemia“ odpovídají obory „diabetes“ a „hypertenze“

Pro každý obor <branch> a klíč je vybrána množina znalostních modelů (GLIKREM), které obsahují tento obor v hlavičce <head> a váha klíče <keyweight> uloženého v modelu je nenulová.

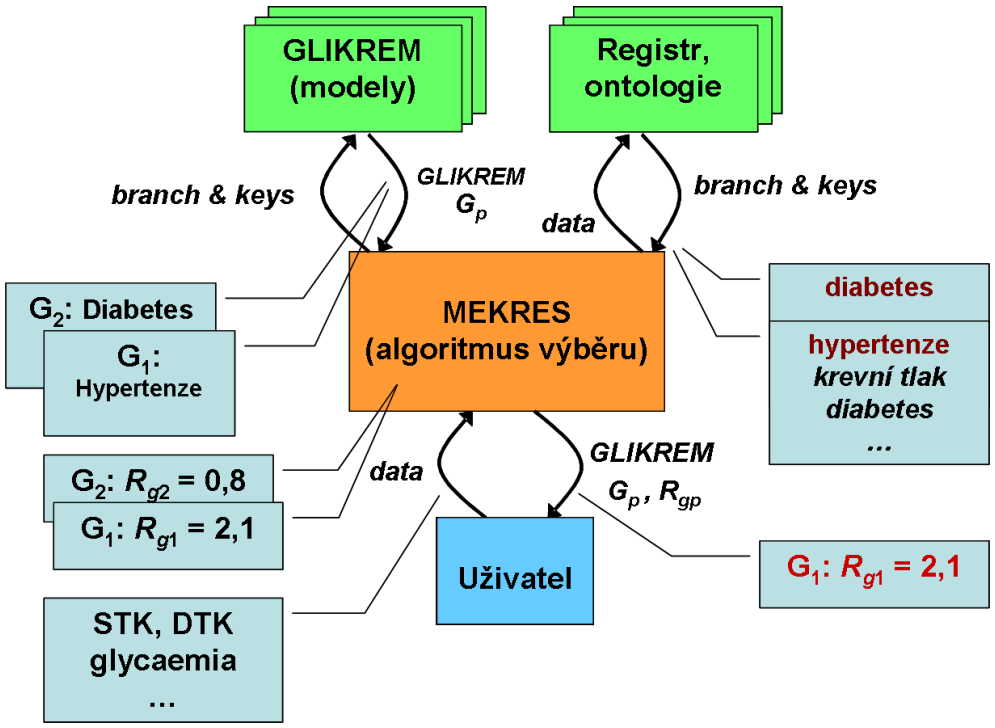
- např. modely G_1 (hypertenze) a G_2 (diabetes)

Pro každý vybraný model je stanovena hodnota obecného agregovaného operátoru (např. $R_g = \sum_k keyweight(k)$) udávající relevantnost daného modelu.

- např. model G_1 má operátor R_{g1} = 2,1 a model G_2 má R_{g2} = 0,8

Uživateli je zpětně zobrazen model s největší hodnotou relevance nebo seznam modelů uspořádaných podle relevance.

- např. uživateli je nabídnut model G_1 s největší hodnotou relevance



obrázek 30 Systém MEKRES

5.5 Model průchodu studenta výukovým systémem

Pomocí znalostního modelu GLIKREM byl navržen model průchodu studenta vyučovaným předmětem s elektronickou podporou výuky. Navržený grafický model byl následně zakódován v XML. Výsledný model byl pak implementován jako modul e-learningového systému Moodle ČZU. Souběžně byl vytvořen i model parametrů (paramodel) a hodnotová mapa, která umožňuje napojit paramodel na skutečná data uložená v databázi Moodle ČZU.

Předmětem zkoumání je student denního studia, jehož cílem je získat během jednoho semestru zápočet daného předmětu. Pro pilotní modul byl vybrán předmět Výpočetní

systémy, který je povinným profilovým předmětem v prvním ročníku bakalářské stupně studia v oboru Informatika na Provozně ekonomické fakultě ČZU v Praze.

Pro absolvování předmětu (dosažení cíle) je třeba splnit následující podmínky:

- prostudování 7 kapitol s teorií týkající se předmětu – na konci každé kapitoly je několik kontrolních otázek (dílčí kontrolní test), na které musí student správně odpovědět (odpovídat může v několika pokusech).
- docházka na cvičení ve 13-ti výukových týdnech – je třeba získat minimálně 70% účast na cvičeních.
- samostatný projekt – odevzdání samotného projektu na zadané téma. Student musí získat hodnocení minimálně 70%. V samostatném projektu je možné provést jednu opravu.
- zápočtový test - kontrolní test z probírané problematiky v celém semestru. Úspěšné absolvování testu znamená získat minimálně 60% z možného hodnocení. Test je možné absolvovat maximálně ve dvou pokusech.

Průběh studia, tj. docházka a výsledky dílčích podmínek, je zaznamenán v systému pro podporu elektronického vzdělávání (e-learningovém systému) Moodle ČZU. Ve stejném systému jsou studentovi k dispozici i potřebné studijní materiály (teoretické kapitoly).

Návrh znalostního modelu a implementace v e-learningovém systému Moodle ČZU je podrobněji popsán v příloze (kapitola 8.1).

6 Závěr

6.1 Shrnutí a diskuse

V dnešním tržním prostředí, kdy úspěch a konkurenceschopnost v řadě oborů lidské činnosti závisí na kreativitě, schopnostech, expertíze, dovednostech, zlepšování a inovacích, jsou informace a znalosti velmi důležitým a cenným faktorem. Úspěšné společnosti jsou dnes převážně ty, které se vyznačují zejména prozíravostí a předvídavostí vývoje, mobilizace a alokace svých informačních a znalostních zdrojů. Je však rozdílné pracovat s informacemi a znalostmi, proto autor věnoval vysvětlení pojmů informace, znalost a typy znalostí kapitoly 2.1 a 2.2 rešeršní části disertační práce.

Znalost vzniká zasazením množství informací do kontextu a nalezením jejich kauzálních vazeb, znalosti jsou tedy založeny na kontextové hodnotě informací. Významná část informačního a znalostního potenciálu by měla být u jednotlivců, organizací, států či jiných skupin uchovávána v digitální podobě a tvořit znalostní bázi podniku, oboru apod.. Znalostní báze, jako součást znalostního či expertního systému, hraje významnou roli v jakémkoliv kvalifikovaném rozhodovacím procesu prováděným člověkem v dané problémové oblasti. Podstatné části a rozdíly mezi znalostními a expertními systémy popsal autor v rešeršní kapitole 2.3.

Zatímco data a informace jsou získávány pozorováním reálných situací spojeným s jejich měřením, znalosti získává člověk vzděláním a zkušenostmi. Ze samotné podstaty znalostí vyplývá jejich výjimečnost a složitost, která se promítá i do procesu jejich získávání. Existují však ověřené techniky, které mohou být při získávání znalostí užitečné. Těmto technikám získávání znalostí věnoval autor kapitolu 2.5. Valná většina údajů a znalostí, které užíváme při rozhodování či řízení, je však do určité míry neurčitá. Problematikou neurčitosti ve znalostních systémech se zabývá kapitola 2.6.

Budování znalostního systému není triviální záležitostí. Při vytváření báze znalostí je velmi důležité, aby použitá reprezentace dovolovala jednoduše upřesňovat znalosti,

tj. jedná se o požadavek modularity reprezentace znalostí. Na druhé straně však existují důvody pro požadavek, aby příbuzné znalosti byly volně sdružovány a nebyly reprezentovány nezávisle. Jedná se o požadavek sémantického sdružování znalostí. Na reprezentaci znalostí jsou tak často kladeny protichůdné požadavky. Popisem a porovnáním vybraných metod reprezentace znalostí se autor zabýval v kapitole 2.7.

Jedním z oborů, kde se setkáme s potřebou reprezentace znalostí, je medicína. Jedná se především o způsob zaznamenání znalostí experta, resp. expertní skupiny, do textové podoby - lékařských doporučení. Oborová (lékařská) doporučení jsou vydávána jako volný text, ve kterém jsou zaznamenány doporučené postupy řešení určitých problémů (např. diagnostika a léčba pacientů). Vzhledem k poměrně obsáhlému a odborně propracovanému souboru lékařských doporučení je pozornost autora zaměřena především na modelování znalostí ve formě doporučených postupů a na v současnosti používané metody. Přehled metod je uveden v kapitole 3.1.

Jedním z nejčastěji používaných modelů reprezentace lékařských doporučení je GLIF (*Guideline Interchange Format*). GLIF modelu a jeho možnostem reprezentace, především v UML (*Unified Modelling Languauge*) a systému Protégé, je v disertační práci věnována kapitola 3.2.

Přesto, že lze GLIF považovat za dobře použitelný nástroj pro reprezentaci znalostí obsažených v lékařských doporučeních, z praktických zkušeností z jeho implementace vyplynulo několik omezení a nedostatků. Jedná se především o absenci obecné definice rozhodování v případě neurčitých (nejistých) hodnot parametrů modelu, řešení opakovaných činností a reprezentace času v modelu a parametrů modelu v různých časových úsecích.

Na základě výše uvedených důvodů byl autorem disertační práce navržen vlastní zobecněný model reprezentace znalostí v oborových (včetně lékařských) doporučeních GLIKREM (*GuideLInes Knowledge REpresentation Model*). GLIKREM poskytuje ucelenou metodu reprezentace doporučení od fáze konstrukce znalostního modelu, přes jeho implementaci v XML (*eXtensible Markup Language*) až po definici datového

rozhraní (paramodelu) a jeho napojení na reálná data uložená v databázi (např. elektronickém zdravotním záznamu pacienta) či informačním systému (např. klinickém informačním systému nemocnice). Fázi konstrukce znalostního modelu věnoval autor kapitolu 4.2 a fázi implementace včetně napojení na reálná data kapitolu 4.3. Implementace modelu ve formě XML schématu je podrobně uvedena i v příloze (kapitola 8.1).

Navržený model reprezentace znalostí oborových doporučení byl použit v praktických aplikacích, na jejichž vývoji se autor také podílel. Jedná se o nástroj GLIKREM editor usnadňující fázi návrh znalostního modelu a jeho zakódování do XML (popsán v kapitole 5.1), obecný prohlížeč znalostního modelu GLIF-VIEW sloužící především jako učební pomůcka začínajících mediků (kapitola 5.2) či několik typů prohlížečů řízených daty popsaných v kapitole 5.3. GLIKREM je součástí i systému reprezentace medicínských znalostí (MEKRES) včetně výběru relevantního modelu, jehož popisu se věnuje kapitola 5.4. Navržený model byl však použit i mimo oblast medicíny, konkrétně pro modelování průchodu studenta elektronickým výukovým systémem. Popisem tohoto použití se autor zabývá v kapitole 5.5 a podrobněji také v příloze v kapitole 8.2.

6.2 Rekapitulace výsledků práce

Cílem předkládané práce bylo především:

- Zmapovat oblast získávání a reprezentace znalostí a představit nejvýznamnější existující metody reprezentace znalostí v oborových (lékařských) doporučeních
- Na základě získaných poznatků a praktických zkušenostní navrhnout vlastní zobecněný model reprezentace znalostí obsažených v oborových doporučeních
- Popsat vhodný způsob kódování (implementace) znalostního modelu pro použití ve znalostních systémech
- Navrhnout datové rozhraní mezi modelem znalostí a reálnými daty uloženými v databázi či informačním systému

- Ověřit použitelnost navrženého modelu a jeho implementace v praktických aplikacích především v oblasti medicíny

Pro reprezentaci znalostí obsažených v oborových doporučeních byl autorem práce navržen vlastní zobecněný model (GLIKREM), který vychází z osvědčeného modelu GLIF. Navržený model představuje ucelenou metodu reprezentace znalostí ve fázích návrhu, implementace i použití modelu. Hlavní přínosy této původní metody lze shrnout následovně:

- Zobecnění znalostního modelu (GLIKREM) a popis všech prvků tohoto modelu je popsáno v kapitole 4.2.1 (typy vrcholů modelu) a v kapitole 4.2.2 (rozhodovací kritéria rozhodovacích vrcholů).
- Upřesnění rozhodování v modelu v případě neurčitosti hodnot vstupních parametrů, konkrétně se jedná o pravidla rozhodování v tří-hodnotové logice (kapitola 4.2.4) a s tím souvisejícím rozšířením výchozího (GLIF) modelu o priority rozhodovacích větví (kapitola 4.2.6)
- Zavedení synchronizačních podmínek pro synchronizační krok (vrchol) modelu. Použití synchronizačních podmínek je popsáno v kapitole 4.2.7.
- Rozšíření modelu o způsob reprezentace opakovaných činností, kterému se věnuje kapitola 4.2.8.
- Původní metoda modelování času a časových transakcí v modelu. S tím souvisí i způsob rozhodování na základě parametrů získaných v různých časových úsecích (transakcích), kterému se autor věnuje v kapitole 4.2.9
- Návrh paramodelu (modelu parametrů), který je použit jako rozhraní mezi znalostním modelem a reálnými daty v informačních systémech. Paramodel je popsán v kapitole 4.2.3 a jeho napojení (transformace) na reálná data v kapitole 4.3.4.

- Formální reprezentace grafického znalostního modelu i modelu parametrů v XML, které je věnována kapitola 4.3. Výsledkem reprezentace je návrh XSD (*XML Schema Definition*), který je uveden v příloze v kapitole 8.1.
- Rozšíření modelu o klíčové atributy a algoritmus výběru relevantního modelu. Toto rozšíření je dále využito v systému reprezentace lékařských znalostí (MEKRES), podrobně popsaného v kapitole 5.4.

6.3 Aplikace výsledků práce

Výsledky předkládané disertační práce je možno uplatnit, a již také uplatněny byly, v následujících oblastech:

- Především se jedná o aplikaci v oblasti medicíny pro elektronickou reprezentaci doporučených postupů diagnostiky a léčby onemocnění z různých oborů. Výsledný model reprezentace lékařských doporučení ve spojení s moderními informačními technologiemi (internet, mobilní telefony s podporou aplikací, elektronické zdravotní knížky apod.) může najít uplatnění ve výuce mladých mediků ale i jako podpora rozhodování především praktických lékařů a lékařů v „terénu“ jako jsou např. operátoři či lékaři záchranné služby.
- Navržený model lze použít pro reprezentaci znalostí v doporučeních z jakéhokoliv oboru lidské činnosti. Jednou z možných (a v práci prezentovaných) oblastí může být i výuka předmětů s pomocí elektronických výukových systémů.
- V akademické sféře, kdy navržená metoda bude představena široké odborné veřejnosti a to nejen informaticky orientované.

6.4 Možnosti dalšího rozvoje

Možnosti dalšího rozvoje spatřuje autor především ve vývoji integrovaného aplikačního nástroje, který by umožňoval implementaci navržené metody reprezentace znalostí a to všech jejích fázích.

Autor předpokládá, že navržený znalostní model bude použit pro formální reprezentaci lékařských doporučení v připravovaném internetovém katalogu českých doporučených postupů v medicíně. Tato rozsáhlejší sada formalizovaných modelů budou následně sloužit jako znalostní báze v systému reprezentace lékařských znalostí (MEKRES).

V současné době je, v oblasti medicíny, nastupujícím trendem ukladání (i zadávání) dat o pacientech ve strukturované podobě (ve formě elektronického zdravotního záznamu) oproti formě volného textu (lékařských zpráv). Další využití navrženého modelu proto autor předpokládá jako součást kontrolního či připomínkového systému, který by kontroloval zadávané údaje (uživatelem / lékařem) nebo existující záznamy v databázi, zda jsou v souladu s příslušnými doporučeními. V případě nesouladu by na to uživatele upozornil.

7 Přehled literatury

7.1 Publikace autora

7.1.1 Impaktované a odborné časopisy

[A1] Veselý, Arnošt; Zvárová, Jana; Peleška, Jan; Buchtela, David; Anger, Zdeněk. Medical Guidelines Presentation and Comparing with Electronic Health Record. *International Journal of Medical Informatics*, 2006, Roč. 75, č. 3-4, s. 240-245. ISSN 1386-5056. **IF** 1.726

[A2] Buchtela, David; Peleška, Jan; Veselý, Arnošt; Zvárová, Jana; Zvolský, Miroslav. Guideline Knowledge Representation Model (GLIKREM). *European Journal for Biomedical Informatics*, 2008, Roč. 4, č. 1, ISSN 1801-5603. Dostupné online: http://www.ejbi.org/articles/200812/34/1.html

[A3] Buchtela, David; Anger, Zdeněk; Peleška, Jan (ed.); Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana. Methods of Medical Guidelines Modelling in GLIF. *IFMBE Proceedings*, 2005, Roč. 11,s. 1529-1532. ISSN 1727-1983.

[A4] Buchtela, David; Peleška, Jan; Veselý, Arnošt; Zvárová, Jana; Zvolský, Miroslav. Model reprezentace znalostí v doporučeních. *Lékař a technika. Biomedicinské inženýrství a informatika*, 2009, Roč. 39, č. 1, s. 39-50. ISSN 0301-5491.

[A5] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Šebesta, K.; Tomečková, Marie; Veselý, Arnošt; Zvára, Karel; Zvárová, Jana. Formalization of Medical Guidelines. *European Journal for Biomedical Informatics*, 2005, Roč. 1, s. 133-141. ISSN 1801-5603.

[A6] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Šebesta, K.; Tomečková, Marie; Veselý, Arnošt; Zvára, Karel; Zvárová, Jana. Formalized Medical Guidelines

and a Structured Electronic Health Record. *IFMBE Proceedings*, 2005, Roč. 11, s. 4652-4656. ISSN 1727-1983.

[A7] Buchtela, David; Brožek, Jiří. Knowledge Representation System or Decision Support. In *Scientia Agriculturae Bohemica*, 2008, 39, s.97 - 101. ISSN 1211-3174

7.1.2 Monografie a kapitoly v monografii

[A8] Zvárová, Jana; Svačina, Š.; Valenta, Zdeněk; Berka, Petr; Buchtela, David; Jiroušek, Radim; Malý, Marek; Papíková, Vendula; Peleška, Jan; Rauch, Jan; Vajda, Igor; Veselý, Arnošt; Zvára, Karel; Zvolský, Miroslav. *Systémy pro podporu lékařského rozhodování*. Praha: Universita Karlova - nakladatelství Karolinum, 2009. 504 s. (Biomedicínská informatika: 3). ISBN 978-80-246-1732-9.

[A9] Buchtela, David; Anger, Zdeněk; Peleška, Jan; Veselý, Arnošt; Zvárová, Jana. Presentation of Medical Guidelines on a Computer. In *Transformation of Healthcare with Information Technologies*. Amsterdam: IOS Press, 2004. s. 166-171. ISBN 1-58603-279-8.

[A10] Buchtela, David; Veselý, Arnošt; Vynikarová, Dana. Guideline Knowledge Representation Model (GLIKREM). In *Knowledge Management and Modern Information Technologies*. 1. vyd. Praha: Alfa Nakladatelství, 2010. s. 26-41. ISBN 978-80-87197-31-8.

7.1.3 Sborníky zahraničních konferencí

7.1.3.1 Příspěvek ve sborníku

[A11] Buchtela, David; Peleška, Jan; Veselý, Arnošt; Zvárová, Jana; Zvolský, Miroslav. Formalization of Clinical Practice Guidelines. In *eHealth Beyond the*

Horizon - Get it There. Amsterdam: IOS Press, 2008. s. 151-156. ISBN 978-1-58603-864-9.

[A12] Buchtela, David; Peleška, Jan; Zvolský, Miroslav; Zvárová, Jana. Medical Knowledge Representation System. In *eHealth Beyond the Horizon - Get it There*. Amsterdam: IOS Press, 2008. s. 377-382. ISBN 978-1-58603-864-9.

[A13] Buchtela, David; Peleška, Jan; Veselý, Arnošt; Zvárová, Jana. Method of GLIF Model Construction and Implementation. In *Connecting Medical Informatics and Bio-Informatics*. Amsterdam: IOS Press, 2005. s. 779-784. ISBN 978-1-58603-549-5. ISSN 0926-9630.

[A14] Zvárová, Jana; Heroutová, Helena; Grünfeldová, Hana; Zvára, Karel; Buchtela, David. Empowering Clinicians by eHealth Technologies in Decision-Making Tasks. In *Medical Informatics in a United and Healthy Europe*. Amsterdam: IOS Press, 2009. s. 683-687. ISBN 978-1-60750-044-5.

[A15] Zvárová, Jana; Veselý, Arnošt; Hanzlíček, Petr; Špidlen, Josef; Buchtela, David. On Direct Comparing of Medical Guidelines with Electronic Health Record. In *Computational Science*. Berlin: Springer, 2004. s. 1133-1139. ISBN 3-540-22129-8.

[A16] Buchtela, David; Vynikarová, Dana. Standardized Knowledge and Data Interface Model of Branch Guidelines. In *Agrarian Perpectives XVIII "Strategies for the Future"*, FEM CULS Prague, 2009. s. 605-610. ISBN 978-80-213-1965-3.

7.1.3.2 Abstrakt ve sborníku

[A17] Buchtela, David; Kiml, Jan; Wunsch, Petr; Peleška, Jan; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana. Medical Guidelines Implementation in Decision Support System. In *International Joint Meeting EuroMISE 2004 Proceedings*. Prague: EuroMISE, 2004. s. 124. ISBN 80-903431-0-4.

[A18] Peleška, Jan; Anger, Zdeněk; Aschermann, Michael; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana. Formalization of Medical Guidelines. In *International Joint Meeting EuroMISE 2004 Proceedings*. Prague: EuroMISE, 2004. s. 114. ISBN 80-903431-0-4.

[A19] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana; Zvolský, Miroslav. Formalization of 2007 ESH/ESC Hypertension Guidelines (HGL). *Journal of Hypertension*, 2008, Roč. 26, Suppl. 1, s. 302-302. ISSN 0952-1178.

[A20] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana. Formalized Combined 2003 European Guidelines on Cardiovascular Disease Prevention and Hypertension. *Journal of Hypertension*, 2005, 23 Suppl. 2, s. 196. ISSN 0263-6352.

[A21] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana. Formalized Interconnected Guidelines on Cardiovascular Disease Prevention and Those for Management of Diabetes, Dyslipidemia and Hypertension. *Journal of Hypertension*, 2006, 24 Suppl. 4, s. 172-172. ISSN 0263-6352.

[A22] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana. Formalized 2003 ESH/ESC Hypertension Guidelines in Hospital and General Practice. *Journal of Hypertension*, 2004, 22 Suppl. 2, s. 253. ISSN 0263-6352.

[A23] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana. Formalized 2003 European Guidelines on Cardiovascular Disease Prevention and for Management of Hypertension. In *Central-European Meeting on Hypertension*. Manchester: European Society of Hypertension, 2005. s. 46.

[A24] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt. Formalized 2003 European Guidelines on Cardiovascular Disease Prevention in Clinical Practice. *European Heart Journal*, 2004, Roč. 25, s. 444. ISSN 0195-668X.

[A25] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie. Is Substantial Improvement in Blood Pressure and Other Risk Factor Control in Primary Care Possible?. *Kidney & Blood Pressure Research*, 2006, Roč. 29, s. 253-253. ISSN 1420-4096.

[A26] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana. Treatment Algorithm for the Hypertension Specialist after the Milan Meeting in 2007. *Kidney & Blood Pressure Research*, 2007, Roč. 30, s. 374-374. ISSN 1420-4096.

[A27] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana. Treatment Algorithm of a Hypertension Specialist. *Journal of Hypertension*, 2007, Roč. 25, Suppl. 2, s. 383 -383. ISSN 0952-1178.

[A28] Veselý, Arnošt; Zvárová, Jana; Peleška, Jan; Anger, Zdeněk; Buchtela, David. Computerized Presentation of Medical Guidelines. In *Proceedings of the 11th World Congress on Medical Informatics*. Edmonton: IMIA, 2004. s. 1894. ISBN 1-58603-444-8.

[A29] Veselý, Arnošt; Zvárová, Jana; Buchtela, David. GLIF: Decision Steps with Missing data. In *ISCB 2009*. Prague, 2009. s. 151.

[A30] Veselý, Arnošt; Zvárová, Jana; Peleška, Jan; Buchtela, David; Anger, Zdeněk Medical Guidelines Presentation and Comparing with Electronic Health Record. In *International Joint Meeting EuroMISE 2004 Proceedings*. Prague: EuroMISE, 2004. s. 53. ISBN 80-903431-0-4.

7.1.4 Sborníky domácích konferencí

7.1.4.1 Příspěvek ve sborníku

[A31] Buchtela, David; Peleška, Jan; Zvolský, Miroslav; Zvárová, Jana. Elektronická zdravotní doporučení ve stomatologii. In *PLATFORMA i2010. Inovace - investice - integrace*. Praha: ČVUT FEL, 2007.

[A32] Buchtela, David. Konstrukce GLIF modelu a znalostní ontologie. Praha: MATFYZPRESS, 2005. In *Doktorandský den '05*. s. 11-15. ISBN 80-86732-56-8.

[A33] Buchtela, David. Řešení problémových situací v GLIF modelu. In *Doktorandský den '04*. Praha: MATFYZPRESS, 2004. s. 172-179. ISBN 80-86732-30-4.

[A34] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana; Zvolský, Miroslav. Formalizace Doporučení diagnostických a léčebných postupů u arteriální hypertenze České společnosti pro hypertenzi – verze 2007. In *Informační technologie pro rozvoj kontinuální sdílené péče o zdraví*. Praha: EuroMISE s.r.o, 2008. s. 63-64. ISBN 80-903431-3-9.

[A35] Buchtela, David; Vynikarová, Dana. E-learningový systém Moodle ČZU. In *Sborník příspěvků ke konferenci Trendy a inovace informačních systémů vysokých škol*. Plzeň: Západočeská univerzita v Plzni, 2010. s. 5-9. ISBN 978-80-7043-892-3.

[A36] Buchtela, David; Vynikarová, Dana; Pavlíček, Josef. Znalostní model průchodu studenta výukou. In *Agrární perspektivy XVII (Výzvy pro 21. století)*. Praha: Česká zemědělská univerzita, 2008. s.537-540. ISBN 978-80-213-1813-7.

[A37] Pavlíček, Josef; Buchtela, David; Vynikarová, Dana. Inteligentní vyhledávač Webových služeb. In *Agrární perspektivy XVII (Výzvy pro 21. století)*. Praha: Česká zemědělská univerzita, 2008. ISBN 978-80-213-1813-7.

[A38] Buchtela, David. Systém reprezentace znalostí pro podporu rozhodování. In *Agrární perspektivy XVI (Evropské trendy v rozvoji zemědělství a venkova)*. Praha: Česká zemědělská univerzita, 2007. s.1711-1717. ISBN 978-80-213-1675-1.

[A39] Buchtela, David. Návrh datového rozhraní mezi GLIF modelem a informačním systémem. In *Agrární perspektivy XV (Zahraniční obchod a globalizační procesy)*. Praha: Česká zemědělská univerzita, 2006. s.769-773. ISBN 80-213-1531-8.

[A40] Buchtela, David; Vynikarová, Dana. Zkušenosti s používáním elektronického výukového systému (ELVYS). In *INFORMATIKA XVIII*. Brno: KONVOJ, spol. s r. o., 2006. s. 22-28. ISBN 80-7302-111-0.

[A41] Buchtela, David. Použití znalostních ontologií při modelování oborových doporučení GLIF modelem. In *Agrární perspektivy XIV (Znalostní ekonomika)*. Praha: Česká zemědělská univerzita, 2005. s. 612-616. ISBN 80-213-1372-2.

[A42] Buchtela, David. Proces modelování oborových doporučení v GLIFu. In *Doktorandský seminář - sborník příspěvků*. Praha: Česká zemědělská univerzita, 2005. s. 325-329. ISBN 80-213-1314-5.

[A43] Vynikarová, Dana; Buchtela, David. Analýza úspěšnosti výuky předmětu Výpočetní systémy II. In *Konference Informatika - sborník příspěvků*. Brno: KONVOJ, spol. s r. o., 2005. s. 206-211. ISBN 80-7302-083-1.

[A44] Buchtela, David. Implementace GLIF modelu v XML. In *Doktorandský seminář - sborník příspěvků*. Praha: Česká zemědělská univerzita, 2004. s. 48. ISBN 80-213-1150-9

[A45] Buchtela, David; Merunka, Vojtěch; Pavlíček, Josef; Pícka, Marek; Vaníček, Jiří; Vynikarová, Dana. Pokus o sondu studijních předpokladů pro informatiku. In *Sborník příspěvků Informatika XV*, Brno: PEF MZLU, 2004. s. 19-23. ISBN 80-7302-006-1.

[A46] Buchtela, David. Rozšíření specifikace GLIF modelu. In *Sborník prací - Agrární perspektivy XIII*. Praha: Česká zemědělská univerzita, 2004. 5s.. ISBN 80-213-1190-8

7.1.4.2 Abstrakt ve sborníku

[A47] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana. Algoritmus specialisty pro léčbu nekomplikované hypertenze. In *Sborník abstrakt 24. konference České společnosti pro hypertenzi, 16. konference pracovní skupiny Preventivní kardiologie ČKS, 12. konference pracovní skupiny Srdeční selhání ČKS*. Brno: Medica Healthworld, 2007. s. 73-73. ISBN 978-80-254-0421-8.

[A48] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt. Algoritmus specialisty pro léčbu nekomplikované hypertenze v lednu 2008. In *XVI. výroční sjezd České kardiologické společnosti*. Brno: Česká kardiologická společnost, 2008. s. 48-48.

[A49] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana; Zvolský, Miroslav. Formalizace doporučení diagnostických a léčebných postupů u arteriální hypertenze - verze 2007. *Cor et Vasa*, 2008, Roč. 50, č. 9, s. 168-168. ISSN 0010-8650.

[A50] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana. Formalizace evropských doporučení o prevenci kardiovaskulárních chorob z roku 2003 a české varianty 2004. *Cor et Vasa*, 2005, Roč. 47, 4 suppl., s. 81. ISSN 0010-8650.

[A51] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana. Formalizace propojených doporučených postupů pro prevenci kardiovaskulárních onemocnění v dospělém věku, hypertenzi, diabetes mellitus, dyslipidemie, obezitu a kouření. *Cor et Vasa*, 2006, Roč. 48, 4 suppl., s. 90-91. ISSN 0010-8650.

[A52] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana. Formalizace spojených doporučených postupů pro prevenci kardiovaskulárních chorob, hypertenzi, DM a dyslipidémie. Český Krumlov: Galén, 2005. ISBN 80-7262-362-1. In *Abstrakta* s. 43.

[A53] Peleška, Jan; Anger, Zdeněk; Aschermann, Michael; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana. Jsou elektronická lékařská doporučení pro nestabilní anginu pectoris přínosem?. *Intervenční a akutní kardiologie*, 2003, Roč. 2, Suppl A, s. 17-18. ISSN 1213-807X.

[A54] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana. Léčebný algoritmus specialisty na hypertenzi. *Vnitřní lékařství*, 2007, Roč. 53, 9 příloha, s. 55-55. ISSN 0042-773X.

[A55] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana. Propojení doporučených postupů pro prevenci kardiovaskulárních chorob, hypertenzi, diabetes mellitus a dyslipidemie - jejich formalizace. *Vnitřní lékařství*, 2005, Roč. 51, č. 10, s. 1178. ISSN 0042-773X.

[A56] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana. Spojení doporučených postupů pro prevenci kardiovaskulárních chorob, diagnostiku a léčbu dyslipidemie, DM a hypertenze. *Metabolismus*, 2005, Roč. 8, 3 Suppl., s. 40. ISSN 1211-9326.

[A57] Peleška, Jan; Anger, Zdeněk; Buchtela, David; Tomečková, Marie; Veselý, Arnošt; Zvárová, Jana; Zvolský, Miroslav. Usnadnění práce lékaře při použití

formalizovaných českých hypertenzních doporučení. *Cor et Vasa*, 2009, Roč. 51, Suppl. 1 - CD ROM, s. 527-527. ISSN 0010-8650.

[A58] Zvárová, Jana; Peleška, Jan; Buchtela, David; Hanzlíček, Petr; Špidlen, Josef; Tomečková, Marie; Veselý, Arnošt. Elektronická lékařská doporučení. In *Sborník abstrakt*. Praha: Galén, 2004. s. 57. ISBN 80-7262-288-9.

[A59] Buchtela, David; Mach, Jiří; Vynikarová, Dana. Integrace systému odevzdej.cz a Moodle ČZU. In *Sborník abstraktů z konference Informatika XXIII*. Brno: Mendelova univerzita v Brně, 2010. s. 17. ISBN 978-80-7375-394-8.

7.1.5 Ostatní publikace

[A60] Buchtela, David; Vynikarová, Dana. Výpočetní systémy – cvičení. Praha: Česká zemědělská univerzita, 2004. s. 1 -198. ISBN 80-213-1170-3

[A61] Buchtela, David; Vynikarová, Dana. Cvičebnice z předmětu Výpočetní systémy. Praha: Česká zemědělská univerzita, 2009. s. 1 -123. ISBN 978-80-213-2017-8.

[A62] Buchtela, David; Vynikarová, Dana. Cvičebnice z předmětu Architektura počítačů. Praha: Česká zemědělská univerzita, 2010. s. 1 -83. ISBN 978-80-213-2073-4.

7.2 Použitá literatura

7.2.1 Knižní a časopisecké publikace

[1] Schmuller, J.: Myslíme v jazyku UML, Praha: Grada Publishing, 2001, 360 s.. ISBN 80-247-0029-8.

[2] Berka, P.: Dobývání znalostí z databází, Academia, Praha, 2003, 366 s., ISBN 80-200-1062-9.

[3] Berka, P.; Jirků, P.; Vejnarová, J.: Expertní systémy, VŠE, Praha, 1998, 166 s., ISBN 80-7079-873-4.

[4] Bury, J.; Fox, J.; Sutton, D.: The PROforma guideline specification language: progress and prospects. In *Proceedings of the First European Workshop, Computer-based Support for Clinical Guidelines and Protocols (EWGLP 2000)*, 2000.

[5] Deerwester, S. and col.: Indexing by Latent Semantic Analysis, In *Journal of the American Society for Information Science*, 41, 1990, p. 391-407, ISSN 0002-8231.

[6] Gennari, J.H.; Musen, M.A.; Fergerson, R.W.; Grosso, W.E.; Crubezy, M.; Eriksson, H.; et al. The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. In *International Journal of Human-Computer Studies*. Academic Press, 2003, vol. 58(1). s. 89-123. ISSN 1071-5819.

[7] Mařík, V. a kol.: Umělá inteligence, díl 1, Academia – nakladatelství AV ČR, Praha, 1993, 264 s., ISBN 80-200-0496-3.

[8] Mařík, V. a kol.: Umělá inteligence, díl 2, Academia – nakladatelství AV ČR, Praha, 1997, 374 s., ISBN 80-200-0504-8.

[9] Mařík, V. a kol.: Umělá inteligence, díl 4, Academia – nakladatelství AV ČR, Praha, 2003, 475 s., ISBN 80-200-1044-0.

[10] Minsky, M.L.: A Framework for Representing Knowledge. In *The Psychology of Computer Vision*, P. H. Winston (ed.), McGraw-Hill, New York, 1975, pp. 211-277, ISSN 70710481.

[11] Novák, V.: Fuzzy množiny a jejich aplikace, Nakladatelství technické literatury, Praha, 1990, 296 s..

[12] Neapolitan, R.E.: Probabilistic Reasoning in Expert Systems, John Wiley & sons, inc., New York, 1989.

[13] Noy, N.F.; Fergerson, R.W.; Musen, M.A.. The knowledge model of Protégé-2000: combining interoperability and flexibility. In *Second International Conference on Knowledge Engineering and Knowledge Management*, Juan-les-Pins, France, 2000.

[14] Ohno-Machado, L.; Gennari, J.H.; Murphy, S.N.; Jain, N.L.; Tu, S.W.; Oliver, D.; et al.. The GuideLine Interchange Format: A model for representing guidelines. In *Journal of the American Medical Informatics Association*. 1998. ISSN 1067-5027.

[15] Boxwala, A.A.; Peleg, M.; Tu, S.W.; Ogunyemi, O.; Zeng, Q.; Wang, D.; et al. GLIF3: A Representation Format for Sharable Computer-Interpretable Clinical Practice Guidelines. In *Journal of Biomedical Informatics*. Academic Press, 2004, vol. 37(3). s.147-61. ISSN 1532-0464.

[16] Johnson, P.D.; Tu, S.W.; Booth, N.; Sugden, B.; Purves, I.N.: Using scenarios in chronic disease management guidelines for primary care. In *Proceedings of the AMIA Annual Symposium*. 2000. s. 389–393.

[17] Peleg, M.; Ogunyemi, O.; Tu, S.; et al. Using features of Arden Syntax with object-oriented medical data models for guideline modeling. In *Proceedings of the 2001 AMIA Annual Symposium*. 2001. s. 523–7. ISBN 1-56053-536-9.

[18] Peleg, M.; Tu, S.W. et al.: Comparing Computer-interpretable Guideline Models: A Case-study Approach. In *The Jurnal of the American Medical Informatics Association*. 2003. Jan–Feb. vol. 10(1). s. 52–68.

[19] Quillian, M.R.: Word Concepts: A Theory and Simulation of Some Basic Semantic Capabilities, In *Behavioral Science*, Sep.12(5), 1967, pp. 410-430.

[20] Quaglini, S.; Stefanelli, M.; Lanzola, G.; Caporusso, V.; Panzarasa, S.: Flexible guideline-based patient careflow systems. In *Artifical Intelligence in Medicine*. 2001. vol.22. s. 65–80.

[21] Shahar, Y.; Miksch, S.; Johnson, P.: The Asgaard Project: a task-specific Framework for the application and critiquing of time-oriented clinical guidelines. In *Artifical Intelligence in Medicine*. 1998. vol. 14. s. 29–51.

[22] Shortliffe, E.H.: MYCIN: Computer-based Medical Consultations. Elsevier Press, New York, 1976, 264 p., ISBN 0444001794.

[23] Sordo, M.; Boxwala, A.; Ogunyemi, O.; Greenes, R.. Description and Status Update on GELLO: a Proposed Standardized Object-oriented Expression Language for Clinical Decision Support. In *Proceedings from Medinfo 2004*, 2004. p. 164-8.

[24] Tu, S.W.; Musen, M.A.: A flexible approach to guideline modeling. In *Proceedings of the AMIA Annual Symposium*. 1999. s. 420–424.

[25] Tu, S.W.; Musen, M.A.: From guideline Modeling to guideline execution: Defining guideline-based decision-support Services. In *Proceedings of the AMIA Annual Symposium*. 2000. s.863–867.

[26] Vaníček, J.; Papík, M.; Pergl, R.; Vaníček, T.: Teoretické základy informatiky. Praha. Kernberg Publishing. 2007. 436 s.. ISBN 978-80-903962-4-1

7.2.2 Internetové a další zdroje

[27] Laleci, G.B.; Dogac, A. et al.: SAPHIRE: A Multi-Agent System for Remote Healthcare Monitoring through Computerized Clinical Guidelines. Available online at http://www.srdc.metu.edu.tr/ webpage/projects/saphire/.

[28] UMLS - Unified Medical Language System. Available online at http://www.nlm.nih.gov/research/umls/.

[29] Zeng, Q.; Peleg, M.; Boxwala, A.; et al.. Guideline Interchange Format 3.5 technical specifications. Available online at: http://www.glif.org.

[30] Francois, M; Netron graph library architecture. Available online at http://netron.sourceforge.net/downloads/NetronGraphLibraryArchitecturev0.3.pdf

[31] ISO/IEC JTC1/SC7 3941 FCD, 24765 Systems and software engineering – Vocabulary, 2008, 298 ps, (pracovní materiál tvůrců norem ISO a IEC).

7.3 Seznam použitých obrázků a tabulek

7.3.1 Seznam obrázků

7.3.2 Seznam tabulek

8 Přílohy

8.1 XML schéma znalostního modelu GLIKREM

8.1.1 XML schéma grafického modelu

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2008 rel. 2 sp2 (http://www.altova.com) by David Buchtela (Ustav informatiky AV CR, v.v.i.) -->
<!-- edited with XMLSPY v5 rel. 4 U (http://www.xmlspy.com) by Registred (Registred) -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
	<xsd:include schemaLocation="glikrem_paramodel.xsd"/>
	<xsd:element name="graph" type="T_graph">
		<xsd:key name="KeyStep">
			<xsd:selector xpath="step"/>
			<xsd:field xpath="name"/>
		</xsd:key>
		<xsd:keyref name="ReferStep" refer="KeyStep">
			<xsd:selector xpath="step/next/option"/>
			<xsd:field xpath="nstep"/>
		</xsd:keyref>
	</xsd:element>
	<xsd:complexType name="T_graph">
		<xsd:sequence>
			<xsd:element name="step" type="T_step" maxOccurs="unbounded"/>
		</xsd:sequence>
	</xsd:complexType>
	<xsd:complexType name="T_step">
		<xsd:sequence>
			<xsd:element name="name" type="xsd:string" nillable="false"/>
			<xsd:element name="type" nillable="false">
				<xsd:simpleType>
					<xsd:restriction base="xsd:string">
						<xsd:enumeration value="action"/>
						<xsd:enumeration value="case"/>
						<xsd:enumeration value="branch"/>
						<xsd:enumeration value="synchronization"/>
						<xsd:enumeration value="state"/>
						<xsd:enumeration value="subgraph"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:element>
			<xsd:element name="text" type="xsd:string"/>
			<xsd:element name="status" default="in" nillable="false">
				<xsd:simpleType>
					<xsd:restriction base="xsd:string">
						<xsd:enumeration value="start"/>
						<xsd:enumeration value="end"/>
```

```
                                        <xsd:enumeration value="in"/>
                                    </xsd:restriction>
                                </xsd:simpleType>
                            </xsd:element>
                            <xsd:element name="operation" type="T_operation"/>
                            <xsd:element name="next" type="T_next" nillable="true"/>
                            <xsd:element name="caption" type="xsd:string"/>
                            <xsd:element name="x" type="xsd:integer"/>
                            <xsd:element name="y" type="xsd:integer"/>
                            <xsd:element name="width" type="xsd:positiveInteger"/>
                            <xsd:element name="height" type="xsd:positiveInteger"/>
                            <xsd:element name="focus" default="not" nillable="false">
                                <xsd:simpleType>
                                    <xsd:restriction base="xsd:string">
                                        <xsd:enumeration value="not"/>
                                        <xsd:enumeration value="auto"/>
                                        <xsd:enumeration value="user"/>
                                    </xsd:restriction>
                                </xsd:simpleType>
                            </xsd:element>
                        </xsd:sequence>
                    </xsd:complexType>
                    <xsd:complexType name="T_operation">
                        <xsd:sequence>
                            <xsd:element name="ops" type="T_ops" minOccurs="0"
maxOccurs="unbounded"/>
                        </xsd:sequence>
                    </xsd:complexType>
                    <xsd:complexType name="T_ops">
                        <xsd:sequence>
                            <xsd:element name="otype">
                                <xsd:simpleType>
                                    <xsd:restriction base="xsd:string">
                                        <xsd:enumeration value="insert"/>
                                        <xsd:enumeration value="get"/>
                                        <xsd:enumeration value="put"/>
                                        <xsd:enumeration value="open"/>
                                    </xsd:restriction>
                                </xsd:simpleType>
                            </xsd:element>
                            <xsd:element name="oparam" type="xsd:string"/>
                            <xsd:element name="ovalue" type="xsd:string"/>
                        </xsd:sequence>
                    </xsd:complexType>
                    <xsd:complexType name="T_next">
                        <xsd:sequence>
                            <xsd:element name="option" type="T_option" minOccurs="0"
maxOccurs="unbounded"/>
                        </xsd:sequence>
                    </xsd:complexType>
                    <xsd:complexType name="T_option">
                        <xsd:sequence>
                            <xsd:element name="nname" type="xsd:string" nillable="false"/>
                            <xsd:element name="nstep" type="xsd:string" nillable="false"/>
```

```
                        <xsd:element name="ntext" type="xsd:string"/>
                        <xsd:element name="npriority" type="xsd:positiveInteger" default="1"/>
                        <xsd:element name="nstrictin" type="xsd:string"/>
                        <xsd:element name="nstrictout" type="xsd:string"/>
                        <xsd:element name="nrulein" type="xsd:string"/>
                        <xsd:element name="nruleout" type="xsd:string"/>
                        <xsd:element name="nnote" type="xsd:string"/>
                        <xsd:element name="nline" type="T_line"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="T_line">
                <xsd:sequence>
                        <xsd:element name="point" type="T_point" minOccurs="0"
maxOccurs="unbounded"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="T_point">
                <xsd:sequence>
                        <xsd:element name="px" type="xsd:positiveInteger"/>
                        <xsd:element name="py" type="xsd:positiveInteger"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="glikrem">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element name="head" type="T_head"/>
                                <xsd:element ref="graph"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:complexType name="T_head">
                <xsd:sequence>
                        <xsd:element name="gname" type="xsd:string" nillable="false"/>
                        <xsd:element name="author" type="xsd:string" nillable="false"/>
                        <xsd:element name="date" type="xsd:date" nillable="false"/>
                        <xsd:element name="BID" type="xsd:string" default="none"
nillable="false"/>
                        <xsd:element name="branch" type="xsd:string" default="none"
nillable="false"/>
                        <xsd:element name="user" default="everybody" nillable="false">
                                <xsd:simpleType>
                                        <xsd:restriction base="xsd:string">
                                                <xsd:enumeration value="patient"/>
                                                <xsd:enumeration value="physician"/>
                                                <xsd:enumeration value="operator"/>
                                                <xsd:enumeration value="everybody"/>
                                        </xsd:restriction>
                                </xsd:simpleType>
                        </xsd:element>
                        <xsd:element name="status" nillable="false">
                                <xsd:simpleType>
                                        <xsd:restriction base="xsd:string">
                                                <xsd:enumeration value="valid"/>
                                                <xsd:enumeration value="expired"/>
```

```
                                        <xsd:enumeration value="draft"/>
                                    </xsd:restriction>
                                </xsd:simpleType>
                            </xsd:element>
                            <xsd:element name="keys" type="T_keys"/>
                        </xsd:sequence>
                    </xsd:complexType>
                    <xsd:complexType name="T_keys">
                        <xsd:sequence>
                            <xsd:element name="key" minOccurs="0" maxOccurs="unbounded">
                                <xsd:complexType>
                                    <xsd:sequence>
                                        <xsd:element name="keyname" type="xsd:string"
nillable="false"/>
                                        <xsd:element name="keyweight" type="xsd:float"
default="0" nillable="false"/>
                                    </xsd:sequence>
                                </xsd:complexType>
                            </xsd:element>
                        </xsd:sequence>
                    </xsd:complexType>
</xsd:schema>
```

8.1.2 XML schéma paramodelu

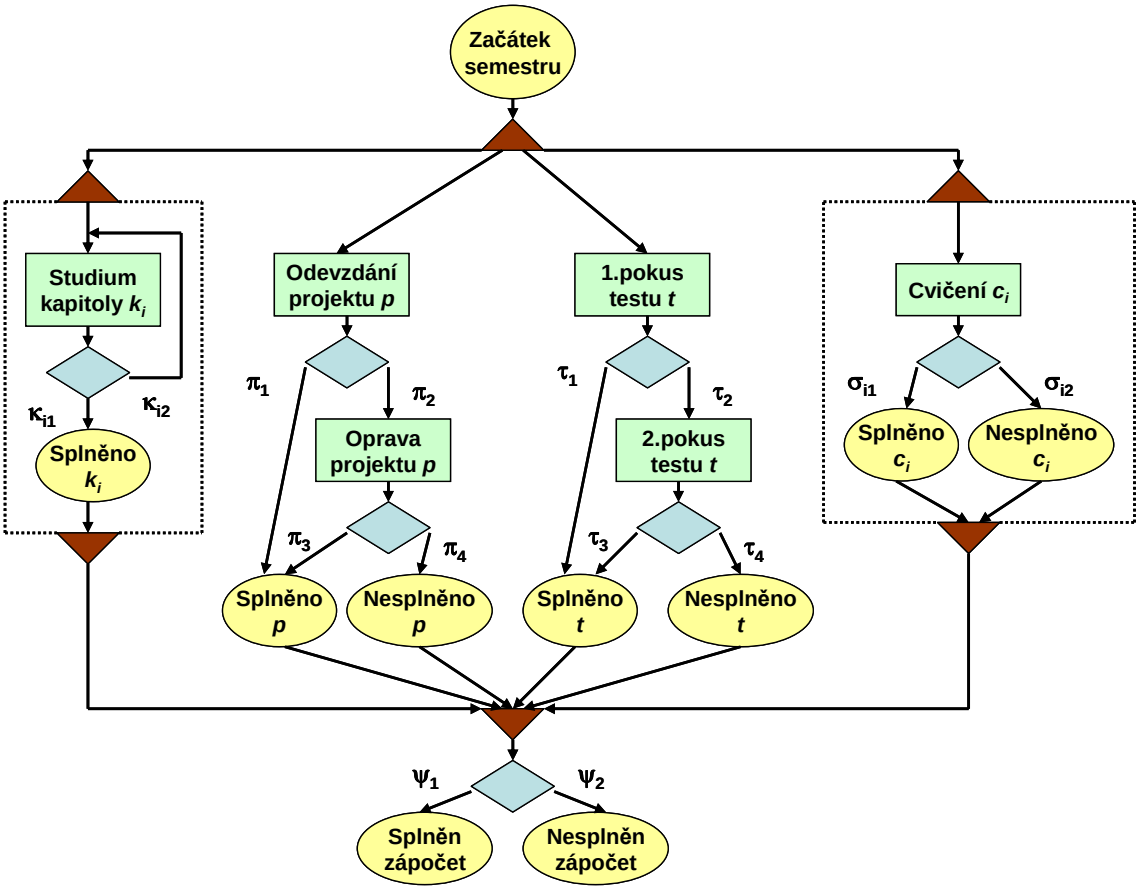
```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2008 rel. 2 sp2 (http://www.altova.com) by David (Ustav informatiky AV CR,
v.v.i.) -->
<!-- edited with XMLSPY v5 rel. 4 U (http://www.xmlspy.com) by Registred (Registred) -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xsd:element name="params" type="T_params">
        <xsd:key name="KeyParam">
            <xsd:selector xpath="param"/>
            <xsd:field xpath="pid"/>
        </xsd:key>
    </xsd:element>
    <xsd:complexType name="T_params">
        <xsd:sequence>
            <xsd:element name="param" type="T_param" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="T_param">
        <xsd:sequence>
            <xsd:element name="pid" type="xsd:string" nillable="false"/>
            <xsd:element name="ptype">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="basic"/>
                        <xsd:enumeration value="derived"/>
                        <xsd:enumeration value="onfly"/>
                    </xsd:restriction>
                </xsd:simpleType>
```

```
                    </xsd:element>
                    <xsd:element name="pname" type="xsd:string"/>
                    <xsd:element name="pdatype" type="xsd:string" nillable="false"/>
                    <xsd:element name="punits" type="xsd:string"/>
                    <xsd:element name="pdef" type="xsd:string"/>
                    <xsd:element name="pquery" type="xsd:string"/>
                    <xsd:element name="pnote" type="xsd:string"/>
                    <xsd:element name="ptrans" type="T_ptrans"/>
                    <xsd:element name="ptepsilon"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="paramodel">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="head" type="T_phead"/>
                        <xsd:element ref="params"/>
                    </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:complexType name="T_phead">
                <xsd:sequence>
                    <xsd:element name="gname" type="xsd:string" nillable="false"/>
                    <xsd:element name="author" type="xsd:string" nillable="false"/>
                    <xsd:element name="date" type="xsd:date" nillable="false"/>
                    <xsd:element name="status" nillable="false">
                        <xsd:simpleType>
                            <xsd:restriction base="xsd:string">
                                <xsd:enumeration value="valid"/>
                                <xsd:enumeration value="expired"/>
                                <xsd:enumeration value="draft"/>
                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="tau"/>
                    <xsd:element name="deltatau"/>
                    <xsd:element name="deltaunit"/>
                    <xsd:element name="starttime"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="T_ptrans">
                <xsd:sequence maxOccurs="unbounded">
                    <xsd:element name="pt">
                        <xsd:complexType>
                            <xsd:sequence>
                                <xsd:element name="ptid"/>
                                <xsd:element name="ptime"/>
                                <xsd:element name="pvalue"
type="xsd:anySimpleType"/>
                            </xsd:sequence>
                        </xsd:complexType>
                    </xsd:element>
                </xsd:sequence>
        </xsd:complexType>
</xsd:schema>
```

8.2 Model průchodu studenta výukovým systémem

8.2.1 GLIKREM

Modelová situace bude v GLIF modelu znázorněna jako čtyři paralelní větve (viz obrázek 31), kde jedna větev představuje studium sedmi kapitol (k_i), druhá větev absolvování 13-ti cvičení (c_i), třetí větev splnění samostatného projektu (p) a poslední větev absolvování zápočtového testu (t). Tečkované čáry znamenají opakování téže struktury, tj. 7 krát studium kapitoly a 13 krát absolvování cvičení.



obrázek 31 Znalostní model (GLIKREM) průchodu studenta výukou

- **Studium kapitoly (k_i).** Student opakuje studium kapitoly k_i dokud neodpoví správně na kontrolní otázky, tj. dokud není *strict-in* kritérium větve κ_{i1} pravdivé.
- **Absolvování cvičení (c_i)**. Každého cvičení c_i se student buď zúčastní (pravdivé *strict-in* kritérium σ_{i1}) nebo nezúčastní (pravdivé *strict-in* kritérium σ_{i2}). Účast (neúčast) je zaznamenána vedoucím cvičení.
- **Splnění projektu (p)**. Student splní projekt v případě, že je splněno *strict-in* kritérium hrany π_1 nebo π_3. Kritéria jsou definována následovně $\pi_1^{SI} = hodnocení(p) \geq 0{,}7$ a $\pi_3^{SI} = hodnocení(p) \geq 0{,}7$. *Strict-in* kritéria větví π_2 a π_4 jsou negací *strict-in* kritérií hran π_1 a π_3.
- **Absolvování testu t**. Student úspěšně absolvuje zápočtový test v případě, že je splněno *strict-in* kritérium hrany τ_1 nebo τ_3. Kritéria jsou definována následovně $\tau_1^{SI} = hodnocení(t) \geq 0{,}6$ a $\tau_3^{SI} = hodnocení(t) \geq 0{,}6$. *Strict-in* kritéria větví τ_2 a τ_4 jsou negací *strict-in* kritérií hran τ_1 a τ_3.
- **Zápočet**. Zápočet student získá, jestliže je splněno *strict-in* kritérium hrany ψ_1, tzn. student splní studium všech kapitol (k_i), absolvuje minimálně 70% cvičení c_i, splní projekt p a úspěšně absolvuje test t. *Strict-in* kritérium hrany ψ_2 je negací *strict-in* kritéria hrany ψ_1.

Všechna kritéria *strict-out* jsou ve všech případech negací *strict-in* kritérií. Kritéria *rule-in* a *rule-out* nejsou v tomto modelu použita vůbec.

8.2.2 Paramodel

V modelové situaci byl stanoven model parametrů (paramodel) podle následující tabulky.

tabulka 3 Průchod studenta výukou - model parametrů (paramodel)

pid	ptype	pdatype	pname	punits	pdef
KAP1	basic	boolean	Kapitola 1	-	-
…	…	…	…	…	…
KAP7	basic	boolean	Kapitola 7	-	-
SKAP	derived	int	Kapitoly	-	/params/param[pid="KAP1"]/pvalue **and** /params/param[pid="KAP2"]/pvalue **and … and** /params/param[pid="KAP7"]/pvalue
CV1	basic	int	Cvičení 1	-	-
…	…	…	…	…	…
CV13	basic	int	Cvičení 13	-	-
DOCH	derived	int	Docházka	%	**((** /params/param[pid="CV1"]/pvalue + /params/param[pid="CV2"]/pvalue + **…** + /params/param[pid="CV13"]/pvalue **)** *100) **div** 13
PROJ	basic	boolean	Projekt	-	-
ZT	basic	int	Záp. test	%	-
OZT	basic	int	Opravný záp. test	%	-

Hodnoty (<pvalue>) u parametrů KAP1 až KAP7 nabývají hodnoty pravda (1), jestliže student úspěšně absolvuje všechny dílčí testy u každé kapitoly. Parametr SKAP je pak konjunkcí všech hodnot KAP1 až KAP7. Hodnoty CV1 až CV13 jsou rovny 1 v případě, že student absolvuje příslušné cvičení, jinak jsou rovny nule. Parametr DOCH je pak procentuální účast na všech cvičeních (CV1 až CV13). Parametr PROJ obsahuje hodnotu 1, jestliže student splní odevzdání samostatného projektu, jinak je roven nule. Parametry ZT a OZT obsahují procentuální úspěšnost studenta u zápočtového testu (ZT) resp. u opravného zápočtového testu (OZT). Student je úspěšný u testu, jestliže alespoň jedna z hodnot ZT nebo OZT je větší nebo rovna 60.

Reálná data studentů modelové situace jsou v systému Moodle ČZU uložena v relační databázi MySQL. Pro propojení modelu parametrů (paramodelu) a dat uložených v systému Moodle ČZU je využito datových struktur stávajících modulů docházka, úkol a test.

Navržené datové rozhraní (hodnotová mapa) mezi modelem průchodu studenta a skutečnými daty v systému Moodle je popsáno v následující tabulce. V tabulce jsou uvedeny pouze základní parametry (typ basic), odvozené parametry nemají přímé napojení na databázi Moodle ČZU.

Při výběru hodnot pro parametry KAP1 až KAP7, ZT a OZT je vybrán identifikátor testu (id) v příslušném kurzu (course) z tabulky quiz. Na základě identifikátoru jsou pak v tabulce quiz_grades vybrány všechny záznamy, resp. hodnocení testu grade, kde quiz = id a to pro studenta s identifikátorem userid.

Při výběru hodnot pro parametry CV1 až CV13 jsou v tabulce attendance_log vybrány záznamy s identifikátorem modulu docházka (sessionid) v kurzu s identifikátorem courseid, pro studenta s identifikátorem studentid. Přítomnost studenta na cvičení je splněna (a hodnocena hodnotou grade), jestliže hodnota statusid odpovídá acronymu P (přítomnost).

tabulka 4 Průchod studenta výukou - datové rozhraní (hodnotová mapa)

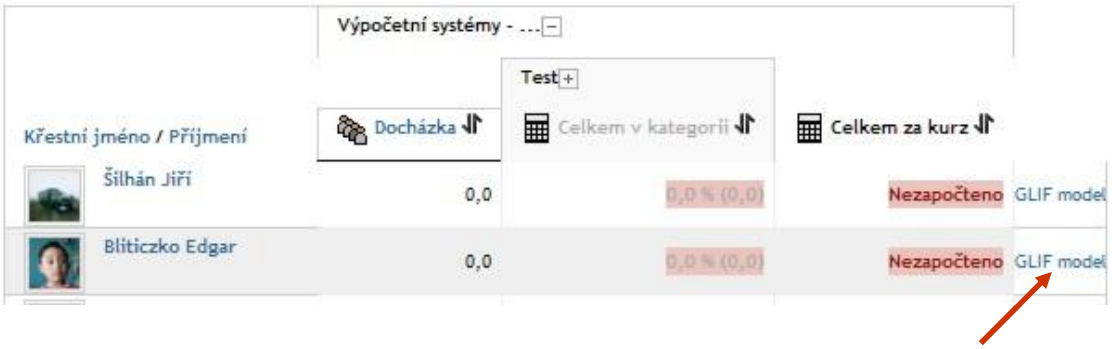
pid	**modul**	**tabulka**	**atribut(y)**
KAP1	quiz	quiz_grades; quiz	quiz, userid, grade; id, course
…	…	…	…
KAP7	quiz	quiz_grades; quiz	quiz, userid, grade; id, course
CV1	attendance	attendance_log; attendance_sessions; attendance_statuses	sessionid, studentid, statusid; id, courseid; id, acronym, grade
…	…	…	…
CV13	attendance	attendance_log; attendance_sessions; attendance_statuses	sessionid, studentid, statusid; id, courseid; id, acronym, grade
PROJ	assignment	assignment_submissions; assignment	assignment, userid, grade; id, course
ZT	quiz	quiz_grades; quiz	quiz, userid, grade; id, course
OZT	quiz	quiz_grades; quiz	quiz, userid, grade; id, course

Pro výběr hodnot pro parametr PROJ jsou v tabulce assignment_submissions vybrány záznamy, kde hodnota assigment se rovná hodnotě id (z tabulky assignment) pro kurz s identifikátorem course. Pro studenta s identifikátorem userid se pak bere hodnocení úkolu grade.

8.2.3 Implementace

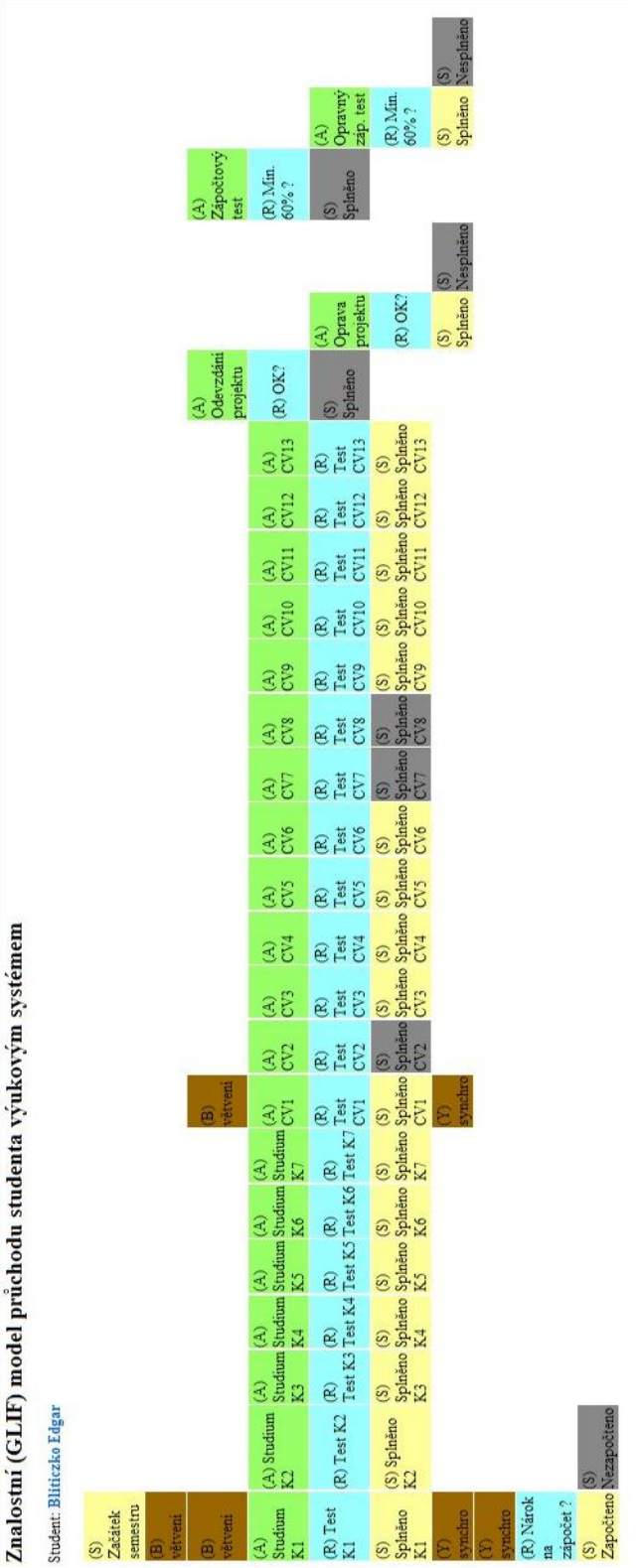
Pilotní modul reprezentace modelu průchodu studenta výukovým systémem byl vytvořen ve skriptovacím jazyku PHP (*Personal Home Pages*) a implementován do e-learningového systému Moodle ČZU jako rozšíření stávajícího modulu Klasifikace (*Grade*).

V seznamu celkového hodnocení studentů v kurzu je u každého studenta možnost zobrazit jeho aktuální pozici ve znalostním modelu (viz obrázek 32).



obrázek 32 Integrace pilotního modulu do modulu Klasifikace

Aktuální pozice je zobrazena ve formě zjednodušeného grafu, ve kterém jsou barevně zobrazeny splněné (prošlé) větve grafu a šedě dosud nesplněné (neprošlé) větve (viz obrázek 33). V zobrazeném příkladu student splnil všechny dílčí testy u kapitol 1 až 7, zúčastnil se všech cvičení až na cvičení č.2, 7 a 8 (celková docházka ale větší než 70%), projekt splnil až po opravě a zápočtový test úspěšně absolvoval v opravném termínu. V celkovém hodnocení má student nárok na zápočet.



obrázek 33 Znalostní model vybraného studenta