TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Tvorba expertního systému pro podporu řešení helpdeskových požadavků

Diplomová práce

Studijní program: N6209 – Systémové inženýrství a informatika
Studijní obor: 6209T021 – Manažerská informatika

Autor práce: **Bc. Michal Dostál**
Vedoucí práce: Ing. Dana Nejedlová, Ph.D.

Liberec 2019

TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Akademický rok 2018 / 2019

Zadání diplomové práce

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Michal Dostál**
Osobní číslo:	E17000266
Studijní program:	N6209 Systémové inženýrství a informatika
Studijní obor:	N6209T021 – Manažerská informatika
Zadávající katedra:	katedra informatiky
Vedoucí práce:	Ing. Dana Nejedlová, Ph.D.
Konzultant práce:	Ing. Petr Motl vedoucí systémový programátor, OR-CZ spol. s r. o.
Název práce:	**Tvorba expertního systému pro podporu řešení helpdeskových požadavků**

Zásady pro vypracování:

1. Teorie expertních systémů.
2. Současné přístupy k zákaznické podpoře v oblasti informačních technologií.
3. Vývoj expertního systému.
4. Zhodnocení vývoje expertního systému.

Seznam odborné literatury:

BRUCKENER, Tomáš, Jiří VOŘÍŠEK, Alena BUCHALCEVOVÁ, Iva STANOVSKÁ, Dušan CHLAPEK a Václav ŘEPA. 2012. *Tvorba informačních systémů: principy, metodiky, architektury*. Praha: Grada Publishing. ISBN 9788024741536.

ALOR-HERNÁNDEZ, Giner a Rafael VALENCI-GARCÍA. 2017. *Current Trends on Knowledge-Based Systems*. Berlin: Springer. ISBN 978-3-319-51904-3.

BAYSAL, Olga, Michael W. GODFREY a Robin COHEN. 2009. A bug you like: A framework for automated assignment of bugs. In: *IEEE International Conference on Program Comprehension*. s. 297–298. ISSN 1092-8138.

BOTHA, R. a A. LEONARD. 2012. Organizational issues and its impact on the performance of service desk staff members in providing quality service. In: *Proceedings of PICMET '12: Technology Management for Emerging Technologies 2012*. s. 3131–3139. ISBN 978-1-890843-25-0.

PROQUEST. 2018. *Databáze článků ProQuest* [online]. Ann Arbor, MI, USA: ProQuest. [cit. 2018-09-30]. Dostupné z: http://knihovna.tul.cz/

Rozsah práce:	min. 65 normostran
Forma zpracování:	tištěná / elektronická
Datum zadání práce:	1. října 2018
Datum odevzdání práce:	31. srpna 2020

Technická univerzita v Liberci Ekonomická fakulta

L.S.

prof. Ing. Miroslav Žižka, Ph.D.
děkan Ekonomické fakulty

doc. Ing. Klára Antlová, Ph.D.
vedoucí katedry

V Liberci dne 31. října 2018

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že texty tištěné verze práce a elektronické verze práce vložené do IS STAG se shodují.

30. 3. 2019 Bc. Michal Dostál

Poděkování

Rád bych tímto poděkoval Ing. Daně Nejedlové, Ph.D. a Ing. Petru Motlovi za jejich čas, pomoc a cenné rady při zpracování této diplomové práce. Zároveň bych rád poděkoval mé rodině za jejich podporu po celou dobu mého studia.

Anotace

Diplomová práce Tvorba expertního systému pro podporu řešení helpdeskových požadavků popisuje tvorbu konkrétního expertního systému HDES pracujícího s helpdeskovými požadavky, napomáhajícího ke kategorizaci těchto požadavků a detekci požadavků duplicitních.

Teoretická část práce obsahuje teoretický základ pro expertní systémy. Jsou popsány různé způsoby reprezentace znalostí v bázi znalostí, jak funguje inferenční mechanismus a jaké jsou další moduly používané v expertních systémech. Tato část se dále zabývá současnými trendy v zákaznické podpoře se zaměřením na aktuální technologie.

V praktické části je popsána analýza, návrh a implementace pravidlového expertního systému HDES. Popisován je proces, jak systém zpracovává data a jakým způsobem jsou technologicky řešeny jeho jednotlivé části.

Klíčová slova

Expertní systém, reprezentace znalostí, báze znalostí, inferenční mechanismus, CLIPS, help desk, zákaznická podpora.

Annotation

This master's thesis Creation of Expert System for the Support of Fulfillment of Helpdesk Service Requests describes the creation of expert system HDES which works with helpdesk requests, helps with their categorization and detection of duplicates.

Theoretical part of this thesis covers theoretical background of expert systems. It describes ways of knowledge representation in the knowledge base, how the inference engine works and which other modules are often used in expert systems. Theoretical part also covers trends in customer support with the focus on present technologies.

Practical part describes the analysis, design and implementation of the rule-based expert system HDES. It covers how data are processed by the system and how individual parts of the system are implemented.

Key Words

Expert System, Knowledge Representation, Knowledge Base, Inference Engine, CLIPS, Help Desk, Customer Support.

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

API	Application Programming Interface
BPMN	Business Process Model and Notation
CSS	Cascading Style Sheets
CSV	Comma Separated Values
FAQ	Frequently Asked Questions
FB	Facebook
GPL	General Public License
GUI	Graphical User Interface
HD	Help Desk
HDES	Help Desk Expert System
HDI	Help Desk Institute
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IM	Instant Messaging
IoT	Internet of Things
IT	Information Technology
ITIL	Information Technology Infrastructure Library
IVR	Interactive Voice Response
JSON	JavaScript Object Notation
OMG	Object Management Group
RDF	Resource Description Framework
SLA	Service Level Agreement
SQA	Social Question and Answer
UML	Unified Modeling Language
XML	eXtensible Markup Language

Úvod

Ve firmách vyvíjejících informační systémy či jiné softwarové produkty je oddělení technické podpory koncových uživatelů vitální součástí. Technická podpora má na starosti helpdeskové požadavky zákazníků a řeší jejich problémy s dodaným softwarem. Správa a řešení těchto helpdeskových požadavků vyžaduje mnoho práce a pokud je jich mnoho, snadno se může stát, že některý z přidělených řešitelů řeší identický problém, který byl již v minulosti vyřešen jiným pracovníkem. Dochází tak ke zbytečné ztrátě času. Řešením může být nasazení metod umělé inteligence. V oboru technické podpory se nabízí použití expertního systému. Administrátor helpdesku i přidělení řešitelé jednotlivých požadavků potřebují k řešení téměř expertní znalosti - mají vynikající znalost v konkrétní doméně řešeného problému.

Cílem této diplomové práce je popsat tvorbu expertního systému, který by měl sloužit jako podpora při řešení helpdeskových požadavků ve firmě zabývající se vývojem informačních systémů a dodávkou IT služeb. Navrhovaný expertní systém by měl pomáhat pracovníkovi helpdesku v procesu řešení požadavků. Soustředí se hlavně na pomoc při detekci duplicitních požadavků, kdy při jejich nedetekování vzniká časová prodleva v opětovném hledání řešení helpdeskového požadavku. Dalším cílem této diplomové práce je čtenáře seznámit s teoretickým základem expertních systémů a současnými trendy v oboru zákaznické podpory.

První část této diplomové práce se nejprve zabývá teorií obecných expertních systémů a poté uvede průzkum současného stavu řešení zákaznické podpory. Druhá část diplomové práce popisuje tvorbu konkrétního expertního systému. Popsána je analýza výchozího stavu technické podpory, analýza navrhovaného systému a konkrétní implementace včetně popisu, jak expertní systém pracuje. V poslední části této práce je provedeno zhodnocení vytvořeného systému.

1 Teorie expertních systémů

Umělá inteligence je dnes velmi důležitým a rychle se vyvíjejícím oborem. Během 20. století vzniklo několik definic. Jedna z nejjednodušších je „*making computers think like people*“ (Giarratano a Riley, 2004), tedy přimět počítače myslet jako lidé. Tato definice má kořeny v Turingově testu (Turing, 1950). Cílem tohoto testu je prověřit schopnost počítače napodobit inteligentní chování člověka. Test probíhá následovně: rozhodčí pokládá řadu otázek subjektu, u kterého neví, zda se jedná o počítač nebo člověka. Pokud rozhodčí nedokáže rozpoznat, zda ten, s kým pomocí klávesnice komunikuje, je počítač nebo člověk, systém prošel Turingovým testem.

O umělé inteligenci jako o vědě jako takové se začalo hovořit v 50. letech 20. století, kdy se konala konference nazvaná „The Dartmouth Summer Research Project on Artificial Intelligence“ pořádaná americkou Dartmouth College v New Hapshire (Mařík et al., 1993). Zde se také tato nová vědní disciplína pojmenovala artificial intelligence – umělá inteligence, a to díky návrhu Johna McCarthyho, organizátora konference. Na konferenci byli pozváni odborníci z různých odvětví lidské činnosti, především to ale byli odborníci z oborů elektrotechniky, psychologie a matematiky. Vývoj umělé inteligence pak probíhal v několika obdobích lišících se optimismem a praktickými výsledky výzkumu. V 50. letech byl vyvinut perceptron - model neuronu schopný simulovat činnost signální soustavy živočichů, čímž byl položen základ výzkumu klasifikace a rozpoznávání (automatického zařazování do tříd). Došlo také ke vzniku programovacího jazyka LISP. Následovalo velmi teoretické období, kdy nebylo dosahováno mnoha reálných výsledků. Přestože je období do druhé poloviny 70. let nazýváno jako „doba ledová“, byl v roce 1972 vyvinut programovací jazyk Prolog a započala tvorba expertních systémů. Jazyk Prolog je oblíbeným logickým programovacím jazykem pro umělou inteligenci. Mezi první expertní systémy se řadí dva diagnostické systémy MYCIN a PROSPECTOR. MYCIN byl využíván na diagnostiku onemocnění z krve a PROSPECTOR v oblasti geologie pro odhalení rudných ložisek. 80. léta se v oblasti umělé inteligence nesla ve znamení komerčního využití expertních systémů. Byl například vyvinut plánovací expertní systém DENDRAL, který se řadí mezi jedny z nejdéle využívaných systémů. Byl určený k vyhodnocování výsledků z hmotnostního spektrogramu.

Jak již bylo uvedeno, vývoj expertních systémů začal v 60. letech 20. století jako speciální typ umělé inteligence za účelem řešení komplexních problémů v úzké doméně. Profesor Edward Feigenbaum (Feigenbaum, 1981) ze Stanfordské univerzity definoval expertní systém jako inteligentní počítačový program, který využívá znalosti a inferenční procedury k řešení složitých problémů vyžadující expertizu řešitele. Jednodušeji lze definici vyjádřit jako počítačový systém, který emuluje rozhodovací schopnosti experta. Termín „emulace“ v tomto smyslu značí chování se jako lidský expert, a to ve všech aspektech. Emulace je významově silnější než simulace, která vyžaduje „chovat se jako expert“ jen v některých ohledech (Giarratano a Riley, 2004, s. 5). Podle profesora Feigenbauma (1992) termín „expert“ odkazuje na záměr tvůrce expertního systému vytvořit systém, jehož úroveň kompetence při řešení dané domény problémů bude schopna konkurovat výkonům lidských expertů (v této doméně problémů). Aby toho bylo docíleno, je zapotřebí, aby měl expertní systém k dispozici takové znalosti, které lidské experty odlišují od laiků. Feigenbaum (1992) dále uvádí, že expertní systémy jsou téměř vždy využívány jako „interaktivní intelektuální pomůcky“ pro osoby provádějící určitá rozhodnutí a nikoli jako autonomní systémy (tzv. agenti) bez jakéhokoli dohledu.

Učební text (Pokorný a Krišová, 2016) uvádí typy (třídy) problémů, jež jsou vhodné k řešení expertním systémem. Jedná se o interpretaci, predikci, diagnostiku, konstruování, plánování, monitorování, ladění či opravování, učení a řízení. Z nich pak vychází dva nejvýznamnější druhy expertních systémů – diagnostické a plánovací systémy. Diagnostický expertní systém se zabývá interpretací dat a snaží se určit nejlepší korespondenci s reálnými (zadanými) daty. Plánovací expertní systém pak pracuje s generátorem možných řešení pro zadaný problém. Systém vytvoří ohodnocený seznam přípustných řešení problému.

Základními dvěma prvky expertního systému jsou *báze znalostí* a *inferenční mechanismus*. Expertní systém ale nejčastěji obsahuje i tzv. vysvětlovací modul, kde systém zdůvodní řešení zadaného problému; či uživatelské prostředí (komunikační modul), skrz které uživatel se systémem komunikuje. Jednotlivými komponentami běžného expertního systému se zabývají podkapitoly 1.1, 1.2, 1.3 a 1.4.

1.1 Báze znalostí

Báze znalostí je základní a vitální součástí každého expertního systému. Oproti běžnému počítačovému programu, který sestává z algoritmů a datových struktur, expertní systém funguje na propojení znalostí a inferenčního procesu.

Báze znalostí v sobě může obsahovat jak faktické znalostí, tak heuristiky (Feigenbaum, 1992). Heuristiku lze jednoduše popsat jako řešení problému pomocí dříve získaných zkušeností (Giarratano a Riley, 2004). Heuristiky jsou významné například v právu nebo v medicíně, kdy je problém řešen na základě dřívějších podobných případů a zkušeností – tzv. precedentů.

Znalosti můžeme rozdělit na *a priori* znalosti a *a posteriori* znalosti (Giarratano a Riley, 2004). A priori znalost je nezávislá na smyslových zkušenostech a vjemech. Je to znalost, která je obecně považována za pravdivou a nemůže být jednoduše popřena. Jedná se zpravidla o logické výroky a matematické zákony. Naproti tomu a posteriori znalosti jsou odvozeny od smyslových zkušeností a vnímání. Zda jsou pravdivé či nepravdivé lze jednoduše dokázat senzorickým vjemem. Ne vždy jsou ale tyto znalosti spolehlivé, a tak je důležité poznamenat, že a posteriori znalosti mohou být popřeny na základě nových znalostí a zkušeností o dané skutečnosti.

Autoři skripta (Pokorný a Krišová, 2016) dělí znalosti na objektivní a subjektivní. Objektivní znalosti, někdy nazývané jako hluboké, jsou dostupné odborníkům například při studiu nějakého oboru. Za objektivní znalosti lze považovat např. matematické a fyzikální zákony. Přirovnat je lze k faktickým znalostem a a priori znalostem. Subjektivní (mělké) znalosti jsou získané poznáním a vlastní zkušeností v praxi.

Literatura (Giarratano a Riley, 2004) dále klasifikuje znalosti na *procedurální, deklarativní* a *tacitní*. Procedurální znalosti se řadí k znalostem o činnostech. Jednoduše je lze popsat jako „vědět, jak něco udělat“. U deklarativních znalostí pak víme, zda je něco pravda nebo nepravda. Tacitní znalost je skrytá, nevyslovená znalost, která se dá přirovnat např. k intuici nebo dovednosti.

1.1.1 Reprezentace znalostí

Formát reprezentace znalostí v bázi znalostí je klíčovým prvkem fungování expertního systému. Existuje několik variant reprezentací znalostí. Na základě těchto variant jsou i rozlišovány expertní systémy. Jedná se například o pravidlové systémy, rámcové systémy, systémy pracující se sémantickými sítěmi nebo s objekty.

Asi nejrozšířenějším způsobem reprezentace v bázi znalostí jsou *pravidla* – také někdy označována jako produkční pravidla. Existují různé formáty pravidel. Autor skripta (Dvořák, 2004) uvádí následující formát - viz Obrázek 1.

```
IF <předpoklad> THEN <závěr>
IF  <situace> THEN <akce>
IF <podmínka> THEN <závěr> AND <akce>
IF <podmínka> THEN <důsledek č. 1> ELSE <důsledek č. 2>
```

Obrázek 1: *Obecné formáty pravidla*
Zdroj: Vlastní zpracování dle Dvořák (2004)

Pravidlo se skládá z antecedentu, což je část za IF – tedy podmínka, předpoklad nebo situace; a konsekventu, části za THEN značící důsledkovou část, která nastane při splnění podmínky či předpokladu. Pravidlo IF E THEN H se často zapisuje jako $E \rightarrow H$, kde E značí anglický výraz evidence (důkaz) a H hypothesis (hypotéza).

Pomocí těchto pravidel jsou z faktů vyvozována další fakta. Vyvozování nových faktů se nazývá inference. U expertních systémů používajících k reprezentaci znalostí pravidla se k inferenci využívá pravidlo *modus ponens*, které říká, že jestliže platí daný předpoklad E a zároveň pravidlo $E \rightarrow H$, pak platí důsledek H (nový fakt). Toto se také nazývá přímé usuzování.

Dalším druhem možné reprezentace znalostí v bázi znalostí jsou *sémantické sítě*. Jedná se o ohodnocený orientovaný graf. Grafem rozumíme reprezentaci objektů, kde jsou znázorněny vztahy mezi jednotlivými objekty. Orientovaný graf znamená, že jednotlivé hrany (šipky) mají danou orientaci – tedy lze přesně identifikovat počáteční a koncový uzel. Díky sémantickým sítím lze lépe porozumět příčinám a událostem vyskytujících se v dané doméně problému a je tak umožněna lepší inference systému o tomto problému. Sémantická síť je považována za reprezentaci deklarativních znalostí (uvádí skutečnost/fakta). Podle (Giarratano

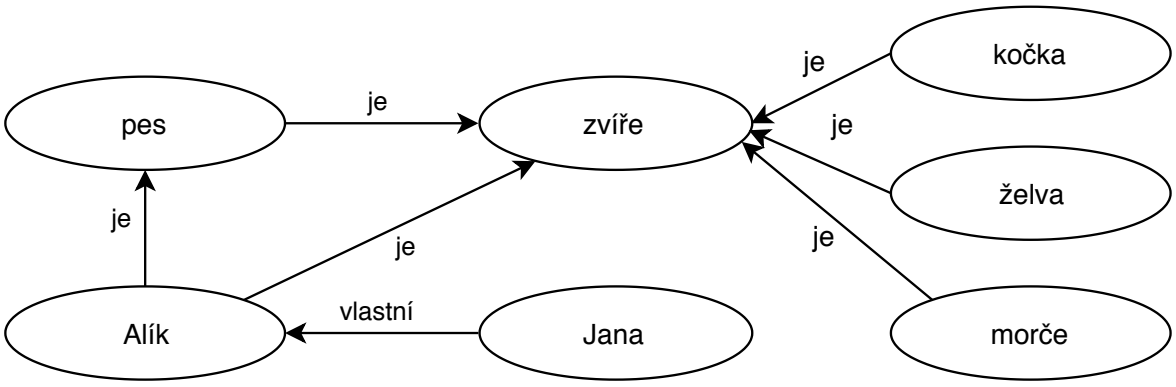
a Riley, 2004) se sémantické sítě nazývají také „propositional nets“, což v překladu může znamenat sítě tvrzení. Tvrzení je vždy pravda nebo nepravda a je atomické, tedy nedá se dále rozložit na řetězec atomických tvrzení spojených logickými operátory (spojkami). Giarratano a Riley dále uvádí, že sémantické sítě byly poprvé vyvinuty pro umělou inteligenci jako reprezentace lidského myšlení a porozumění jazyka. Při pohledu na Obrázek 2 lze vidět, že sémantická síť se skládá z jednotlivých uzlů (objektů) a šipek (vztahů). Uzly většinou představují fyzické objekty, koncepty nebo situace. Šipky pak značí vztahy mezi jednotlivými vztahy. V sémantické síti je také podporována dědičnost či tranzitivita. Tranzitivity si lze na Obrázku 2 všimnout u uzlů Alík, pes a zvíře. Pokud bychom použili klasickou notaci:

$$A \rightarrow B, B \rightarrow C, \text{ takže } A \rightarrow C \tag{1}$$

dostali bychom následující notaci:

$$\text{Alík} \rightarrow \text{pes}, \text{pes} \rightarrow \text{zvíře}, \text{takže Alík} \rightarrow \text{zvíře} \tag{2}$$

Na tomto příkladě lze také naznačit dědičnost. Je jasné, že pes dědí vlastnosti zvířete - tedy např. je živé, dýchá, k životu potřebuje potravu. To samé platí u Alíka, který samozřejmě dědí všechny vlastnosti psa - 4 nohy, ocas aj. Dědičnost však neplatí u všech uzlů v síti. V této ukázkové síti lze deklarovat, že mezi těmito uzly, kde existuje vztah nazvaný „je“, platí dědičnost z více obecného uzlu na uzel specializovaný.



Obrázek 2: *Příklad sémantické sítě*

Zdroj: Vlastní zpracování pomocí nástroje draw.io

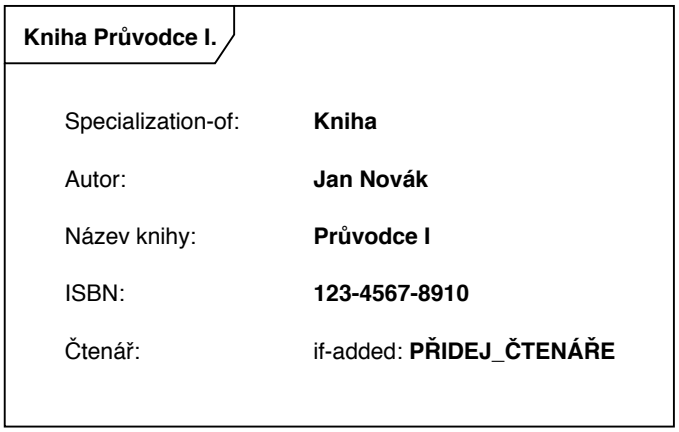
Giarratano a Riley (2004) uvádí, že je vhodné pro reprezentaci používat určité typy a názvy vztahů mezi uzly, a to z důvodu lepšího porozumění neznámé sítě pro různé lidi. Uvádí, že

dva běžné názvy vztahů jsou IS-A a AKO, což je zkrácený výraz pro A-KIND-OF. První z nich značí vztah „je“ použitý v Obrázku 2 a má znamenat „je instancí [třídy]“. Znamená to tedy, že tento vztah značí vztah instance k obecné třídě. Oproti tomu AKO má vyjadřovat vztah mezi jednotlivými obecnými třídami (uzly).

Dvořák (2004) uvádí, že výhodou sémantických sítí je jasné vyjádření faktů a vztahů. Naopak za nevýhodu považuje neexistenci standardizace jmen vazeb a interpretace.

Dalším způsobem reprezentace znalostí je reprezentace pomocí tzv. *rámců*. Rámcová reprezentace byla představena v odborném článku (Steels, 1978). Jedná se o deklarativní formu reprezentace. Rámec představuje informační strukturu, obsahující především název rámce a tzv. sloty, které jsou naplňovány informacemi souvisejícími s daným rámcem. Sloty lze považovat za atributy (vlastnosti, data) objektu. Obsahem slotu může být například aktuální hodnota, implicitní hodnota nebo rozsah možných hodnot. Pomocí rámců lze reprezentovat stereotypní situace činnosti (tzv. scénáře). Rámce dále mohou obsahovat speciální sloty se zvláštními procedurami (v tomto případě rámce spojují deklarativní a procedurální formu reprezentace znalostí), jako jsou například `if-added` nebo `if-deleted`, které jsou aktivovány v případě, že nastane daná situace. Rámec může být také generický nebo specifický. Generický rámec může představovat jakousi třídu či druh entity. Entita v tomto případě znamená objekt, který je pomocí rámce zachycen (popsán). Specifický rámec je pak konkrétní instancí třídy čili konkrétní entitou. U rámců je možné také implementovat dědičnost, kdy je u specializovaného rámce použit slot `specialization-of` či naopak v případě zobecnění `generalization-of`. Na Obrázku 3 je možné vidět jednoduchou formu rámce, která reprezentuje specializaci rámce `Kniha`. Konkrétní názvy slotů, jejich náplň a terminologie záleží na konkrétním použití a implementaci interpretace pomocí rámců. Tento příklad obsahuje sloty s jednoduchými údaji a speciální slot `Čtenář`, který, pokud je přidán, je aktivována procedura `PŘIDEJ_ČTENÁŘE`.

Velmi podobným stylem reprezentace znalostí je použití *objektů*. Objekty v sobě obsahují data (atributy) a metody, které pracují s danými daty. Metody a data jsou zapouzdřena v objektu. Generalizací objektu získáme třídu, která slouží jako jakýsi předpis, jak bude instance (objekt) této třídy vypadat a jaké jeho metody bude možné využívat. Důležitým vztahem je pak dědičnost tříd, kdy je možné od rodičovské třídy zdědit data a metody s možností přidat další. Jednotlivé objekty mezi sebou komunikují posíláním zpráv, a to voláním metod druhého objektu.



Obrázek 3: *Ukázka rámce*

Zdroj: Vlastní zpracování pomocí nástroje draw.io

Expertní systém ale nemusí vždy využívat jen jeden způsob reprezentace dat ve své bázi. Existují tzv. hybridní expertní systémy, které pracují s více než jedním formátem reprezentace. Kombinují se pravidlové, rámcové nebo objektové techniky. Příkladem může být programové prostředí CLIPS, které bude podrobněji popsáno dále v této práci, a ve kterém lze používat jak pravidla, tak objekty či oboje společně.

Kromě pravidel, rámců či sémantických sítí, lze znalosti reprezentovat pomocí *logických symbolů* a *výroků*. Prostředkem k tomu může být například logický programovací jazyk Prolog, o kterém bude více uvedeno dále v této práci. Podle (Giarratano a Riley, 2004) lze první formalizovanou (= logika se zabývá spíše formou výrazu než jeho významem) logiku datovat do 4. století před n. l. k řeckému filosofovi Aristotelovi. Aristotelova logika je založena na tzv. sylogismech, což je druh konečného logického výroku odvozeného z ostatních logických výroků - neboli inference. Základní sylogismus má dvě premisy a jeden závěr, který je odvozen (inferován) z těchto dvou premis. Premisy v sylogismu představují důkazy, ze kterých musí vycházet závěr. Pomocí sylogismu lze reprezentovat znalosti.

Výroková logika se zabývá deklarativními (oznamovacími) větami, které mohou být klasifikovány buď jako pravda nebo nepravda. Jedná se o symbolickou logiku pro manipulaci s logickými proměnnými. Pro analýzu obecnějších případů se používá predikátová logika prvního řádu. Prvního řádu, protože kvantifikuje pouze objekty a ne jejich vztahy nebo funkce těchto objektů, což řeší logika vyššího řádu (Russell a Norvig, 1995, s. 195). Tato logika je základem programovacího jazyka Prolog. Výroková logika je podmnožinou predikátové logiky. Predikátová logika se zabývá vnitřní strukturou věty. V predikátové logice se používa-

jí speciální slova – kvantifikátory. Univerzálním kvantifikátorem rozumíme „pro každý/pro všechny“. Značí se ∀. Dalším kvantifikátorem je existenční kvantifikátor, který vyjadřuje výrok, který je pravdivý pro alespoň jednoho člena domény. Lze ho také vyjádřit jako „existuje alespoň jeden objekt, pro který platí “ a je značen ∃.

1.1.2 Jazyky pro reprezentaci znalostí

Pro reprezentaci a tvorbu expertních systémů existuje několik programovacích jazyků a softwarových nástrojů. Jedná se o například o jazyky CLIPS, Jess, Loom či Prolog.

Jedním z nejznámějších je nástroj *CLIPS*, jehož název je zkratkou pro „C Language Integrated Production System“. Byl vyvinut v americkém NASA Johnson Space Center a naprogramován v jazyce C. Jedná se o multiparadigmový programovací jazyk, což znamená, že podporuje více než jedno programové paradigma, čili podporuje více programovacích stylů. Konkrétně u jazyka CLIPS to znamená, že podporuje tzv. rule-based programování (založené na pravidlech neboli deklarativní paradigma), objektově orientované programování, ale i procedurální programování. Vývojář Gary Riley, který je zároveň spoluatorem publikace (Giarratano a Riley, 2004) se stará o vývoj public domain verze, která byla uveřejněna v roce 2005. Jazyk CLIPS má několik verzí, ale až ve verzi 5 do něj byla implementována další programovací paradigmata - původně byl jazyk založen pouze na rule-based programování.

CLIPS pracuje s tzv. COOL - CLIPS Object Oriented Language (Giarratano a Riley, 2004), který je hybridní kombinací vlastností a funkcionalit z jiných objektově orientovaných jazyků jako je Common Lisp Object System (CLOS) a SmallTalk. Procedurální programovací jazyk má vlastnosti podobné jazykům C nebo Pascal a syntakticky je podobný LISPu.

Aby byl program napsaný v CLIPS schopen řešit problém, musí mít dostupná data v takové formě, aby byl schopen usuzovat. Informace jsou v CLIPS reprezentovány pomocí tzv. faktů. Tyto fakty obsahují název a několik slotů s údaji. Příklad takového faktu je vidět na Ukázce kódu č. 1.

```
(student (jmeno "Jan")
         (prijmeni "Novak")
         (vek 21)
         (titul "Bc."))
```

Ukázka kódu 1: Příklad faktu zapsaného ve formátu přijatelném prostředím CLIPS

Jess je pravidlový programovací jazyk obsahující pravidlový mechanismus a skriptovací prostředí, které je kompletně napsané v programovacím jazyku Java. Právě i pro tento jazyk je Jess určen. Jess byl vyvinut v roce 1995 Ernestem Friedman-Hillem ze Sandia National Laboratories v Kalifornii (Friedman-Hill, 2013). Jess využívá deklarativní paradigma, což znamená, že neustále aplikuje kolekci pravidel na kolekci faktů, což se také nazývá jako *pattern matching*, tedy něco jako porovnání se vzory v databázi. Autor projektu Friedman-Hill uvádí, že Jess využívá vylepšenou formu Rete algoritmu, což je výkonný mechanismus pro již zmíněný pattern-matching. Rete algoritmus sestavuje z pravidel v paměti síť (proto je pojmenován rete - latinsky síť) (Russell a Norvig, 1995). Rete algoritmus dokáže řešit náročné „many to many“ porovnávací problémy. Jazyk, který Jess používá, je velmi podobný jazyku používanému v prostředí CLIPS. Kód napsaný v CLIPSu lze také rozběhnout v systému Jess. Oproti CLIPS, které je vydáno jako open-source, je Jess publikován jako volný (free) pro akademické a vládní účely. Pro použití v komerční sféře je ale zapotřebí získat licenci od autora projektu.

Zástupcem „frame-based“ programovacích jazyků pro reprezentaci znalostí je jazyk *Loom*. Dle webové stránky projektu (Information Sciences Institute, 2007) je Loom jazyk pro konstrukci inteligentních aplikací. Jeho základem je jazyk pro reprezentaci znalostí umožňující deduktivní funkcionalitu systému. Deklarativní znalostí jsou v Loomu sestavovány z definic, pravidel, faktů a výchozích pravidel. Loom dále obsahuje tzv. classifier, což je deduktivní mechanismus využívající dopředné řetězení, sémantické sjednocení (sjednocení reprezentací, které mají stejný sémantický význam) a objektově orientované techniky tzv. truth maintenance, které udržují konzistenci mezi deklarovanými znalostmi a znalostmi odvozenými. Loom umožňuje programátorovi v jedné aplikaci využít metod logického programování, produkčních pravidel a objektově orientovaného programování. Zajímavým faktem je, že mnoho projektů využívající systém Loom, bylo sponzorováno agenturou DARPA (Defense Advanced

Research Projects Agency) Ministerstva obrany USA a Loom tak našel využití i v projektech zapojených do obrany Spojených států.

Nelze také opomenout velmi známý jazyk *Prolog*, jehož historie sahá až do 70. let 20. století. Jedná se o logický programovací jazyk. V tomto jazyce se nepopisuje program, ale jednotlivé vztahy popisovaných faktů. Prolog je deklarativním programovacím jazykem, což znamená že programátor definuje, co má program udělat, ale již nedefinuje, jak to program provede. Název Prolog byl definován v roce 1972 jako zkratka pro „*PROgrammation en LOGique*“ (Colmerauer a Roussel, 1996). V témže roce byl Philippem Rousselem implementován první systém Prolog. Jeho kolegové zároveň pracovali na komunikačním systému pro člověka s počítačem (man-machine communication system) pracujícím s francouzským jazykem. Byl to první velký prologový systém, který byl do té doby kdy vytvořen. Systém měl v sobě celkem 610 klauzulí, které lze roztřídit do tří kategorií. První sada klauzulí se řadila k analytické části, druhá sada byla dedukční a třetí sada je spojena s morfologií francouzského jazyka. S touto prvotní verzí Prologu byly vytvořeny další dva systémy. První z nich byl systém pro symbolické výpočty a druhý z nich obecný systém pro řešení problému nazvaný Sugiton. Dnes má Prolog několik implementací, jako například SWI-Prolog nebo GNU Prolog, což jsou interpreti tohoto jazyka.

1.2 Inferenční mechanismus

Pomocí inferenčního mechanismu v expertním systému je zabezpečeno využívání znalostí v bázi znalostí. Cílem expertního systému je nalezení té hypotézy, která nejlépe datově koresponduje s daným případem. Posloupnost jednotlivých inferencí se nazývá řetězení. Existují dvě strategie – dopředné a zpětné řetězení. Při dopředném řetězení se proces vyvozování řídí od faktů k závěrům a pomocí stávajících faktů vyvozuje fakty nové. Dopředné řetězení je běžně vyvoláno přidáním nového faktu do báze znalostí. Základem je hledání důsledku na základě daného předpokladu (faktu). Dopředné řetězení sestavuje obraz situace postupně s tím, jak do báze přicházejí nová data (fakty). Inferenční procesy v tomto případě nesměřují k řešení konkrétního problému (Russell a Norvig, 1995). Dvořák (2004) popisuje, že algoritmus dopředného řetězení pracuje na bázi opakování 3 základních kroků.

1. Nejprve dochází k porovnání, kdy jsou fakta porovnána s pravidly ze znalostní báze. Dojde tak ke zjištění pravidel se splněnými předpoklady (důkazová část pravidla).

2. V druhém kroku řetězení se provádí řešení konfliktu. U pravidel se splněnými předpoklady systém vybírá na základě priority jednotlivých pravidel. V případě, že nastane konflikt – najdou se dvě a více pravidel splňujících předpoklady se stejnou prioritou – systém použije jednu ze strategií řešení. Může se jednat o prohledávání do hloubky nebo do šířky, či strategie složitosti nebo jednoduchosti. Při prohledávání do hloubky jsou preferována pravidla, která používají data vyskytující se v bázi faktů kratší dobu. Při prohledávání do šířky jsou naopak preferována data starší. Podobný princip strategií je u strategie složitosti a jednoduchosti, kdy strategie vybírá na základě složitosti nebo jednoduchosti pravidel.

3. Třetím krokem dopředného řetězení je provedení pravidla vybraného v předchozím kroku nějakou ze strategií. Výsledek provedení pravidla může být přidání či smazání faktu či pravidla.

Cílem zpětného řetězení je najít odpovědi na otázku podanou proti bázi znalosti. Algoritmus zpětného řetězení nejprve zkontroluje, zda odpověď nemůže být vyvozena okamžitě na základě pravidel uložených v databázi znalostí. Následně najde v bázi všechny implikace (důsledky pravidel), které souhlasí s daným dotazem a pokouší se vytvořit předpoklady těchto implikací také pomocí zpětného řetězení (Russell a Norvig, 1995). Při zpětném řetězení tedy probíhá usuzování od hypotéz k faktům. Algoritmus zpětného řetězení funguje trochu na bázi zásobníkového automatu, kdy se na počátku utváří zásobník požadovaných hypotéz (Dvořák, 2004). Systém pak provádí shromáždění všech pravidel, která jsou schopna splnit hypotézu na vrcholu zásobníku. V dalším kroku se kontrolují předpoklady (důkazy) jednotlivých pravidel zda souhlasí s danými fakty. Pokud ne, průzkum pravidla je ukončen. Pokud se shodují, provede se pravidlo (vyvodí se důsledek). Hypotéza je odstraněna ze zásobníku a pokračuje se další hypotézou (cílem).

V pravidlových systémech je inference založena na *modus ponens*, které představuje přímé usuzování. Pokud platí důkaz (předpoklad) A a pravidlo $A \rightarrow B$, pak platí i hypotéza (závěr) B. Představitelem nepřímého usuzování je pravidlo *modus tollens*, které říká, že pokud důsledek není pravdivý, není pravdivý ani důkaz, tedy pokud neplatí hypotéza B při pravidlu $A \rightarrow B$,

pak neplatí ani důkaz *A*. Příkladem může být: „Jestliže venku sněží, je tam zima. Venku je teplo, takže tam nesněží“.

Autor (Dvořák, 2004) uvádí, že pravidlové systémy pracují dvěma základními způsoby. Jako inferenční síť anebo jako systém pro porovnávání se zdrojem. Inferenční sítě jsou vhodné pro řešení diagnostických a klasifikačních problémů. Fakta vychází jako závěr (důsledek) pravidla a fungují jako důkaz (předpoklad) pro pravidla jiná. Lze tak vytvořit orientovaný graf, ve kterém hrany (šipky) představují pravidla. Při použití systému porovnávání se zdrojem se vztahy mezi pravidly a fakty ustavují až v průběhu inference. Po úspěšném porovnání faktů s levou stranu pravidel (s tzv. vzory) a pokud je ve všech shoda, provedou se akce v pravé části pravidel. Tento systém se používá spíše v oblastech, kde existuje větší počet řešení či je tento počet neomezený.

1.3 Vysvětlovací modul

Vysvětlovací modul, z anglického „explanation facility“, má jako součást expertního systému na starosti vysvětlení nebo zdůvodnění rozhodnutí, ke kterému systém došel. Vysvětlovací moduly mají svůj původ v expertních systémech pracujících na bázi pravidel (rule-based expertní systémy) (Darlington, 2013), a to díky procesu řetězení pravidel v průběhu inference, což zanechává určitou trasu či stopu toho, jak systém rozhodoval a došel ke svému rozhodnutí. Darlington (2013) ve svém článku dále uvádí, že existují dva přístupy k vysvětlovacím modulům. V prvním případě si uživatel vysvětlení od systému vyžádá, ten ho tedy na základě toho zobrazí. V druhém případě je vysvětlovací modul přímo zapojen do procesu rozhodování a vysvětlení je zobrazováno automaticky. Může se jednat o inteligentní vysvětlování, tedy zobrazení vysvětlení v případech, kdy je to relevantní a užitečné. Vysvětlovací modul může nabízet vysvětlení po hledání (nalezení) řešení nebo ještě před zahájením procesu. V první variantě, se jedná o zpětnou vazbu (tzv. „feedback“), kde modul může zobrazit sekvenci pravidel, které byly použity v procesu inference a uživateli tak umožňuje vidět historii akcí v průběhu řešení problému a jakým způsobem bylo vlastně docíleno daného výsledku. Pokud modul podává vysvětlení dopředu (tzv. „feedforward“), je to většinou v případech, kdy odůvodňuje položení aktuální otázky a pomáhá tak uživateli lépe pochopit otázku a jak na ni má odpovědět. Tato vysvětlení mívají podobu popisu technických termínů, které je potřeba znát k rozumné odpovědi.

Darlington (2013) dále popisuje, jak závisí správná funkčnost na důvěře, kterou mají uživatelé k expertnímu systému. Vnímání expertního systému uživatelem je důležitým faktorem budoucího používání systému. Vjemy získané z práce vysvětlovacího modulu zahrnují důvěru uživatele, jeho spokojenost, jak systém přijímá a jak je systémem přesvědčený o výsledku řešení.

Moore a Paris (1991) ve svém článku shrnují charakteristiky dobrého vysvětlovacího modulu. Dobrý vysvětlovací modul by měl:

- sestavit své vysvětlení ze stejných znalostních zdrojů jako při procesu řešení problému - neměl by se tedy spoléhat na nějaké předem vytvořené šablony,
- čerpat z různých zdrojů znalostí, mezi které se řadí například terminologie, doména faktů či historie průběhu programu,
- užívat přirozeného jazyka uživatele - měl by tedy být schopen vytvořit vysvětlení ve formě, která je pro uživatele přirozená a tedy mít prostředky k tomu, aby tohoto docílil,
- být responsivní - uživatel by měl mít možnost systému položit doplňující otázky, na které by měl být systém schopen odpovědět,
- být flexibilní - je důležité, aby měl modul k dispozici několik strategií odpovědi na otázku, pro případ, že jeden způsob odůvodnění nebyl uživatelem pochopen,
- být schopen vysvětlení uzpůsobit znalostem uživatele,
- být jednoduše rozšiřitelný, pro případ potřeby rozšíření typu otázek od uživatele aj.,
- a být schopen adaptovat své stávající strategie na základě nějakého učícího procesu.

1.4 Komunikační modul

Komunikační modul je prostředek pro komunikaci expertního systému s uživatelem a naopak. Komunikační modul je také označován jako uživatelské prostředí, které obsahuje různá menu a grafické prostředí ve snaze učinit dialog mezi uživatelem a systémem více přívětivý (Tripathi, 2011). Giarratano (2004) označuje uživatelské prostředí jako „top-level“, tedy

nejvyšší úroveň systému a dodává, že mnohdy je mnohem více snahy věnováno návrhu a implementaci uživatelského rozhraní než samotné bázi znalostí. Implementace uživatelského prostředí může být provedena například pomocí pravidel nebo jiného programovacího jazyka. Příklad uživatelského rozhraní řešeného pomocí pravidel je vidět na ukázce kódu č. 2.

```
(defrule ask-spec-soft "Dotaz na specialni SW."
(not (spec-soft ?))
=> (assert (spec-soft (ano-ne "Pouzivate specialni SW? (ano/ne) =>
")))
)
```

Ukázka kódu 2: Pravidlo pro zobrazení dotazu v prostředí CLIPS

Zjednodušený popis: je deklarováno pravidlo s názvem `ask-spec-soft`, které nejprve zjistí, zda již otázka nebyla zodpovězena kontrolou na naplněnost proměnné `spec-soft`. Poté zobrazí dotaz v následujícím tvaru: „`Používáte speciální SW? (ano/ne) =>` “. Uživatel na něj odpovídá zadáním `ano` nebo `ne`. Hodnota je pak uložena do proměnné `spec-soft` a systém dále pokračuje v inferenci a pokládá další dotazy na základě plnění pravidel. Otázka vyvolaná tímto pravidlem již vyvolána nebude, protože daná proměnná je již naplněna a systém s ní pracuje.

Pokud je expertní systém naprogramován v jazyce, který podporuje tvorbu grafického uživatelského rozhraní, je možné vytvořit komunikační modul jako grafickou nadstavbu. K tomu je například vhodný programovací jazyk Jess, který je velmi podobný jazyku CLIPS ale napsán je v jazyce Java, což umožňuje využívat knihovny Javy pro tvorbu GUI.

1.5 Současné trendy v expertních systémech

Tato podkapitola se zabývá stručným souhrnem současných trendů v oboru expertních systémů a reprezentace znalostí. V době, kdy je internet nezbytnou součástí každodenního života, není divu, že i expertní systémy začaly pracovat s internetovými technologiemi. Autoři Verhodubs a Grundspenkis (2011) ve svém článku navrhují expertní systém založený na technologiích tzv. sémantického webu, který popisují jako jakousi superstrukturu nad klasickým internetem, která umožňuje znalosti uložené na klasickém internetu reprezentovat takovým

způsobem, aby byly „strojově čitelné“ a počítače s nimi mohly pracovat. Tohoto je docíleno mj. i díky sémantickým sítím a ontologiím. Za ontologii je v informatice považován hiearchicky strukturovaný popis domény - je to způsob, jak popsat určitou hiearchii tak, aby byla „pochopitelná“ pro počítače. Technologie sémantického webu zahrnuje například standardy XML (eXtensible Markup Language) nebo RDF (Resource Description Framework) - sadu specifikací deklarující způsob modelování dat umožňující kombinaci strukturovaných i nestrukturovaných dat, a to tak aby byla čitelná jak lidmi, tak počítači. Aplikacím využívajícím technologie sémantického webu se věnuje celá jedna část publikace (Alor-Hernández a Valencia-García, 2017), kde například autoři Noguera-Arnaldos et al. (2017) zpracovali systém pro ovládání tzv. IoT (Internet of Things) zařízení, které na základě hlasových pokynů uživatele detekuje činnost, kterou má systém vykonat, poté pomocí sémantické sítě dokáže zjistit jednotlivé vztahy mezi termíny v pokynu uživatele a díky ontologickému modelu dostupných zařízení detekuje, které zařízení má pokyn vykonat. Jejich projekt spojuje dohromady Natural Language Processing (zpracování přirozené řeči) a technologii sémantického webu.

Zajímavým konceptem je spolupráce či kombinace expertního systému a tzv. recommender systému (doporučovacího systému). Buyya et al. (2016) popisují doporučovací systémy jako nástroje a techniky nabízející uživateli nějaká doporučení. Nejčastěji jsou používány jako systémy a služby doporučující nějaký produkt, film či knihu, a to na základě uživatelova hodnocení či zájmů podobných uživatelů. Spolu s expertním systémem jej například používají Walek a Spackova (2018) k doporučení produktů na e-shopu. Expertní systém je zde použit pro výpočet popularity daného produktu. Vstupními daty pro expertní systém jsou: čas, který uživatelé stráví zobrazováním produktu, jak často byl daný produkt uživatelem zobrazen a kolik podobných produktů z určité kategorie si uživatel prohlížel. Na základě výstupu expertního systému pak doporučovací systém doporučí uživateli možné vhodné produkty.

Rubio et al. (2017) ve svém článku popisují doporučovací systém pro montéry chytrých čítačů spotřeby elektrické energie zvaných Smart Meter. Jelikož sběr takovýchto dat může být bezpečnostní hrozbou (data z těchto čítačů obsahují osobní informace o majitelích), systém si klade za cíl doporučit vhodné bezpečnostní řešení. Expertní systém je v tomto případě založen na tzv. Bayesian Network (Bayesiánské síti), což je pravděpodobnostní model založený na podmíněných pravděpodobnostích.

Současným trendem je také upouštění od klasických pojetí expertních systémů. Začínají se objevovat tzv. hybridní expertní systémy (Wagner, 2017), které zapojují další metody umělé inteligence jako například neuronové sítě. Neuronové sítě napodobují biologické neurony v mozku - jedná se o jejich matematickou reprezentaci. Používají se například ke klasifikaci, kde každá kategorie je reprezentována pomocí jednoho neuronu. Neuronové sítě lze „učit“ tak, aby byl jejich výstup co nejlepší. Učení je zajištěno pomocí upravování vah spojů mezi jednotlivými neurony, a to na základě zpracovávaných dat a z nich získaných „zkušeností“.

2 Zákaznická podpora

Zákaznická podpora je velmi důležitou součástí firemního ekosystému. Pokud firma něco produkuje, je zásadní, aby měla kvalitní zákaznické služby. Díky tomu je zákazník loajální a k firmě se při dalším nákupu produktu nebo služby vrací. Tato kapitola se zabývá historickým vývojem zákaznické podpory, jejími současnými trendy a seznamuje s některými procesy oblíbené metodiky ITIL, které se týkají zákaznické podpory.

Se zákaznickou podporou se pojí několik důležitých pojmů, které je vhodné vysvětlit.

Help Desk Pojem help desk zahrnuje konkrétní bod kontaktu, kde zákazník či uživatel získává pomoc s produktem. Tato pomoc může zahrnovat hlášení chybného chování produktu, dotazy ohledně fungování a používání produktu či požadavky na různé úpravy, které zákazník potřebuje. Zákazník se na helpdesk může obracet pomocí různých komunikačních kanálů: emailem, telefonicky, prostřednictvím speciální aplikace nebo pomocí uživatelského fóra. Podle (ManagementMania, 2019) může helpdesk řešit i jiné záležitosti, jako sledování stavu objednávky či příjem objednávky od zákazníka. Jak zdroj dále uvádí, helpdesk je pro uživatele velmi výhodný, protože se jedná o jedno konkrétní místo, kam se obrátí se svým problémem či požadavkem a nemusí tak sám hledat odborníka, který mu pomůže. Pracovníci helpdesku si s problémem buď poradí, nebo jej delegují dále k osobě, která bude schopna problém vyřešit.

Service Desk Metodika ITIL (Cartlidge et al., 2012) definuje service desk jako: „*The single point of contact between the service provider and the users. A typical service desk manages incidents and service requests, and also handles communication with the users.*“ (Hanna a Rance, 2011). V překladu to znamená, že service desk je jednotný bod kontaktu mezi poskytovatelem služby a uživatelem. Typický service desk spravuje incidenty, servisní požadavky a řeší komunikaci s uživateli. Oproti termínu help desk se jedná o novější termín, který se zrodil z frameworku ITIL, který prosazuje provozování IT jako služby (viz kap. 2.5). Jak uvádí Mann (2015), oproti helpdesku, který je považován za více se soustředící na incident management (viz kap. 2.5.1), se service desk kromě řešení chyb zaměřuje také na požadavky uživatelů – ať už na nové služby nebo jen

informace. Service desk je zabudován jako součást dodávky IT služeb a ekosystému IT podpory.

Aby byla firemní technická podpora funkční a plnila správě svou roli, musí se umět vypořádat s organizačními chybami, které mnohdy ve firmách nastávají. Této problematice se věnují autoři Botha a Leonard (2012) ve svém článku, kde zkoumají dopady organizačních problémů na výkon pracovníků help desku. Identifikovali základní organizační problémy, které lze rozdělit do následujících skupin: komunikace, řízení (tzv. macromanagement), technické problémy a jazykové problémy. Příkladem organizačních problémů může být nedostatečný dohled nad pracovníky, nespolehlivé a zastaralé operační systémy či žádná koordinace managementu změn. Problémy způsobují také zaměstnanci, kteří mají povoleno si na své pracovní stanice instalovat software, odinstalovávat jej a jinak měnit operační systém. Pracovníkům podpory tak vzniká dodatečná a zbytečná práce.

2.1 Historický vývoj zákaznické podpory

Historie zákaznické podpory sahá až do konce 19. století, kdy došlo k vynálezu telefonu. Do té doby se vztah prodejce a zákazníka zakládal na osobní interakci. Pokud měl zákazník nějaký problém, musel se osobně dostavit (ať už s produktem či bez něj) k danému prodejci, který jeho problém vyřešil či zodpověděl jeho dotaz. S vynálezem telefonu přišel i další způsob komunikace mezi prodejcem a jeho zákazníkem. První telefony se prodávaly pouze v párech a fungovaly pouze mezi sebou (Zendesk.com, 2019). Když se používání telefonů začalo rozmáhat, došlo k vývoji telefonních ústředen („telephone switchboard“), což umožnilo uskutečňovat několik telefonních hovorů najednou. Pokud měl tedy zákazník i prodejce telefon, komunikace mezi zákazníkem a prodejcem tak mohla být, v případě potřeby řešit nějaký dotaz zákazníka, jednodušší.

Zákaznická podpora se v průběhu 20. století dočkala dalšího vývoje, a to především zavedením tzv. call center. V 60. letech se firmy snažily zvýšit efektivitu, a tak začaly investovat peníze do oddělení, která byla zaměřená na odpovídání a přijímání dotazů od zákazníků. Tím se zrodila centra pro podporu zákazníků.

Zajímavým milníkem je rok 1962 kdy na World‘s Fair v Seattlu představili American Bell Telephone System – americká síť společností zajišťujících telefonní služby „The Bell Sys-

tem“ představila první počítač (systém), který dokázal vytáčet čísla pouze za použití tónů. Do té doby bylo možné telefonní čísla vytáčet pouze mechanicky pomocí točení číselníkem. Tento nový způsob vytáčení byl nazván Touch-tone dialing.

Svůj největší rozsah zažila call centra v 70. letech, kdy došlo k vývoji a rozšíření tzv. IVR (Interactive Voice Response). Jedná se o systém, který umožňuje počítači komunikovat s člověkem s pomocí hlasu a tónů, které vydávají číselná tlačítka telefonu. Zjednodušeně se jedná o automaty, které vybízejí ke stisknutí tlačítka pro zvolení určité volby. Tyto systémy byly zakomponovány do zákaznické podpory. Z počátku ale byly limitovány slabou slovní zásobou, byly drahé a velmi náchylné na chyby.

80. léta jsou dle Small (2013) zrodem klasického IT helpdesku. V těchto letech se do firem dostal operační systém MS-DOS a osobní stolní počítače od IBM. Pracovníci helpdesku v těchto dobách neměli o moc lepší vybavení než to, které měli uživatelé. Měli dostatek znalostí a zkušeností s počítači a operačními systémy. V těchto dobách ještě neexistovaly technologie, které by dokázaly práci helpdeskového týmu zefektivnit. V roce 1989 byla publikována první verze příručky ITIL – Information Technology Infrastructure Library, ve které byl poprvé formálně popsán koncept tzv. Service desku. Ve stejném roce také Ron Muns založil HDI - Help Desk Institute (Edwards, 2016). HDI se dodnes zabývá vzděláváním v oblasti zákaznické podpory a celkově se snaží udržovat standard tohoto odvětví (HDI, 2019).

V 90. letech došlo k velkému rozšíření outsourcingu, kdy bylo pro firmu výhodnější, aby převedla svá call centra a zákaznickou podporu do zahraničí. Díky nízkým nákladům a dobré vzdělanosti pracovníků se největší oblibě pro outsourcing těšila Indie či Filipíny (CompareCamp.com, 2015). S nástupem nového tisíciletí je spojen velký rozvoj softwaru pro správu help desku. Po roce 2008 se významným trendem v tomto odvětví staly sociální sítě, viz kap. 2.2.3 - především Facebook nebo Twitter. Uživatelé začali pomocí těchto kanálů hodnotit produkty, hlásit chyby a pokládat firmám své dotazy. Firmy jsou díky tomu schopny rychle reagovat na dotazy a požadavky zákazníků.

Oblíbeným nástrojem se také stává používání softwaru pro vzdálenou správu počítače. Zákazníkův problém s počítačem je řešen okamžitě v reálném čase a je tak ušetřena někdy zdlouhavá vzájemná komunikace mezi zákazníkem a operátorem, kdy zákazník musí dostatečně dobře popsat svůj problém, aby mohl být operátorem efektivně vyřešen. Když tato „podmínka“ není splněna, operátor musí pokládat dodatečné dotazy a čas řešení problému

se prodlužuje. Při použití softwaru pro vzdálenou správu počítače je operátor schopen vidět problémové chování a chybu rychle opravit.

2.2 Současné trendy zákaznické podpory

V současné době moderních komunikačních technologií není divu, že zákazník od svého provozovatele služby očekává prakticky 24hodinovou dostupnost zákaznické podpory (Baucom, 2018). S tím také souvisí potřeba podpory skrze mobilní zařízení, jejichž využití neustále roste a mobilní zařízení se stávají nedílnou součástí nejen osobního, ale i pracovního života. Mobilní řešení zákaznické podpory s sebou přináší i příležitosti ve formě cloudových řešení, kdy data zákazníků nejsou uložena na jednom místě a je tak docíleno většího zabezpečení dat proti ztrátě.

Podle Baucoma (2018) je dnes poptávka po tzv. self-service (samoobslužných) řešeních servisní podpory. V tomto přístupu jsou zákazníkovi k dispozici potřebné informace vedoucí k řešení jeho problému. Může se jednat například o nějakou formu online katalogu, ve kterém si sám zákazník hledá informace a postupy. Tímto způsobem lze výrazně snížit množství práce, kterou musí pracovníci podpory vykonávat a mohou se soustředit na větší problémy. Díky neustálému rozvoji technik umělé inteligence se do oboru zákaznické podpory také dostávají i metody umělé inteligence, viz kap. 2.2.4, například ve formě chatbotů, viz kap. 2.2.2.

Postupně se mění i přístup k zákaznické podpoře. Reaktivní přístup je nahrazován proaktivním (Fujitsu Services, 2016). Organizace začínají využívat technologie pro prediktivní analýzu a různé senzory, čímž se snaží zjistit, kde a jaké problémy s IT by mohly nastat. Díky tomu je možné detekovat události, které v konečném důsledku vedou k nějakému problému. Pokud je taková událost odchycena, může být zahájeno řešení ještě před tím, než se problém projeví. Proto lze s proaktivním přístupem výrazně snížit pravděpodobnost nastání IT incidentů.

2.2.1 Call centra

Přestože se může zdát, že call centra jsou dnes trochu zastaralým řešením, jsou stále důležitým komunikačním kanálem zákaznické podpory. V posledních letech začaly v oboru podpory objevovat nástroje datové analýzy, techniky strojového učení a tzv. boti. Dle (Scott, 2018)

jsou dnes rutinní úkoly řešeny mnohem více efektivněji a zaměstnanci jsou tak schopni řešit složitější problémy a požadavky. Zákazníci komunikující s kontaktním centrem chtějí stejnou kvalitu služby online, jakou by dostali při osobním nákupu v kamenném obchodě.

Vzrůstajícím trendem v oblasti chytrých zařízení je tzv. voice activated technology - nejvíce známou implementací jsou virtuální asistenti jako je Siri nebo Alexa. Tyto technologie nabízejí příležitosti i pro call centra, kde za pomoci dalších technologií, jako metody umělé inteligence, je možné vytvořit efektivní systém pro komunikaci zákazníka s firmou.

Call centra také ustupují od modelu rovnoměrného rozdělení volajících mezi pracovníky call centra. Moderním trendem je přidělování volajících na základě specializace, schopností a dovedností jednotlivých operátorů. V praxi to tedy může vypadat následovně: Zákazník zavolá na linku zákaznické podpory, kde sdělí operátorovi či počítačovému systému, jaká je povaha jeho problému či dotazu. Operátor nebo systém pak vyhodnotí vhodného operátora, na kterého je zákazník přepojen a se kterým řeší svůj problém. Může se tedy stát, že někteří operátoři jsou vytíženější než ti ostatní. Pro zákazníka je ale mnohem lepší, že s ním řeší jeho problém člověk, který má dostatečné znalosti a zkušenosti z domény jeho problému.

Simons (2015) ve svém internetovém článku popisuje inovace, které by současná call centra měla implementovat. Pracovníci call centra by měli například být schopni komunikovat nejen pomocí telefonu, ale používat další komunikační kanály jako email, chat či video hovory. Pomocí těchto technologií může zákazník kontaktovat centrum i v jiných než běžných pracovních hodinách. Simons dále uvádí, že pokud je pro zákazníka snadné kontaktovat firmu v jakoukoli dobu a skrze jemu výhodný komunikační kanál, existuje větší pravděpodobnost, že firmě zůstane déle loajální.

2.2.2 Chatboti

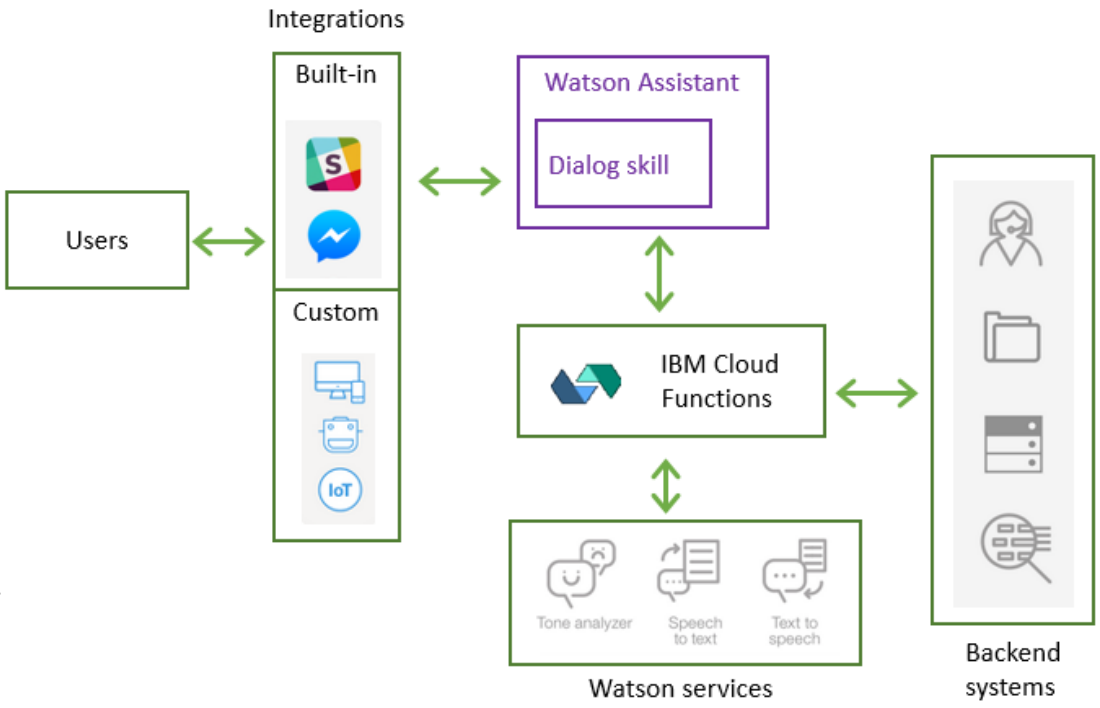
V oblasti inteligentních chatovacích robotů existuje mnoho zajímavých technologií a produktů. Mezi takové chatboty lze zařadit i hlasově ovládané asistenty Alexa, Siri nebo Google Home. Kloeckner et al. (2018) tyto asistenty označují jako technologie konverzační umělé inteligence (conversational AI). Ve své práci dále popisují systém pracující s technologií konverzačního chatbota pro samoobslužný (self-service) systém, který uživateli nabídne přesné odpovědi na jeho otázku z určité domény problémů. Zaměřují se především na management

IT služeb. Jak text (Kloeckner et al., 2018) uvádí, proto, aby chatbot pracoval dostatečně efektivně a správně, je zapotřebí se zaměřit na dvě zásadní oblastí, a to znalosti z domény otázky, na kterou má chatbot odpovědět, a dostatečné schopnosti chatbota rozpoznat lingvistický projev osoby podávající dotaz. Chatboti jako např. Alexa fungují přibližně na následujícím principu:

1. uživatel osloví zařízení či jinak aktivuje hlasového asistenta a položí mu určitý dotaz,
2. asistent tento dotaz zpracuje, a to tak, že z požadavku extrahuje klíčová slova, na která se uživatel ptá – například, pokud asistenta požádám o informaci, jaké bude dnes v Jablonci nad Nisou počasí, systém rozpozná klíčová slova jako „dnes“, „Jablonci nad Nisou“ a „počasí“. Na základě těchto klíčových slov určí, že se uživatel ptá na informace o počasí, že časový parametr je „dnes“ a lokace je „Jablonec nad Nisou“.
3. Na základě rozpoznaných informací provede chatbot dotaz na specifickou službu podávající potřebné informace. V tomto případě se může jednat o webovou službu meteorologického serveru.
4. Vrácenou odpověď serveru zpět reprezentuje uživateli – ať už v podobě hlasového projevu (v případě hlasových asistentů jako Alexa) nebo v podobě textu (v případě běžných chatbotů).

Pro tvorbu chatbotů již existují speciální nástroje a frameworky, jako například IBM Watson Assistant (původně Watson Conversation). Online dokumentace projektu (IBM Cloud, 2018) uvádí, že se jedná o kognitivního „bota“, který je přizpůsobitelný konkrétním potřebám podniků a je použitelný skrze více komunikačních kanálů. Obrázek 4 znázorňuje, jak služba funguje.

Uživatel s asistentem komunikuje skrze jeden ze vstupních bodů, kterým může být chatbot zabudovaný do služeb jako Slack (cloudový software pro týmovou spolupráci) či Messenger od Facebooku, základní rozhraní skrze další produkt IBM či chatbot integrovaný do vlastního projektu – mobilní aplikace nebo hlasově ovládaný asistent. Obsah komunikace je předán službě Watson Assistent, která skrze další komponenty prostředí IBM a jiných systémů zpracuje odpověď. Ta je následně předána zpět uživateli. Na schématu na Obrázku 4 je vidět tzv. dialog skill. V doslovném překladu se jedná o schopnost dialogu, kterou Watson Assistant



Obrázek 4: *Schéma fungování služby IBM Watson Assistant*
Zdroj: IBM Watson Assistant (2018)

má. Podle oficiální dokumentace (IBM Cloud, 2018) tato „schopnost dialogu" obsahuje trénovací data a programovou logiku. Obsahem jsou 3 základní artefakty: intent (záměr), dialog a entity. Intent, jak překlad napovídá, znamená záměr uživatele. Správce služby nadefinuje předem dané záměry, na které by měl být chatbot schopen odpovědět (může se jednat například o již zmiňované počasí nebo aktuální dopravní situaci a události v kalendáři). Dialog v sobě obsahuje jakýsi scénář odpovědí na určené záměry. Určuje, jak by měl chatbot odpovídat na dané záměry spolu s dalšími entitami, které reprezentují objekty nebo termíny – ty souvisí se záměrem a určují další vlastnosti dotazu, jako je místo pro zjištění počasí nebo datum, ze kterého chce zjistit naplánované události.

Základní součástí služby IBM Watson Assistant je systém IBM Watson. IBM Watson je tzv. kognitivní systém, což je systém, který své chování upravuje na základě zkušeností. Superpočítač Watson, který byl pojmenován po zakladateli společnosti IBM Thomasu J. Watsonovi, vznikl z projektu DeepQA, který zpracovával otázky zadané počítači v přirozeném jazyce (Collinaszy et al., 2017). Významným okamžikem v historii IBM Watson je jeho vítězství v americké soutěži „Jeopardy!", kde její účastníci hledají otázky na zadané odpovědi z domény obecných znalostí. Od roku 2017 název IBM Watson neoznačuje jen superpočítač, ale

celou škálu kognitivních služeb fungujících na IBM Bluemix Open Cloud Architecture. IBM Bluemix je cloudovou službou (firma pronajímá výpočetní kapacitu svých serverů), umožňující vývojářům vytvářet, nasazovat a spravovat cloudové aplikace.

Pokud bychom uvažovali použití v prostředí zákaznické podpory, autoři textu (Kloeckner et al., 2018) popisují zajímavý modul nazvaný „Guided Troubleshooting“, což v doslovném překladu znamená řízené řešení problémů. Jedná se o postup krok po kroku, jakým expert řeší problém nahlášený dotazem zákazníka. Řešení problému je prováděno takovým způsobem, kdy jsou kladeny otázky, které zpřesňují popis problému v případě, že uživatel položil dotaz nedostatečně detailně. Chatbot emulující experta může také klást otázky, ověřující určité podmínky. Autoři (Kloeckner et al., 2018) uvádí jako příklad hlášení problému s bootováním serveru. Systém položí upřesňující otázky na operační systém daného serveru a může se například dotázat, zda je schopen uživatel spustit určitý příkaz a na základě odpovědi uživatele chatbot rozhodne o dalším postupu a otázkách. Tuto strategii řešení problému autoři popisují jako založenou na tzv. grafu znalostí (knowledge graph-based). Graf znalostí znázorňuje entity, jejich vlastnosti a vztahy mezi nimi. V tomto konkrétním případě se jedná o graf zobrazující možné cesty řešení určitého problému, které mohou být omezeny určitými podmínkami a atributy (např. operační systém, zda má systém spuštěnou určitou službu aj.).

2.2.3 Sociální sítě

Sociální síť definují Lam a Hannah (2017) jako „*computer-mediated networks that enable individuals to build, share, and exchange information or knowledge in a global society*“, v překladu počítači zprostředkované sítě, které umožňují budovat, sdílet a vyměňovat informace či znalosti v globální společnosti. K využití sociálních sítí v oboru zákaznické podpory lze přihlížet minimálně dvěma způsoby. Prvním z nich je použití již vytvořených a veřejně známých sociálních sítí. V takovém případě je zásadním prvkem přítomnost dané firmy na dané sociální síti. Zákazník tak může firmu jednoduše vyhledat a okamžitě kontaktovat. Za velmi známé sociální sítě jsou považovány například Facebook (FB) nebo Twitter. Oproti Twitteru zprostředkovává Facebook více možností, jak zákazník může s firmou komunikovat. Pokud si firma vytvoří tzv. stránku na FB, uživatel má možnost na tuto stránku přidat příspěvek se svým názorem, hodnocením nebo nějakým dotazem. FB také umožňuje stránky hodnotit a recenzovat.

Sociální síť Twitter je v několika ohledech jednodušší. Každý uživatel, který si vytvoří v síti profil, získá osobní id (v terminologii sociálních sítí je to „handle“, v překladu rukojeť), pomocí které je v síti identifikován. Uživatel může přidávat příspěvky i s multimediálním obsahem. Dříve byl Twitter znám svým omezením délky příspěvku na 140 znaků. Toto omezení však bylo v roce 2017 lehce uvolněno a maximální počet znaků jednoho příspěvku se posunul[1] na 280 znaků.

Lam a Hannah (2017) provedli průzkum používání sociální sítě Twitter jako prostředek helpdeskové podpory. Zjistili, že ve většině případů lze rozdělit typ použití této sociální sítě pro komunikaci s firmou na dva případy. V prvním případě zákazník kontaktuje firmu skrze Twitter za účelem hledání řešení svého problému. V druhém případě jej využije k tomu, aby firmě vyjádřil stížnost nebo názor na ní. V článku je dále řešen vztah těchto dvou typů komunikace a míra úspěšného řešení daného problému či stížnosti. Bylo zjištěno, že stížnosti byly mnohem méně úspěšně adresovány a řešeny než běžné technické problémy. Lam a Hannah to vysvětlují tím, že pokud si uživatel na sociální síti na firmu stěžuje, existuje předpoklad, že je naštvaný a nebude ochoten řádně dodržovat postup, který mu firma dodá jako možné řešení jeho problému.

Druhým přístupem k sociálním sítím je tvorba vlastní sociální sítě pro potřeby zákaznické podpory. Autoři textu (Ortbach et al., 2014) řeší zajímavé téma zabývající se tzv. sociálními stránkami pro otázky a odpovědi a jak tyto stránky podporují user-to-user podporu. Tyto tzv. SQA (Social Question and Answer) lze definovat jako webové stránky, na kterých uživatel pokládá své dotazy a hledá informace od uživatelů v dané síti. Uživatelé si tak navzájem pomáhají. Ortbach et al. dále uvádí, že na internetu existují další 2 druhy stránek pro otázky a odpovědi. První z nich je jakousi digitální referenční službou (jako lze najít v knihovnách), kdy uživatel položí dotaz expertovi, který ho navede na ty správné zdroje informací. Druhým typem jsou tzv. expertní služby, které se zaměřují na určitá témata a zaměstnávají experty v těchto tématech, kteří dodávají potřebné informace.

[1] `https://blog.twitter.com/official/en_us/topics/product/2017/Giving-you-more-characters-to-express-yourself.html`

2.2.4 Umělá inteligence v technické podpoře

Výzkum v oboru umělé inteligence zasahuje do mnoha odvětví, a tak není divu, že jsou metody umělé inteligence využívány i v oboru uživatelské podpory. V uživatelské podpoře lze najít mnoho míst, kde je možné umělou inteligenci využít. Jedním z nich může být například inteligentní přidělování helpdeskových tiketů, viz kap. 2.4. Přidělování řešitelů je většinou manuální, časově náročný úkol, kdy zadávající musí mít dostatečné informace o řešitelích (zda již nemají jiné úkoly k řešení, zda mají dostatečné znalosti jak problém nebo požadavek vyřešit). Tento problém se snaží řešit např. Baysal et al. (2009), a to pomocí frameworku sestávajícího ze 3 částí. První část má na starost doporučení řešitele na základě jeho úrovně expertizy. Tuto úroveň určí na základě historie řešení minulých požadavků. Systém sestaví ohodnocený seznam vývojářů, kteří by se mohli problému zhostit. Druhá část frameworku se zabývá získáním informací o preferencích řešitele v oblasti jednotlivých problémů a požadavků. Pokud řešitel neměl problém s řešením požadavku, ohodnotí jej jako preferovaný a systém toto hodnocení příště vezme v úvahu při procesu vyvozování, kdo by měl daný požadavek řešit. Poslední část frameworku se zabývá alokací daných požadavků řešitelům. Snaží se vzít v úvahu, zda potenciální řešitel právě neřeší mnoho úkonů a pracuje s jejich „rozvrhy“.

Problém efektivnosti komunikace zákazníka s podporou při použití telefonní komunikace řeší Weerawarna et al. (2011). Podle nich je komunikace zákazníka s callcentrem neefektivní - uživatelé mnohdy volají, protože potřebují poradit se svým produktem a mnohdy mají stejný typ otázek. Pracovník call centra tak odpovídá na stejné dotazy. Toto příliš ani neřeší tzv. FAQ stránky (Frequently Asked Questions). Ve svém článku tedy navrhují speciální systém CyberMate, který umožňuje komunikovat s uživatelem pomocí IM (Instant Messaging). Systém pracuje s několika IM protokoly jako Skype, Google Talk (předchůdce dnešního Google Hangouts) či Jabber. CyberMate pomocí umělé inteligence odpovídá na dotazy uživatele na základě znalostí z konkrétní domény. Výhodou systému je neustálá dostupnost v kterémkoli čase a z jakéhokoli zařízení schopného komunikovat skrze IM protokoly. Zároveň jsou eliminovány zákazníkovy náklady na uskutečnění telefonních hovorů.

2.3 Úrovně technické podpory

Počet a vlastnosti úrovní technické podpory se v literatuře liší. Například (Windley, 2002) popisuje 5 různých úrovní takzvaného „tiered support modelu“ (v překladu přibližně víceúrovňového modelu podpory). Jedná se o:

- **Úroveň 1** - Zákaznická podpora a help desk. Jedná se o tzv. první linii technické podpory, která přijímá požadavky zákazníků jako první. Pracovníci na této úrovni podpory se zabývají tříděním požadavků a řeší jednoduché úkoly například pomocí informací z databáze znalostí. Zbytek úkolů je eskalován do 2. úrovně. Pracovníci na této úrovni mohou mít menší technické znalosti (ty jsou důležité ve vyšších úrovních).
- **Úroveň 2** - Technická podpora. Na této úrovni se řeší eskalované požadavky z první úrovně. Pracovníci řeší problémy sami s použitím dostupných materiálů. Mají dostatečné technické znalosti týkající se produktů a mohou přímo komunikovat s pracovníky na vyšších úrovních podpory. Tito pracovníci většinou mají začátečnické technické znalosti, ale je zvykloství, že postupně získávají další zkušenosti a dovednosti, takže se posouvají na pozice na vyšších úrovních technické podpory. Problémy, které nedovedou na této úrovni vyřešit nebo nejsou z jejich domény problémů, které řeší, eskalují do vyšších úrovní.
- **Úroveň 3** - Systémoví a síťoví administrátoři. Řeší eskalované požadavky z předchozí úrovně. Zaměřují se především na infrastrukturu fungování služeb podniku. Starají se o síťovou infrastrukturu, servery aj.
- **Úroveň 4** - Product Operations Engineering. Pro toto oddělení se používá zkratka ProdOps. Pracovníci na této úrovni podpory se řadí k velmi technicky zdatným. Starají se o konkrétní operace individuálních produktů firmy. Může se jednat o systémové inženýry, administrátory databází nebo síťové inženýry. Jejich hlavním úkolem je zajišťovat spolehlivost a dostupnost produktů a služeb. Kontrolují a monitorují také nová vydání zdrojového kódu, které jde do produkční fáze (poslední fáze vývoje softwaru).
- **Úroveň 5** - Engineering. Do této úrovně podpory se řadí softwaroví a systémoví inženýři, kteří mají na starost vývoj a udržování produktů a služeb. Zpracovávají požadavky eskalované z úrovní 2-4 a mají na starosti proces tzv. bug trackingu.

Jiný, trochu zjednodušený přístup k úrovni technické podpory, představuje Hertvik (2016):

- **Úroveň 0** - Svépomoc a samostatné hledání řešení. Uživatel sám získává informace prostřednictvím internetových stránek, mobilních aplikací, zpracované a publikované produktové dokumentace, FAQ či pomocí manuálů a internetových vyhledávačů. Uživatelé své problémy řeší sami, tudíž nekomunikují s IT oddělením. V případě, že problém sami vyřešit nedokážou, eskalují svůj problém prostřednictvím webových formulářů nebo sociálních sítí pracovníkům help desku (1. úroveň).
- **Úroveň 1** - Základní helpdesk. Pracovníci helpdesku reagují na požadavky získané skrze jejich webové stránky, sociální sítě nebo na zákaznických fórech. Na této úrovni se řeší převážně známé problémy.
- **Úroveň 2** - Technická podpora, která má hlubší znalosti a dovednosti pro řešení problémů.
- **Úroveň 3** - Expertní produktová a servisní podpora.
- **Úroveň 4** - Externí podpora pro problémy, které jsou nepodporované organizací. To může znamenat záležitosti, se kterými si podpora neporadí nebo je nemůže nijak ovlivnit.

Jak je vidět, přístup Hertvika je odlišný. Přidává nultou úroveň, ve které ještě není vůbec technická podpora přímo zapojena. Vliv na tuto úroveň má stupeň dostupnosti informací k danému produktovému portfoliu - ať už se jedná o servisní informace oficiální (např. v servisním katalogu společnosti) nebo informace ostatních uživatelů.

2.4 Ticket management

Ticket management je proces řízení a správy problémů a požadavků. Jako ticket označujeme jakýkoli požadavek nebo problém, který potřeba vyřešit. Tickety (lístky) lze rozdělit na tzv. trouble tickety a support tickety. V předkladu se jedná o tiket týkající se problému a tiket týkající se interního nebo zákaznického požadavku.

Podle (Xu a He, 2018) je tiket vytvořen buď monitorovacím systémem nebo přímo zákazníkem. Poté „cestuje“ sítí expertů, dokud není vyřešen. To nastává v případě, že některý

z řešitelů není schopen problém vyřešit a tak jej deleguje dále, buď někomu, na stejné úrovni podpory nebo jej eskaluje na vyšší úroveň podpory. Toto se nazývá „ticket routing“, v překladu směrování tiketu - tiket cestuje od zákazníka přes několik řešitelů i přes několik úrovní podpory. Čím delší směrování, tím se prodlužuje doba, za kterou je tiket vyřešen, což může, jak uvádí Xu a He, vést k porušení SLA (Service Level Agreement) viz kap.2.5 - smlouvě o úrovni poskytovaných služeb i ke zbytečnému plýtvání prostředky. V článku představují model sítě pro spolupráci expertů na řešení tiketu a algoritmus pro doporučení vhodného experta.

2.5 Metodika ITIL

Jak uvádí oficiální příručka ITIL (Cartlidge et al., 2012), ITIL (Information Technology Infrastructure Library) nabízí soubor návodů a tzv. best-practices (osvědčených postupů) v oblasti řízení IT služeb. První publikace metodiky ITIL se datuje do let 1989–1995, kdy byla vydána ve formě 31 knih. Zrevidovaná verze vyšla v letech 2000–2004 ve formě 7 knih a s názvem ITIL v2. Další revize se metodika dočkala v roce 2007, kdy byl počet knih snížen na 5. Zatím poslední revize je datována do roku 2011, kdy byla verze 2007 doplněna o aktuální revize a je označována jako ITIL 2011 Edition. Každá z oněch 5 knih se zabývá jednou částí životního cyklu IT služby. Životní cyklus sestává ze:

- ITIL Service Strategy - strategie služby,
- ITIL Service Design - návrh služby,
- ITIL Service Transition - přechod služby,
- ITIL Service Operation - provoz služby a
- ITIL Continual Service Improvement - neustále zlepšování služeb.

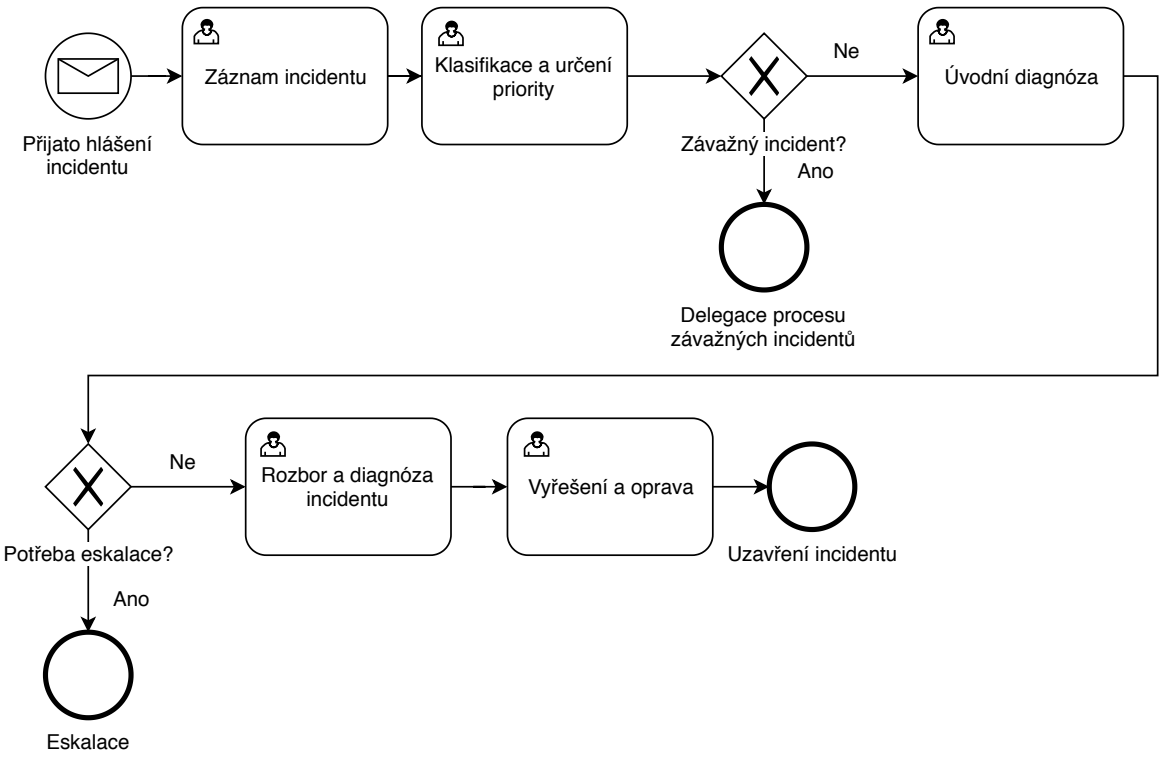
Některým částem metodiky se věnují následující podkapitoly. Nejprve je ale důležité představit a vysvětlit základní pojmy, které se v metodice ITIL vyskytují. Základní z nich je *služba*. V oficiálním glosáři metodiky ji Hanna a Rance (2011) definují jako „*A means of delivering value to customers by facilitating outcomes customers want to achieve without the*

ownership of specific costs and risks.“ To lze volně přeložit jako pomoc zákazníkům v dosahování cílů převzetím zodpovědnosti za určité náklady a rizika. Příkladem může být dodávka a správa počítačového vybavení firmy, kdy dodávající firma dodává službu, díky které se zákaznická firma nemusí starat o správu počítačového vybavení a její „starosti“ jsou přeneseny na dodávající firmu, a to na úrovni odpovídající smlouvě SLA. SLA je dohoda zákazníka a dodavatele IT služby, ve které jsou specifikovány poskytované IT služby a jejich úroveň podpory a definuje jednotlivé povinnosti - zákazníka i dodavatele.

2.5.1 Incident Management

Řízení incidentů neboli incident management se řadí do provozu služby, tedy části životního cyklu služby ITIL Service Operation. Incident Management rozlišuje mezi tzv. incidenty a požadavky. Incidentem se rozumí neplánované přerušení či snížení provozu nebo kvality služby. Za servisní požadavek se rozumí formální požadavek uživatele o informaci, radu či nějaký úkon (Cartlidge et al., 2012). V řízení incidentů existují 4 základní role. První z nich, *incident manager* (manažer incidentů), se zabývá efektivní implementací politiky incident managementu a má na starosti první část eskalace incidentu v případě nemožnosti řešení v rámci dohodnuté úrovně podpory. Další dvě role tvoří dvě úrovně podpory. Podpora 1. úrovně je zodpovědná za většinu úkonů v procesu zpracování incidentů a zpracovává incidenty, které lze řešit okamžitě. Pokud incident nelze vyřešit ihned, převezme jej podpora 2. úrovně a jejím úkolem je co nejrychleji incident vyřešit a stav služby obnovit do funkčního stavu. Čtvrtou rolí v řízení incidentů je Major Incident Team, který řeší vážné incidenty.

Proces řízení incidentů podle ITIL je znázorněn na Obrázku 5. Uživatel (zákazník) nejprve podá hlášení o incidentu prostřednictvím webu, emailu či telefonu. Toto hlášení je přijato oddělením řízení incidentů, které toto hlášení zaznamená, provede jeho klasifikace a následně stanoví prioritu tohoto incidentu. Poté, pokud se jedná o tzv. *major incident* - tedy závažný incident, tak je delegován do speciálního procesu pro řešení těchto závažných incidentů. Pokud se nejedná o závažný incident, pokračuje dále k úvodní diagnostice incidentu. Pokud je zjištěno, že problém je nad síly daného oddělení, je aktivován proces eskalace incidentu (viz kap. 2.3). V opačném pokračuje běžný postup k rozboru a diagnóze daného incidentu. Poté, co je incident vyřešen a je vydána oprava, se incident uzavírá.



Obrázek 5: *Proces ITIL Incident Management (v notaci BMPN)*

Zdroj: Vlastní zpracování dle Cao a Zhang (2016) pomocí nástroje draw.io

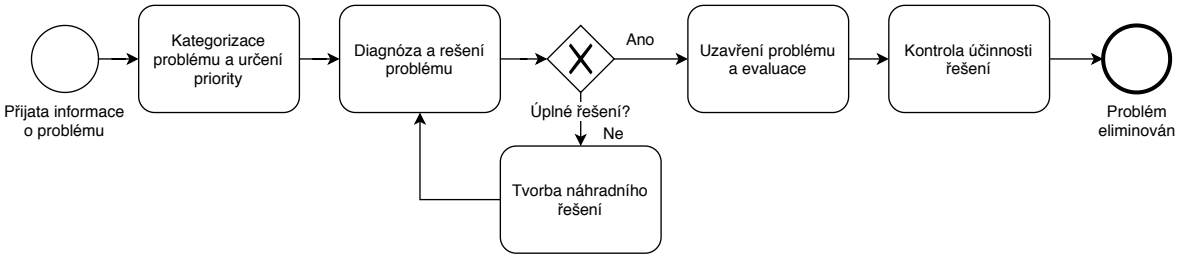
V popisu proces je zmíněn termín závažný incident (major incident). Ten je identifikován jako incident, který zabrání značnému množství lidí či důležitým zákazníkům v užívání služeb nebo systémů. Náklady vyvolané tímto incidentem jsou nebo budou značné a reputace dodavatele služby může být poškozena (S. Kempter a A. Kempter, 2006–2018). Proto je v případě identifikace závažného incidentu sestaven speciální tým (Major Incident Team), který má na starosti rychlou obnovu služby. Pokud je potřeba, jsou k řešení přizváni specialisté nebo dodavatelé z třetí strany (third-party suppliers). Pokud náprava původní příčiny není možná, je vytvořen záznam o problému (Problem Record) a oprava je převedena do Řízení problémů (Problem Management), které je popsáno v další podkapitole.

2.5.2 Problem Management

Problémem je v terminologii metodiky ITIL (Cartlidge et al., 2012, s. 43) označována příčina nějakého incidentu (viz předchozí podkapitola 2.5.1). Souvisejícími termíny jsou známá chyba (Known Error) a náhradní řešení (Workaround). Známá chyba je problém, u kterého byla popsána jeho příčina a bylo nalezeno náhradní řešení. Náhradní řešení má za cíl snížit nebo odstranit dopady známé chyby, pro kterou ještě nebylo nalezeno dostatečné úplné

řešení. Tato řešení jsou mnohdy aplikována, pokud nelze pohotově incident nebo problém odstranit či identifikovat (S. Kempter a A. Kempter, 2006–2018).

Úkolem problem managementu je řídit životní cyklus všech problémů. Hlavní rolí v procesu je Problem Manager (vedoucí problémů), který je zodpovědný za řízení celého životního cyklu problému, který je popsán na Obrázku 6. Jeho hlavním cílem je předcházet incidentům a zmírňovat jejich důsledky, pokud jim nelze předcházet.



Obrázek 6: *Proces ITIL Problem Management (v notaci BMPN)*

Zdroj: Vlastní zpracování dle Kempter a Kempter (2006–2018) pomocí nástroje draw.io

Problém je procesem řízení problémů přijat na základě delegace incidentu nebo na základě tzv. proaktivního identifikování problémů, které má za cíl identifikovat problémy, které by mohly být přehlédnuty, a včas tak předcházet incidentům. Problém je následně v procesu klasifikován a je mu přiřazena priorita. Poté dochází k jeho diagnóze a hledání řešení. Pokud nelze problém vyřešit ihned, je vydáno dočasné řešení, které se snaží minimalizovat dopady problému a následně se problém vrací zpět na řešení. Pokud již byla nalezena příčina, je problém zaveden do databáze známých chyb (Known Error Databse). Jakmile je dosaženo úplného řešení problému, je uzavřen a hodnocen. V tomto podprocesu se kontrolují záznamy problému (Problem Record) obsahující celou historii problému v procesu a jeho rozřešení. Nakonec je provedeno přezkoumání problému (Major Problem Review), které má za cíl zajistit, aby se chyba již neopakovala a aby z ní bylo dostatečné poučení. Úkolem řízení problémů je samozřejmě také neustále informovat a komunikovat s ostatními procesy řízení služeb, a to především o stavu aktuálních problémů a dostupných dočasných řešeních.

3 Vývoj expertního systému

Cílem této diplomové práce je popsat tvorbu expertního systému. Pro expertní systém byla zvolena problematika běžného podniku v oboru vývoje informačních systémů. Daný podnik má vlastní zákaznické helpdeskové oddělení. Následující kapitola se zabývá současným stavem této helpdeskové podpory a analyzuje problém, který by měl být expertním systémem řešen. Tato kapitola dále obsahuje analýzu požadavků v textové i grafické podobě a popis implementace expertního systému.

K analytickému modelování expertního systému jsou v této diplomové práci použity modely jazyka UML (Unified Modeling Language). UML je v současnosti standardem pro objektově orientovanou analýzu a návrh. Byl vytvořen v 90. letech 20. století a o jeho rozvoj se stará skupina Object Management Group (OMG). V jeho aktuální verzi (2.0) lze k popisu chování a struktury systému použít 13 diagramů. V této práci jsou použity následující: diagram případů užití, diagram aktivit a diagram tříd.

Další model, který je použit v této kapitole, je BPMN (Business Process Model and Notation). Řadí se k modelům pro procesní modelování a jedná se o otevřený standard, který je také spravován skupinou OMG. Business Process definujeme jako „*A defined set of business activities that represent the steps required to achieve a business objective. It includes the flow and use of information and resources.*“ (Object Management Group, 2011, s. 499). V překladu se jedná o definovanou sadu business aktivit reprezentujících kroky potřebné ke splnění cíle daného procesu. Zahrnuje v sobě tok a užití informací a zdrojů.

3.1 Analýza současného stavu

Firma, pro kterou je navrhován a vytvářen expertní systém, se zabývá vývojem informačních systémů a dodávkou IT služeb zákazníkům z oblastí výroby, logistiky, ale i zdravotnictví. Navrhovaný expertní systém, tak jak je popisován v této diplomové práci, je uzpůsoben pro potřeby konkrétní divize, jež má na starosti vývoj vlastního informačního systému. Lze jej ale snadno rozšířit i pro potřeby divizí ostatních a zapojit jeho fungování do dalších aktivit firmy. To však z důvodu zachování jednoduchosti této diplomové práce není její součástí.

Množina zákazníků tvořících helpdeskové požadavky zahrnuje společnosti z různých odvětví průmyslu.

V současné době je systém helpdeskové podpory postaven na integraci s firemním softwarem Lotus Notes od firmy IBM. Většina činností divize je na toto prostředí napojena, a to platí i pro správu helpdesku. Zákazník firmy má pomocí svých přihlašovacích údajů přístup k webovému prostředí helpdeskové aplikace fungující jako součást firemního workflow v IBM Lotus Notes. Produkt IBM Lotus Notes se řadí mezi tzv. groupware, což je software pro podporu spolupráce pracovních skupin. Zahrnuje v sobě moduly různých kategorií. Mezi nejzákladnější patří emailová schránka, kalendář, adresář kontaktů, plánování a správa porad, správa úkolů aj. Firma vytvořila a upravila několik dalších modulů - například správa zákaznických požadavků na aplikační software, záznamy o pracovní činnosti či záznamy o komunikaci se zákazníky. Díky tomu se jedná o vitální součást pracovního prostředí společnosti. Vytvořen byl i helpdeskový modul, který je do ekosystému Lotus Notes plně zapojen (zasílání emailových upozornění, svázání helpdeskové činnosti se záznamy o pracovní činnosti, aj.).

Zákazníci firmy mají k dispozici přístup do webového formuláře helpdeskové aplikace. Pomocí tohoto formuláře zadávají nové požadavky, sledují aktuální stav svých požadavků a v případě potřeby i doplňují další informace nezbytné k uspokojení svých požadavků. Proces práce s požadavkem, od nastalé potřeby až po vyřešení, je znázorněn na BPMN diagramu na Obrázku 7. Proces je pro potřeby diplomové práce zjednodušen. Část procesu, které se bude týkat expertní systém, je v diagramu uvedena jako „Zpracování požadavku“ a je detailněji popsána BPMN diagramem na Obrázku 8.

Poté, co zákazník odešle požadavek skrze webové rozhraní helpdeskové aplikace, je tento požadavek přijat pracovníkem, který identifikuje cílovou divizi, které se požadavek týká, a tento požadavek jí deleguje. Osob, které působí ve firmě v roli první linie podpory, je několik: pracovníci recepce, určení pracovníci helpdesku v jednotlivých divizích, ale i firemní helpdeskové oddělení, které má na starosti jeho správu. Může se také jednat o osoby, které mají tzv. pohotovostní službu. Úrovně podpory v dané firmě dle modelu popsaného v kap. 2.3, jsou následující:

Úroveň 1 - pracovníci recepce či pracovníci s pohotovostní službou. Jedná se o první linii podpory. Tito pracovníci požadavek přijmou a eskalují jej na určeného pracovníka jednotlivých divizí.

Úroveň 2 - jedná se o pracovníka, který přijímá eskalované požadavky zákazníků a provádí jeho prvotní zpracování, kdy se snaží požadavku porozumět a vyřešit jej. Pokud je požadavek mimo jeho kompetence, provede jeho eskalaci do další úrovně.

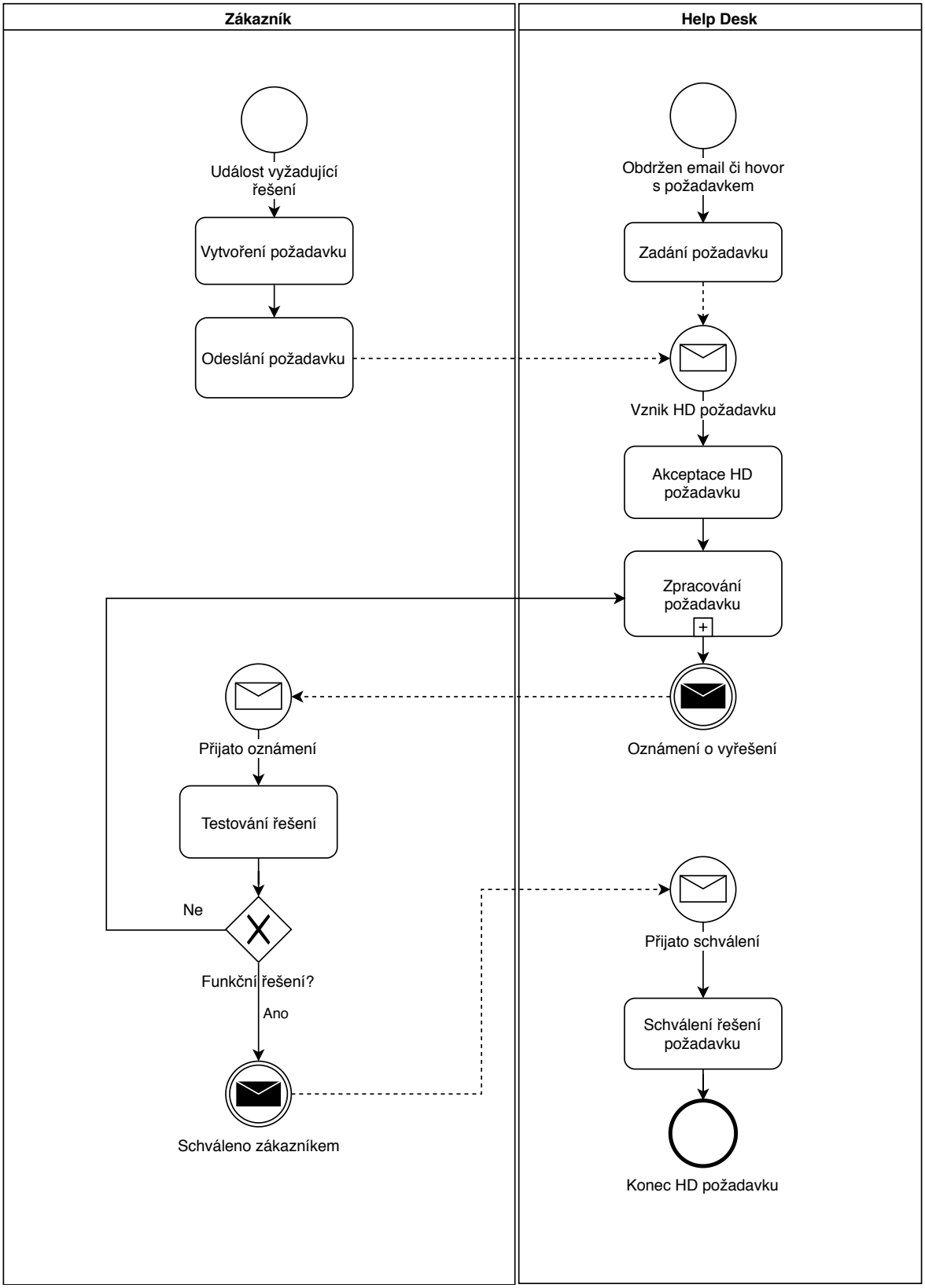
Úroveň 3 - tato úroveň v sobě v případě popisované firmy zahrnuje úrovně 3-5, protože se jedná o firmu vyvíjející informační systém, a tak její pracovníci zpracovávají relevantní požadavky. Jedná se o softwarové inženýry, systémové programátory, databázové inženýry či síťové administrátory.

Pracovník konkrétní divize, na kterého je požadavek delegován, má na starosti přidělení řešitele a v mnoha případech i řešení daných požadavků. Proces tohoto zpracování zákaznického požadavku je popsán diagramem BPMN na Obrázku 8.

V této části životního cyklu zákaznického požadavku může dojít k časové prodlevě. Pokud se pracovník helpdesku snaží požadavek analyzovat, hledá při tom podobné požadavky a snaží se též najít, zda již takovýto požadavek nebyl v minulosti řešen. Pokud takový najde, může aplikovat jeho řešení. Pokud takový ale nenajde a duplicitní požadavek opravdu existuje, může dojít k řešení požadavku, který má již potenciálně řešení, a tím ke zbytečné ztrátě času. K časové prodlevě dochází také v případě, kdy pracovník vyhledává relevantní požadavky. V doméně požadavků jednoho zákazníka to není tak náročné, ale pokud by měl pracovník prohledávat i ostatní požadavky zákazníků, bylo by to neefektivní.

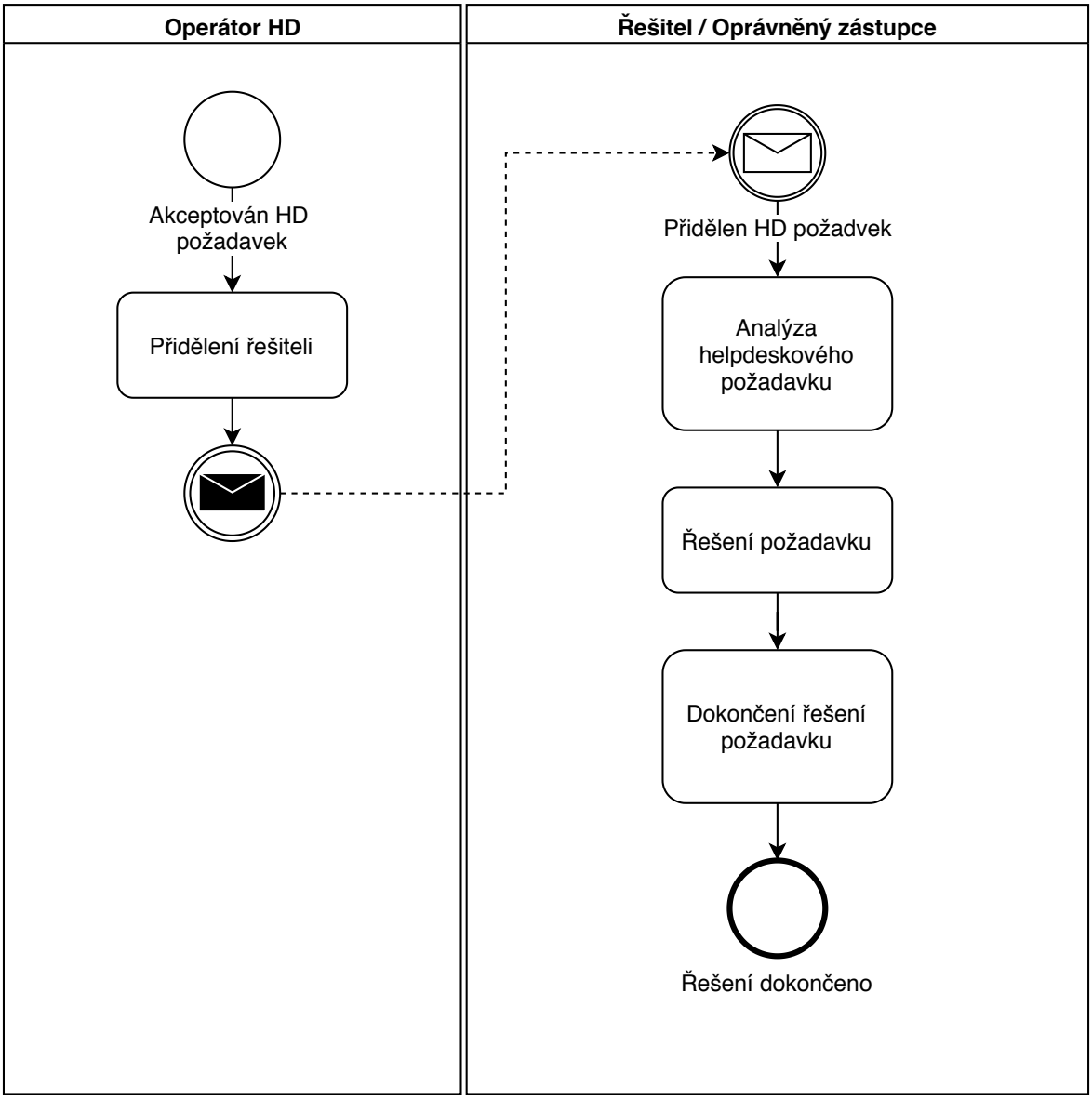
Tomu se snaží pomoci navrhovaný expertní systém. Jeho cílem je působit jako podpora při řešení helpdeskových požadavků, a to ve formě, která je specifikovaná v dalších podkapitolách.

V průběhu řešení je častým jevem, že řešitelé zákaznického požadavku potřebují dodatečné informace, a tak dochází k další komunikaci řešitele a zákazníka. Důvodem může být nedostatečný popis v původním požadavku nebo potřeba, která vyvstala v průběhu řešení. Po dokončení řešení požadavku je zákazníkovi zasláno oznámení a zákazník je požádán, aby řešení implementoval nebo vyzkoušel. Pokud zákazník potvrdí jeho funkčnost, je požadavek uzavřen. V opačném případě zákazník dále komunikuje s helpdeskovým oddělením, které se požadavek opět snaží prověřit a vyřešit.



Obrázek 7: *Zjednodušený proces životního cyklu zákaznického požadavku*

Zdroj: Vlastní zpracování pomocí nástroje draw.io



Obrázek 8: *Detail podprocesu Zpracování požadavku*

Zdroj: Vlastní zpracování pomocí nástroje draw.io

3.2 Specifikace požadavků na expertní systém

Specifikace požadavků obsahuje dvě základní části: model požadavků a model případů užití. Popisem těchto modelů se zabývají následující podkapitoly.

3.2.1 Model požadavků

Softwarové požadavky dělíme na tzv. funkční a nefunkční. Funkční požadavky popisují chování, které by měl navrhovaný systém nabízet. Oproti tomu nefunkční požadavky popisují

jeho vlastnosti či omezující podmínky (jako například v jakém programovacím jazyce má být software naprogramován). Veškeré činnosti spojené s definicí požadavků na systém lze shrnout termínem *requirements engineering*, čili inženýrství požadavků. Požadavky definují pouze to, co by měl daný systém dělat, ale již neřeší, jak by toho měl dosáhnout.

Pro navrhovaný expertní systém lze definovat následující funkční požadavky:

F.1 *Expertní systém bude kategorizovat nové helpdeskové požadavky.*

F.2 *Expertní systém bude uživateli zobrazovat seznam relevantních požadavků.*

F.3 *Expertní systém bude detekovat možné duplicitní požadavky.*

F.4 *Expertní systém bude umožňovat uživateli jednoduchou administraci báze znalostí.*

Nefunkční požadavky jsou pak následující:

N.1 *Expertní systém bude dostupný jako webová aplikace.*

N.2 *Expertní systém bude mít uživatelské rozhraní s formulářovým vstupem.*

N.3 *Expertní systém bude mít API pro vzdálenou komunikaci.*

N.4 *Expertní systém by měl být univerzálně použitelný.*

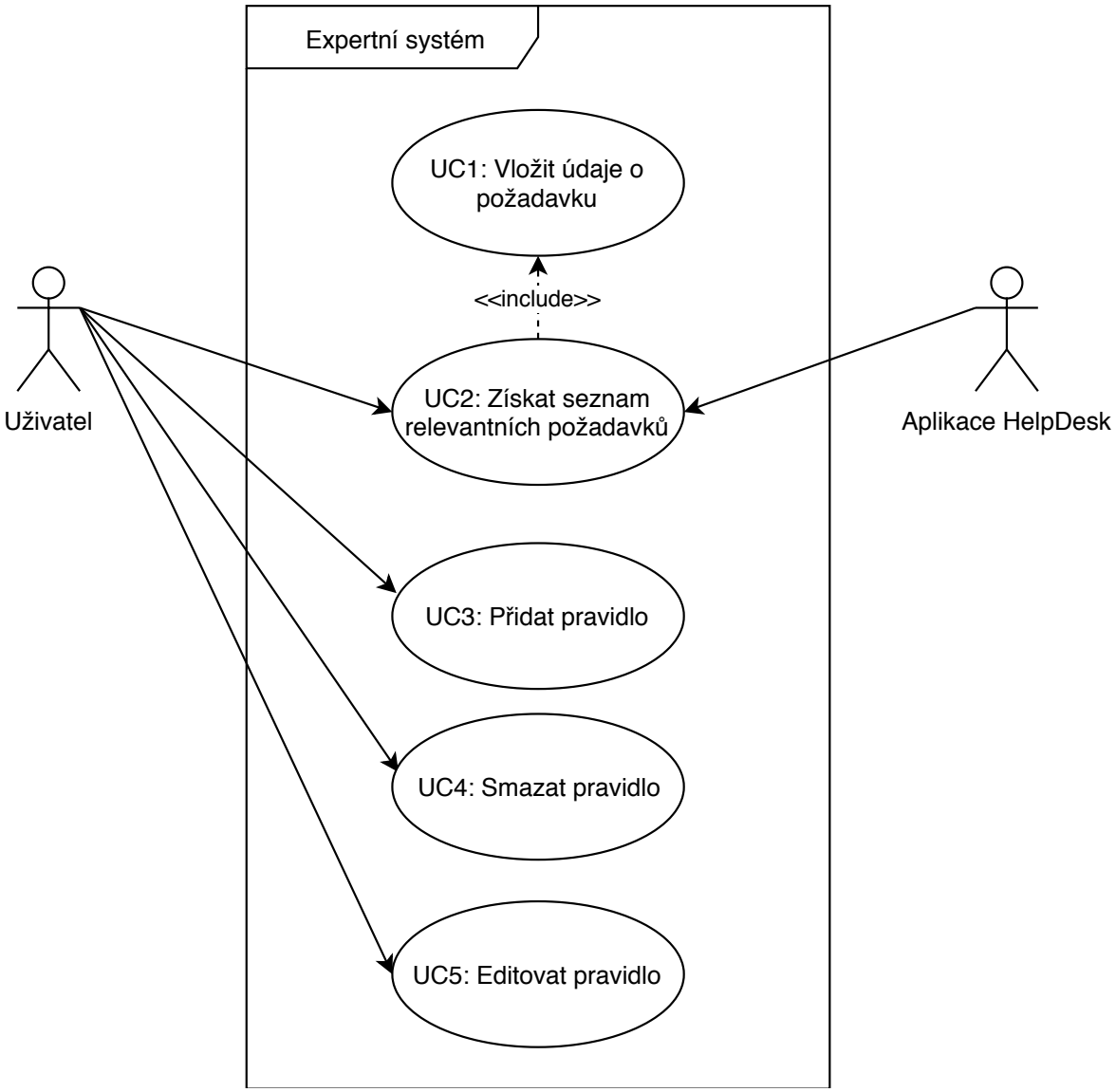
První tři funkční požadavky F.1, F.2 a F.3 se přímo týkají práce expertního systému, čtvrtý funkční požadavek je zaměřený na bázi znalostí, se kterou expertní systém pracuje.

Dle úrovňového modelu zákaznické podpory by měl navrhovaný expertní systém pomáhat pracovníkům helpdesku na 2. úrovni zákaznické podpory. Konkrétně pracovníkovi, který má na starosti příjem požadavku na úrovni konkrétní divize. Jemu jsou delegovány požadavky z 1. úrovně podpory (viz kap. 3.1)

3.2.2 Model případů užití

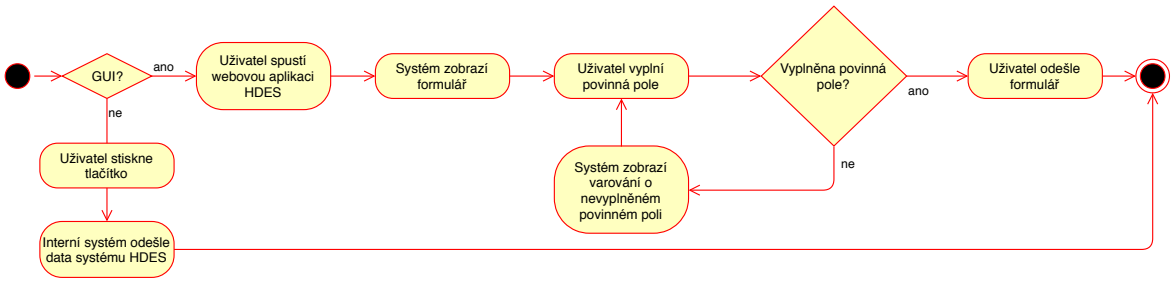
Model případů užití je dalším způsobem, jak modelovat požadavky na navrhovaný systém. Diagram případů užití sestává ze 2 základních komponent: *aktérů* a jednotlivých *případů užití*.

Aktéři jsou v případech užití navrhovaného systému dva - pracovník helpdesku (zjednodušeně Uživatel) a interní helpdesková aplikace (v diagramu případu užití značena jako „Aplikace HelpDesk“), která do systému jako aktér vstupuje v případě, kdy jsou funkce expertního systému volány vzdáleně s pomocí API (viz kap. 3.4.4.4). Diagram případů užití lze vidět na Obrázku 9.



Obrázek 9: *Diagram případů užití expertního systému HDES*
Zdroj: Vlastní zpracování pomocí draw.io

Všechny případy užití jsou v této podkapitole znázorněny pomocí diagramů aktivit standardu UML, které reprezentují jednotlivé činnosti, jež musí aktéři provést, aby bylo dosaženo cílů definovaných případem užití. K většině případů užití je zde popsána i jejich specifikace. Aplikací HDES je v nich myšlen vyvíjený expertní systém (více o názvu v kapitole 3.3).



Obrázek 10: *Diagram aktivity pro UC1: Vložit údaje o požadavku*

Zdroj: Vlastní zpracování pomocí draw.io

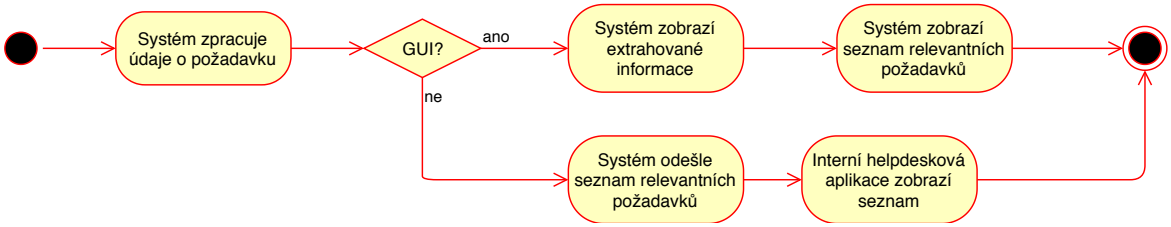
Prvním případem užití je *UC1: Vložit údaje o požadavku*, který je znázorněn diagramem aktivity na Obrázku 10. Specifikace tohoto případu užití je definována v Tabulce č. 1. Cílem tohoto případu užití je úspěšné vložení dat do expertního systému, který s nimi dále pracuje.

Tabulka 1: *Specifikace případu užití UC1: Vložit údaje o požadavku*

Případ užití:	UC1: Vložit údaje o požadavku
Aktéři:	Uživatel, aplikace HelpDesk
Popis:	Umožňuje uživateli či interní helpdeskové aplikaci vložit do expertního systému data, na základě kterých bude usuzovat.
Počáteční podmínka:	Expertní systém běží a uživatel má přístup k GUI nebo je funkční API.
Hlavní scénář:	1. Uživatel spustí webovou aplikaci HDES 2. Systém zobrazí formulář pro vstup dat 3. Uživatel vyplní povinná pole 4. Uživatel odešle formulář
Alternativní scénář:	1a. Uživatel používá pro komunikaci rozhraní API.

Zdroj: Vlastní zpracování

Dalším případem užití je *UC2: Získat seznam relevantních požadavků*. Tento případ užití je znázorněn pomocí diagramu aktivity na Obrázku 11. Nejprve dojde ke zpracování dat, která byla do systému předána pomocí vstupního formuláře nebo pomocí volané funkce API. Na základě typu vstupu dat se systém rozhodne, zda výsledek zpracování a seznam relevatních požadavků zobrazí pomocí GUI nebo odešle zpět systému, který jej volal, a to ve formátu JSON (viz kapitola 3.4.4.4). V Tabulce č. 2 je popsána specifikace tohoto případu užití.



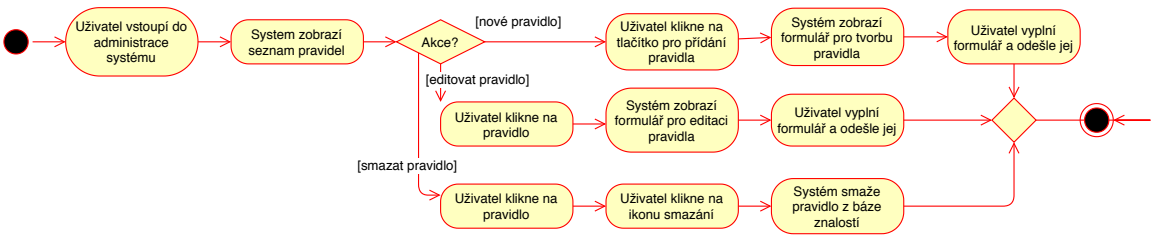
Obrázek 11: *Diagram aktivity pro UC2: Získat seznam relevantních požadavků*

Zdroj: Vlastní zpracování pomocí draw.io

Tabulka 2: *Specifikace případu užití UC2: Získat seznam relevantních požadavků*

Případ užití:	UC2: Získat seznam relevantních požadavků
Aktéři:	Uživatel, aplikace HelpDesk
Popis:	Umožňuje uživateli či interní helpdeskové aplikaci získat seznam relevatních požadavků na základě předtím vložených dat.
Počáteční podmínka:	Uživatel do systému vložil požadované údaje.
Hlavní scénář:	1. Uživatel potvrdí vstupní formulář s daty 2. Systém zobrazí výstup ES a seznam relevantních požadavků
Alternativní scénář:	1a. Uživatel odešle požadavek na funkci API 2a. Systém vrátí uživateli výstup ES a seznam relevantních požadavků

Zdroj: Vlastní zpracování



Obrázek 12: *Diagram aktivity pro UC3, UC4 a UC5*

Zdroj: Vlastní zpracování pomocí draw.io

Případy užití UC3, UC4 a UC5 se týkají administrace báze znalostí, kdy uživatel přidává nové pravidlo či manipuluje s pravidly stávajícími. Tyto případy užití jsou všechny zahrnuty do diagramu aktivit na Obrázku 12. Pro ilustraci je zde také uvedena specifikace případu užití UC3: Přidat pravidlo (viz Tabulka 3). Další dva případy jsou velmi podobné a není je tedy pro potřeby této diplomové práce nutno rozepisovat.

Tabulka 3: *Specifikace případu užití UC3: Přidat pravidlo*

Případ užití:	UC3: Přidat pravidlo
Aktéři:	Uživatel
Popis:	Umožňuje uživateli přidat nové pravidlo do báze znalostí.
Počáteční podmínka:	Uživatel vstoupil do administrace expertního systému.
Hlavní scénář:	1. Uživatel klikne na tlačítko pro vložení požadavku 2. Systém zobrazí modální okno s jednoduchým formulářem 3. Uživatel vyplní údaje o pravidlu a potvrdí formulář 4. Systém uloží nové pravidlo do báze znalostí

Zdroj: Vlastní zpracování

3.3 Návrh

V této podkapitole jsou popisovány návrhy jednotlivých činností expertního systému. Autor této diplomové práce navrhovaný expertní systém pojmenoval HDES, což je zkratka pro „Help Desk Expert System“. Proto bude ve zbývajících částech diplomové práce odkazováno na expertní systém právě tímto názvem.

Dle požadavků definovaných v kapitolách 3.2.1 a 3.2.2 je zde rozpracován návrh hlavních komponent, které bude systém HDES obsahovat. V první řadě se jedná o způsob, jakým bude provádět kategorizaci a detekci duplicitních požadavků.

Požadavky zákazníků, podávané skrze firemní helpdeskovou aplikaci, mají následující strukturu: automaticky generované ID, nadpis požadavku shrnující celý požadavek a detailní popis požadavku. Obsahuje i další data, ta jsou ale pro práci systému HDES nepodstatná nebo se o jejich zapojení do procesu práce systému HDES v této verzi neuvažuje.

3.3.1 Kategorizace požadavku

Každý nadpis i popis požadavku v sobě obsahuje informace, které jej do nějaké míry kategorizují. U požadavku týkající se informačního systému se může například jednat o určitý modul, podprogram nebo databázovou tabulku. Tato klíčová slova by měla být systémem HDES detekována a měla by sloužit ke kategorizaci zkoumaného požadavku. S těmito daty by pak měl systém pracovat v procesu hledání relevantních požadavků. Díky použití klíčových slov tak lze vytvořit jakýsi „recept“, podle kterého lze další požadavky přiřadit do stejné kategorie.

3.3.2 Relevantní požadavky

Díky kategorizaci požadavku pomocí klíčových slov lze získat seznam podobných a souvisejících požadavků - tedy například požadavků, které se týkají stejného podprogramu nebo databázové tabulky. Tento seznam by měl být určitým způsobem ohodnocen a na základě toho seřazen. Zde přichází v úvahu podobnost textu zpracovávaného požadavku a jednotlivých požadavků v seznamu. Díky tomu by uživatel mohl rychleji detekovat duplicitní požadavek nebo alespoň najít řešení z jiných požadavků.

3.3.3 Komunikace se systémem

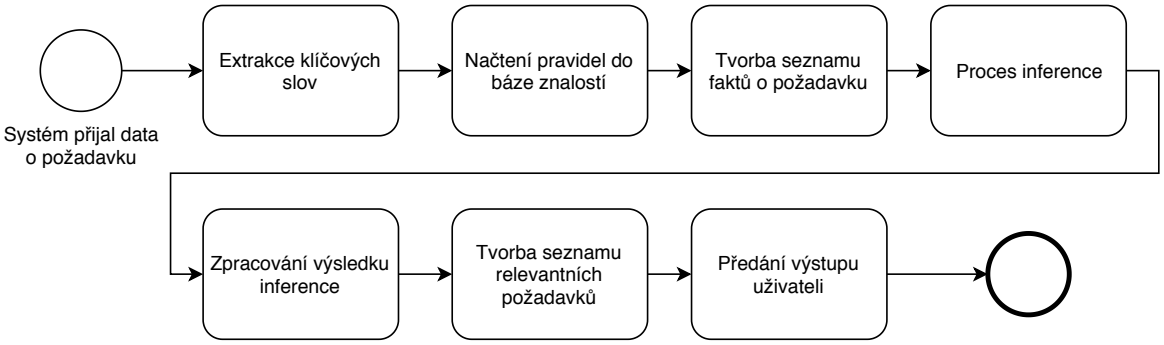
Jak je definováno v požadavcích, systém by měl fungovat jako webová aplikace, kterou lze ovládat a komunikovat s ní přímo nebo pomocí aplikačního programovacího rozhraní (API). Pro přímou komunikaci se jako nejvhodnější zdá být klasický formulář pro vkládání dat. Systém pak uživateli zobrazí výsledek své práce. Pro vzdálenou komunikaci je pak vhodné vytvořit určité funkce, jejichž volání je možné uskutečnit vzdáleně nějakým požadavkem. Výsledek práce systému je pak zaslán uživateli zpět a volající aplikace zpracuje tato návratová data.

Celý proces práce s požadavkem by se tedy dal popsat následovně:

- požadavek je odeslán zákazníkem,
- požadavek je přijat konkrétním oddělením,

- obsluha zadá požadované údaje do systému HDES nebo obsluha helpdesku spustí akci, která předá data (ID, název a popis) expertnímu systému pomocí volání API,
- expertní systém nejprve provede kategorizaci na základě metadat (klíčových slov) - jedním z nich by měl být název modulu, na základě něhož se provede filtrace a vznikne množina relevantních požadavků,
- z extrahovaných klíčových slov a dalších metadat se vytvoří nové pravidlo, které bude mít jako název ID požadavku
- v dalším kroku se v množině relevantních požadavků provede porovnání podobnosti textu požadavku vůči ostatním požadavkům v rámci zúžené domény.

Práce expertního systému je pak znázorněna na BPMN diagramu na Obrázku 13



Obrázek 13: *Proces práce expertního systému*

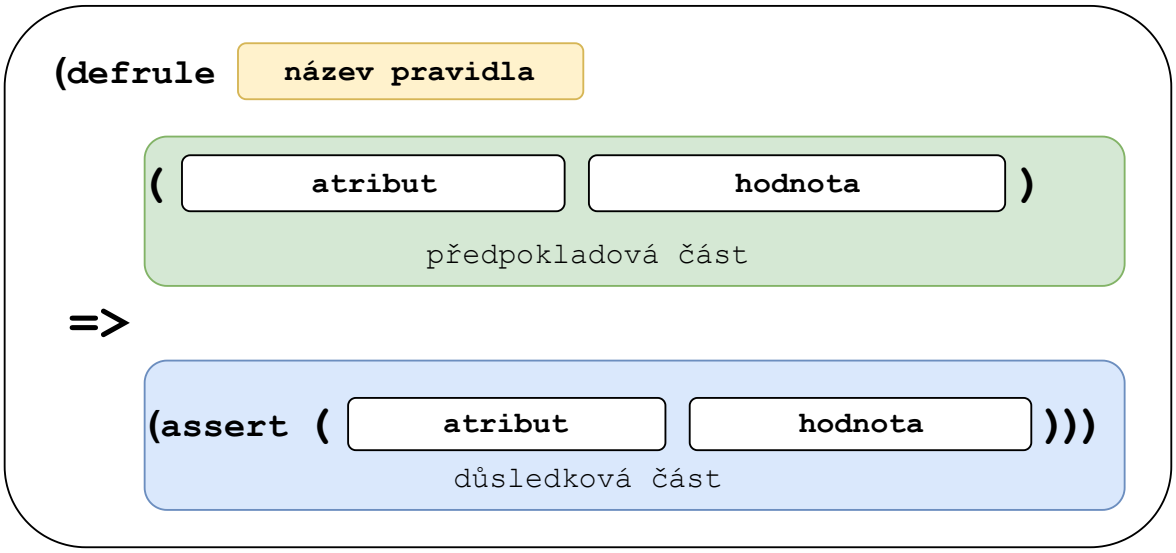
Zdroj: Vlastní zpracování pomocí nástroje draw.io

3.4 Implementace

V této kapitole a podkapitolách se čtenář dozví, jak je expertní systém HDES konkrétně implementován. Při výběru technologií a prostředků bylo přihlíženo k faktu, že by navrhovaný expertní systém měl být webovou aplikací. Jako programovací jazyk bylo tedy zvoleno PHP, což je skriptovací a programovací jazyk, který se řadí k interpretovaným programovacím jazykům a pracuje na straně serveru. Uživatel tedy ve svém prohlížeči nevidí obsah skriptu, ale pouze jeho výstup. Spolu se značkovacím jazykem HTML a kaskádovými styly (CSS) tvoří oblíbenou trojici pro tvorbu webových stránek a webových aplikací. Jazyk PHP byl zvolen především pro jeho rozšířenost, rozšiřitelnost, rozsáhlou dokumentaci a širokou základnu uživatelů.

3.4.1 Báze znalostí a jejich reprezentace

V teoretické části této diplomové práce se čtenář může dočíst o různých způsobech, jak v expertním systému reprezentovat znalosti. Při tvorbě báze znalostí expertního systému HDES bylo přihlíženo k tomu, aby byla reprezentace znalostí pro uživatele jednoduše pochopitelná i použitelná, aby bylo možné znalosti dobře rozšiřovat a aby měla reprezentace optimální technologickou implementaci.



Obrázek 14: *Struktura pravidla v bázi znalostí*
Zdroj: Vlastní zpracování pomocí nástroje draw.io

Jako vhodný způsob reprezentace znalostí byla zvolena pravidla. Vyvíjený expertní systém HDES se tedy řadí do kategorie tzv. pravidlových expertních systémů. Strukturu pravidel, se kterými systém pracuje, je možné vidět na Obrázku 14. Při pohledu na tuto strukturu lze jako první vidět výraz `defrule` - ten značí, že se jedná o definici pravidla. Po něm následuje „název pravidla“, který jednoznačně identifikuje dané pravidlo. Po něm pravidlo obsahuje předpokladovou část, která charakterizuje atributy a jejich hodnoty, které musí být splněny, aby bylo pravidlo aktivováno a tedy byla spuštěna důsledková část. Ta obsahuje výraz `assert`, což lze přeložit jako „tvrdit“. Celé pravidlo lze tedy zjednodušeně popsat následovně: pokud detekované slovo splňuje podmínku v předpokladové části, systém o něm „tvrdí“, že se jedná o modul X, tabulku Y nebo klíčové slovo Z (v závislosti na typu pravidla - viz níže).

Předpokladová i důsledková část pravidla obsahuje vždy nějaký atribut a jeho hodnotu. Tyto atributy jsou popsány v Tabulce 4, která obsahuje seznam atributů, s nimiž expertní systém HDES v současnosti pracuje.

Tabulka 4: *Přehled možných atributů v bázi znalostí*

Atribut	Popis
`keyw-is`	„Klíčové slovo je“ - hodnotou tohoto atributu je každé detekované klíčové slovo v procesu extrakce klíčových slov.
`module-id`	„Zkratka modulu je“ - zkratka detekovaného modulu.
`module-is`	„Název modulu je“ - obsahuje plný název detekovaného modulu.
`table-id`	„Zkratka tabulky je“ - zkratka detekované tabulky.
`table-is`	„Název tabulky je“ - plný název detekované tabulky.
`keyword`	„Obecné klíčové slovo“ - označení pro další klíčová slova.
`relevantni-pozadavek`	„Související požadavek“ - obsahuje ID požadavku, který byl vyhodnocen jako relevantní.

Zdroj: Vlastní zpracování

I na základě toho lze pravidla uložená v bázi znalostí rozdělit do několika skupin:

- *moduly* - pravidla, která dešifrují názvy a ID jednotlivých podprogramů a modulů informačního systému, kterého se požadavek týká. Příklad takového pravidla:

```
(defrule sgui
  (keyw-is "sgui")
  => (assert (module-is "Ulozene GUI masek"))
     (assert (module-id "sgui")))
```

Ukázka kódu 3: Pravidlo dešifrující zkratku `sgui`

Toto pravidlo říká, že pokud je při extrakci klíčových slov detekováno slovo „sgui“ (což je zkratka modulu), tak se do seznamu faktů uloží dvě nové informace: `(module-is „Uložené GUI masek")`, které říká, že detekovaný modul má takovýto úplný název a `(module-id „sgui")`, říkající systému, že toto je zkratka detekovaného modulu a slouží hlavně k záznamu, že došlo k detekci právě tohoto modulu, s čímž pak pracují další části inferenčního mechanismu (více v kapitole 3.4.3).

- *tabulky* - pravidla dešifrující zkratky databázových tabulek. Důsledkem těchto pravidel je vložení názvu tabulky a její zkratky do seznamu faktů. Jedná se o stejný princip, jako u pravidel týkajících se modulů.

```
(defrule lpe
  (keyw-is "lpe")
  => (assert (table-is "Sklady"))
     (assert (table-id "lpe")))
```

Ukázka kódu 4: Pravidlo dešifrující zkratku `lps`

- *obecná klíčová slova* - pravidla sloužící k vyhodnocení obecného klíčového slova (definovaného uživatelem). Výsledkem aktivace těchto pravidel je atribut `keyword` obsahující obecný tvar klíčového slova. Podmínka (předpoklad) obsahuje slovo ve svém základním tvaru (slovo prošlo procesem tzv. lemmatizace - viz kapitola 3.4.2). Příklad je možné vidět na Ukázce kódu 5, kde v předpokladové části je uvedeno slovo „databáz“, což je základem slova „databáze“, ale i jeho dalších tvarů jako „databázový, databázového“, díky čemuž je systém schopen je detekovat:

```
(defrule databaze
  (keyw-is "databáz")
  => (assert (keyword "databáze")))
```

Ukázka kódu 5: Pravidlo detekující klíčové slovo „databáze“

Toto pravidlo říká, že pokud při procesu extrakce klíčových slov bude nalezeno slovo, jehož základem je slovo „databáz“, systém jej označí jako klíčové slovo a uloží tuto informaci do seznamu faktů.

- *relevantní požadavky* - pravidla sloužící k označení relevantních požadavků. Příklad takového pravidla je uveden na Ukázce kódu 6.

```
(defrule HD-2019-1
  (module-id "sgui")
  (keyword "databáze")
  => (assert (relevantni-pozadavek HD-2019-1)))
```

Ukázka kódu 6: Pravidlo označující relevantní požadavek

Smyslem těchto pravidel je najít v množině již řešených požadavků požadavky relevantní. Toho je docíleno tím, že každý požadavek zpracovaný expertním systém má vytvořené své pravidlo, které ve své předpokladové části obsahuje detekovaná klíčová slova. Důsledkem po aktivaci tohoto pravidla je pak označení požadavku za relevantní, a to uložením informace do seznamu faktů. Expertní systém HDES pak z tohoto seznamu čte všechny výskyty atributu `relevantni-pozadavek` a může tak zpracovat seznam relevantních požadavků.

Veškerá pravidla z báze znalostí jsou uloženy v databázi (viz kapitolu 3.4.5). Systém při každém spuštění tato pravidla z databáze načte do paměti a dále s nimi pracuje inferenční mechanismus (viz kapitolu 3.4.3). V současné době tvoří bázi znalostí přes 2100 pravidel a uživatel ji může snadno administrovat skrze dedikovanou část webové aplikace (viz kapitolu 3.4.4 popisující uživatelské prostředí).

3.4.2 Extrakce klíčových slov

Důležitým prvkem expertního systému HDES je proces extrakce klíčových slov ze zadaného požadavku. Jedná se o první akci, která je s vloženými informacemi o požadavku provedena. Výstupem tohoto procesu je seznam faktů o zadaném požadavku. Jedná se o jakousi bázi pro systém aktuálně známých informací. S tímto seznamem faktů později pracuje inferenční mechanismus, který jej dále rozšiřuje pomocí procesu usuzování.

Proces extrakce je řešen pomocí skriptu PHP, ve kterém jsou potřebné funkce k zajištění detekce klíčových slov. Tento proces je zjednodušeně popsán BPMN diagramem na Obrázku 15. Nejprve dojde ke spojení obsahu nadpisu a popisu požadavku. Následně systém provede operace, kdy jsou odstraněna interpunkční znaménka, text je převeden na malá písmena a je rozdělen na jednotlivá slova. V následném kroku dojde k tzv. lemmatizaci, kdy jsou všechna slova převedena na jejich základní tvar (systém tak dokáže detekovat klíčová slova, jež mohou mít podobné tvary). Seznam slov je následně porovnán s množinou známých klíčových slov, na základě čehož je sestaven seznam faktů, se kterými expertní systém pracuje v procesu inference.

Tento proces lze ilustrovat na následujícím příkladu. Mějme požadavek s následujícím textem: „Dobrý den, při pokusu o spuštění podprogramu SGUI se zobrazilo hlášení o chybě

v databázi. Prosím o rychlou opravu.“. Systém jej nejprve spojí s nadpisem, který v našem případě může být například „Chyba v databázi“. Poté dojde k odstranění interpunkčních znamének, převedení na malá písmena a celý řetězec je rozdělen do pole řetězců dle jednotlivých slov. V notaci JSON to pak vypadá následovně:

```
["dobrý", "den", "při", "pokusu", "o", "spuštění", "podprogramu",
"sgui", "se", "zobrazilo", "hlášení", "o", "chybě", "v", "databázi",
"prosím", "o", "rychlou", "opravu", "chyba", "v", "databázi"]
```

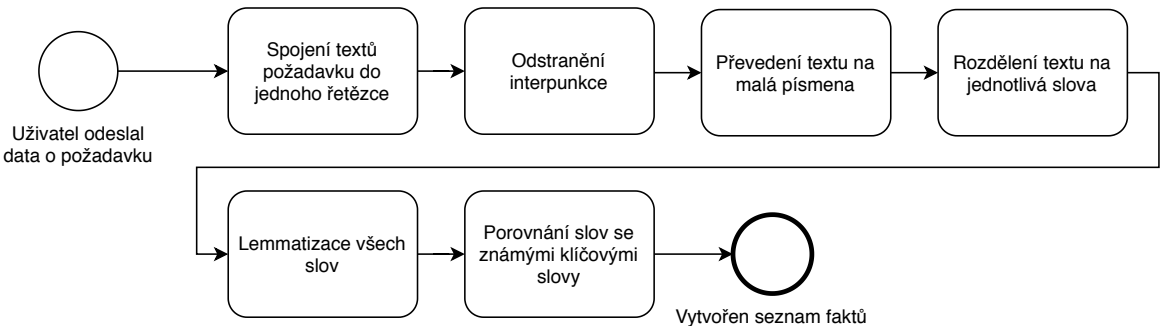
Ukázka kódu 7: Pole slov požadavku v notaci JSON

Po tzv. lemmatizaci má pole následující tvar:

```
["dobr", "den", "při", "pokus", "o", "spuštěn", "podprogram",
"sgui", "se", "zobraz", "hlášen", "o", "chyb", "v", "databáz",
"prosí", "o", "rychl", "oprav", "chyb", "v", "databáz"]
```

Ukázka kódu 8: Pole slov požadavku po lemmatizaci v notaci JSON

Nyní dochází k porovnání všech řetězců v poli s polem známých klíčových slov. Byla nalezena shoda ve dvou klíčových slovech, a to „sgui“ a „databáz“. Do seznamu faktů jsou tedy uloženy dva fakty: `(keyw-is „sgui")` a `(keyw-is „databáz")`. S ostatními slovy systém již nepracuje a pokračuje dále procesem inference.



Obrázek 15: *Proces extrakce klíčových slov*

Zdroj: Vlastní zpracování pomocí draw.io

3.4.3 Inferenční mechanismus

Inferenční mechanismus může programátor vytvořit sám, to je ale nákladná a časově náročná záležitost. Proto je vhodnější použít již vytvořené řešení a držet se tzv. přístupu „Do not

reinvent the wheel“. Jelikož bylo v předchozí kapitole deklarováno, že bude expertní systém pracovat se znalostmi reprezentovanými pomocí pravidel, přichází v úvahu následující řešení: nástroj pro tvorbu expertních systémů CLIPS. Běžná verze tohoto nástroje pracuje jako desktopová aplikace, kterou tvoří jakýsi shell (interpret příkazů), pomocí něhož uživatel nástroj ovládá (přidává fakty, spouští proces inference aj.). CLIPS má ale mnoho dalších implementací, kdy jej lze například použít v Java nebo .NET aplikacích. Důležitá je existence PHP pluginu pro CLIPS zvaného `php-clips`[2]. Tento plugin se instaluje jako plugin instalace PHP na webovém serveru, na kterém běží systém HDES. Díky tomu jsou pak přístupné metody pro ovládání nástroje CLIPS v prostředí, ve kterém běží PHP skripty zodopvědné za běh aplikace.

CLIPS byl v základním měřítku popsán v kapitole 1.1.2 popisující jazyky a nástroje pro reprezentaci znalostí. Velkou výhodou je, že jeho inferenční mechanismus je založen na Rete algoritmu, viz kapitola 1.1.2 (jazyk Jess), čímž se výrazně snižuje čas práce inferenčního mechanismu. Rete algoritmus totiž nejprve vyhodnotí všechna aplikovatelná pravidla, jejichž předpoklady odpovídají aktuálním faktům, se kterými systém pracuje. Následně algoritmus vyhodnotí první pravidlo na seznamu aplikovatelných pravidel a provede znovu celý proces hodnocení aplikovatelných pravidel. Tímto způsobem se může seznam vyhovujících pravidel měnit. Pokud by tento proces měl být řešen klasickým sekvenčním způsobem, který zahrnuje řazení a vnořování tzv. IF-THEN-ELSE výrazů, nelze dosáhnout stejných výsledků. Problém nastává obzvláště v případě, kdy bychom požadovali inferenci - tedy že je určité pravidlo aktivováno na základě aktivace pravidla jiného. To by se klasickým přístupem těžko řešilo, obzvláště v případě, kdy je v bázi znalostí přítomno velké množství pravidel. Toto je také důvod, proč byl zvolen nástroj CLIPS. V bázi znalostí je přítomno velké množství pravidel popisující jednotlivé moduly, tabulky a předchozí zpracované požadavky. Jak již bylo zmíněno, v současné době báze znalostí obsahuje přes 2100 pravidel. Vytvořit tedy síť IF-THEN-ELSE příkazů by bylo velmi náročné.

Důležité je i hledisko výkonu inferenčního mechanismu. V porovnání se sekvenčním přístupem, kdy jsou data a fakty kontrolovány se všemi předpoklady, algoritmus Rete nejprve vybere pravidla, jejichž předpoklady odpovídají dostupným faktům. Tím je zajištěna dostatečná rychlost zpracování inferenčním mechanismem.

[2] `https://github.com/guitarpoet/php-clips`

Pokud se vrátíme k ilustračnímu příkladu z minulé podkapitoly, kdy byla z požadavku extrahována klíčová slova, proces zpracování expertním systémem dále pokračuje procesem inference.

1. Uživatel vloží do formuláře data požadavku, ze kterých expertní systém HDES extrahuje následující klíčová slova: „databáze“ a „SGUI“ (viz kapitolu 3.4.2). To tedy znamená, že se požadavek nějakým způsobem týká modulu SGUI a databáze. Tato klíčová slova jsou systému předána v seznamu faktů, a to v následujícím tvaru: `(keyw-is „sgui")` a `(keyw-is „databáz")`

2. Odesláním formuláře je spuštěn proces inference zajištěný rozhraním CLIPS, kdy se nejprve vyberou pravidla, jejichž předpoklady `(keyw-is „sgui")` a `(keyw-is „databáze")` jsou obsaženy v seznamu faktů. Jsou tedy vybrána následující pravidla:

```
(defrule sgui
  (keyw-is "sgui")
  => (assert (module-is "Ulozene GUI masek"))
     (assert (module-id "sgui")))
(defrule databaze
  (keyw-is "databáz")
  => (assert (keyword "databáze")))
```

Ukázka kódu 9: Předpis pro definici pravidel `sgui` a `databaze`

Význam těchto pravidel byl již popsán v kapitole 3.4.1.

3. Do seznamu faktů jsou tedy díky aktivaci pravidel vloženy následující nové fakty: `(module-is „Ulozene GUI masek")`, `(module-id „sgui")` a `(keyword „databáze")`. Seznam faktů má tedy následující podobu:

```
(keyw-is "sgui")
(keyw-is "databáz")
(module-is "Ulozene GUI masek")
(module-id "sgui")
(keyword "databáze")
```

Ukázka kódu 10: Seznam faktů po aktivaci první sady pravidel

Rete algoritmus v inferenčním mechanismu opět provede vybrání pravidel, pro jejichž předpoklady jsou dostupné fakty v seznamu faktů. Tím, že do seznamu přibyly další dva fakty, objevují se ve výčtu vhodných pravidel další pravidla, a to pravidla popisující již zpracované požadavky. Může se jednat například o následující pravidlo:

```
(defrule HD-2019-1
  (module-id "sgui")
  (keyword "databáze")
  => (assert (relevantni-pozadavek HD-2019-1)))
```

Ukázka kódu 11: Definice pravidla popisujícího požadavek `HDES-2019-1`

Pravidlo sestává ze své jednoznačné identifikace, v našem případě `HD-2019-1`. Dále ze dvou předpokladů `(module-id „sgui")` a `(keyword „databáze")`, které předpokládají, že byla v systému aktivována pravidla, která do seznamu faktů uložila tyto hodnoty.

4. Jelikož byly splněny oba předpoklady pravidla, pravidlo bylo aktivováno. Tím se do seznamu faktů vloží nový fakt `(relevantni-pozadavek HD-2019-1)`. Pokud existuje více takových pravidel (= více relevantních požadavků), tato pravidla se aktivují a rozšíří seznam faktů. Po dokončení procesu inference si aplikace HDES z prostředí CLIPS vyžádá seznam aktuálních faktů, se kterými dále pracuje vysvětlovací modul a proces porovnání s aktuálním požadavkem. Finální podoba seznamu faktů je následující:

```
(keyw-is "sgui")
(keyw-is "databáz")
(module-is "Ulozene GUI masek")
(module-id "sgui")
(keyword "databáze")
(relevantni-pozadavek HD-2019-1)
(relevantni-pozadavek HD-2019-5)
(relevantni-pozadavek HD-2019-8)
(relevantni-pozadavek HD-2019-4)
```

Ukázka kódu 12: Podoba seznamu faktů po dokončení inference

O zobrazení výsledku práce inferenčního mechanismu se stará uživatelské rozhraní, které je popisováno v následující kapitole.

3.4.4 Uživatelské rozhraní

Uživatelské rozhraní je jedno ze dvou možností interakce s navrhovaným expertním systémem HDES. Druhá možnost, komunikace pomocí API, je popsána v kapitole 3.4.4.4. Jelikož se jedná o webovou aplikaci, je grafické uživatelské rozhraní řešeno pomocí jazyka HTML, kaskádových stylů CSS a skriptovacího programovacího jazyka PHP. Jedná se tedy o sadu webových stránek, pomocí kterých uživatel systém ovládá a komunikuje s ním. Expertní systém HDES má několik základních pohledů:

- vstupní formulář pro manuální vstup dat (viz kapitola 3.4.4.1),
- výstup expertního systému a vysvětlovacího modulu (viz kapitola 3.4.4.2),
- administrační pohled s možností editace báze znalostí (viz kapitola 3.4.4.3) a
- nastavení rozhraní API (viz kapitola 3.4.4.4).

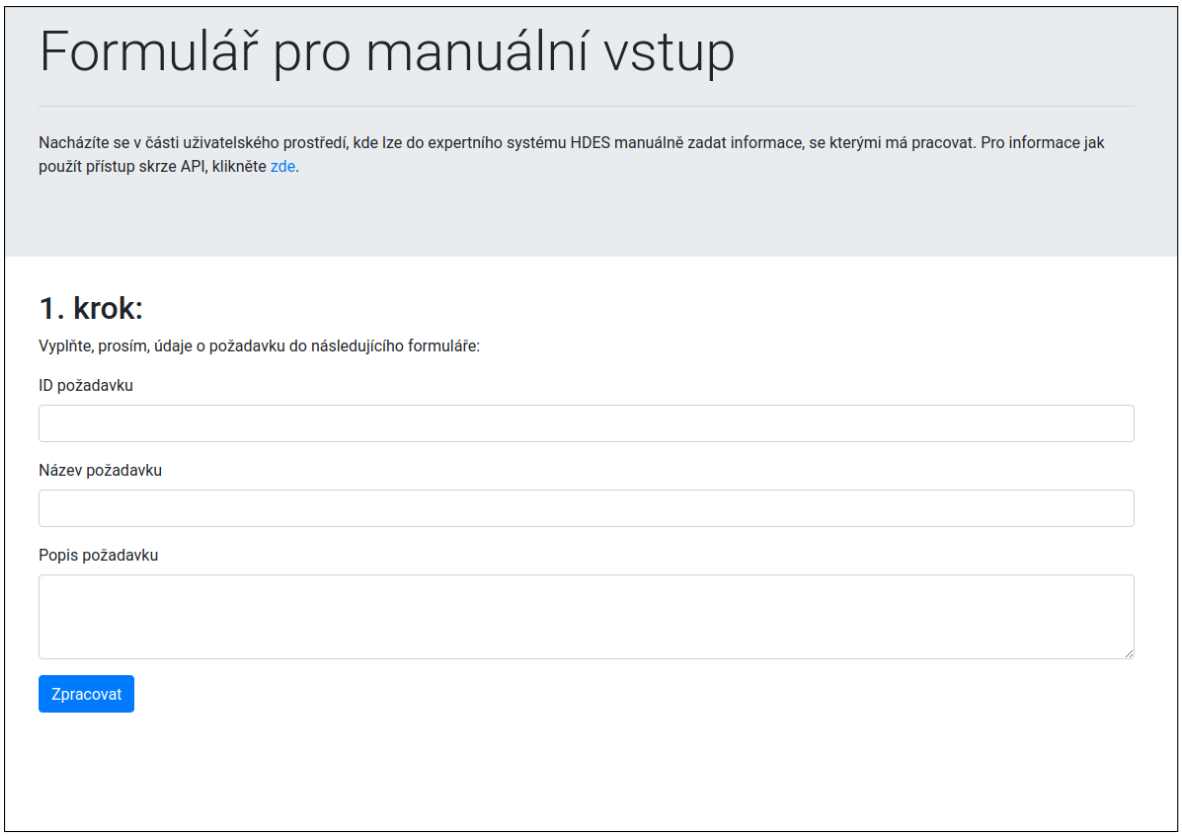
3.4.4.1 Vstupní formulář

Protože se s expertním systémem dá komunikovat i skrze uživatelské rozhraní, je potřeba aby byl v projektu přítomen také vstupní formulář, skrze který uživatel zadá potřebné informace, se kterými následně expertní systém pracuje. Jak je vidět na Obrázku 16, vstupní formulář obsahuje následující pole:

ID Požadavku - jedná se o jednoznačnou identifikaci požadavku, převzatou z interní helpdeskové aplikace. S pomocí tohoto ID expertní systém vytvoří pro nový požadavek nové pravidlo v bázi znalostí, který je pak díky tomu možné identifikovat v procesu hledání duplicit a relevantních požadavků.

Název požadavku - název požadavku je jedním ze 2 vstupních polí, ze kterých jsou extrahována klíčová slova charakterizující jednotlivé požadavky. Zároveň také stručně popisují obsah požadavku.

Popis požadavku - jedná se detailní rozpis zákaznického požadavku. Toto je druhé vstupní pole, ze kterého systém extrahuje klíčová slova a vytváří tak seznam faktů, se kterými inferenční mechanismus systému pracuje.



Obrázek 16: *Vstupní formulář expertního systému*

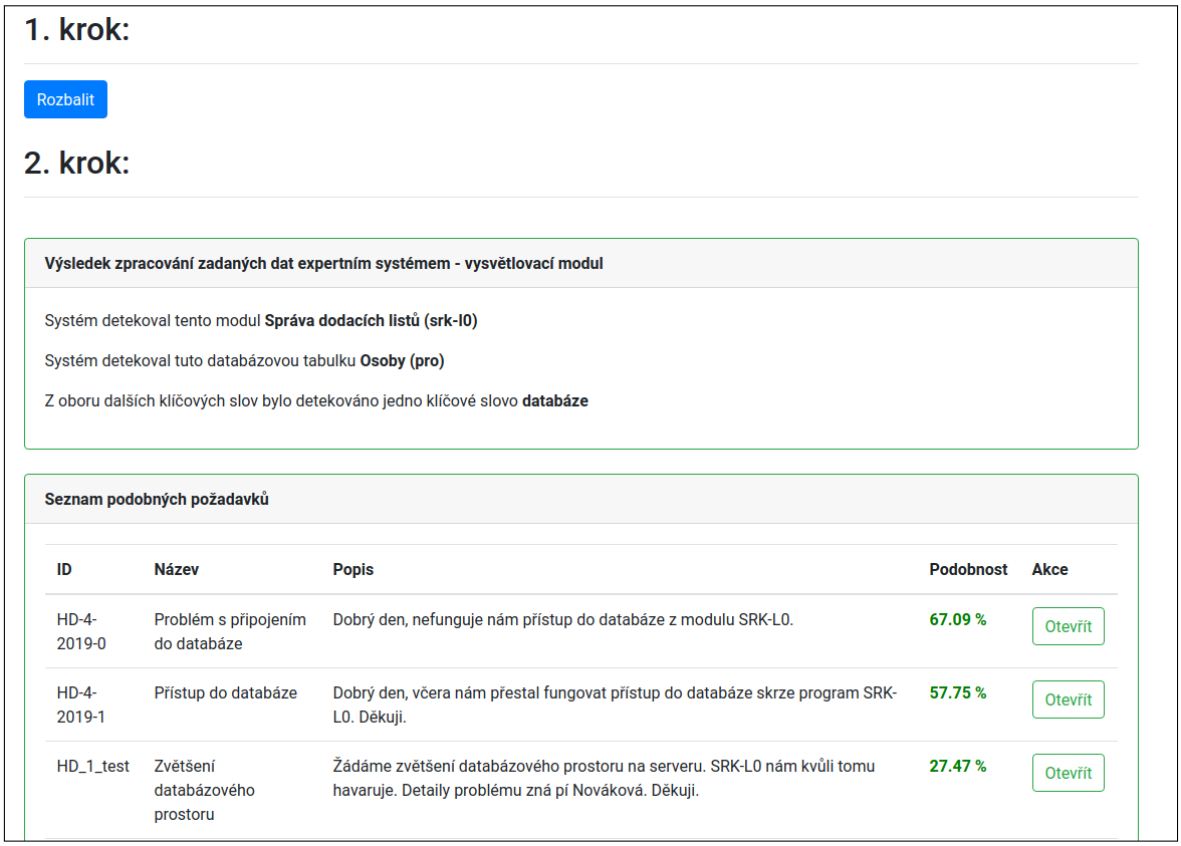
Zdroj: Vlastní

Tato část expertního systému slouží pouze jako vstup a žádné operace v této fázi ještě neprobíhají. Spuštění procesu usuzování je iniciováno kliknutím na tlačítko „Zpracovat“. Tento proces je již popsán v kapitole 3.4.3. Po dokončení procesu usuzování, je uživateli zobrazen pohled s výstupem expertního systému.

3.4.4.2 Výstup expertního systému a vysvětlovací modul

Uživateli se po dokončení práce inferenčního mechanismu zobrazí pohled rozdělený na 3 části (viz Obrázek 17). První část je vstupní formulář z předchozího pohledu, kde jsou vyplněny informace, které uživatel v tomto kroku zadal. Tento formulář je ale skrytý, aby zbytečně nezaplňoval plochu pohledu. Uživatel jej jednoduše zobrazí kliknutím na tlačítko „Rozbalit“. Druhou část tvoří tzv. vysvětlovací modul, který má na starosti uživateli „vysvětlit“, na základě čeho a jak experntní systém HDES rozhodoval. Popisuje zde detekovaná klíčová slova

ve formě názvů modulů, tabulek a uživatelem definovaných klíčových slov. Třetí část tvoří seznam podobných helpdeskových požadavků, které byly v minulosti řešeny. Tyto požadavky jsou řazeny podle procent jejich podobnosti s nově zadaným požadavkem. U každého z požadavku v seznamu je tlačítko „Otevřít“, které otevře modální okno (viz Obrázek 18), ve kterém jsou zobrazeny uložené informace o daném požadavku.

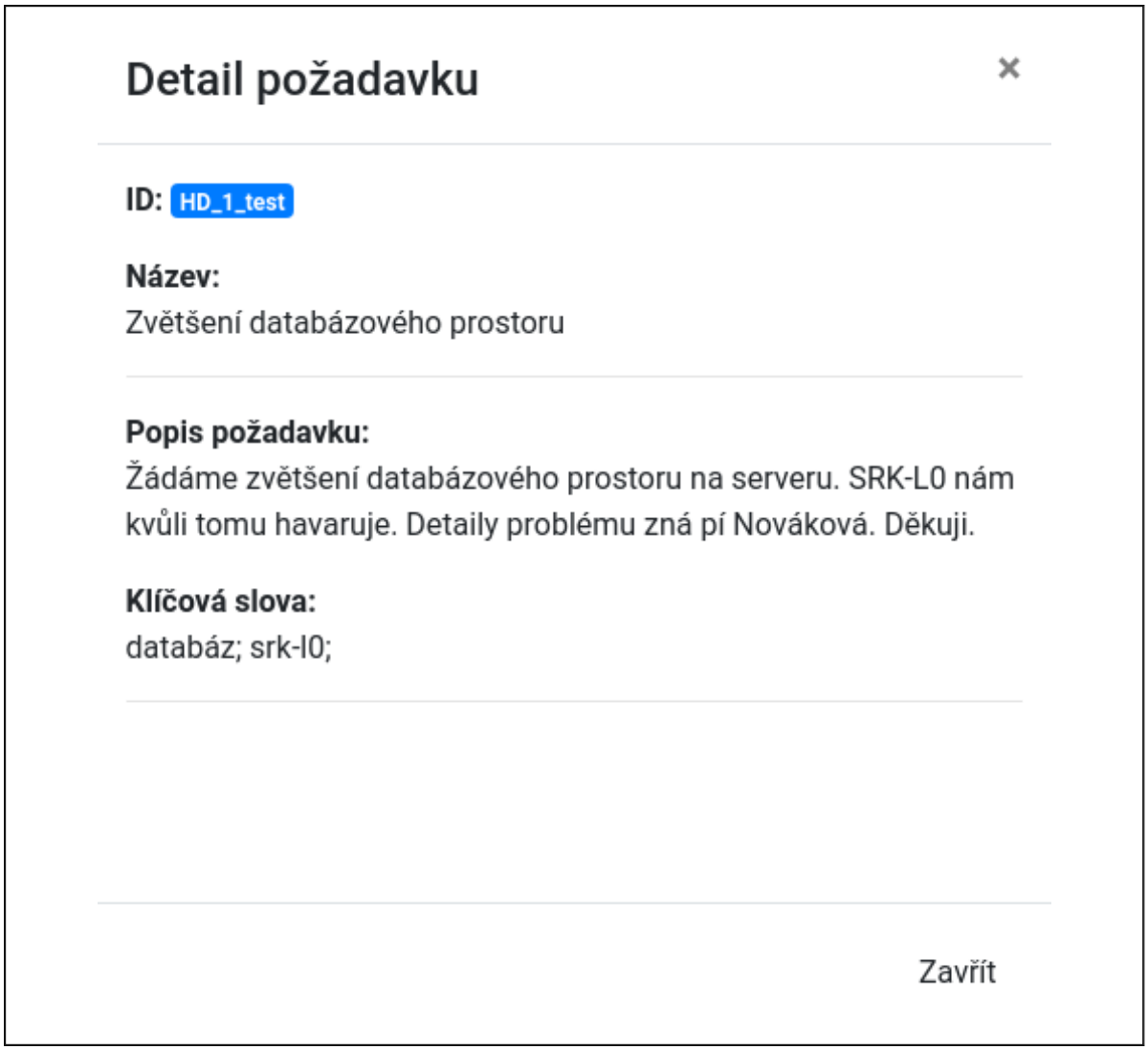


Obrázek 17: *Výstup expertního systému*

Zdroj: Vlastní

3.4.4.3 Administrace báze znalostí

Základní funkcí tohoto pohledu je uživateli ve srozumitelném formátu zobrazit obsah báze znalostí expertního systému HDES. K dosažení srozumitelnosti se snaží přispět rozdělení seznamů pravidel dle jednotlivých kategorií, kterých se konkrétní pravidla týkají. Jedná se tedy o pravidla vygenerovaná na základě zpracovaných požadavků a na základě nichž je pak systém schopen detekovat relevantní požadavky. Další kategorií jsou pravidla popisující jednotlivé moduly a názvy databázových tabulek, které jsou součástí informačního systému, který firma vyvíjí. Poslední kategorií jsou uživatelsky definovaná klíčová slova. Administraci báze znalostí lze vidět na Obrázku 19.

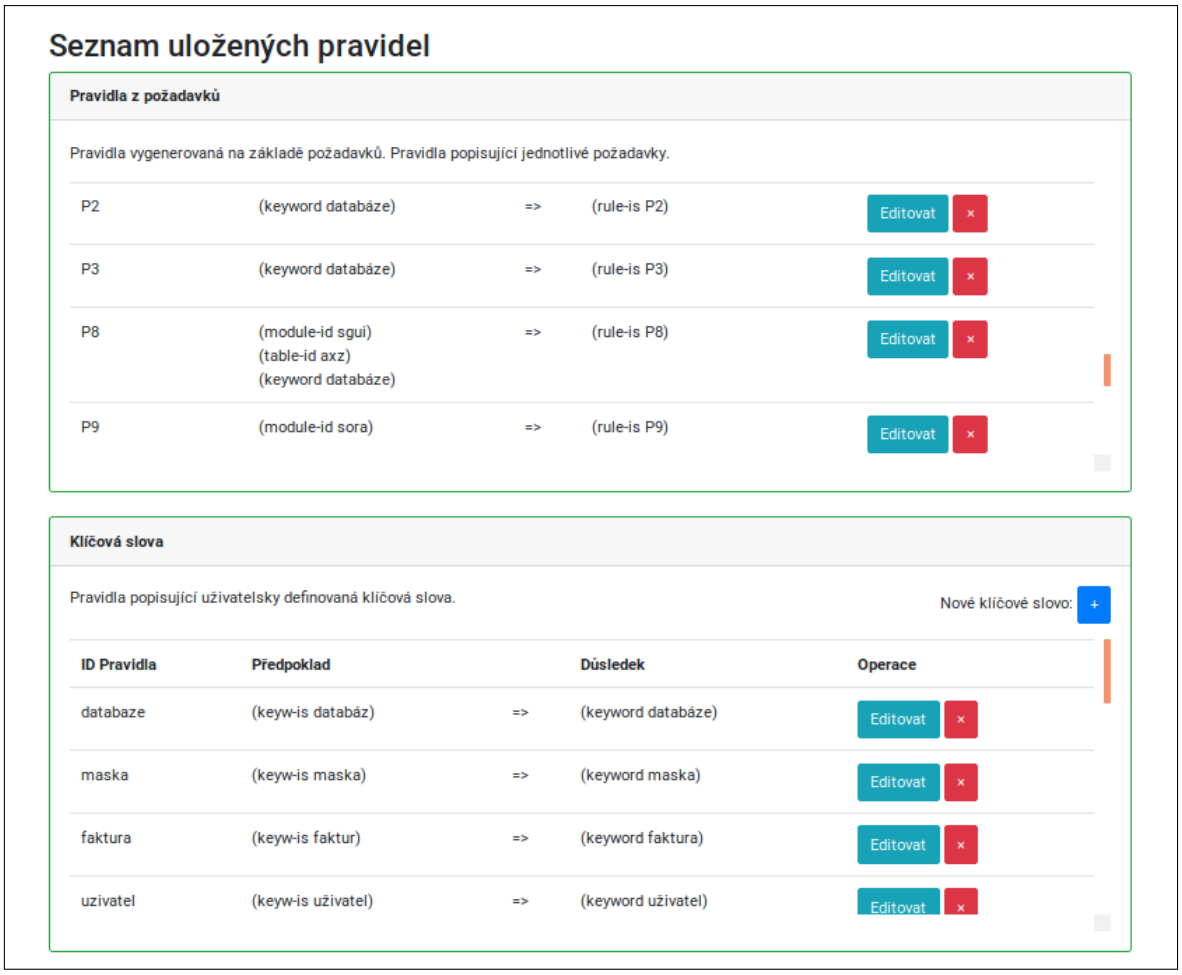


Obrázek 18: *Detail relevantního požadavku*

Zdroj: Vlastní

U každé z kategorií pravidel má uživatel možnost vytvořit a vložit do báze znalostí nové pravidlo. Toho docílí kliknutím na tlačítko se znaménkem „+“. Uživateli se tak zobrazí modální okno s nabídkou vstupních polí. Tato nabídka je upravena na míru každé kategorii pravidla. Zobrazí se tedy daná proměnná, se kterou předpoklad pracuje, a uživatel jen vyplní její obsah. Příkladem tedy může být nové uživatelsky definované klíčové slovo - zobrazí se formulář se vstupním polem, které bude označeno jako `keyword`, kde uživatel vyplní dané klíčové slovo a po stisku potvrzovacího tlačítka dojde k vytvoření nového pravidla a jeho nahrání do databáze (více o způsobu uložení pravidel v databázi v kapitole 3.4.5).

V administraci báze znalostí se také nachází možnost hromadně nahrát pravidla do systému. Uživatel tak má možnost hromadně nahrát pravidla pro dešifrování zkratek modulů a tabulek.



Obrázek 19: *Administrace báze znalostí - seznam pravidel*

Zdroj: Vlastní

Tuto možnost může uživatel využít například v případě, že je potřeba znovu naplnit bázi znalostí a má k dispozici seznam modulů a tabulek uložených v souboru ve formátu CSV (Comma Separated Values).

3.4.4.4 Rozhraní API a návrh jeho použití

Jak již bylo několikrát zmíněno v předchozích kapitolách, expertní systém HDES nabízí i funkci API (lze také nazývat HDES API), což je aplikační programové rozhraní, skrze které lze ovládat expertní systém z jiné aplikace. Základní myšlenkou je, že z jedné aplikace je odeslán požadavek obsahující zakódovaná data, která přijímající aplikace zpracuje a provede požadovanou operaci. Není tedy potřeba přímý přístup k aplikaci pomocí nějakého formuláře či jiné formy uživatelského prostředí.

HDES API nabízí několik funkcí, které lze zavolat „z vnějšku“. Základní funkcí je celý proces zpracování požadavku expertním systémem. Jedná se o operaci, kdy jsou skrze HDES API předány expertnímu systému potřebné informace, které následně automaticky zpracuje a volající aplikaci zpět pošle výsledek zpracování, to jest detekovaná klíčová slova a vygenerovaný seznam relevantních požadavků.

Pomocí dalších funkcí lze například do báze znalostní expertního systému vložit nové pravidlo a provést úpravy pravidel jiných. To však za předpokladu, že uživatel přesně ví, co a jak chce editovat. V opačném případě je výhodnější využít grafického uživatelského prostředí přímo ve webové aplikaci, které nabízí intuitivnější prostředí pro editaci pravidel.

Díky rozhraní API lze navrhnout, jak by se mohl expertní systém HDES zapojit do současné aplikace pro správu helpdeskových požadavků. U nově přijatého požadavku by mohlo být přítomno tlačítko, které po jeho stisknutí vytvoří HTTP požadavek na adresu směřující ke konkrétní funkci HDES API. V těle požadavku by pak měly být informace o požadavku, konkrétně tedy jeho ID, nadpis a jeho popis. Tyto údaje musí být zakódovány ve formátu JSON (JavaScript Object Notation). Po odeslání je HTTP požadavek přijat na straně webové aplikace HDES, kde je přijat konkrétním PHP skriptem, který dekóduje JSON data uložená v těle HTTP požadavku. S těmito daty pak provede extrakci klíčových slov a proces inference, na základě kterého vznikne seznam relevatních požadavků. Tento seznam je zakódován do tvaru JSON a odeslán zpět helpdeskové aplikaci. Seznam obsahuje seznam ID požadavků, díky kterým by pak mohla helpdesková aplikace zobrazit jejich obsah a informace o nich.

3.4.5 Databázová vrstva

Tato podkapitola se zabývá reprezentací zpracovaných požadavků a pravidel expertního systému HDES v databázi. Jako databázová technologie byla zvolena MySQL. MySQL je produkt v současnosti vlastněný společností Oracle a dostupný pod licencí GPL 2 (General Public License) - licencí pro svobodný software. MySQL je databázová technologie pracující s relačním databázovým modelem. Byla zvolena z důvodu své otevřenosti, snadné instalace, rozsáhlé dokumentace a dobré spolupráce se skriptovacím jazykem PHP.

Jelikož nahrávání pravidel z CSV souboru při každém spuštění systému není příliš praktické řešení, a to hlavně pro případ potřeby editace pravidel a jejich přidání nebo smazání, byl zvolen způsob reprezentace a uložení v databázi. Inferenční mechanismus v systému HDES

pracuje s pravidly popsanými jazykem LISP. Jak tedy uložit takové pravidlo napsané v jazyce LISP do běžné MySQL databáze tak, aby bylo jednoduše rozšiřitelné a editovatelné?

Každé pravidlo se skládá ze dvou hlavních částí - předpokladu (antecedentu) a důsledku (konsekventu). Jak je vidět na Ukázce kódu 9, pravidlo v expertním systému může obsahovat několik částí své důsledkové části (ale i několik dílčích předpokladů). Pro co nejjednodušší uložení v databázi byl zvolen databázový model znázorněný diagramem tříd na Obrázku 20. Každá jednotlivá část předpokladu i důsledku je reprezentována jedním řádkem v tabulce Pravidla.

Obrázek 20: *Diagram tříd popisující použité databázové tabulky*
Zdroj: Vlastní zpracování pomocí draw.io

Jednotlivé atributy v databázové tabulce Pravidla:

- **id** - automaticky generované ID, expertní systém HDES s ním nepracuje.
- **id_pravidla** - tento atribut popisuje jednoznačnou identifikaci pravidla v bázi znalostí. Jelikož je pravidlo reprezentováno v databázi několika řádky, je hodnota atributu opakována minimálně 2×, protože alespoň jeden řádek je antecedent a druhý konsekvent. Pro pravidla generovaná na základě zpracovaného požadavku, je použito jednoznačné ID požadavku. Pro moduly a tabulky je použit název konkrétního modulu a tabulky.
- **typ** - pomocí tohoto atributu je u záznamu rozlišeno, zda se jedná o předpoklad pravidla nebo o jeho důsledek. Předokladová část je reprezentována číslem 0 a důsledková část pomocí čísla 1.
- **atribut** - tento atribut obsahuje informaci o proměnné, ve které je uložena hodnota, reprezentována dalším atributem. Příkladem může být `(keyw-is „databáze")`.

- **hodnota** - zde je uložena hodnota atributu. Dle předchozího příkladu se jedná o tuto část: `(keyw-is „databáze")`
- **vztah** - atribut obsahuje informaci o vztahu, který mezi sebou mají jednotlivé části předpokladu. Může tedy nabývat například hodnot „and“ nebo „or“ vyjadřující logické operátory AND a OR.
- **skupina** - tento atribut značí, do jaké skupiny pravidel zaznamenané pravidlo patří. Značí, zda se jedná o pravidlo o modulu, tabulce, klíčovém slovu a nebo pravidlo popisující požadavek.
- **poznamka** - atribut může obsahovat uživatelem definovanou poznámku.

O požadavku jsou pak ukládány následující informace:

- **id_pozadavku** - ID požadavku, jeho jednoznačná identifikace. Jedná se o stejné ID, které má požadavek přidělen ve firemní helpdeskové aplikaci.
- **nazev** - obsahuje název (nadpis) požadavku, tak jak jej definoval zákazník při založení požadavku.
- **popis** - obsahuje text popisu požadavku.
- **klic_slova** - obsahuje středníkem oddělená klíčová slova, která byla expertním systémem HDES vygenerována při zpracování daného požadavku.
- **id_pravidla** - jedná se o ID pravidla, které bylo na základě daného požadavku vygenerováno.

3.5 Zhodnocení

Vytvořený expertní systém HDES je v době odevzdání diplomové práce ve stavu předběžného testování a plnění báze znalostí dalšími pravidly. Z každého testování vzejde nová potřeba na úpravu pravidla či nějaké funkce systému. Tvorba takového expertního systému je na delší dobu a stav systému popisovaný v této diplomové práci je jakousi jeho první iterací.

V analytické sekci praktické části této diplomové práce byly definovány funkční a nefunkční požadavky. Tyto požadavky byly splněny. Expertní systém pomocí extrakce klíčových slov a sady pravidel dokáže kategorizovat požadavek podle toho, jakého modulu, tabulky či jiného tématu se požadavek týká. Systém dále uživateli zobrazuje seznam relevantních požadavků, čímž napomáhá k nalezení možného duplicitního požadavku. Uživateli je dále umožněno spravovat bázi znalostí - tedy jednotlivá pravidla, se kterými expertní systém HDES pracuje. Tímto jsou splněny požadavky funkční.

Nefunkční požadavky deklarují, aby systém byl dostupný jako webová aplikace. Toho je docíleno jeho provozem na webovém serveru s připojením k vlastní MySQL databázi. Jako webové technologie jsou dále použity skriptovací jazyk PHP, značkovací jazyk HTML a kaskádové styly CSS. Další dva nefunkční požadavky definují způsob přístupu uživatele k expertnímu systému. Je tedy vypracováno grafické uživatelské prostředí s klasickým formulářovým vstupem. Dále je uživateli dostupné ovládání systému pomocí HDES API, které umožňuje pomocí HTTP požadavků odeslat data a získat je zpracovaná zpět. Posledním nefunkčním požadavkem je, aby byl systém univerzálně použitelný. Přestože je jeho grafické prostředí zaměřeno přímo na práci s helpdeskovými požadavky, struktura báze znalostí a inferenční mechanismus na něm nejsou závislé. Pokud by tedy byla báze znalostí expertního systému naplněna jinou sadou pravidel popisující jinou problematiku, systém by stále byl schopen usuzovat. Jeho grafické prostředí a vysvětlovací modul by musely být upraveny dle dané problematiky.

Expertní systém HDES, který je v současnosti naplněn cca 2100 pravidly, dokáže v nadpisu a popisu požadavku detekovat klíčová slova, díky kterým požadavek kategorizuje a vygeneruje seznam relevantních požadavků. Díky tomu, že je expertní systém umístěn na webovém serveru, je rychlost jeho zpracování úměrná dostupným prostředkům daného serveru. Systém byl testován na soukromém virtuálním serveru, který měl přidělenou 1 GB RAM a 1 virtuální CPU. V porovnání s provozem na lokální stanici, kde probíhal vývoj, byl provoz expertního systému HDES na virtuálním serveru podstatně rychlejší. Proto lze doporučit provoz na dedikovaném stroji (ať už virtuálním nebo fyzickém).

3.5.1 Ekonomické zhodnocení

Jelikož je celý expertní systém HDES postaven jen na otevřených technologiích, nejsou pro jeho provoz potřeba žádné licenční produkty a jiné placené služby. Systém využívá open source technologie jak pro aplikační, tak pro databázovou vrstvu. Pořizovací cena je tedy nulová, pouze je potřeba vzít v potaz čas potřebný na správnou konfiguraci systému, na kterém expertní systém poběží.

Výhodou navrhovaného řešení je i využití webových technologií, tedy vytvoření webové aplikace. Tato aplikace je dostupná ze všech zařízení, které mají přístup k internetu (což je dnes většina zařízení). To ovšem není podmínkou, stačí mít expertní systém nainstalovaný na interním webovém serveru, který je přístupný pouze z firemního intranetu. K webové aplikaci se přistupuje vzdáleně a není tedy potřeba lokální instalace nějakého softwaru. V případě nějakého problému s aplikací (např. nalezení nějaké chyby) je tento problém řešen na jednom místě a není potřeba změnu nasazovat na všechny instalace aplikace, jako by tomu bylo u klasického softwaru. To samé platí pro aktualizace systému HDES. Pokud dojde k nějaké změně, je tato změna provedena pouze ve webové aplikaci. Na prostředky koncového uživatele to nemá vliv.

Nyní k samotnému přínosu expertního systému HDES. Jeho ekonomické přínosy tkví především v úspoře času v procesu analýzy nového požadavku a případné detekci duplicitního požadavku - je tedy eliminována potřeba řešit požadavek znovu a jsou tím ušetřeny nejen časové prostředky, které by pracovník mohl investovat do dalších důležitých pracovních činností. Potenciálně se také jedná o pozitivní krok pro zákazníka, protože řešitel má k dispozici seznam podobných požadavků a může snadno identifikovat řešení a díky tomu požadavek zákazníka vyřešit rychleji. Je tím tedy snížena celková doba řešení požadavku. Pokud je zákazník spokojen se zákaznickou podporou, je utvrzována jeho loajalita a je velká pravděpodobnost, že se znovu na firmu obrátí se zájmem o nový produkt či funkci.

Expertní systém HDES je možné implementovat i do jiných problémových domén, což přináší možnost jeho snadné komercionalizace. Úpravy uživatelského prostředí a vysvětlovacího mechanismu pro potřeby dané domény jsou jednoduše proveditelné.

Závěr

Cílem, definovaným v úvodu této diplomové práce, bylo seznámit čtenáře s teorií expertních systémů, současnými trendy v oboru zákaznické podpory a spojením obou těchto oborů ve formě tvorby expertního systému pro podporu řešení helpdeskových požadavků.

První část této práce se zabývala teorií obecných expertních systémů. Byla popsána jejich stručná historie, jejich obvyklé součásti a teoretický popis těchto součástí. Pozornost byla věnována různým formám reprezentace znalostí v bázi znalostí expertního systému. Čtenář byl dále seznámen s tím, jak pracuje inferenční mechanismus tvořící jakési jádro expertního systému. Nechybí ani stručné seznámení se současnými trendy v oboru expertních systémů.

Druhá část diplomové práce popisovala teorii zákaznické podpory a její současné trendy, mezi něž patří nové technologie ve formě inteligentních chatbotů či sociálních sítí. Spolu s využitím metod umělé inteligence v oboru zákaznické podpory je v práci dále zmíněna metodika ITIL a její relevantní součásti.

Třetí, praktická část, se zabývala tvorbou expertního systému pojmenovaného HDES. Cílem tohoto expertního systému bylo usnadnit práci řešiteli helpdeskového požadavku, a to pomocí jeho kategorizace a zobrazení požadavků jemu podobných. Řešitel je s pomocí systému schopen identifikovat případný duplicitní požadavek a ušetřit tak čas zbytečně věnovaný již vyřešenému požadavku. Tato třetí část diplomové práce je rozdělena na jednotlivé části procesu vývoje expertního systému HDES. V první, analytické části, byla provedena analýza současného stavu a požadavků na vytvářený systém. V druhé, návrhové části, bylo stručně charakterizováno, jakým způsobem by mohly základní části systému fungovat. V třetí, implementační části, bylo rozepsáno, jakým způsobem byl expertní systém implementován a jak jsou technologicky řešeny jeho jednotlivé části.

Výsledkem této diplomové práce je funkční expertní systém HDES, který pracuje s daty helpdeskového požadavku. Z těchto dat extrahuje klíčová slova, na základě nichž je požadavek kategorizován a porovnán s ostatními požadavky v dané kategorii, které byly již v minulosti systémem zpracovány. Díky použití pravidel, jako způsobu reprezentace znalostí, je systém snadno rozšiřitelný o další znalosti, a může tak být zapojen i do jiných procesů. Protože je

systém implementován jako webová aplikace, je jeho výpočetní logika přesunuta na stranu serveru, takže není ovlivněno zařízení, ze kterého je k expertnímu systému přistupováno.

Přínosem této diplomové práce je především úspora času a urychlení procesu analýzy nově příchozího helpdeskového požadavku. Úspora času tkví ve snadnější detekci duplicitních požadavků. Jelikož je expertní systém vytvořen v otevřených technologiích, není potřeba na jeho provoz vynakládat prostředky tak, jako je tomu u produktů zatížených nesvobodnými licencemi. Za technologický přínos lze považovat popis způsobu uložení pravidel báze znalostí v databázi, které bylo vytvářeno s důrazem na jeho budoucí rozšířitelnost a komplexitu.

V diplomové práci je navržen způsob, jakým by mohl být expertní systém HDES zapojen do současného workflow řešení helpdeskových požadavků, a to konkrétně implementace do současného softwarového řešení. Expertní systém HDES zatím pracuje se znalostmi týkajícími se konkrétní divize firmy, pro kterou je vytvářen. Do budoucna by ale mohl být upraven tak, aby byl schopen být nápomocen i jiným divizím, či dokonce našel využití v jiných aktivitách firmy. Značná část zákazníků podávajících helpdeskové požadavky přikládá i logy popisující danou situaci či nějaký snímek obrazovky, na kterém je případný problém ilustrován či zaznamenán. Systém by mohl být rozšířen i o nějakou formu text miningu zpracovávající přiložené logy nebo schopnost detekovat ze snímku obrazovky aktuálně přihlášeného uživatele a spuštěný modul informačního systému, kterého se popisovaný požadavek týká. Zajímavým rozšířením by byla aplikace samoučících metod ve formě neuronových sítí a dalších technologií umělé inteligence.

Seznam použité literatury

Citace

ALOR-HERNÁNDEZ, Giner a VALENCIA-GARCÍA, Rafael. 2017. *Current trends on knowledge-based systems*. Springer. Dostupné z DOI: 10.1007/978-3-319-51905-0.

BAUCOM, Jay. 2018. *The Top 7 Global Service Desk Trends for 2018* [online] [cit. 2019-01-25]. Dostupné z: https://www.slideshare.net/Alphanumeric_IT/the-top-7-global-service-desk-trends-for-2018.

BAYSAL, Olga a GODFREY, Michael W a COHEN, Robin. 2009. A bug you like: A framework for automated assignment of bugs. In: *Program Comprehension, 2009. ICPC'09. IEEE 17th International Conference on*, s. 297–298. Dostupné z DOI: 10.1109/ICPC.2009.5090066.

BOTHA, Rupert a LEONARD, Awie. 2012. Organizational issues and its impact. In: *Technology Management for Emerging Technologies (PICMET), 2012 Proceedings of PICMET'12:* s. 3131–3139. ISBN 978-1-890843-25-0.

BUYYA, Rajkumar a CALHEIROS, Rodrigo N a DASTJERDI, Amir Vahid. 2016. *Big data: principles and paradigms*. Morgan Kaufmann. ISBN 9780128093467.

CAO, JQ a ZHANG, SH. 2016. ITIL Incident Management Process Reengineering in Industry 4.0 Environments. In: *Proceedings of the 2nd International Conference on Advances in Mechanical Engineering and Industrial Informatics (AMEII 2016)*. Sv. 73, s. 1011–1016.

CARTLIDGE, Alison a RUDD, Colin a SMITH, Marco a WIGZEL, Paul a RANCE, Stuart a SHAW, Sue a WRIGHT, Theresa. 2012. *An Introductory Overview of ITIL® 2011* [online]. TSO [cit. 2019-01-29]. Dostupné z: https://itsmf.cz/wp-content/uploads/2017/08/itSMF_An_Introductory_Overview_of_ITIL_V3_2011.pdf.

COLLINASZY, Juraj a BUNDZEL, Marel a ZOLOTOVA, Iveta. 2017. Implementation of intelligent software using IBM Watson and Bluemix. *Acta Electrotechnica et Informatica*. **17**(1), 58–63. Dostupné z DOI: 10.15546/aeei-2017-00.

COLMERAUER, Alain a ROUSSEL, Philippe. 1996. The birth of Prolog. In: *History of programming languages—II*, s. 331–367. Dostupné z DOI: 10.1145/234286.1057820.

COMPARECAMP.COM. 2015. *History of Help Desk Software: Evolution and Future Trends* [online] [cit. 2019-01-23]. Dostupné z: http://comparecamp.com/history-of-help-desk-software-evolution-and-future-trends/.

DARLINGTON, Keith. 2013. Aspects of intelligent systems explanation. *Universal Journal of Control and Automation*. **1**(2), 40–51. Dostupné také z: http://www.hrpub.org/download/201309/ujca.2013.010204.pdf.

DVOŘÁK, Jiří. 2004. *Expertní systémy* [online] [cit. 2018-11-20]. Dostupné z: http://www.uai.fme.vutbr.cz/~jdvorak/Opory/ExpertniSystemy.pdf.

EDWARDS, Jamie. 2016. *The History of Customer Service: Ticket Troubleshooting to Proactive and Personal* [online] [cit. 2019-01-23]. Dostupné z: https://www.kayako.com/blog/history-of-customer-service/.

FEIGENBAUM, Edward A. 1981. Expert Systems in the 1980s. *State of the art report on machine intelligence. Maidenhead: Pergamon-Infotech* [online] [cit. 2019-01-22]. Dostupné z: https://stacks.stanford.edu/file/druid:vf069sz9374/vf069sz9374.pdf.

FEIGENBAUM, Edward A. 1992. *Expert Systems: Principles and Practice* [online] [cit. 2019-01-22]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.34.9207.

FRIEDMAN-HILL, Ernest. 2013. *Jess, the Rule Engine for the Java Platform* [online] [cit. 2019-01-22]. Dostupné z: https://www.jessrules.com/.

FUJITSU SERVICES. 2016. *The White Book Of: The Next-Generation Service Desk* [online] [cit. 2019-04-11]. Dostupné z: https://www.fujitsu.com/uk/Images/White-Book-next-generation-service-desk.pdf.

GIARRATANO, Joseph D. a RILEY, Gary D. 2004. *Expert Systems: Principles and Programming, Fourth Edition*. 4th edition. TBS. ISBN 978-81-315-0167-2.

HANNA, Ashley a RANCE, Stuart. 2011. *ITIL® Glossary of English Terms* [online]. 2. vyd. AXELOS Limited [cit. 2019-01-24]. Dostupné z: https://www.axelos.com/corporate/media/files/glossaries/itil_2011_glossary_gb-v1-0.pdf.

HDI. *About HDI* [online] [cit. 2019-01-30]. Dostupné z: https://www.thinkhdi.com/about.aspx.

HERTVIK, Joe. 2016. *IT Support Levels* [online] [cit. 2019-01-27]. Dostupné z: https://www.bmc.com/blogs/support-levels-level-1-level-2-level-3/.

IBM CLOUD. 2018. *IBM Watson Assistant* [online] [cit. 2019-01-26]. Dostupné z: https://console.bluemix.net/docs/services/assistant/index.html#about.

INFORMATION SCIENCES INSTITUTE. 2007. *Loom* [online] [cit. 2019-01-22]. Dostupné z: https://www.isi.edu/isd/LOOM/.

KEMPTER, Stefan a KEMPTER, Andrea. 2006. *IT Process Wiki - the ITIL® Wiki* [online] [cit. 2019-01-28]. Dostupné z: https://wiki.en.it-processmaps.com/index.php/Main_Page.

KLOECKNER, Kristof a DAVIS, John a FULLER, Nicholas C. a LANFRANCHI, Giovanni a PAPPE, Stefan a PARADKAR, Amit a SHWARTZ, Larisa a SURENDRA, Maheswaran a WIESMANN, Dorothea. 2018. Conversational IT Service Management. In: KLOECKNER, Kristof a DAVIS, John a FULLER, Nicholas C. a LANFRANCHI, Giovanni a PAPPE, Stefan a PARADKAR, Amit a SHWARTZ, Larisa a SURENDRA, Maheswaran a WIESMANN, Dorothea (ed.). *Transforming the IT Services Lifecycle with AI Technologies*, s. 75–93. Dostupné z DOI: 10.1007/978-3-319-94048-9_5.

LAM, Chris a HANNAH, Mark A. 2017. The social help desk. *Communication Design Quarterly Review*. **4**(2), 37–51. ISSN 21661200. Dostupné z DOI: 10.1145/3068698.3068702.

MANAGEMENTMANIA. *Help Desk* [online] [cit. 2019-01-24]. Dostupné z: https://managementmania.com/cs/help-desk.

MANN, Stephen. 2015. *IT Help Desk vs. Service Desk vs. ITSM - What's Different* [online] [cit. 2019-01-24]. Dostupné z: https://www.atlassian.com/it-unplugged/itsm/help-desk-vs-service-desk-vs-itsm.

MAŘÍK, V. a ŠTĚPÁNKOVÁ, O. a LAŽANSKÝ, Jiří. 1993. *Umělá inteligence (1)*. Praha: Academia. ISBN 978-80-200-0496-3.

MOORE, Johanna D a PARIS, Cécile L. 1991. Requirements for an expert system explanation facility. *Computational Intelligence*. **7**(4), 367–370.

NOGUERA-ARNALDOS, José Ángel a PAREDES-VALVERDE, Mario Andrés a SALAS-ZÁRATE, María Pilar a RODRÍGUEZ-GARCÍA, Miguel Ángel a VALENCIA-GARCÍA, Rafael a OCHOA, José Luis. 2017. im4Things: An Ontology-Based Natural Language Interface for Controlling Devices in the Internet of Things. In: *Current Trends on Knowledge-Based Systems*. Ed. ALOR-HERNÁNDEZ, Giner a VALENCIA-GARCÍA, Rafael. Cham: Springer International Publishing, s. 3–22. ISBN 978-3-319-51905-0. Dostupné z DOI: 10.1007/978-3-319-51905-0_1.

OBJECT MANAGEMENT GROUP. 2011. *Business Process Model And Notation, Version 2.0* [online] [cit. 2019-03-14]. Dostupné z: https://www.omg.org/spec/BPMN/2.0/PDF.

ORTBACH, Kevin a GAß, Oliver a KÖFFER, Sebastian a SCHACHT, Silvia a WALTER, Nicolai a MAEDCHE, Alexander a NIEHAVES, Bjoern. 2014. Design Principles for a Social Question and Answers Site: Enabling User-to-User Support in Organizations. In: TREMBLAY, Monica Chiarini a VANDERMEER, Debra a ROTHENBERGER, Marcus a GUPTA, Ashish a YOON, Victoria (eds.). *Advancing the Impact of Design Science: Moving from Theory to Practice*. Springer International Publishing, s. 54–68. Lecture Notes in Computer Science. ISBN 978-3-319-06701-8.

POKORNÝ, Miroslav a KRIŠOVÁ, Zdeňka. 2016. *Znalostní systém* [online]. Olomouc [cit. 2018-11-27]. ISBN 978-80-7455-065-2. Dostupné z: https://mvso.cz/wp-content/uploads/2017/10/Znalostn%c3%ad-syst%c3%a9my.pdf.

RUBIO, Juan E a ALCARAZ, Cristina a LOPEZ, Javier. 2017. Recommender system for privacy-preserving solutions in smart metering. *Pervasive and Mobile Computing*. **41**, 205–218. Dostupné z DOI: 10.1016/j.pmcj.2017.03.008.

RUSSELL, Stuart J. a NORVIG, Peter. 1995. *Artificial Intelligence: A Modern Approach*. Prentice Hall. ISBN 0-13-1038052.

SCOTT, Rob. 2018. *Contact Centre Trends 2019* [online] [cit. 2019-01-24]. Dostupné z: https://www.uctoday.com/contact-centre/contact-centre-trends-2019/.

SIMONS, George. 2015. *11 Innovations Your Call Center Needs Right Now* [online] [cit. 2019-01-25]. Dostupné z: https://www.gcsagents.com/blog/2015/11/11-innovations-your-call-center-needs-right-now.

SMALL, Karen. 2013. *Help Desk vs. Service Desk: A Brief History* [online] [cit. 2019-01-23]. Dostupné z: https://blog.samanage.com/it-service-management/help-desk-vs-service-desk-a-brief-history/.

STEELS, Luc. 1978. *Frame-Based Knowledge Representation* [online] [cit. 2018-11-22]. Dostupné z: http://dspace.mit.edu/handle/1721.1/41133. Working Paper. MIT Artificial Intelligence Laboratory.

TRIPATHI, KP. 2011. A review on knowledge-based expert system: Concept and architecture. *IJCA Special Issue on Artificial Intelligence Techniques-Novel Approaches & Practical Applications*. **4**, 19–23.

TURING, A. M. 1950. Computing Machinery and Intelligence. *Mind*. **59**(236), 433–460. ISSN 0026-4423, 1460-2113. ISSN 0026-4423, 1460-2113. Dostupné z DOI: 10.1093/mind/LIX.236.433.

VERHODUBS, Olegs a GRUNDSPENKIS, Janis. 2011. Towards the semantic web expert system. *Scientific Journal of Riga Technical University. Computer Sciences*. **44**(1), 116–123.

WAGNER, William P. 2017. Trends in expert system development: A longitudinal content analysis of over thirty years of expert system case studies. *Expert Systems with Applications*. **76**, 85–96. ISBN 978-80-247-4153-6. ISSN 09574174. Dostupné z DOI: 10.1016/j.eswa.2017.01.028.

WALEK, Bogdan a SPACKOVA, Petra. 2018. Content-Based Recommender System for Online Stores Using Expert System. In: *2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, s. 164–165. Dostupné z DOI: 10.1109/AIKE.2018.00036.

WEERAWARNA, N.T. a HATHTHELLA, H.M.H.R.B. a AMBADENIYA, A.R.G.K.B.R. a CHANDRASIRI, L.H.S.S. a BANDARA, M.S.L. a THELIJJAGODA, S.S. 2011. CyberMate Artificial Intelligent business help desk assistant with instance messaging services. In: *Industrial and Information Systems (ICIIS), 2011 6th IEEE International Conference on*, s. 420–424. Dostupné z DOI: 10.1109/ICIINFS.2011.6038105.

WINDLEY, Phillip J. 2002. Delivering high availability services using a multi-tiered support model. *Windley's Technometria*. **16**, 1–9.

XU, Jian a HE, Rouying. 2018. Expert recommendation for trouble ticket routing. *Data & Knowledge Engineering*. **116**, 205–218. Dostupné z DOI: 10.1016/j.datak.2018.06.004.

ZENDESK.COM. *The History of Customer Support* [online] [cit. 2019-01-23]. Dostupné z: https://www.zendesk.com/resources/the-history-of-customer-support/.

Bibliografie

ARLOW, Jim a Ila NEUSTADT. 2007. *UML a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky.* 2. Brno: Computer Press. ISBN 978-80-251-1503-9.

BRUCKENER, Tomáš, Jiří VOŘÍŠEK, Alena BUCHALCEVOVÁ, Iva STANOVSKÁ, Dušan CHLAPEK a Václav ŘEPA. 2012. *Tvorba informačních systémů: principy, metodiky, architektury.* Praha: Grada Publishing, 357 s. Management v informační společnosti. ISBN 978-80-247-4153-6.