VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

DIAGNOSTICKÝ EXPERTNÍ SYSTÉM

DIAGNOSTIC EXPERT SYSTEM

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Michal Krechler

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Václav Jirsík, CSc.

SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Michal Krechler ***ID:*** 78222

Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Diagnostický expertní systém

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s problematikou diagnostických expertních systémů.
2. Zaměřte se na problematiku realizace expertních systémů na internetu.
3. Realizujte expertní systém NPS32 tak, aby umožňoval konzultaci v prostředí www.
4. Proveďte test funkčnosti realizace a rozbor dosažených výsledků.

DOPORUČENÁ LITERATURA:

Popis programu NPS32.

Termín zadání: 6. 2. 2017 ***Termín odevzdání:*** 15.5.2017

Vedoucí práce: doc. Ing. Václav Jirsík, CSc.

Konzultant: Ing. Petr Petyovský

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta elektrotechniky a komunikačních technologií
Ústav automatizace a měřicí techniky

UPOZORNĚNÍ:

Autor diplomové práce práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Abstrakt

Diplomová práce se zabývá návrhem a tvorbou nové verze výpočetního jádra expertního systému vycházejícího z expertního systému NPS32. Dále se práce zabývá možnostmi a podmínkami provozu expertního systému na internetu. Část práce se věnuje testování navrženého SW a demonstraci jeho možností.

Klíčová slova

Expertní systém, NPS32, C++, Apache, Web Server, PHP, Testování SW

Abstract

The master's theses deal with concept and creation of a new version of the compute kernel in expert system based on NPS32. This paper is focused on possibilities and requirements of the expert system's operation on the Internet, too. One part of the study is devoted to testing the designed SW and demonstrating of its facilities.

Keywords

Expert System, NPS32, C++, Apache, Web Server, PHP, software testing

Bibliografická citace:

KRECHLER, M. *Diagnostický expertní systém*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 73s. Vedoucí práce: doc. Ing. Václav Jirsík, CSc.

Prohlášení

„Prohlašuji, že svou závěrečnou práci na téma Diagnostický expertní systém jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.“

V Brně dne **15. května 2017** ……
podpis autora

Poděkování

Děkuji vedoucímu diplomové práce doc. Ing. Václavu Jirsíkovi, CSc. a konzultantovi Ing. Petru Petyovskému, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne **15. května 2017** ……

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 ÚVOD

Expertní systémy jsou jedním z druhů umělé inteligence. Jsou schopné zaznamenat znalosti experta a transformovat je do takové podoby, kdy uživatel může bázi znalostí využívat pro své potřeby bez nutnosti konzultace s expertem. Této vlastnosti se využívá v celé řadě oborů – od lékařství, geologie, zemědělství až po různé servisní nástroje a mnoho dalších. Hlavní výhodou expertních systémů pak je možnost specializace na některou z oblastí, pro kterou není dostatek kvalifikovaných expertů, nebo je příliš rozsáhlá a výchova experta pro tuto oblast není ekonomická, avšak je potřebná.

V současné době navíc dochází k pomalému přesunu aplikací z prostředí lokálního počítače do prostředí webových aplikací. Je to zapříčiněno především jednodušší správou verzí dané aplikace a snadnější dostupností z různých moderních komunikačních prostředků, jako jsou tablety či mobilní telefony, které mnohdy obsahují různé operační systémy (Android, iOS, Windows atd.). Dále je v prostředí webových aplikací snazší správa licencí pro přístup k danému software, kdy odpadá pirátské kopírování a pozměňování aplikace. Ovšem samozřejmě pokud nebudeme počítat náhodnou nebo úmyslnou ztrátu přihlašovacích údajů k dané aplikaci.

V souvislosti s tímto trendem je pochopitelné, že i aplikace expertních systémů se postupně přesunou do prostředí webových aplikací. Díky tomu je možné mít k dispozici experta na danou problematiku neustále on-line v případě potřeby. Tento přesun však nikdy nebude možné udělat tak, že konzultaci s expertním systémem bude provádět kdokoliv. Stejně jako v reálném životě musí mít uživatel o dané problematice odpovídající míru vlastních znalostí tak, aby byl schopný zodpovědně vést konzultaci – odpovídat na otázky expertního systému. Z tohoto důvodu bude vždy odpovědnost za výsledek konzultace ležet minimálně z části na uživateli a jeho schopnostech.

V rámci této práce se upraví výpočetní jádro expertního systému NPS32 a připraví se pro nasazení v rámci webových aplikací. V teoretické části je popsán matematický aparát a syntaxe báze znalostí původního expertního systému.

V praktické části se tato práce poté zabývá zapracováním požadavků na výpočetní systém a jeho popis. V poslední kapitole se poté tato práce zabývá ověřením funkčnosti a nových vlastností nové aplikace.

2 TEORETICKÁ ČÁST

V této části práce je rozebrána základní definice expertních systémů, jejich vlastnosti a základní části. Následuje přehled systému NPS32 včetně souhrnu matematického aparátu a přehled inferenčního mechanismu.

Dále se práce zabývá podmínkami a prostředky pro tvorbu webových aplikací. Poslední část této kapitoly je věnována testování software.

2.1 Teoretický úvod k expertním systémům

Expertní systémy spadají do kategorie systémů umělé inteligence. Jejich hlavní úlohou je shromáždit specifické informace a znalosti z dané oblasti o nějakém problému, tyto informace zpracovat a poskytnout je uživateli v takové formě a takovým způsobem, že expertní systém může zastoupit experta (znalce) na danou oblast či problematiku.

Expertem (znalcem) je člověk, který disponuje znalostmi z oblasti, jež jsou z nějakého důvodu cenné a žádoucí. Typickými oblastmi, kde je potřeba využít znalosti experta, může být například diagnostika, správná interpretace výsledků, vyhledávání, učení a další. Pro detailnější přehled teorie expertních systémů je vhodná například kniha [1].

2.1.1 Vlastnosti expertních systémů

Základními výhodami expertních systémů – na rozdíl od konzultace s odborníkem – jsou:

Dostupnost – informace jsou v jeden okamžik dostupné většímu množství lidí, navíc jsou dostupné kdykoliv.

Opakovatelnost – stejnou konzultaci je možné provést i s větším odstupem času.

Dohledatelnost – většinou bývá součástí expertního systému i forma protokolu, kde se dá přesně doložit, na základě jakých podkladů se dospělo k řešení.

Nestrannost – každý, kdo projde konzultaci, dostane stejné výsledky a informace.

Nevýhodou expertních systémů naopak je:

Úzká oblast znalostí – zaměřují se pouze na specifickou malou část oboru. Navíc je pro snížení složitosti často tendence systém navrhnout pouze na nekompletní (majoritní) část cílových hypotéz.

Nutnost definovat znalosti – podmínkou tvorby expertního systému je, že znalosti z dané problematiky lze popsat a strojově zpracovat. Mezi znalosti tedy nelze započítat zkušenost experta umožňující využití intuice či pocitu.

Z těchto důvodů jsou expertní systémy stále pořád spíše doplňkovým nástrojem, kdy obsahují specifické detailní informace na danou problematiku, ale výsledky konzultace stejně nějakým způsobem vyhodnocuje osoba, která využila konzultace s expertním systémem.

2.1.2 Základní části expertního systému

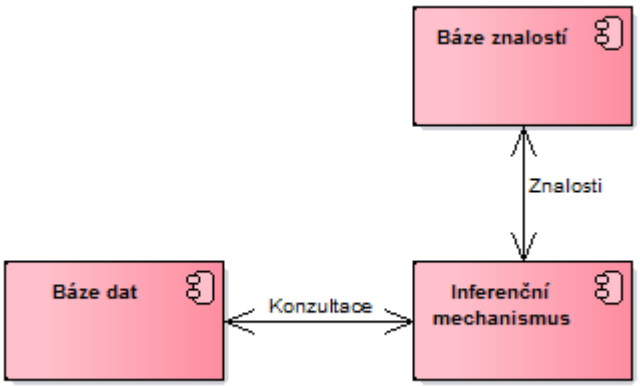
V průběhu času se u expertních systémů utvořily tři základní části:

Báze znalostí – obsahuje samotné specifické znalosti z oblasti, kterou se navrhovaný expertní systém zabývá. V současnosti se tvorbou báze znalostí zabývá obor znalostní inženýrství (knowledge engineering). Pro zpracování báze znalostí se užívají matematická pravidla, sémantické či neuronové sítě a další vhodné prvky z oblasti umělé inteligence.

Báze dat – obsahuje dvě základní složky:

- předem daná data závislá na neměnných či málo měnících se podmínkách.
- informace vstupující do inferenčního mechanismu v průběhu konzultace od uživatele.

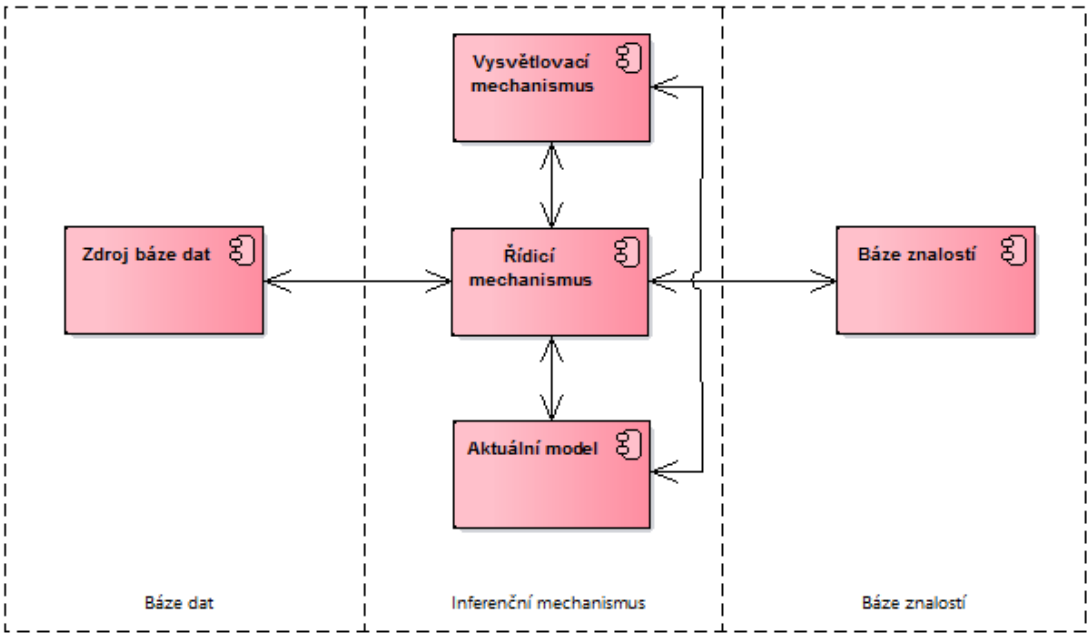
Inferenční mechanismus – slouží k samotnému řízení konzultace, kdy na základě báze znalostí a báze dat vyhodnocuje jednotlivé hypotézy a rozhoduje o dalším postupu konzultace.



Obrázek 1 Základní prvky expertního systému

2.1.3 Základní kategorie expertních systémů

Diagnostický expertní systém – úkolem diagnostického expertního systému je na základě vstupních informací porovnávat a určovat, které z předem definovaných cílových hypotéz jsou nejvhodnější.



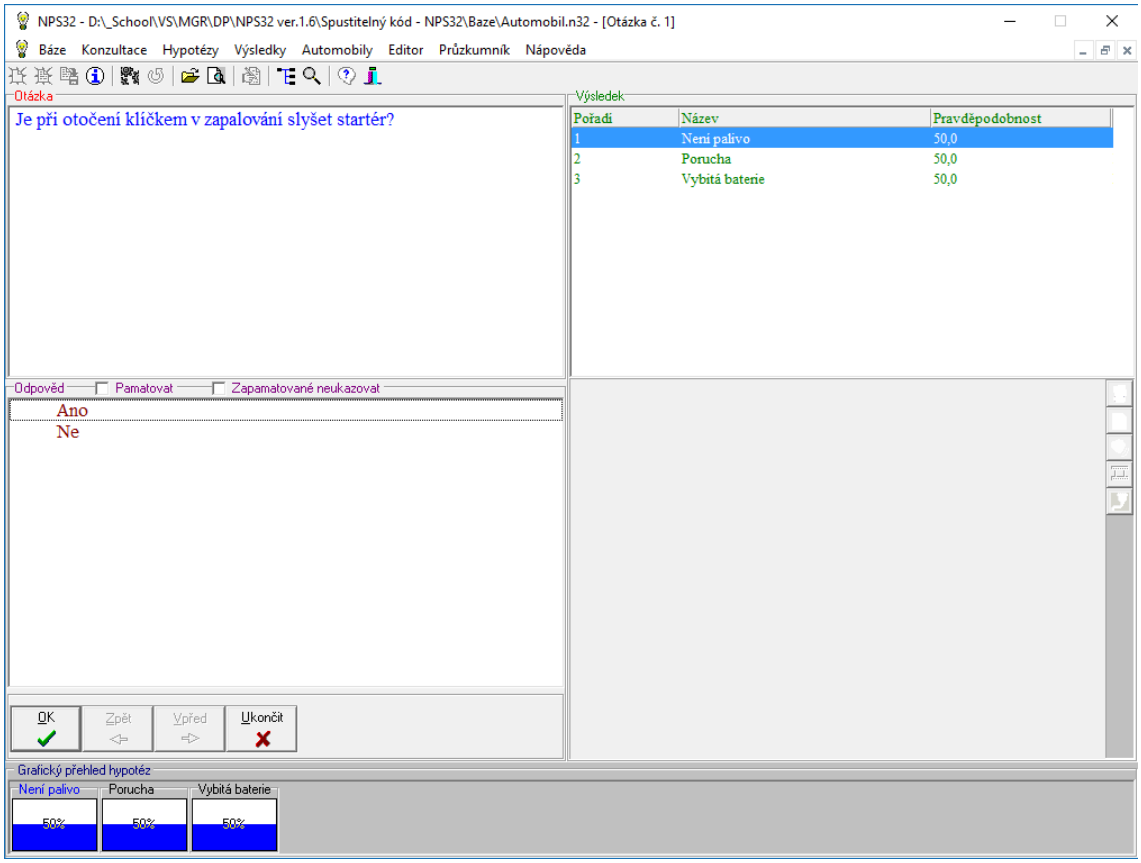
Obrázek 2 Základní prvky diagnostického expertního systému

Plánovací expertní systém – prakticky jde o inverzní úlohu k diagnostickým systémům, kdy je zadán cíl, kterého se má dosáhnout, a plánovací expertní systém má za úkol v průběhu konzultace vytvořit posloupnost kroků, jak daného cíle dosáhnout.

Specifické expertní systémy – jde o zbývající expertní systémy, které nelze zahrnout do předchozích kategorií.

2.2 Expertní systém NPS32

Na VUT Brno fakultě FEI následně FEKT je již řadu let vyvíjen diagnostický expertní systém NPS32 a jednotlivé báze znalostí pro tento systém. V následující kapitole je popis základních prvků a matematického aparátu tohoto systému. Tato práce vychází z předchozích prací např.: [2]:



Obrázek 3 Hlavní okno konzultace systému NPS32

2.3 Definice pojmů v expertním systému NPS32

V této kapitole jsou popsány základní pojmy používané při popisu expertního systému NPS32.

Pravděpodobnost – Systém NPS32 pracuje s pravděpodobnostmi. Pravděpodobnost se používá pro vyjádření míry důvěry v řešení konzultace cílovou hypotézou. Pro vyjádření výsledku je použita pravděpodobnost v rozsahu 0...100 %. Interně je v bázi znalostí pro výpočty použita pravděpodobnost vyjádřená dvojicí hodnot (T, F).

Hypotéza – jde o druh uzlu, který odpovídá některému z možných hledaných řešení během konzultace.

Báze znalostí – Reprezentuje znalosti z určité problematiky v takové formě, aby umožňovala strojové zpracování a průběh konzultace. Pro NPS32 jde o soubor obsahující definice uzlů, jejich atributů a vazeb mezi nimi.
Uzel – Báze znalostí je realizována uzly. Každý uzel má jedinečný identifikátor – Literál. Vedle identifikátoru má každý uzel pravděpodobnost, která vyjadřuje významnost daného uzlu v průběhu konzultace.
Literál – Každý uzel má jedinečný identifikátor, který je využíván v bázi znalostí pro definování vazby mezi jednotlivými uzly.
Uživatel – klient expertního systému, který provádí konzultaci pomocí odpovědí na položené dotazy.
Vazba – je realizována buď pravidlem, nebo kontextem. Zajišťuje šíření informací mezi uzly. Vazba je vždy spojena s uzlem a ovlivňuje hodnotu pouze tohoto uzlu.
Pravidlo – pomocí konjunkce nebo disjunkce mezi jednotlivými uzly zajišťuje základní šíření informací v bázi znalostí.
Kontext – je součástí pouze dotazovatelných uzlů. Kontext definuje podmínku závislou na ostatních uzlech, která určuje, zda bude uzel při konzultaci položen či nikoliv. Pokud podmínka nejprve negativní a poté je v průběhu konzultace dodatečně splněna, je dotaz položen.
Konzultace – proces plnění báze znalostí informacemi, které jsou získány pomocí pokládání dotazů uživateli. Informace jsou přes bázi znalostí distribuovány až k cílovým hypotézám. Na konci konzultace je žádoucí mít co nejmenší množinu cílových hypotéz s co nejvyšší pravděpodobností a u zbývajících hypotéz naopak pravděpodobnost co nejnižší.

2.4 Popis matematického aparátu systému NPS32

(Čerpáno z [2]) Matematický aparát systému NPS32 pracuje s pravděpodobností jednotlivých uzlů v rozsahu <0,100> [%] (P) případně <0,1> [-] (p). Tyto pravděpodobnosti jsou však interně vyjádřeny v složeném formátu pomocí dvou hodnot (T, F). Kombinací dvou hodnot T a F je možné vyjádřit, vedle samotné pravděpodobnosti, i spor a nepřítomnost jakékoliv informace.
Výraz (1-T) vyjadřuje míru důvěry v pravdivost tvrzení.
Výraz (1-F) vyjadřuje míru důvěry v nepravdivost tvrzení.
Dále pak tedy následující kombinace hodnot (T, F) znamenají:

(0,1) odpovídá pravdivému tvrzení
(1,0) odpovídá nepravdivému tvrzení
(1,1) znamená nepřítomnosti informace

(0,0) znamená spor v bázi znalostí nebo odpovědích uživatele.

Pro přepočet poté platí následující vztah:

$$T = \frac{F(1-p)}{p} \quad resp: F = \frac{pT}{1-p} \tag{2.1}$$

kde:

p je pravděpodobnost $p \in < 0, 1 >$

T, F jsou složky pravděpodobnosti $T, F \in < 0, 1 >$.

Hodnoty T, F se volí dle následujícího pravidla:
Pokud je p < 0,5, volí se T = 1 a F se dopočítá dle vztahu (2.1). Pokud je p ≥ 0,5, volí se F = 1 a T se dopočítá. Dále jsou definovány následující operace:

Unární operace:
Operace negace označená symbolem „-“:

$$-(T, F) = (F, T) \tag{2.2}$$

Binární operace:
Operace konjunkce označená symbolem „&“:

$$(T1, F1)\&(T2, F2) = (max(T1, T2), min(F1, F2)) \tag{2.3}$$

Operace disjunkce označená symbolem „|“:

$$(T1, F1)|(T2, F2) = (min(T1, T2), max(F1, F2)) \tag{2.4}$$

Operace skládání s odpovědí uživatele označená „·“:
Tento vztah vyjadřuje vliv odpovědi uživatele na pravděpodobnost zodpovězeného uzlu.

$$(T_1, F_1) = (T_0, F_0) \cdot (T_{odp}, F_{odp}) = (T_0 T_{odp}, F_0 F_{odp}) \tag{2.5}$$

kde:

(T_0, F_0) je původní pravděpodobnost uzlu
(T_1, F_1) je nová pravděpodobnost uzlu
(T_{odp}, F_{odp}) je pravděpodobnost odpovědi.

Operace skládání silou vazby označená symbolem „$\overset{\alpha}{\rightarrow}$“:
Tato operace popisuje vliv změny pravděpodobnosti uzlu na všechny zbývající uzly, které jsou s tímto uzlem svázány. Pomocí této vazby tedy dochází k šíření informací mezi uzly. Skládání silou vazby je realizováno vztahem:

$$(T_A, F_A) \overset{\alpha}{\rightarrow} (T_B, F_B) = \big(\big((1-\alpha) + \alpha T_A\big)T_B, \big((1-\alpha) + \alpha F_A\big)F_B\big) \quad (2.6)$$

kde:
(T_A, F_A) je pravděpodobnost zdrojového (výchozího) uzlu
(T_B, F_B) je pravděpodobnost ovlivněného (cílového) uzlu
α je síla vazby $\alpha \in < 0, 1 >$.

Vztah *(2.6)* je pro výpočet v programu možné nahradit upraveným vztahem *(2.7)*, který umožňuje získat hodnoty pravděpodobnosti i v průběhu konzultace.

$$(T_{B1}, F_{B1}) = \left(\frac{T_{A0}}{T_{A1}}, \frac{F_{A0}}{F_{A1}}\right) \overset{\alpha}{\rightarrow} (T_{B0}, F_{B0}) \quad (2.7)$$

$$T_{B1} = \left(\frac{\alpha T_{A1} - \alpha + 1}{\alpha T_{A0} - \alpha + 1}\right) T_{B0}$$

$$F_{B1} = \left(\frac{\alpha F_{A1} - \alpha + 1}{\alpha F_{A0} - \alpha + 1}\right) F_{B0}$$

kde:
(T_{A0}, F_{A0}) je pravděpodobnost zdrojového (výchozího) uzlu před změnou
(T_{A1}, F_{A1}) je pravděpodobnost zdrojového (výchozího) uzlu po změně
(T_{B0}, F_{B0}) je pravděpodobnost ovlivněného (cílového) uzlu před změnou
(T_{B1}, F_{B1}) je pravděpodobnost ovlivněného (cílového) uzlu po změně.

Hodnota neurčitosti dané pravděpodobnosti se určí dle vztahu:

$$Q = TF \quad (2.8)$$

Hodnota pravděpodobnostní šance se určí dle vztahu:

$$O = \frac{F}{T} = \frac{p}{1-p} \quad (2.9)$$

2.5 Inferenční mechanismus

Základním stavebním prvkem inferenčního mechanismu je uzel. Každý uzel je reprezentován identifikátorem a hodnotou pravděpodobnosti. Přenos informací mezi uzly zajišťují vazby.

2.5.1 Popis uzlů

Inferenční mechanizmus systému NPS32 pracuje s následujícími typy uzlů:
Pomocný uzel – obsahuje pouze identifikátor, hodnotu pravděpodobnosti, a může obsahovat pravidla vazeb na jiné uzly.
Cílový uzel – oproti pomocnému uzlu obsahuje navíc popis cílové hypotézy.
Dotazovatelný přímý uzel – u tohoto typu uzlu je navíc definována otázka, která se v průběhu konzultace položí uživateli a po jejím zodpovězení se změní hodnota pravděpodobnosti pomocí vztahu (2.5). Pro tento typ uzlu se hodí otázky typu „výběr míry souhlasu s otázkou“, kdy uživatel vybere prostřednictvím odpovědi hodnotu pravděpodobnosti, do jaké míry s tvrzením v otázce souhlasí. Otázka se v průběhu konzultace pokládá pouze tehdy, pokud je kontext kladný. Po zodpovězení otázky je aktualizována hodnota všech uzlů, které závisí dle pravidel na zodpovězeném uzlu.
Dotazovatelný přímý kvantitativní uzel – tento dotazovatelný uzel slouží k položení otázky typu „výběr jedné možnosti ze seznamu“. Po zvolení odpovědi uživatelem je automaticky vždy přiřazena uzlu odpovídajícímu zvolené odpovědi hodnota 100 % a ostatní uzly odpovídající otázkám zůstanou nezměněny. Otázka se v průběhu konzultace pokládá pouze tehdy, pokud je kontext kladný. Po zodpovězení otázky jsou aktualizovány hodnoty všech uzlů dle vztahu (2.10).
Dotazovatelný běžný uzel – tento typ uzlu nemá možnost definovat vlastní kontexty ani pravidla. Na rozdíl od dotazovatelného přímého uzlu se tento typ pokládá až v momentě, kdy byly zodpovězeny všechny vhodné dotazovatelné přímé uzly. V tento moment se z pravidel všech cílových uzlů vybere nezodpovězený běžný uzel, který má největší sílu vazby a jde o vhodnou vazbu.

Vhodná vazba se určí tak, že předchozí hodnota pravděpodobnosti vazby je větší než aktuální hodnota pro disjunkci, respektive menší pro konjunkci.

Každý dotazovatelný uzel může mít atribut „Speciální“; tento atribut ovšem nemá v současnosti v inferenčním mechanismu implementovánu jakoukoliv funkci.

2.5.2 Vazby mezi uzly

Vazby mezi uzly jsou realizovány pomocí pravidel nebo kontexty.

Pravidla – zajišťují základní šíření informací mezi uzly a změny pravděpodobností dle zodpovězených otázek. Pravidla jsou uložena u uzlu a definují vliv pravděpodobnosti jiného uzlu na pravděpodobnost tohoto uzlu. Pravidlo může být buď konjunkční nebo disjunkční. Každý uzel může obsahovat libovolné množství pravidel.

Pro šíření informace mezi uzly pomocí pravidel je použit vztah:

$$(T,F)_B^{[k]} = \left(\frac{T^{[k-1]}}{T^{[k]}}, \frac{F^{[k-1]}}{F^{[k]}}\right)_n \overset{\alpha}{\rightarrow} (T,F)_B^{[k-1]} \qquad (2.10)$$

kde:

$(T,F)_B^{[k]}$ je pravděpodobnost uzlu obsahujícího pravidlo n v k-tém přepočítávání báze znalostí.

n je index pravidla z celkového souboru všech pravidel definovaných pro daný uzel.

Hodnota pravděpodobnosti pravidla *n* se určí dle vztahu:

$$pro\ konjunkci: (T,F)_n = \left((-)_{(0)}(T,F)_{(0)}\right)\& \ldots \&((-)_{(m)}(T,F)_{(m)}) \qquad (2.11)$$
$$pro\ disjunkci: (T,F)_n = \left((-)_{(0)}(T,F)_{(0)}\right)| \ldots |((-)_{(m)}(T,F)_{(m)})$$

kde:

$(T,F)_n$ je výsledná pravděpodobnost vazeb pravidla *n*

m je počet uzlů pravidla *n*.

Dle definice v bázi znalostí se případně použije negovaná hodnota pravděpodobnosti příslušného uzlu.

Kontexty – definují podmínky, za kterých se má otázka uzlu během konzultace položit. Opět jsou definovány u uzlu, jenž se má položit. Kontext obsahuje vazbu na ostatní uzly, kterou vyhodnocuje tak, že porovnává aktuální hodnoty pravděpodobnosti uzlů vazby s definovaným prahem a pokud všechny kontextové vazby vyhovují podmínce, je možné tento uzel položit při konzultaci. Jsou dva druhy kontextových prahů – větší než „>“ nebo menší než „<“. Pro porovnání se vždy používá hodnota pravděpodobnosti v procentech.

2.5.3 Externí načítání odpovědí

Každý dotazovatelný uzel může mít definovaný seznam pravidel, na jejichž základě inferenční mechanismus sám určí odpověď a otázka se během konzultace uživateli nepoloží. Systém NPS32 má definovány dva typy takovýchto pravidel:

Externí textové odpovědi – porovnávají text odpovědi uložený v pomocném souboru s textem některého z nadefinovaných pravidel. V případě, že najde první shodu, je zvolena příslušná odpověď. Porovnávání není závislé na velikosti písmen.

Externí číselné odpovědi – inferenční mechanismus načte číslo z pomocného souboru a porovná jej postupně s definovanými pravidly. V případě, že najde první podmínku, které toto číslo vyhovuje, je zvolena příslušná odpověď. Pro externí číselné odpovědi jsou možné následující podmínky:

Interval – podmínka je splněna, pokud číslo leží v intervalu ohraničeném spodní a horní mezí.

Spodní mez – podmínka je splněna, pokud je číslo větší než spodní mez.

Horní mez – podmínka je splněna, pokud je číslo menší než horní mez.

Oba typy externích odpovědí mohou obsahovat navíc „bezpodmínečnou“ volbu odpovědi. V takovém případě se rovnou zvolí odpověď dle indexu a inferenční mechanismus pokračuje v dotazování další otázkou.

2.6 Formát báze znalostí *.n32

Báze znalostí je uložena v dokumentu s příponou *.n32. Aby bylo možné tento soubor editovat, je uložen v textové podobě, ovšem jeho formát je poměrně striktně kontrolován. Obsah báze znalostí je rozdělen do dvou částí. První částí je hlavička báze znalostí. Druhou částí je Definice uzlů a vazeb mezi nimi.

2.6.1 Obecná část báze znalostí *.n32

Hlavička báze znalostí obsahuje následující informace:

Název báze znalostí – každá báze znalostí může obsahovat až jeden tento řádek obsahující text názvu báze znalostí. Formát tohoto řádku je:

```
";.{nazev}\r\n"
```

(pozn: bez uvozovek, znaky \r\n odpovídají konci řádku. Text uvedený v závorkách {} obsahuje libovolný text)

Popis báze znalostí – obsahuje textový popis báze znalostí. Tento řádek je nepovinný a může se opakovat maximálně 255krát. Formát tohoto řádku je:

```
";;{text}\r\n"
```

Počet implicitně nabízených odpovědí – pokud není v bázi znalostí definováno jinak, je uživateli během konzultace nabídnuta množina možných odpovědí v rozsahu definovaném tímto parametrem. Možné hodnoty jsou 3, 5 nebo 7. Pokud není tento parametr definován, je výchozí hodnota nastavena na 7 odpovědí. Možné hodnoty implicitních odpovědí a příslušné pravděpodobnosti jsou uvedeny v Tab. 2.1 Implicitní hodnoty odpovědí
Formát tohoto řádku je:

```
";?{357}\r\n"
```

Tab. 2.1 Implicitní hodnoty odpovědí

Text odpovědi	Počet odpovědí					
	3		5		7	
Ano	100 %	+1	-	-	-	-
Určitě ano	-	-	100 %	+2	100 %	+3
Spíše ano	-	-	-	-	75 %	+2
Snad ano	-	-	66.7 %	+1	60 %	+1
Nevím	50 %	0	50 %	0	50 %	0
Asi ne	-	-	33.3 %	-1	40 %	-1
Spíše ne	-	-	-	-	25 %	-2
Určitě ne	-	-	0 %	-2	0 %	-3
Ne	0 %	-1	-	-	-	-

Konec báze znalostí – Báze znalostí je ukončena znakem "#"

2.6.2 Definice uzlů báze znalostí *.n32

V této části jsou definovány všechny zbývající informace o bázi znalostí. Tato část začíná tehdy, když se pro čtení báze znalostí nalezne první definice uzlu. Od tohoto okamžiku již není možné uvést některou z obecných informací o bázi znalostí popsaných v kapitole 2.6.1.

V bázi znalostí musí být jednotlivé uzly seřazeny tak, aby byla nejprve uvedena jejich definice a až poté se na ně jiné uzly odkazovaly. Zpravidla jsou nejprve uvedeny dotazovatelé uzly, poté pomocné uzly a nakonec cílové hypotézy.

Uzel – definice každého uzlu má následující tvar:

```
".{id} p={pr}% {typ}\r\n"
```

kde:

`id` je unikátní identifikátor uzlu s délkou 1…8 znaků mezi identifikátorem a znaky `p=` je mezera. Znaky `p=` jsou tedy vždy 10. a 11. znak na řádku.

`pr` je hodnota pravděpodobnosti uzlu na začátku konzultace. Jde o dvoucifernou hodnotu v procentech. Rozsah je tedy 00..99 %.

`typ` definuje typ uzlu
- `""` – pomocný uzel
- `"G"` – hypotéza (cílový)
- `"D"` – dotazovatelný přímý
- `"DK"` – dotazovatelný přímý kvantitativní
- `"A"` – dotazovatelný běžný

Každý dotazovatelný uzel může navíc mít atribut `"S"` označující tento uzel jako speciální.

Komentář uzlu – tato část obsahuje samotný text otázky zobrazený při konzultaci. Řádek je nepovinný a může se opakovat maximálně 255krát. Formát tohoto řádku je:

```
";{text}\r\n"
```

Explicitně definované pravděpodobnostní odpovědi – pokud se pro text odpovědi na dotazovatelný uzel nedá použít některá ze skupin implicitních odpovědí uvedených v Tab. 2.1 Implicitní hodnoty odpovědí je možné definovat vlastní odpovědi s vhodnou pravděpodobností pomocí tohoto řádku. V případě, že je u uzlu definována alespoň jedna explicitní odpověď, jsou automaticky nabízeny již jen uživatelské odpovědi. Maximální počet uživatelských odpovědí je 99. Formát tohoto řádku je:

```
";*{text}[{pp}]*\r\n"
```

kde:

`text` je text uživatelské odpovědi

`pp` je hodnota pravděpodobnosti uživatelské odpovědi ve formátu:

1. 00, 01 .. 99, ++ (++ odpovídá hodnotě 100 %)
2. -3, -2, … +2, +3 dle Tab. 2.1 Implicitní hodnoty odpovědí

Dotazy kvantitativního uzlu – U kvantitativního uzlu jsou otázky definovány ve formě dvojic: otázka – ovlivněný uzel. U textu otázky je použit následující formát řádku:

```
";*{text}[{id}]*\r\n"
```

kde:

`text` je text odpovědi

`id` je pořadové číslo následujících uzlů v rozsahu 1 … 99.

Formát reference na uzel je:

```
" {uzel}\r\n"
```

Kontextová vazba – u uzlu je kontextová vazba definována pomocí řádku ve formátu:

```
"{<>} {pr}%      {uzel}\r\n"
```

kde:

<|> je příznak vazby. Položí otázku pouze tehdy, když je hodnota uzlu, na který odkazuje kontextová vazba, větší než (>) respektive menší než (<) hodnota `pr`.

Pravidla – pravidla definují vazby mezi uzly. Každý uzel může obsahoval libovolné množství pravidel. Každé pravidlo může obsahovat libovolné množství uzlů, se kterými pracuje. Formát zápisu je následující:

```
"{&|} {alfa}%     {+-}{uzel}\r\n"
"                 {+-}{uzel}\r\n"
```

kde:

`&|` je znak vazby – konjunkce nebo disjunkce

`alfa` je síla vazby v rozsahu 1 … 99

`+-` je znak negace hodnoty pro "–" je použita při výpočtu negovaná hodnota pravděpodobnosti.

Externí načítání textových odpovědí – v tomto případě jsou možné dvě podmínky označené znaky:

`"n"` porovnává se text s textem v souboru a v případě shody je zvolena odpověď dle indexu

`"a"` automaticky je zvolena odpověď dle indexu.

Podmínky typu `"n"` se mohou opakovat s různým textem a indexem. Jako odpověď je vybrán index, pro který je jako první splněna podmínka. Odpověď typu `"a"` je vždy poslední. Začátek každé podmínky je uvozen středníkem. Externí textové odpovědi mají následující formát:

```
";@s {soubor}{podminka}\r\n"
```

Podmínka typu `"n": "; n {odpoved};;{id}"`

Podmínka typu `"a": "; a{id}"`

kde:

`soubor`	je cesta k souboru obsahující text porovnávané odpovědi
`odpoved`	je text odpovědi, se kterou se porovnává text ze souboru
`id`	je index odpovědi zvolené v případě, že jsou texty shodné.

Externí načítání číselných odpovědí – v tomto případě jsou možné čtyři podmínky označené znaky:

`"x"`	porovnává číslo v souboru s intervalem
`">"`	porovnává číslo v souboru se spodní mezí
`"<"`	porovnává číslo v souboru s horní mezí
`"a"`	automaticky je zvolena odpověď dle indexu.

První tři typy podmínek se mohou opakovat pro různé meze. Jako odpověď je vybrán index, pro který je jako první splněna podmínka. Odpověď typu `"a"` je vždy poslední. Začátek každé podmínky je uvozen středníkem. Externí číselné odpovědi mají následující formát:

`";@r {soubor}{podminka}\r\n"`

Podmínka typu `"x": "; x{ll};{ul};{id}"`

Podmínka typu `">": "; >{ll};{id}"`

Podmínka typu `"<": "; <{ul};{id}"`

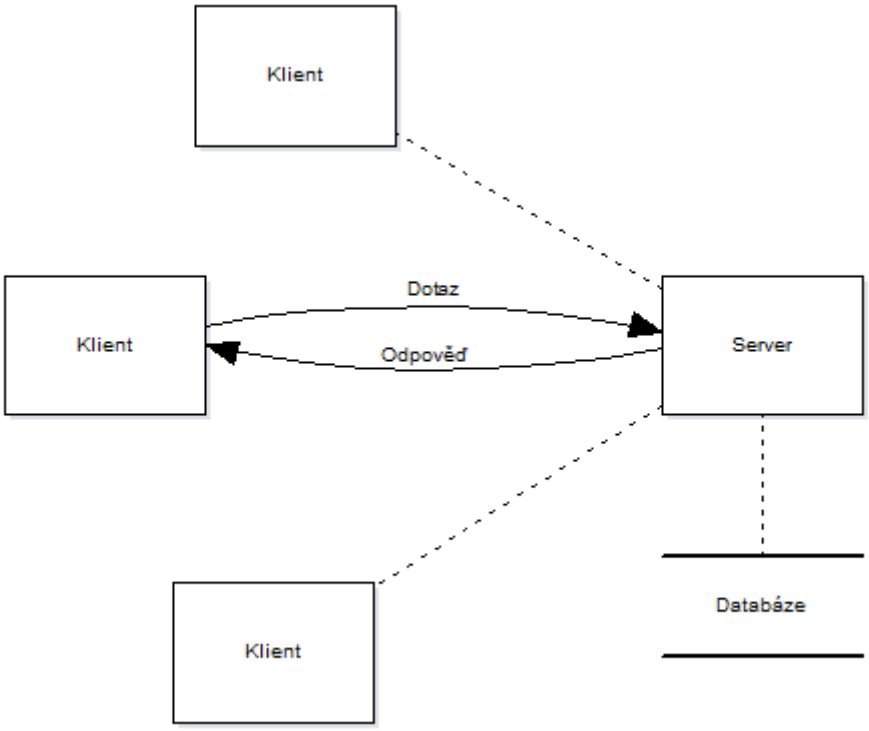
Podmínka typu `"a": "; a{id}"`

kde:

`ll`	je spodní mez v rozsahu 1 … 99 %
`ul`	je horní mez v rozsahu 1 … 99 %

2.7 Webové aplikace

Webovou aplikací je aplikace, která zprostředkovává nějakou službu dostupnou prostřednictvím počítačové sítě. K tomuto účelu jsou využívány webové technologie a nástroje pro správu dat. Základní rozdělení webových technologií je na Server-Side technologie a Client-Side technologie.



Obrázek 4 Základní komponenty webového prostředí

Komunikace mezi klientem a severem probíhá prostřednictvím sítě ethernet a protokolu HTTP, kdy klient zasílá požadavky na obsah webové stránky nebo provedení operace a server na tyto požadavky odpovídajícím způsobem zareaguje. Obsah je serverem zasílán zpravidla ve formě HTML dokumentu, ovšem protokol HTTP může z principu přenášet jakýkoliv obsah.

2.7.1 Server-Side technologie

Jedná se o technologie, které se vykonávají na straně serveru a zajišťují tak správu a předávání dat prostřednictvím webových protokolů. Příkladem těchto technologií jsou: PHP, Python, ASP.NET či Java nebo Node.js. Mezi tyto technologie můžeme teoreticky zařadit i systémy pro řízení báze dat (SŘBD) jako jsou například MySQL.

V rámci Server-Side technologií se časem vyvinula ucelená řešení integrující interpretry skriptovacích jazyků (SŘBD) a diagnostické nástroje. Uceleně se pro tyto celky zavedlo označení Webové servery. Mezi nejrozšířenější webové servery patří Apache a IIS, NGINX.

Apache

Je open-source multiplatformní webový server podporující většinu základních programovacích jazyků pro web (Perl, PHP, Python). Vývoj tohoto serveru probíhá již přes dvacet let, proto je pro něj k dispozici velké množství distribucí a rozšiřujících modulů.

IIS (Internet Information Services)

Je Webový server určený pro operační systém Windows. Bývá dodáván jako součást operačního systému MS Windows Server nebo vyšších edicí OS Windows.

NGINX

Je opět open-source multiplatformní webový server. Mezi jeho přednosti patří vysoká rychlost (díky tomu, že požadavek se zpracovává asynchronně jako event a ne jako proces/vlákno u Apache).

2.7.2 Client-Side technologie

Jde o technologie, které se vykonávají na straně klienta, jejich účelem je především formátování obsahu a vykonávání skriptů, které mají za účel formátovat obsah do požadované podoby, nebo upravovat obsah tak, aby byl závislý na klientovi.
Příkladem těchto technologií jsou: JavaScript či CSS.

CSS (Kaskádové styly)

Kaskádové styly slouží především pro statickou úpravu vzhledu webových stránek. Díky velkému množství parametrů, které lze nastavit u elementu HTML, je rozsah možností nastavení obrovský. V současnosti (2017) je k dispozici již verze CSS3 a podpora kaskádových stylů je integrována snad do všech nejrozšířenějších webových prohlížečů.

JavaScript

JavaScript je skriptovací jazyk, jehož interpret je rovněž podporován v naprosté většině současných prohlížečů. Na rozdíl od CSS slouží především k tvorbě složitějších prvků webu vykonávaných na straně klienta.

2.8 Testování SW

Vývoj každého software se řídí určitými pravidly, která vycházejí ze zkušeností a z inženýrského přístupu k řešení vývoje software. Pro proces, který zahrnuje návrh realizaci a nasazení, se zavedl pojem životní cyklus software. Životní cyklus software obecně obsahuje následující etapy:

1) specifikace požadavků,
2) návrh řešení (design),
3) implementace,
4) testování,
5) uvolnění,
6) údržba.

U vývoje každého software se tyto etapy vyskytují, ovšem v závislosti na zvolené metodice vývoje je každá z kapitol realizována jiným způsobem. Může být zvýrazněna nebo naopak dokonce zcela potlačena (například z důvodu nedostatku podkladů od zákazníka, přílišné náročnosti ekonomické nebo časové).

Tento text nemá za cíl provádět popis každé z těchto etap vývoje. K tomuto účelu může sloužit dostatek kvalitní odborné literatury, jako je například [6].

Můžeme se však částečně zaměřit na etapu, která se týká testování software. Před tím je ovšem potřeba si nejprve definovat základní pojmy.

Chyba

Chybou v software může být nazván jev, který splňuje alespoň jednu z následujících podmínek [2]:

1. Software nedělá něco, co by podle specifikace požadavků měl dělat.
2. Software dělá něco, co by podle specifikace požadavků neměl dělat.
3. Software dělá něco, co není nikde ve specifikaci požadavků uvedeno.
4. Software dělá něco, co není nikde ve specifikaci požadavků uvedeno, ale uvedeno by být mělo.
5. Software je těžko srozumitelný, těžkopádně se s ním pracuje nebo vykazuje takové vlastnosti, které brání v jeho správném a efektivním užívání.

Testování

Testování software je taková činnost, kdy se během vývoje systematicky a účelně vyhledávají a napravují možné chyby software.

Validace a verifikace

Verifikací se rozumí ověření, zda software odpovídá specifikaci požadavků.

Validací se zkoumá, zda program vyhovuje požadavkům uživatele.

Správnost

Program pracuje správně, pokud na svých vstupech/výstupech provádí takové operace, které jsou v souladu se specifikací požadavků.

Přesnost

Program pracuje přesně, pokud jsou jeho výstupy v naprosté shodě s předpokládaným výstupem, bez ohledu na to, jestli provedl správnou operaci.

Syntaktická chyba

Jedná se o chybu v zápisu strojového kódu. Zpravidla se jedná o jednodušší druh chyby, protože překladač ji zpravidla odhalí během překladu a nedovolí tak vytvořit spustitelný kód.

Sémantická chyba

Jedná se o chyby, které vzniknou až za běhu programu.

2.8.1 Kategorie jednotlivých testů software

Problematika testování software je poměrně rozsáhlá a cílem této práce není ani lehký přehled všech typů chyb, problémů a nástrojů, které slouží k jejich odstranění. Práce se omezuje pouze na kapitoly, které byly v průběhu její tvorby řešeny.

Rozdělení testů z hlediska etap vývoje software

Toto rozdělení zahrnuje typy testů, které by se měly realizovat během etapy implementace a testování v životním cyklu a pak v samozřejmě v případě změn. Rozdělení říká, jaká role, v které fázi vývoje sw, by měla dělat příslušné druhy testů.

Unit testy (Jednotkové testování)

Jedná se v naprosté většině případů o automatizované testování nejmenších dílčích částí – jednotek (Unit) kódu, které již lze samostatně testovat. Touto jednotkou může být libovolná část programu – funkce, procedura nebo program samotný, pokud je například dílčí částí většího systému. Detailnost nebo hloubka unit testů záleží na specifikaci požadavků, které jsou na daný software během vývoje kladeny.
Unit testy většinou provádí programátor během tvorby zdrojového kódu.

Integrační testy

Zatímco jednotkové testování je testování dílčích částí software, u integračních testů se již testují různé kombinace těchto jednotek a ověřuje se, zda interakce mezi nimi probíhá správným způsobem. V této části se tedy ověřují především rozhraní jednotlivých jednotek. Na úrovni integračních testů se již může zapojovat role testera.

Systémové testy

Systémové testy již testují program (systém) jako celek. Jejich účelem je ověřit, zda jsou naplněny požadavky kladné na software. Tento typ testů by již měl provádět tester a zapojuje se do této fáze role vlastníka (product ownera) daného software.

Akceptační testy

Akceptační testy jsou realizovány zpravidla ve finální části vývoje. Jejich smyslem je ověřit, zda software splňuje všechny požadavky na něj kladené. Mohou probíhat i za účasti zákazníka nebo koncového uživatele. Do této kategorie patří například beta testy.

Členění z hlediska způsobu provádění testů.

Na každé z předešlých úrovní je možné provádět testy rozdělitelné do dvou následujících hlavních kategorií:

Statická analýza

Statická analýza kódu se provádí zpravidla na zdrojových kódech, bez toho, aniž by byl program spuštěn. Pomocí statické analýzy se dají odhalit chyby typu překrývání proměnných, možné problémy při přetypování, memory leaky či například nesplnitelné podmínky a mnoho dalšího. Dále v rámci statické analýzy mohou být počítány i určité metriky kódu a tím pádem i sledována jeho kvalita.
Statickou analýzu kódu v omezené míře provádí již samotný překladač (např: Clang Static Analyzer) nebo vývojové prostředí (např: Visual Studio Code Analysis), Tyto prostředky odhalí řadu možných problémů, jako jsou například nesplnitelné podmínky nebo chyby při přetypování a zahlásí je jako varování nebo rovnou jako chybu překladu.
Vedle překladače existuje pro statickou analýzu kódu samozřejmě i celá řada specializovaných nástrojů. Jako příklad nástrojů pro jazyk C/C++ mohou složit například:

PRQA

Jedná se o komerční produkt určený pro jazyky C(QA·C)/C++(QA·C++), domovská stránka programu je http://www.programmingresearch.com/. Program podporuje kontrolu kódu dle standardů MISRA-C / MISRA-C++.

CppCheck

Jedná se o open-source nástroj, dostupný je na domovské stránce programu http://cppcheck.sourceforge.net/. Má podporu v podobě zásuvných modulů do mnoha vývojových nástrojích jako je Eclipse, Visual Studio či Git. Provádí přes 200 různých typů kontrol kódu.
Pro integraci CppCheck do prostředí MS Visual Studia slouží Cppcheck add-in dostupný z adresy:
https://marketplace.visualstudio.com/items?itemName=Alexium.Cppcheckadd-in.

Dynamická analýza

Na rozdíl od statické analýzy se dynamická analýza kódu provádí za běhu programu. Testuje se především práce s pamětí: memory leaky, čtení/zápis z již uvolněných objektů, práce s vlákny a prostředky (soubory, rozhraní).
Jedním z nejrozšířenějších nástrojů pro dynamickou analýzu je například Valgrind. Jedná se o open-source projekt určený pro prostředí OS založených na UNIX. Tento nástroj přeloží program do vlastní formy strojového kódu, který poté kontrolovaným způsobem běží ve virtuálním prostředí.

3 PRAKTICKÁ ČÁST

Praktická část se zabývá především tvorbou programu NPSCore, což je základní výpočetní jádro expertního systému vycházejícího ze systému NPS32. Matematický aparát a inferenční mechanismus zůstaly zachovány, ovšem byly kompletně přepsány a zapouzdřeny do samostatné knihovny. Vedle popisu programu je v této části práce uveden i kompletní popis rozhraní knihovny a popis jednotlivých funkcí. V poslední části se pak práce zabývá ověřením funkčnosti knihovny a nových možností.

3.1 Základní požadavky na NPSCore

V rámci této diplomové práce měla být provedena především příprava jádra programu NPS32 pro možnost použití v prostředí webového serveru. Program NPS32 byl vyvinut před více než 15 lety v rámci předchozích diplomových prací. Byl vytvořen ve vývojovém prostředí Borland C++ pro Windows.
Vzhledem k tomu, že během vývoje NPS32 nebylo zcela dodrženo striktní oddělení výkonné části kódu od prezentační části a bylo velmi těžké se v daném programu zorientovat, byla zvolena cesta provést nový objektový návrh výkonného jádra, jeho zapouzdření a doplnění podpory pro obecný obsah.
V rámci této práce mělo být tedy na výkonném jádře provedeno:

1. refaktoring kódu s větším důrazem na OOP
2. vytvoření dynamické knihovny s rozhraním umožňujícím komunikaci jak v rámci aplikace, tak i pro připojení v rámci webového rozhraní
3. vícejazyčná podpora
4. zobecnění obsahu zobrazovaného uživateli na libovolný obsah
5. ověření funkčnosti
6. pro překlad použít překladač Clang.

V následujících kapitolách budou jednotlivé body podrobněji rozebrány.

3.2 Formáter/parser XML dokumentů

Vzhledem k tomu, že knihovna NPSEngineLibrary poskytuje rozhraní prostřednictvím textu ve formátu XML, je vhodné pro práci s tímto souborem použít některou z již existujících a ověřených knihoven.
Základními požadavky na Formáter/Parser XML jsou:

- otevřený kód s přenositelností mezi operačními systémy,
- volná licence pro použití v komerční i nekomerční sféře,
- podpora textů v kódování UTF-8,
- stabilita,
- rychlost,
- není nutná podpora namespace v XML.

Po prozkoumání dostupných knihoven na internetu se nabízí hned několik možností například: LibXML2, TinyXML, RapidXML či PugiXML. Po ověření možností jednotlivých knihoven byla nakonec pro další práci zvolena knihovna RapidXML. Tato knihovna je v současnosti volně dostupná na adrese http://rapidxml.sourceforge.net/.

Knihovna RapidXML obsahuje dvě hlavní třídy:

```
rapid::xml_document<> // zastřešujících celý XML dokument
rapid::xml_node<>     // realizuje jeden XML element dokumentu.
```

Pomocí těchto tříd se skládá XML dokument do požadovaná struktury. Doplňující třídou je:

```
rapid::xml_attribue<> // realizuje jeden atribut XML elementu.
```

3.3 Diagram hlavních komponent

Hlavní komponenty programu jsou:
NPSCore – hlavní program zajišťující zprostředkování komunikace mezi knihovnou NPSEngineLibrary a okolním prostředím.
NPSEngineLibrary – knihovna realizující funkce pro inferenční mechanismus expertního systému NPS.

3.4 Program NPSCore

Slouží především pro předávání příkazů z vnějšího prostředí programu a standardních kanálů std::cin a std::cout do knihovny NPSEngineLibrary a zpět.

3.4.1 Objektový model uzlů inferenčního mechanismu

Základem inferenčního mechanismu jsou uzly. Uzly jsou třídy odvozené od třídy Node, která obsahuje základní obecné funkce dostupné pro práci s nody. Základní hierarchie mezi jednotlivými uzly je znázorněna na obrázku uvedeném v příloze 5.2Hierarchie dědičnosti uzlů.
`Node` – bázová abstraktní třída pro práci s jednotlivými uzly báze znalostí.
`NodeAncilliary` – třída realizující uzel inferenčního mechanismu typu pomocný uzel.
`NodeHypothesis` – třída realizující uzel inferenčního mechanismu typu cílová hypotéza.
`NodeQuery` – třída realizující uzel inferenčního mechanismu typu dotazovatelný uzel přímý.
`NodeQueryUsual` – třída realizující uzel inferenčního mechanismu typu dotazovatelný uzel běžný.
`NodeQueryQuantitative` – třída realizující uzel inferenčního mechanismu typu dotazovatelný uzel kvantitativní.

Odpovědi u jednotlivých uzlů jsou realizovány pomocí tříd:
`Answer` – bázová abstraktní třída pro práci s uživatelskými odpověďmi.
`AnswerQuery` – odpověď pro dotaz přímý a běžný.
`AnswerQuntitative` – odpověď pro dotaz kvantitativní.

3.4.2 Vazby mezi uzly

Vazby mezi uzly jsou především pravidla realizovaná poté třídou `Relationship` a kontext, který je realizován třídou `Context`.

3.4.3 Inferenční mechanismus

Inferenční mechanismus zajišťuje samotnou funkci konzultace, je realizován především třídou `NPSInferenceMechanism`, která poskytuje všechny potřebné metody pro administraci konzultace.

3.4.4 Formatery/Parsery

Pro převod báze znalostí v podobě souboru do vnitřní reprezentace tvořené instancemi příslušných tříd slouží formatery/parsery. V rámci této práce bylo potřeba vytvořit celkem čtyři konverzní funkce. Každá z nich je realizována jednou samostatnou třídou:

`ParserN32` – slouží pro načtení báze znalostí z vstupního std::istream ve formatu *.n32 na vnitřní reprezentaci báze znalostí.

`FormaterN32` – slouží pro převod vnitřní reprezentace báze znalostí do výstupního proudu std::ostream ve formatu *.n32.

`FormaterNPS` – slouží pro převod vnitřní reprezentace báze znalostí do výstupního proudu std::ostream ve formatu *.n32.

`ParserNPS`– slouží pro načtení báze znalostí z vstupního std::istream ve formatu *.nps na vnitřní reprezentaci báze znalostí.

3.4.5 Třída NPSEngine

Třída `NPSEngine` realizuje rozhraní samotné knihovny NPSEngineLibrary. Její hlavní veřejnou metodou je metoda Execute, která zpracovává příkazy pro práci s bází dat a samotnou konzultací. Popis funkce jednotlivých příkazů (metod) je uveden v kapitole 3.5 Komunikační rozhraní NPSEngineLibrary.dll.

3.5 Komunikační rozhraní NPSEngineLibrary.dll

Dynamická knihovna NPSEngineLibrary.dll má jako rozhraní zvolenu třídu NPSEngine. Každá instance třídy `NPSEngine` obsahuje jednu instanci Inferenčního mechanismu a tím pádem může obsahovat pouze jednu aktuálně rozpracovanou konzultaci.

Konstruktor třídy `NPSEngine` obsahuje jeden parametr typu `std::string`, který obsahuje cestu k souboru s defaultními odpověďmi pro inferenční mechanismus. Pokud tento parametr není zadán nebo je zadaná cesta neplatná, použijí se interní hodnoty defaultních odpovědí. Soubor obsahující defaultní odpovědi a jeho parametry je popsán v kapitole 3.6 Externí soubor s defaultními odpověďmi.

Po vytvoření instance třídy `NPSEngine` se veškerá komunikace s bází dat provádí prostřednictvím metody:

```
void NPSEngine::Execute(std::string &response, std::string message);
```

která přijímá příkazy pro vykonání operace s bází dat prostřednictvím parametru `message` a odpověď na příkaz vrací pomocí parametru `response`. Knihovna

standardně přijímá příkazy v textovém podobě se syntaxí XML. Pokud příkaz neobsahuje znakovou stránku, počítá se s kódováním UTF-8.
Konkrétní příkazy, které se dají odeslat do třídy `NPSEngine`, jsou rozděleny do dvou kategorií:

Příkaz typu `cmd` – tento typ příkazů slouží pro operace nad bází dat.

Příkaz typu `base` – tento typ příkazů je určený přímo k inferenčnímu mechanismu a probíhající konzultaci.

Seznam příkazů, které je aktuálně třída `NPSEngine` ve verzi 1.0.0.1 schopná zpracovat, je uveden v následující tabulce Tab. 3.2:

Tab. 3.1 Příkazů typu `base`

Typ	Příkaz	Krátký popis
`base`	`set_state`	Příkaz slouží k načtení stavu konzultace a nastavení konzultace do uloženého stavu
`base`	`get_language`	Příkaz vrátí seznam jazyků, které jsou v aktuální bázi znalostí k dispozici
`base`	`set_language`	Příkaz nastaví jazyk, se kterým báze znalostí pracuje a ve kterém předává texty
`base`	`get_info`	Příkaz vrátí název aktuální báze znalostí a její popis
`base`	`get_hypotesis`	Příkaz vrátí hodnoty hypotéz odpovídající stavu konzultace
`base`	`get_query`	Příkaz vrátí aktuální text dotazu a množinu možných odpovědí na něj
`base`	`set_answer`	Příkaz nastaví odpověď na aktuální dotaz
`base`	`reset_query`	Příkaz slouží pro reset aktuální otázky
`base`	`reset_base`	Příkaz slouží pro reset celé konzultace do počátečního stavu
`base`	`go_back`	Příkaz slouží pro posun zpět na předchozí otázku
`base`	`go_ahead`	Příkaz slouží pro posun vpřed na následující otázku

Tab. 3.2 Příkazů typu `cmd`

Typ	Příkaz	Krátký popis
`cmd`	`load_base`	Příkaz slouží pro načtení báze znalostí a její inicializaci
`cmd`	`close_base`	Příkaz slouží pro ukončení konzultace a smazání aktuální báze znalostí
`cmd`	`convert_base`	Příkaz slouží pro konverzi mezi bází znalostí ve formátu *.n32 a formátu *.nps
`cmd`	`exit`	Příkaz slouží k ukončení programu

3.5.1 Příkaz cmd load_base

V následujících kapitolách jsou jednotlivé příkazy rozebrány a podrobně popsány. Všechny příkazy, včetně chybných, jsou zpracovány a knihovna na ně předá odpověď v následujícím jednotném formátu:

```
<reply type="{type}" id="{id}">
     <reply_message>
          {msg}
     </reply_message>
     {prm}
</reply>
```

kde:

{type}	je výsledek zpracování příkazu `OK` nebo `error`
{id}	je unikátní kód chyby viz číselník chyb v kapitole 3.5.19 Popis kódů chyb
{msg}	případný krátký popis chyby
{prm}	další případné XML elementy odpovědi, například text dotazu. Detailněji jsou tyto elementy uvedeny v popisu každého příkazu.

3.5.2 Obecný formát zobrazovaných informací

Oproti předchozí verzi programu NPS32 byla u programu NPSCore snaha zobecnit obsah všech informací zobrazovaných uživateli tak, aby bylo možné vedle prostého textu zobrazit během konzultace i multimediální obsah. Z tohoto důvodu byl obsah uzavřen do obecného elementu `content` s metainformacemi o obsahu a tento element program NPSCore zpracová tak, že jej v nezměněné formě předává v rámci odpovědi během konzultace. Tento formát je zvolen pro následující texty:

Nadpis,

Popis báze znalostí,

Obsah otázky,

Obsah odpovědi,

Obsah hypotézy,

Element `content` má následující formát:

```
<content culture="{culture}" type="text">
     {text}
</content>
```

kde:

`type`	je označení typu obsahu; aktuálně pouze typ `text`
`{culture}`	je označení jazyka, ve kterém je tento text předáván
`{text}`	je samotný text zobrazený během konzultace.

V následujícím textu označen jako *`{content}`*.

3.5.3 Příkaz cmd load_base

Příkaz slouží pro načtení báze znalostí. V aktuální verzi je k dispozici možnost načíst bázi znalostí ve formátu *.n32 nebo z nového formátu typu *.nps. Jako zdroj dat je aktuálně implementován soubor. Struktura příkazu pro načtená báze dat je následující:

```
<cmd type="load_base" src="file">
     {path}
</cmd>
```

kde:

`type`	je povinný název příkazu
`src`	je povinný typ zdroje aktuálně pouze typ `file`
`{path}`	je povinná cesta k souboru s bází znalostí s příponou *.n32 nebo *.nps.

Tento příkaz může vrátit následující kódy chyb:

0, 1011, 1012, 1013, 1014, 1017

Popis kódů je uveden v kapitole 3.5.19 Popis kódů chyb.

3.5.4 Příkaz cmd close_base

Příkaz slouží pro uzavření konzultace a zrušení případné načtené báze znalostí. Příkaz neobsahuje žádné potvrzování a po jeho přijetí dojde k smazání i případného neuloženého stavu probíhající konzultace. Struktura příkazu pro uzavření konzultace je následující:

```
<cmd type="close_base"/>
```

Tento příkaz může vrátit následující kódy chyb:

0, 2000

Popis kódů je uveden v kapitole 3.5.19 Popis kódů chyb.

3.5.5 Příkaz cmd exit

Příkaz slouží pro ukončení provozu programu. Uzavře se konzultace a zruší se případné načtené báze znalostí. Struktura příkazu pro ukončení provozu je následující:

```
<cmd type="exit"/>
```

Tento příkaz může vrátit následující kódy chyb:

0

Popis kódů je uveden v kapitole 3.5.19 Popis kódů chyb.

3.5.6 Příkaz cmd convert_base

Příkaz slouží pro konverzi mezi formáty bází znalostí. *.n32 a *.nps. V aktuální verzi je jako zdroj dat aktuálně implementován pouze soubor (`file`). Struktura příkazu pro konverzi bází znalostí je následující:

```
<cmd type="convert_base">
     <src type="file">{srcPath}</src>
     <dst type="file">{dstPath}</dst>
</cmd>
```

kde:

`type` je povinný typ zdroje aktuálně pouze typ `file`

`{srcPath}` je povinná cesta k souboru s výchozí bází znalostí s příponou *.n32 nebo *.nps.

`{dstPath}` je povinná cesta k souboru s cílovou bází znalostí s příponou *.n32 nebo *.nps.

Tento příkaz může vrátit následující kódy chyb:

0, 1010, 1013, 1015, 1016

Popis kódů je uveden v kapitole 3.5.19 Popis kódů chyb.

3.5.7 Příkaz base get_query

Příkaz slouží pro načtení aktuálního dotazu konzultace. Jako odpověď je odeslán text dotazu a příslušné možné odpovědi. Texty jsou předávány v nastaveném jazyce konzultace. Struktura příkazu je následující:

```
<base type="get_query"/>
```

Odpověď na tento příkaz poté má navíc parametr *{prm}* ve tvaru:

```
<question queryNumber="{queryNumber}">
     {content}
     < answers>
          <answer answer_id="{answID}">
               {content}
          </answer>
          …
     </answers>
</question>
```

kde:

{queryNumber}	je pořadové číslo dotazu od začátku konzultace
{answID}	je pořadové číslo možné odpovědi na dotaz.

Tento příkaz může vrátit následující kódy chyb:

0, 100

Popis kódů je uveden v kapitole 3.5.19 Popis kódů chyb.

3.5.8 Příkaz base get_hypotesis

Příkaz slouží pro načtení aktuálního stavu cílových hypotéz. Jako odpověď je odeslána množina všech hypotéz. Texty jsou předávány v nastaveném jazyce konzultace. Struktura příkazu je následující:

```
<base type="get_hypotesis"/>
```

Odpověď na tento příkaz poté má navíc parametr *{prm}* ve tvaru:

```
<hypoteses>
     <hypotesis literal="Hrac1" probability="0.417910"
     uncertainty="0.717949" chance="0.717949">
```

```
            {hypotesis_sumary}
            {content}
        </hypotesis>
            …
    </hypoteses>
```

kde:

`literal`	je název uzlu
`probability`	hodnota pravděpodobnosti v intervalu <0,1>, která odpovídá hodnotě pro aktuální dotaz před jeho zodpovězením
`uncertainly`	je hodnota neurčitosti v intervalu <0,1>
`chance`	je hodnota šance v intervalu <0, Inf>.

Vedle těchto parametrů je u hypotézy uvedena tato množina parametrů ještě jednou v případě, že je aktuálně zobrazovaný dotaz již zodpovězen. V takovém případě je přidán ještě element *{hypotesis_sumary}* ve tvaru:

```
<hypotesis_sumary literal="Hrac1" probability="0.417910"
uncertainty="0.717949" chance="0.717949"/>
```

Tento příkaz může vrátit následující kódy chyb:

0

Popis kódů je uveden v kapitole 3.5.19 Popis kódů chyb.

3.5.9 Příkaz base get_info

Příkaz slouží pro načtení informací o aktuální konzultaci. Jako odpověď je odeslán název báze znalostí a její popis. Texty jsou předávány v nastaveném jazyce konzultace. Struktura příkazu je následující:

```
<base type="get_info"/>
```

Odpověď na tento příkaz poté má navíc parametr *{prm}* ve tvaru:

```
<name>
    {content}
</name>
<description>
    {content}
</description>
```

kde:

`name`	je název aktuálně načtené báze znalostí
`description`	je popis aktuálně načtené báze znalostí.

Tento příkaz může vrátit následující kódy chyb:

0

Popis kódů je uveden v kapitole 3.5.19 Popis kódů chyb.

3.5.10 Příkaz base get_language

Příkaz slouží pro načtení informací o dostupných jazykových verzích. Jako odpověď je odeslána množina jazyků, které obsahuje aktuální báze znalostí a aktuálně nastavený jazyk konzultace. Je předán pouze identifikátor jazyka. Struktura příkazu je následující:

```
<base type="get_language"/>
```

Odpověď na tento příkaz poté má navíc parametr *{prm}* ve tvaru:

```
<languages>
     <language>
          {langID}
     </language>
     …
</languages>
<current_language>
     {langID}
</current_language>
```

kde:

`language`	je dostupný jazyk báze znalostí
`current_language`	je aktuálně nastavený jazyk konzultace
{langID}	je identifikátor jazyka. Text musí být pro jazyk unikátní, ovšem konkrétní forma již povinná není. Pro testovací účely byla zvolena forma odpovídající RFC 5646 ve formátu odpovídajícím češtině „cs-CZ“.

Tento příkaz může vrátit následující kódy chyb:

0

Popis kódů je uveden v kapitole 3.5.19 Popis kódů chyb.

3.5.11 Příkaz base get_state

Příkaz slouží pro načtení kompletního stavu konzultace. Uložen stav tak, aby bylo možné v konzultaci po načtení stavu pokračovat ze stejného bodu jako při uložení.

Stav konzultace je možné aktuálně uložit dvěma způsoby: buď do souboru nebo předat stav konzultace jako element odpovědi na příkaz.

Struktura příkazu je následující:

```
<base type="get_state" dst="{dstType}">
     {dstPath}
</base>
```

kde:

{dstType} je parametr určující, jak se předává stav konzultace
- `file` - stav konzultace se ukládá do souboru
- `direct` - stav konzultace je předáván v odpovědi na příkaz

{dstPath} pokud je zvoleno uložení do souboru, je potřeba předat i cestu k souboru, do kterého se má stav konzultace uložit.

V případě, že je stav konzultace předáván v odpovědi, je stav předán jako parametr *{prm}*. Popis uloženého stavu konzultace je popsán v kapitole 3.7 Stav konzultace báze.

Tento příkaz může vrátit následující kódy chyb:

0

Popis kódů je uveden v kapitole 3.5.19 Popis kódů chyb.

3.5.12 Příkaz base set_answer

Příkaz slouží pro nastavení odpovědi na aktuálně platný dotaz. Aby byla odpověď nastavena správnému uzlu, je potřeba, aby před zavoláním příkazu `set_answer` byl zavolán příkaz `get_query` s kódem chyby 0. V takovém případě je pro aktuální nezodpovězený uzel nastavena odpověď a konzultace je přepočítána dle typu uzlu a zvolené odpovědi. Následně je pro další příkaz `set_answer` nutné opět zavolat příkaz `get_query`. Tím je zajištěno, že dotaz byl položen.

V případě, že se listuje v historii konzultace pomocí příkazu `go_back` a `go_ahead`, není nutné volat příkaz `get_query`, protože text dotazu i příslušné odpovědi je předáván v odpovědi na tyto příkazy.

Texty jsou předávány v nastaveném jazyce konzultace. Struktura příkazu je následující:

```
<base type="set_answer">
     {answID}
</base>
```

Tento příkaz může vrátit následující kódy chyb:

0, 2002, 2003

Popis kódů je uveden v kapitole 3.5.19 Popis kódů chyb.

3.5.13 Příkaz base reset_base

Příkaz slouží pro uvedení celé konzultace do výchozího stavu před zodpovězením prvního dotazu.

Struktura příkazu je následující:

```
<base type="reset_base" />
```

Tento příkaz může vrátit následující kódy chyb:

101

Popis kódů je uveden v kapitole 3.5.19 Popis kódů chyb

3.5.14 Příkaz base reset_query

Příkaz slouží pro reset aktuálně zobrazeného dotazu do stavu před zodpovězením. Spolu s tímto uzlem jsou uvedeny do výchozího stavu i všechny případné následující již zodpovězené dotazy v konzultaci, pokud byl posunut kurzor konzultace pomocí příkazů `go_back` a `go_ahead`.

Struktura příkazu je následující:

```
<base type="reset_query" />
```

Tento příkaz může vrátit následující kódy chyb:

100, 102, 2006

Popis kódů je uveden v kapitole 3.5.19 Popis kódů chyb.

3.5.15 Příkaz base go_back

Příkaz slouží pro posun kurzoru konzultace o jednu otázku zpět. Pokud se nejedná o první dotaz v konzultaci, je v odpovědi na příkaz vrácena otázka včetně všech odpovědí. Texty jsou předávány v nastaveném jazyce konzultace.

Struktura příkazu je následující:

```
<base type="go_back" />
```

V případě kódu chyby 0 je obsah odpovědi stejný jako v případě příkazu `get_query`, který je uvedený v kapitole 3.5.7 Příkaz base `get_query`. Vedle elementu `question` je však navíc uveden i element `user_answer`, který obsahuje následující informace:

```
<user_answer answer_id="{answID}"
answeredBy="{answBy}" answerSource="{answSrc}"
timeStamp="{timeSt}"/>
```

kde:

`answer_id`	je id zvolené odpovědi na tento dotaz
`answeredBy`	je id zdroje odpovědi. Možné zdroje odpovědí odpovídají datovému typu `NodeMetadataType`.
`answerSource`	je případný název zdroje odpovědi. Jde o proměnnou typu `std::string.`
`timeStamp`	je časová značka, odpovídající času, kdy byla odpověď zvolena. Čas je předáván jako UTC čas ve formátu ISO 8601. Například: `2017-05-03 19:48:59Z.`

Tento příkaz může vrátit následující kódy chyb:

0, 100, 2005

Popis kódů je uveden v kapitole 3.5.19 Popis kódů chyb.

3.5.16 Příkaz base go_ahead

Příkaz slouží pro posun kurzoru konzultace o jednu otázku vpřed. V odpovědi na příkaz je vždy vrácena otázka včetně všech odpovědí a pokud již odpověď byla zvolena, tak jsou předány i metadata o odpovědi pomocí elementu `user_answer.` Tyto parametry jsou popsány v kapitole 3.5.15 Příkaz base go_back. Texty jsou předávány v nastaveném jazyce konzultace.

Struktura příkazu je následující:

```
<base type="go_ahead" />
```

Tento příkaz může vrátit následující kódy chyb:

0, 100, 2005

Popis kódů je uveden v kapitole 3.5.19 Popis kódů chyb.

3.5.17 Příkaz base set_language

Příkaz slouží pro nastavení aktuálního jazyka konzultace. Pokud aktuální báze znalostí zvolený jazyk podporuje, je tento jazyk nastaven a další texty jsou již předávány v tomto jazyce.

Struktura příkazu je následující:

```
<base type="set_language">{langID}</base>
```

Tento příkaz může vrátit následující kódy chyb:

0, 2007

Popis kódů je uveden v kapitole 3.5.19 Popis kódů chyb.

3.5.18 Příkaz base set_state

Příkaz je opakem příkazu pro uložení stavu get_state. Slouží pro nastavení stavu konzultace do uloženého stavu. Obdobně jako u příkazu get_state je možné stav načíst přímo z příkazu „`direct`“ nebo z uloženého souboru „`file`“.

Struktura příkazu je následující:

```
<base type="set_state" src="{srcType}">
     {prmState}
</base>
```

kde:

`{srcType}` je parametr určující, jak se předává stav konzultace

- `file` - stav konzultace se ukládá do souboru. Pro tento parametr je v proměnné *`{`*`prmState`*`}`* uložena cesta k souboru, ve kterém se očekává stav konzultace.
- `direct` - stav konzultace je předáván v odpovědi na příkaz. Pro tento parametr je v proměnné *`{`*`prmState`*`}`* uložen stav konzultace odpovídající kapitole 3.7 Stav konzultace báze.

Tento příkaz může vrátit následující kódy chyb:

0, 2003, 2008, 2009, 2010, 2011, 2012, 2013, 2016, 2017

Popis kódů je uveden v kapitole 3.5.19 Popis kódů chyb.

3.5.19 Popis kódů chyb

Po zaslání příkazu do inferenčního mechanismu odpoví program NPSCore na daný příkaz vedle požadovaných informací nebo provedení nějaké operace i pomocí krátkého návratového kódu chyby. Následující tabulka obsahuje souhrnný přehled všech možných kódů chyby a jejich popis.

Tab. 3.3 Obecné a nechybové kódy

Kód chyby	Popis
0	Příkaz se vykonal v pořádku.
Výjimečné stavy, u kterých nejde o chybu	
100	Konzultace byla dokončena a nejsou k dispozici žádné další dotazy, které by bylo možné uživateli položit. Je možné pouze listovat v historii konzultace pomocí příkazů go_back a go_ahead a případně zvolit jinou odpověď na aktuální otázku, čímž se následující dotazy resetují.
101	Byl zavolán příkaz na reset celé konzultace.
102	Byl zavolán příkaz na reset aktuálního dotazu.
Obecné chyby	
1001	Byl zaslán příkaz, který není typu cmd nebo base.

Tab. 3.4 Kódy chyb příkazu `cmd`

Kód chyby	Popis
1002	Nebyl specifikován nebo není uveden druh příkazu pro práci s bází znalostí.
1010	Při konverzi báze znalostí se objevila chyba. V return_message je detailněji popsán obsah této chyby.
1011	V příkazu chybí typ předání souboru.
1012	Typ zdroje báze znalostí není definovaný nebo není podporovaný. Aktuálně je podporovaný pouze zdroj typu `file`.
1013	Soubor na dané cestě neexistuje nebo se jej nepodařilo otevřít.
1014	Chyba při načítání souboru s bází znalostí – chyba syntaxe chybějící parametry, hodnoty parametrů mimo rozsah. Detailnější popis je uveden v souboru.
1015	Nepodporovaný formát cílového souboru pro konverzi mezi bázemi znalostí. Podporovanými formáty jsou *.n32 nebo *.nps.
1016	Nepodporovaný formát zdrojového souboru pro konverzi mezi bázemi znalostí. Podporovanými formáty jsou *.n32 nebo *.nps.
1017	Předaný název souboru nemá příponu, která by odpovídala podporovaným tj. *.n32 nebo *.nps.

Tab. 3.5 Kódy chyb příkazu base

Kód chyby	Popis
2000	Báze znalostí není načtena. Před všemi příkazy typu base musí být úspěšně vykonán příkaz typu load_base.
2001	Nebyl specifikován nebo není uveden typ příkazu pro práci s bází znalostí.
2002	Nelze nastavit odpověď na dotaz, protože dotaz nebyl položen. Před zasláním příkazu set_answer je potřeba nejprve zavolat příkaz get_query, go_back, nebo go_ahead.
2003	Index zvolené odpovědi pro aktuální otázku není definován. Index odpovědi musí být v rozsahu odpovídajícím odpovědím v aktuálním dotazu.
2004	Kurzor posuvu v historii odpovědí je na první otázce konzultace nelze se posunout.
2005	Kurzor posuvu v historii odpovědí je na poslední – aktuální otázce konzultace, nelze se posunout dále bez zodpovězení otázky.
2006	Otázku nelze resetovat. Nebyla ještě zodpovězena.
2007	Jazyk, který se pokoušíte nastavit, není v aktuální konzultaci dostupný.
2008	Stav báze znalostí a Aktuální načtená báze znalostí nejsou shodné. Aktuálně se jako identifikátor báze bere název souboru.
2009	Časová značka začátku konzultace neodpovídá požadovanému formátu. Aktuálně jediný platný formát je dle normy ISO8601 například: `2017-05-03 19:48:59Z`.
2010	Časová značka doby zodpovězení otázky neodpovídá požadovanému formátu. Aktuálně jediný platný formát je dle normy ISO8601 například: `2017-05-03 19:48:59Z`.
2011	Typ zdroje odpovědi v stavu konzultace. Aktuálně jsou podporovány pouze zdroje odpovídající `NodeMetadataType`.
2012	Uzel, jehož hodnota odpovědi se má nastavit ze stavu konzultace, není dotazovatelný uzel.
2013	Chybný formát souboru se stavem konzultace. Ve zdroji je chyba syntaxe nebo se nepodařilo nalézt a otevřít uvedený soubor.
2014	Soubor, do kterého se má uložit aktuální stav konzultace, nelze otevřít nebo vytvořit.
2015	Typ cíle pro uložení stavu konzultace není uveden nebo není podporován. Aktuálně jsou podporovány pouze zdroje `direct` a `file`.

2016	Typ zdroje stavu konzultace není uveden nebo není podporován. Aktuálně jsou podporovány pouze zdroje `direct` a `file`.
2017	Chyba v obsahu stavu konzultace. V return_message je detailněji popsán obsah této chyby.
6666	Obecně nějaká neočekávaná chyba. V return_message je detailněji popsán obsah této chyby.
9999	Chyba při inicializaci knihovny NPSEngine, program je potřeba restartovat.

3.6 Externí soubor s defaultními odpověďmi

Soubor NPSCore má v sobě napevno zabudované defaultní odpovědi pro úrovně detailnosti odpovědí 3, 5 a 7 tak, jak jsou popsány v tabulce Tab. 2.1 Implicitní hodnoty odpovědí. Tyto defaultní odpovědi jsou však k dispozici pouze v češtině.

Aby bylo možné definovat vlastní obsah odpovědí pro jednu ze tří úrovní detailnosti, případně měnit celý obsah, je možné definovat vlastní soubor s defaultními odpověďmi. V tomto souboru musí být zachováno rozdělení detailnosti na 3, 5 a 7 odpovědí. Soubor musí být umístěn ve stejném adresáři jako je program NPSCore. V tomto případě program po startu zkusí vyhledat soubor DefaultAnswers.xml a načíst z něj defaultní odpovědi. Pokud se soubor nepodaří najít nebo nemá platný formát, je jeho obsah ignorován a použijí se defaultní odpovědi definované interně v programu.

Soubor s defaultními odpověďmi je opět ve formátu XML s následující syntaxí:

```
<?xml version="1.0" encoding="utf-8"?>
<DefaultAnswers version="{version}">
    <DefaultAnswerThreefold>
        <answer probability="{p}">
            {content}
            …
        </answer>
        …
    </DefaultAnswerThreefold>
    <DefaultAnswerFivefold>
        …
    </DefaultAnswerFivefold>
```

```
        <DefaultAnswerSevenfold>
            …
        </DefaultAnswerSevenfold>
    </DefaultAnswers>
```

kde:

`DefaultAnswers`	je vrcholový element obsahující všechny defaultní odpovědi
`version`	je atribut obsahující verzi formátu se souborem defaultních odpovědí
`DefaultAnswerThreefold`	je element obsahující defaultní odpovědi se stupněm detailnosti 3
`DefaultAnswerFivefold`	je element obsahující defaultní odpovědi se stupněm detailnosti 5
`DefaultAnswerSevenfold`	je element obsahující defaultní odpovědi se stupněm detailnosti 7

3.7 Stav konzultace báze

Stav konzultace obsahuje kompletní seznam všech relevantních informací o konzultaci v takovém stavu, aby bylo možné konzultaci uložit a po opětovném načtení pokračovat v konzultaci ve stejném bodě. Stav báze znalostí je opět uložen ve formátu XML. Obecný obsah stavu konzultace je následující:

```
<state baseID="{baseID}" start_time="{startTime}"
current_language="{langID}" offset="{offset}">
    <nodes>
        <node literal="{ID}" answer="{answID}"
        answeredBy="{answBy}"
        answerSource="{answSrc}"
        timeStamp="{timeSt}"/>
    </nodes>
</state>
```

kde:

`state`	je vrcholový element obsahující kompletní uložený stav konzultace
`baseID`	je atribut obsahující identifikátor báze znalostí. Aktuálně se používá název báze znalostí.

`Start_time`	je atribut obsahující čas, kdy byla konzultace spuštěna
`current_language`	je atribut obsahující aktuální nastavený jazyk konzultace
`offset`	je atribut určující ofset kurzoru v konzultaci
`nodes`	je element obsahující odpovědi na jednotlivé uzly
`node`	je element obsahující id odpovědí pro jednotlivé dotazovatelné uzly, které byly během konzultace vybrány jako odpověď. Dále jsou součástí tohoto elementu i metadata o odpovědi.

3.8 Formát báze znalostí *.nps

Změna formátu báze znalostí z formátu *.n32 popsaného v kapitole 2.6 Formát báze znalostí *.n32 byla provedena především z důvodu umožnění podpory více jazykových verzí, umožnění zpracování báze znalostí pomocí standardních nástrojů a zobecnění textů tak, aby bylo možné vedle klasických textů zobrazit i jiné informace než prostý text. Zároveň bylo potřeba zachovat původní matematický aparát a informace z již odladěných bází znalostí.

Z těchto důvodů byl jako základní formát báze znalostí zvolen formát XML. Jako kódování byl zvolen formát UTF-8, čímž se zajistí jazyková nezávislost a jednotnost formátu báze znalostí. Formát XML zároveň umožňuje hierarchické uspořádání informací a je kompatibilní s formátem (X)HTML.

Formát XML navíc do jisté míry podporuje zpětnou kompatibilitu formátu, protože informace, které mohou být v novější verzi doplněny, mohou být starší verzí programu NPSCore jednoduše ignorovány. Z tohoto důvodu je u vrcholového elementu uvedena i verze formátu báze znalostí.

V novém formátu *.nps bylo zachováno rozdělení z původního formátu. Stejně jako *.n32 obsahuje záhlaví obsahující metadata o konkrétní bázi znalostí a tělo obsahující samotnou definici uzlů báze znalostí a jejich parametry.

Obecný obsah formátu báze znalostí je tedy v následující podobě:

```
<?xml version="1.0" encoding="utf-8"?>
<knowledge_base version="1.0">
     <head default_answers="3">
     …
     </head>
     <body>
     …
     </body>
</knowledge_base>
```

kde:

`knowledge_base`	je vrcholový element souboru s bází znalostí
`version`	je verze formátu souboru aktuálně ve verzi 1.0
`head`	je element obsahující metadata o bázi znalostí
`default_answers`	je atribut obsahující počet defaultních odpovědí
`body`	je element obsahující definici uzlů báze znalostí.

V následujících kapitolách je uveden popis formátu báze znalostí a význam jednotlivých elementů.

3.8.1 Hlavička báze znalostí

Obdobně jako u formátu *.n32 jsou v záhlaví souboru uvedeny následující informace: Název báze znalostí, Popis báze znalostí a Defaultní počet odpovědí. Je zde doplněn seznam podporovaných jazyků. Veškeré texty jsou uvedeny v elementu `content` popsaném v kapitole 3.5.2 Obecný formát zobrazovaných informací. Obecně je tedy formát záhlaví v následujícím tvaru:

```
<head default_answers="3">
     <cultures>
          <culture>cs-CZ</culture>
          …
     </cultures>
     <name>
          {content}
          …
     </name>
     <description>
          {content}
          …
     </description>
</head>
```

kde:

`default_answers`	je implicitní počet nabízených odpovědí v případě, že u dotazu není odpověď definována
`name`	je element obsahující název báze znalostí. Obsah je uzavřen do elementu *`{content}`* popsaného

	v předchozích kapitolách. Pokud je k dispozici více jazykových verzí, opakují se elementy *{content}* na stejné úrovni pro všechny jazykové mutace.
`cultures`	je množina identifikátorů všech jazykových mutací podporovaných v dané bázi znalostí
`description`	je element obsahující popis báze znalostí, obsah je formátován stejně jako v případě elementu `name`.

3.8.2 Definice uzlů báze znalostí v souboru *.nps

V této části je definována množina uzlů znalostní báze. Jedná se o pole elementů, které obsahují jednotlivé atributy. Obecně je tedy formát těla v následujícím tvaru:

```
<body>
     <node{nodeattributes}>
          {nodeSubElements}
     </node>
     …
</body>
```

kde:

`node`	je element obsahující kompletní definici jednoho uzlu v bázi znalostí
{nodeattributes}	jsou parametry uzlu, jejich popis je v následující
{nodeSubElements}	kapitole.

3.8.3 Společné parametry uzlů

Základní společné prvky

Všechny typy uzlů mají společnou základní množinu parametrů, které musí nebo mohou být uvedeny. Následující příklad uzlu obsahuje všechny společné parametry:

```
<node literal="{ID}" probability="{p}" type="{type}">
     <comment>
          {content}
          …
     </comment>
</node>
```

kde:

`literal` je atribut obsahující element obsahující jednoznačný identifikátor uzlu. Jeho forma zůstala stejná jako v případě souboru *.n32. Jde o text obsahující velká a malá písmena bez diakritiky, číslice a znak „_“. Literál nesmí začínat číslicí. Délka literálu je omezena na maximálně 8 znaků.

`probability` je výchozí hodnota pravděpodobnosti uzlu před započetím konzultace. Povolený rozsah je v intervalu <0,00, 100,00> včetně.

`comment` je nepovinný komentář uzlu. V případě dotazovatelného uzlu se jedná o obsah (text) otázky a v případě cílové hypotézy se jedná o zobrazovaný obsah odpovědi.

`type` je identifikátor typu uzlu. Je uveden v textové podobě a může nabývat jedné z následujících hodnot:

`"ancillary"`	– pomocný uzel
`"hypotesis"`	– cílová hypotéza
`"query"`	– dotazovatelný uzel přímý
`"question"`	– dotazovatelný uzel
`"query_quantitative"`	– dotazovatelný uzel kvantitativný
`"query_usual"`	– dotazovatelný uzel běžný

Pravidla

U všech typů uzlů je možné navíc definovat i pravidla, která ovlivňují hodnotu uzlu na základě změn hodnoty pravděpodobnosti jiných uzlů. Všechna pravidla jsou uzavřena v elementu `relationships`. Formát pravidel je definovaný s následující syntaxí:

```
<relationships>
     <relationship type="{relType}" alpha="{alpha}">
          <negative literal="{ID}"/>
          <positive literal="{ID}"/>
          …
     </relationship>
     …
```

```
</relationships>
```

kde:

`relationships`	je nepovinný element obsahující všechna pravidla daného uzlu
`relationship`	představuje jedno konkrétní pravidlo
`type`	představuje typ pravidla. Hodnota je textová a může nabývat dvou hodnot: `"conjunction"` – konjunkce „&“ `"disjunction"` – disjunkce „|“
`alpha`	představuje sílu vazby mezi pravidlem a ovlivňovaným uzlem. Povolený rozsah je v intervalu <0,00, 100,00> včetně.
`positive`	je element obsahující literál, který vstupuje do vazby v nezměněné formě. Atribut `literal` je poté identifikátor tohoto uzlu.
`negative`	je element obsahující literál, jehož hodnota pravděpodobnosti je před vstupem do vazby negována. vstupuje do vazby v nezměněné formě. tribut `literal` je poté identifikátor tohoto uzlu.

3.8.4 Kontextové vazby

U všech typů dotazovatelných uzlů je možné definovat kontext otázky. Formát kontextu je definovaný jako element `contexts` s následující syntaxí:

```
<contexts>
     <less threshold="50.00" literal="{ID}"/>
     <greather threshold="50.00" literal="{ID}"/>
     …
</contexts>
```

kde:

`contexts`	je nepovinný element obsahující všechny kontextové vazby daného uzlu
`less`	představuje jednu kontextovou vazbu typu menší než. Ve verzi *.n32 odpovídá znaku “<“. Dotaz se tedy může položit tehdy, pokud hodnota

uzlu *{ID}* v parametru `literal` je větší nebo rovna hodnotě v atributu `threshold`.

3.8.5 Odpovědi u dotazovatelného uzlu běžného a přímého

U dotazovatelného uzlu běžného a přímého je možné využít buď defaultní odpovědi popsané v předchozích kapitolách nebo definovat text odpovědi explicitně. V takovém případě se uživateli během konzultace zobrazí pouze explicitně definované odpovědi. Tyto odpovědi se definují pomocí elementu `answers` s následující syntaxí:

```
<answers>
     <answer probability="95.00">
          {content}
          …
     </answer>
          …
</answers>
```

kde:

`answers`	je nepovinný element obsahující všechny definované explicitní odpovědi daného dotazovatelného uzlu
`answer`	představuje jednu z možných odpovědí daného uzlu
`probability`	je hodnota pravděpodobnosti, která se použije pro složení s pravděpodobností uzlu v případě, že uživatel zvolí během konzultace tuto odpověď. Povolený rozsah je v intervalu <0,00, 100,00> včetně.

3.8.6 Odpovědi u dotazovatelného kontextového uzlu

U dotazovatelného uzlu kontextového je nutné vždy explicitně definovat odpovědi. Tyto odpovědi se stejně jako v předchozím případě definují pomocí elementu `answers`, ovšem místo pravděpodobnosti je uveden literál uzlu, kterému se v případě zvolení odpovědi nastaví pravděpodobnost o hodnotě 100.

```
<answers>
     <answer literal="{ID}">
          {content}
          …
     </answer>
     …
</answers>
```

4 OVĚŘENÍ FUNKČNOSTI SW

4.1 Statická analýza

V průběhu vývoje se pro překlad programu používal překladač Clang včetně vlastní statické analýzy kódu. Během tvorby programu byly samozřejmě postupně napravovány všechny varování překladače. To ovšem nedává dostatečnou záruku, že v programu se nebudou vyskytovat chyby, které je schopná statická analýza odhalit. Z tohoto důvodu byl pravidelně program kontrolován pomocí nástroje statické analýzy CppCheck, zmíněném v předchozích kapitolách.
Tento nástroj umožňuje integraci do vývojového prostředí Visual Studio. Díky této integraci je možné automaticky spustit analýzu kódu v momentě, kdy dojde k uložení změn v libovolném souboru projektu. Díky tomu je prakticky okamžitě možné dostat zpětnou vazbu o případných potenciálních chybách v zdrojových kódech.
Během vývoje byly všechny potenciální chyby v kódu odstraněny. Zůstaly zde ovšem chyby, které CppCheck odhalil v knihovně RapidXML. Tyto chyby byly po prozkoumání považovány za bezvýznamné a jejich opravou by se v budoucnu znemožnil případný upgrade na novější verzi knihovny RapidXML.

4.2 Systémové testy

V rámci systémových testů bylo provedeno ověření pomocí porovnání původním systémem NPS32 a nového programu NPSCore jednak s bází ve formátu *.n32 a jednak ve formátu *.nps. V případě rozdílu ověříme správnou hodnotu pomocí matematického výpočtu.
Pro ověření byly zvoleny dvě báze znalostí:

Automobil.n32 (demonstrační báze znalostí)

Brambory 2000.n32 (reálná báze znalostí; vzhledem k tomu, že v této bázi znalostí je velké množství cílových hypotéz, budeme sledovat pouze vývoj u tří libovolně zvolených hypotéz).

Pro každou z bází byly zvoleny série různých odpovědí a výsledné hodnoty hypotéz poté zaneseny do tabulky pro přehledné porovnání.

4.2.1 Ověření prostřednictvím báze Automobil.n32

V této bázi jsou celkem 3 otázky:

Tab. 4.1 Otázky v bázi znalostí Automobil.32

Otázka	Uzel	Text otázky
1	`Starter`	Je při otočení klíčkem v zapalování slyšet startér?
2	`Radio`	Hraje po zapnutí autorádio?
3	`Nadrz`	Co ukazuje měřič paliva v nádrži?

Sekvence 1

Pro dané otázky zvolíme následující sekvenci odpovědí:

Tab. 4.2 1. Sekvence testovacích odpovědí pro bázi Autmobil.n32

Otázka	Uzel	Zvolená odpověď
1	`Starter`	Ne
2	`Radio`	Ano
3	`Nadrz`	Nádrž je prázdná

Po vyhodnocení všech tří konzultací můžeme vytvořit následující tabulku:

Tab. 4.3 Tabulka výsledků hypotéz pro 1. Sekvenci testovacích odpovědí báze Autmobil.n32

	NPS32			NPSCore (*.n32)			NPSCore (*.nps)		
Hypotéza	Vybitá baterie	Palivo	Porucha	Vybitá baterie	Palivo	Porucha	Vybitá baterie	Palivo	Porucha
Začátek	50	50	50	50	50	50	50	50	50
Otázka 1	50	50	50	50	50	50	50	50	50
Otázka 2	9,1	50	50	9,1	50	50	9,1	50	50
Otázka 3	9,1	90,9	11,3	9,1	90,9	11,3	9,1	90,9	11,3

Z výsledků konzultace vidíme, že koncové hodnoty hypotéz i dílčí výsledky se shodují. Tento test nám tedy prošel úspěšně.

Sekvence 2

Opět zvolíme sekvenci odpovědí:

Tab. 4.4 2. Sekvence testovacích odpovědí pro bázi Autmobil.n32

Otázka	Uzel	Zvolená odpověď
1	`Starter`	Ano
2	`Nadrz`	Paliva je dostatek

Po vyhodnocení všech tří konzultací pro druhou sekvenci máme hodnoty hypotéz:

Tab. 4.5 Tabulka výsledků hypotéz pro 2. Sekvenci testovacích odpovědí báze Autmobil.n32

	NPS32			NPSCore (*.n32)			NPSCore (*.nps)		
Hypotéza	Vybitá baterie	Palivo	Porucha	Vybitá baterie	Palivo	Porucha	Vybitá baterie	Palivo	Porucha
Začátek	50	50	50	50	50	50	50	50	50
Otázka 1	9,1	50	50	12,8	50	50	12,8	50	50
Otázka 2	9,1	9,1	88,7	12,8	9,1	85,3	12,8	9,1	85,3

Z této tabulky vidíme, že výsledky pro program NPS32 a NPSCore se rozcházejí v případě hypotézy `Vybitá baterie` a `Porucha`. Zkusíme tedy spočítat hodnotu hypotézy Vybitá baterie:

Do cílové hypotézy pomocí pravidla vstupují uzly `Starter` a `Radio`. Pro tyto uzly tedy nejprve určíme hodnoty po zodpovězení otázky pomocí vztahů (2.1) a (2.5):

Pro uzel `Starter`:

$$(T,F)^{[1]}_{Starter} = (T,F)^{[0]}_{Uzel} \cdot (T,F)^{[1]}_{Odp} = (1,1) \cdot (0{,}05263, 1)$$
$$(T,F)^{[1]}_{Starter} = (0{,}05263, 1) \qquad (4.1)$$

Pro uzel `Radio` je hodnota původní, protože u něho je kontextová vazba, která nebyla naplněna, a proto se dotaz uzlu `Radio` během konzultace nepoložil a hodnota zůstala na (1,1). Následně pomocí vztahu (2.11) spočítáme hodnotu pravidla:

$$(T,F)^{[1]}_{Pravidlo} = \left(-(T,F)^{[1]}_{Starter}\right) \& \left(-(T,F)^{[1]}_{Radio}\right)$$
$$(T,F)^{[1]}_{Pravidlo} = (1, 0{,}05263) \& (1,1) \qquad (4.2)$$
$$(T,F)^{[1]}_{Pravidlo} = (1, 0{,}05263)$$

Následně dle vztahu (2.7) konečně můžeme spočítat hodnotu cílové hypotézy `Vybitá baterie` $\alpha = 0{,}9$:

$$(T,F)^{[1]}_{Baterie} = \left(\frac{T^{[0]}_{Pravidlo}}{T^{[1]}_{Pravidlo}}, \frac{F^{[0]}_{Pravidlo}}{F^{[1]}_{Pravidlo}}\right) \xrightarrow{\alpha} (T,F)^{[0]}_{Baterie} \qquad (4.3)$$

$$T_{B1} = \left(\frac{0{,}9.1 - 0{,}9 + 1}{0{,}9.1 - 0{,}9 + 1}\right) 1 = 1$$

$$F_{B1} = \left(\frac{0{,}9.0{,}05263 - 0{,}9 + 1}{0{,}9.1 - 0{,}9 + 1}\right) 1 = 0{,}1474$$

Pokud využijeme vztah (2.1), tak výsledná hodnota pravděpodobnosti hypotézy Vybitá baterie potom je: 12,8. Pokud bychom tento výpočet provedli pro uzel porucha, dostali bychom hodnotu 85,3. Z těchto výpočtů vyplývá, že hodnota, kterou dává program NPSCore, je matematicky správná. Ovšem rozdíl mezi programem NPS32 a NPSCore je podstatný.

Dohledáváním možných příčin se podařilo zjistit, že v programu N32 je chyba, která zapříčiní, že pokud se použije přesná hodnota pravděpodobnosti v explicitně definovaných odpovědích, je tato hodnota ignorována a použije se hodnota 100 %. Pokud tuto hodnotu dosadíme do báze znalostí, začnou nám výsledky konzultace pro 2. sekvenci odpovědí odpovídat hodnotám z programu NPSCore.

4.2.2 Ověření prostřednictvím báze Brambory 2000.n32

V této bázi je celkem 20 otázek:

Tab. 4.6 Otázky v bázi znalostí Brambory 2000

Otázka	Uzel	Text otázky
1	`OBLAST`	Zvolte si pěstitelskou oblast, pro niž chcete provést výběr!
2	`VYNOS`	Bude na honu využit vysoký výnosový potenciál odrůdy?
3	`RANOST`	Požadujete omezení výběru odrůdy podle skupiny ranosti?
4	`SKLIZEN`	Máte zájem upřednostnit odrůdy pro:
5	`PRODHL`	Jakou produkci hlíz máte zájem?
6	`VHODNOST`	Chcete upřednostnit odrůdy vhodné jako:

7	VARTA	Jakou bližší specifikaci konzistence hlíz požadujete
8	VARTC	Jakou bližší specifikaci konzistence hlíz požadujete
9	VYROBA	Pro jakou výrobu chcete upřednostnit odrůdy?
10	INFOLOUP	V současné době je nejvhodnější odrůdou pro loupání odrůda
11	POM1_	V této skupině ranosti není povolena žádná vhodná odrůda
12	PLISEN	Chcete bez ohledu na jiné vlastnosti upřednostnit odrůdy
13	STRUP	Je možno s ohledem na pH, strukturu půdy a další podmiňující faktory předpokládat zvýšený výskyt obecné strupovitosti ?
14	POSKOZ	Chcete upřednostnit odrůdy odolnější mechanickému poškození pro případ poranění hlíz při sklizni ?
15	POKRAC1	Požadujete ukončit expertízu nebo vybrané odrůdy dále setřídit:
16	POKRAC2	Požadujete odrůdy dále setřídit podle:
17	VELHLIZ	Velikost Hlíz
18	TVARHLIZ	Tvar Hlíz
19	HLOCEK	Hloubka oček
20	BARDUZ	Barva dužniny

Pro dané otázky zvolíme následující sekvenci odpovědí:

Tab. 4.7 1. Sekvence testovacích odpovědí pro bázi Brambory 2000

Otázka	Uzel	Zvolená odpověď
1	OBLAST	Ranobramborářská
2	VYNOS	Snad Ano
3	RANOST	Letní
4	PRODHL	Zušlechtěné produkty
5	VYROBA	Lupínky
6	PLISEN	Určitě ne
7	STRUP	Nevím
8	POSKOZ	Spíše ne
9	POKRAC1	Jeden Znak
10	POKRAC2	Tvar Hlíz
11	TVARHLIZ	Nepožaduji

Po vyhodnocení všech tří konzultací můžeme vytvořit následující tabulku:

Tab. 4.8 Tabulka výsledků hypotéz pro Sekvenci testovacích odpovědí báze Brambory 2000

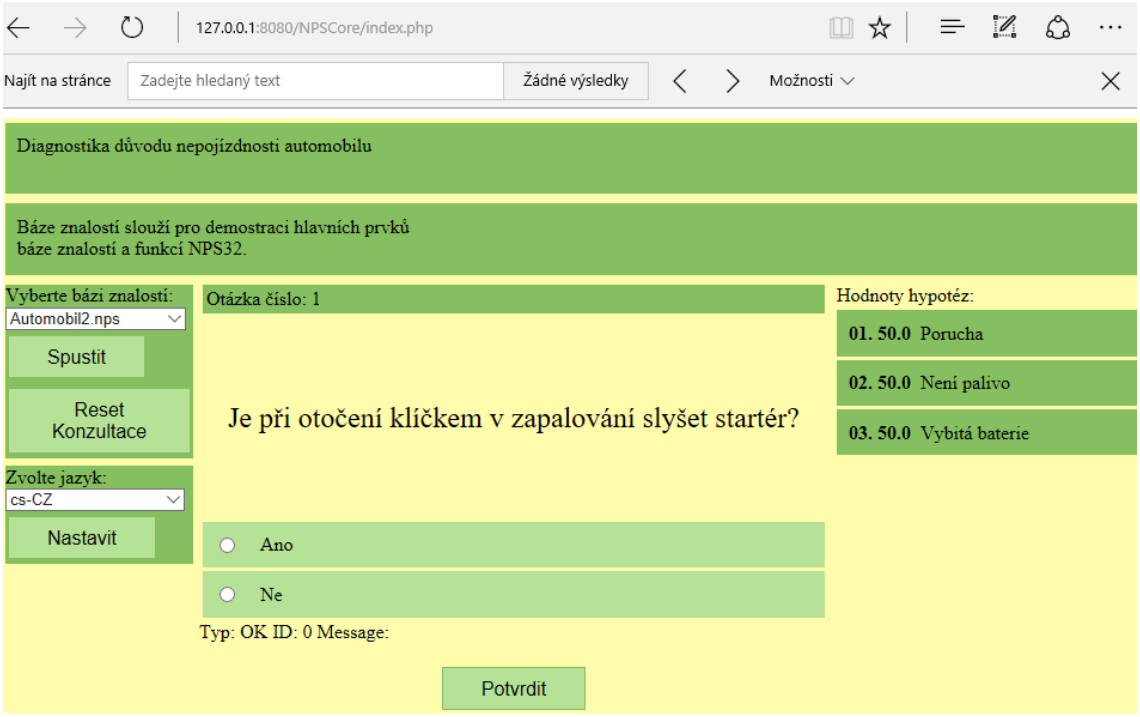
	NPS32			NPSCore (*.n32)			NPSCore (*.nps)		
Hypotéza	ACCENT	VERA	FAMBO	Vybitá baterie	Palivo	Porucha	Vybitá baterie	Palivo	Porucha
Začátek	30	30	30	30	30	30	30	30	30
Otázka 1	45,2	45,2	36,1	45,2	45,2	36,1	45,2	45,2	36,1
Otázka 2	52,0	47,3	34,2	52,0	47,3	34,2	52,0	47,3	34,2
Otázka 3	79,5	76,2	0,5	79,5	76,2	0,5	79,5	76,2	0,5
Otázka 4	79,5	76,2	0,5	79,5	76,2	0,5	79,5	76,2	0,5
Otázka 5	3,7	92,0	0,0	3,7	92,0	0,0	3,7	92,0	0,0
Otázka 6	3,8	92,2	0,0	3,8	92,2	0,0	3,8	92,2	0,0
Otázka 7	3,8	92,2	0,0	3,8	92,2	0,0	3,8	92,2	0,0
Otázka 8	3,8	92,4	0,0	3,8	92,4	0,0	3,8	92,4	0,0
Otázka 9	3,8	92,4	0,0	3,8	92,4	0,0	3,8	92,4	0,0
Otázka 10	3,8	92,4	0,0	3,8	92,4	0,0	3,8	92,4	0,0
Otázka 11	3,8	92,4	0,0	3,8	92,4	0,0	3,8	92,4	0,0

Z výsledků konzultace vidíme, že koncové hodnoty hypotéz i dílčí výsledky se shodují. Tento test nám tedy prošel úspěšně.

4.3 Demonstrační část webové aplikace.

Pro snadnější práci a demonstraci funkčnosti nového jádra NPSCore bylo vytvořeno demonstrační rozhraní v PHP, umožňující provést konzultaci se zvolenou bází znalostí. Tato aplikace slouží pouze k testování daného rozhraní a ověření funkčnosti za pomoci reálné konzultace. Nehodí se pro distribuci a veřejné použití. Jako webový server pro tuto aplikaci posloužil Apache v distribuci EASYPHP DEVSERVER. Tento program je volně ke stažení, disponuje jednoduchou administrací a podporou PHP, Python, MySQL a mnoho dalších.

V jazyce PHP byla vytvořena testovací aplikace, která spouští program NPSCore a předává mu sérii příkazů, které zajišťují konzultaci. Z výsledků konzultace poté sestaví webovou stránku a předají ji klientu.



Obrázek 5 Příklad konzultace v okně prohlížeče

Obrázek 6 Příklad konzultace v okně prohlížeče obsahujícím grafické prvky

5 ZÁVĚR

Protože dostupné materiály popisující chování programu NPS32 neobsahovaly informace potřebné pro navázání práce na programu NPS32, byly v rámci této práce nejprve detailně zmapovány funkce a chování programu NPS32 a formát báze znalostí *.n32. Tyto informace jsou shrnuty v teoretické části. Následně byl vytvořen návrh nové koncepce nástupce programu NPS32 a vytvořen program NPSCore a knihovna NPSEngineLibrary. Tyto programy využívají původní matematiku z programu NPS32, ovšem jedná se o zcela novou implementaci. Jsou plně zpětně kompatibilní a umožňují práci se soubory v původním formátu *.n32 i v novém formátu *.nps, který byl v rámci této práce rovněž navržen. Nový formát je ve standardní syntaxi XML a umožňuje další rozšiřitelnost a podporu více jazyčných verzí. Pro knihovnu NPSEngineLibrary bylo zvoleno rozhraní pro komunikaci s okolím prostřednictvím dotazů a odpovědí ve formátu XML. Pro ověření správné funkce byly použity testy na úrovni kódu - statická analýza kódu a systémové testy, kdy byl do systémů NPS32 a NPSCore vložena stejná báze znalostí a byly porovnávány výsledky konzultace. Vzhledem k tomu že se výsledky obou systémů víceméně shodují, dá se říct, že se podařilo dokázat, že daný program funguje správně. Jedinou výjimkou je problém v případě explicitně definovaných odpovědí u dotazovatelného uzlu. Pokud je u tohoto typu uvedena pravděpodobnost s přesnou hodnotou, je tato hodnota ignorována a použije se automaticky 100. Pokud tuto chybu v původním programu NPS32 kompenzujeme, shodují se výsledky konzultací úplně.

5.1 Návrh pro další práci:

V rámci této práce byl vytvořen a ověřen program NPSCore a knihovna NPSLibrary a ověřena použitelnost těchto komponent pro prostředí webových technologií. V další práci by měl být vytvořen informační systém pro webové rozhraní a business logika, která zajistí provoz pro více klientů a umožní administraci a autorizaci klientů ke konzultacím.

V rámci této úpravy by bylo vhodné přeložit program NPSCore i pro prostředí UNIX.

Dále by bylo vhodné doplnit načtení stavu z více než jednoho souboru a uložení pouze stavu konzultace dle požadovaného zdroje odpovědí. V současnosti jsou všechny tyto informace vloženy do jednoho souboru odpovídajícího celé historii konzultace. Tím se nahradí funkce šablon v původní verzi programu NPS32.

V rámci této práce nebyly řešeny externí odpovědi, pouze byl detailně popsán formát příslušných řádků. V rámci další práce by možná bylo vhodné začlenit tuto funkcionalitu do stavu báze znalostí jako další typ zdroje dat.

Literatura

[1] MAŘÍK Vladimír, ŠTĚPÁNKOVÁ Olga, LAŽANSKÝ Jiří: *Umělá inteligence 2.* Academia, Praha, 1997. ISBN 80-200-0504-8

[2] BARTOŠÍK Martin: *Tvorba Báze znalostí - Diplomová práce*. VUT, Brno, 2004.

[3] PATTON,R.: *Testování softwaru*, vyd. Computer Press, Praha, 2002, ISBN 80-7226-636-5

[4] PRATA Stephen: *Mistrovství v C++ 4. aktualizované vydání*, vyd. Computer Press, Praha, 2013, ISBN 978-80-251-3828-1

[5] DLOUHÝ Radek: *PHP v příkladech*, vyd. Computer Media, Praha, 2007, ISBN 80-86686-83-3

[6] ŠOCHOVÁ Zuzana, KUNCE Eduard: *Agilní metody řízení projektů*, vyd. Computer Press, Praha, 2014, ISBN 9788025142479

Seznam symbolů, veličin a zkratek

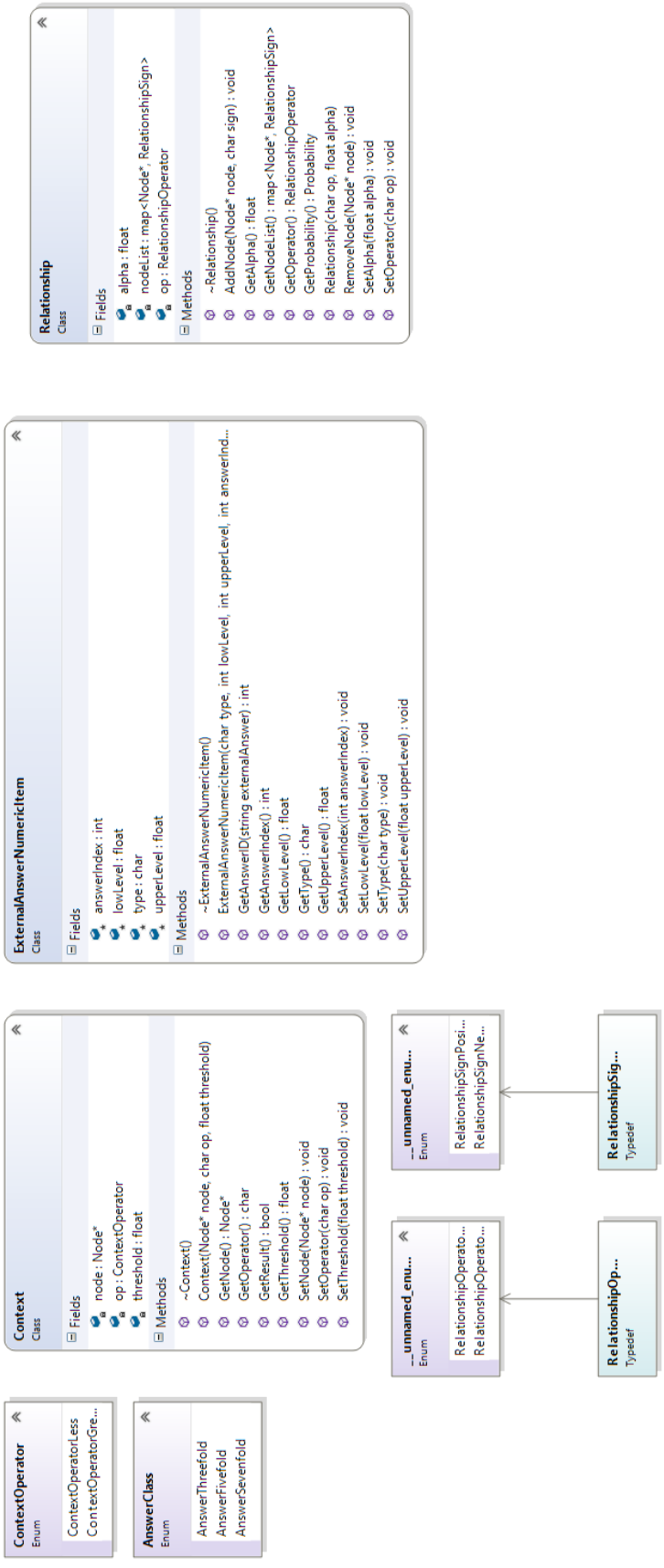
CSS	-	Cascading Style Sheets
FEKT	-	Fakulta elektrotechniky a informatiky
FEKT	-	Fakulta elektrotechniky a komunikačních technologií
HTML	-	HyperText Markup Language
HTTP	-	Hypertext Transfer Protocol
IIS	-	Internet Information Services
SŘBD	-	Systém řízení báze dat
UTF-8	-	Unicode Transformation Format
VUT	-	Vysoké učení technické v Brně
XML	-	eXtensible Markup Language
XHTML	-	eXtensible HyperText Markup Language

Seznam příloh

5.2 Hierarchie dědičnosti uzlů

5.3 Hierarchie uživatelských odpovědí

5.4 Vazby mezi uzly



5.5 Třídy pro kontrolu konzultace a báze znalostí