Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra systémového inženýrství

Znalosti a znalostní jednotky v podmínkách neurčitosti

Disertační práce

Autor: Ing. Michal Peták

Školitel: doc. Ing. Milan Houška, Ph.D.

Odborný konzultant: Ing. Martina Houšková Beránková, Ph.D.

Poděkování

Děkuji na tomto místě svému školiteli doc. Ing. Milanu Houškovi, Ph.D., a odborné konzultantce Ing. Martině Houškové Beránkové, Ph.D., za vedení při zpracování disertační práce. Dále děkuji kolegyním a kolegům z katedry systémového inženýrství za podporu a spolupráci během celého mého doktorského studia a zejména prof. RNDr. Heleně Brožové, CSc., za zpracování katedrového posudku, na jehož základě bylo možné disertační práci dále zkvalitnit.

Abstrakt

V současném světě narůstá potřeba rychlého řešení složitých rozhodovacích situací, které se neobejde bez hlubokých expertních znalostí. Stále větší důraz je kladen na rozvoj expertních systémů, které dokáží do určité míry zastoupit lidského experta a poskytovat podporu rozhodovateli. Využití expertních systémů je široké v různých vědních oborech i průmyslových aplikacích a jejich počet se stále zvyšuje. Znalosti doménových expertů lze v expertních systémech reprezentovat různě; přitom volba reprezentace může zásadně ovlivnit efektivnost nasazení expertního systému v praxi. Z těchto důvodů se práce zabývá znalostmi v expertních systémech.

V literární rešerši jsou vymezeny znalosti a jejich reprezentace spolu s interoperabilitou. Z přehledu vyplývají také způsoby uchovávání znalostí používané v současné době, mezi které patří rozhodovací stromy, produkční pravidla, která jsou modifikována na pravděpodobnostní pravidla, a zejména na fuzzy produkční pravidla.

Ve výzkumné části jsou řešeny modifikace metod dopředného a zpětného řetězení vybraných elementů znalostních jednotek. Problematika řešení neurčitosti vyústila v řešení a definici fuzzy znalostních jednotek, které umožňují s neurčitostí pracovat. Byly vytvořeny modelové situace v prostředí MATLAB, konkrétně v Simulink a Fuzzy Logic Toolbox a byl potvrzen předpoklad klasické inference, dopředné a zpětné řetězení se znalostní jednotkou. Celá simulace je koncipovaná jako základní model, na kterém byl vyvinut postup inference s fuzzy znalostními jednotkami. Tento postup byl aplikován na případovou studii, ve které byly simulovány scénáře a výsledek simulace byl porovnán s názory expertů.

Klíčová slova

Znalostní jednotka, inferenční mechanismus, fuzzy znalostní jednotka, simulace, grafická reprezentace znalostní jednotky, model fuzzy znalostních jednotek.

Abstract

In the real world there is an increasing need for rapid solutions to assist in complex decision-making during situations that cannot be dealt with without expert knowledge. More emphasis is placed on the development of expert systems, which can substitute a human expert and provide support to decision-makers. Their use has become widespread in multiple scientific disciplines and industrial fields and their number and capability are increasing. The knowledge of relevant experts can be represented differently in expert systems, but the choice of representation can have a fundamental influence on the effectiveness of the system in practice. Therefore, this thesis deals with knowledge in expert systems.

In the literature review, expert knowledge is defined and its representation and interoperability are presented there. This is followed by delineating the current methods of knowledge preservation, such as decision trees, production rules modified to probability rules, and especially fuzzy production rules.

Modifications of the classical inference, such as forward and backward chaining methods of knowledge unit elements, are dealt with in the research portion. Resolving the aforementioned uncertainty issues resulted in the fuzzy knowledge unit definition which allows work to proceed despite uncertainty. Model situations were created in the MATLAB, specifically in the Simulink and Fuzzy Logic Toolbox add-ins. The assumption of classical inference, the forward and backward chaining of the knowledge unit, was confirmed. The entire simulation is built as a basic model, where the inference procedure of fuzzy knowledge units is demonstrated. This procedure is applied in a case study, where different scenarios were simulated and the results were discussed with knowledge domain experts.

Key words

Knowledge unit, inference mechanism, fuzzy knowledge unit, simulation, graphical representation of the knowledge unit, model of the fuzzy knowledge units.

Obsah

1 Úvod

Základními důvody pro vybudování a rozvoj expertních systémů je předpoklad, že informace a znalosti distribuované jejich prostřednictvím mají přidanou hodnotu pro jejich uživatele. Společně se zvyšující se poptávkou po znalostech se také vyvíjí softwarové a výpočetní programy pro aplikace těchto systémů. Expertní systém obecně obsahuje vědomosti určité oblasti vědění - aplikační domény. Výhodou expertních systémů je získání informací a znalostí od lidských expertů a jejich uložení do databází expertního systému. To má za význam jak uchování znalostí z různých oborů, tak zároveň slouží pro dynamický a uživatelsky přívětivý vizuální výstup, který již nemá nevýhody lidského experta a je neustále dostupný. Jedním z dalších výstupů, kterým expertní systém disponuje, je zobrazení důvodů, resp. podkladů, ze kterých se rozhodnutí předkládané expertním systémem generuje. Přesnost odpovědí expertních systémů je značně závislá na zvoleném postupu při vyrovnávání se s neurčitostí (řešení rozhodovacích situací). Mezi postupy, jakými lze řešit rozhodovací situace v expertních systémech, patří pravděpodobnostní (tzv. Bayesovské) metody, hybridní metody a metody využívající fuzzy logiky. Speciálním typem expertních systémů jsou tedy fuzzy expertní systémy (FES), které mají interdisciplinární charakter s úzkou vazbou na znalostní inženýrství a informační technologie. Slovy autorů Nilashi et al. (2015):

„Fuzzy expertní systém kombinuje schopnost expertního systému simulovat rozhodovací proces s nejistotou typickou pro lidské uvažování, která je přítomna ve fuzzy logice.[1]*“*

Při práci se znalostmi je třeba, aby byly nějakou vhodnou formou uchopeny a reprezentovány. Pro automatické zpracování expertním systémem jsou z důvodu přehlednosti a uchopitelnosti používána produkční pravidla, rozhodovací stromy či myšlenkové mapy. V originální části této práce je rozpracována myšlenka použití znalostních jednotek v expertních systémech, jelikož z pohledu architektury expertních systémů může být výhodné zakomponovat znalostní jednotku jako stavební kámen expertního systému. Znalostní jednotka může zpřehlednit architekturu celého řešení a také být uvedena v textové podobě, jak uvádí Rauchová a Houška (2013), což má význam při

[1] „*The Fuzzy Expert System combines the ability of an expert system to simulate the decision-making process with the uncertainty typical of human reasoning, which is present in fuzzy logic“.*

interpretaci výsledků. Podle Bertalanffy (2006) je v každém systému důležitý princip ekvifinality, který říká že *„V systému existuje řada rozdílných cest vedoucích k dosažení požadovaného stavu“.* V podstatě nezáleží na způsobu dosažení, ale na výsledku. Přehlednost řešení problému, kterému může znalostní jednotka prospět a jeho zdůvodnění rozhodnutí jsou klíčovými prvky expertních systémů (Kendal a Creen, 2007).

Podstatou originální části disertační práce bude navrhnout způsob implementace znalostních jednotek jako specifické formy reprezentace procedurálních znalostí do prostředí expertních systémů, a to včetně operací se znalostními jednotkami a inferenčních metod. To znamená analyzovat aktuální postupy inferenčních mechanismů a reprezentací znalostí v expertních systémech a vytvořit tak potenciál pro výzkum v oblasti reprezentací procedurálních znalostí a operací s těmito znalostmi v prostředí expertních systémů.

V současné době vznikají studie jako například celoplošné předpovědi povodní (Zhang et al., 2018), v nichž jako jeden z benefitů autoři zmiňují přehlednost řešení, oddělení znalostní báze a infernenčního mechanismu a snadnou aktualizaci znalostní báze. Zároveň upozorňují na obtížnost modernizace znalostní databáze jejich řešení. Zmíněná obtížnost se skrývá v každé změně databáze, například vložení dalších znalostních entit, například informaci „o nové průtokové kapacitě“, vyžaduje úpravu implicitního operačního kódu v syntaxi PyKE.

V případě navrhovaného simulačního modelu fuzzy znalostních jednotek se jedná o vytvoření fuzzy znalostní jednotky a jejího zakomponování do řešené problémové situace pomocí již předdefinovaných objektů a případné ladění v grafickém režimu v kontextu celého problému. Rozdíl oproti systému pro celoplošné předpovědi povodní (Zhang et al., 2018) a nutností úpravy implicitního kódu v syntaxi PyKE je v tom, že je vytvořena fuzzy znalostní jednotka například v objektu toolboxu, která je zakomponována do řešení. Změna databáze tedy neprobíhá v kódu, ale pomocí fuzzy znalostní jednotky.

Mezi aktuálně používané způsoby uchovávání a reprezentaci znalostí patří rozhodovací stromy (Kumar et al., 2016), produkční pravidla, která jsou modifikována na pravděpodobnostní pravidla, Bayesovský přístup (Moreno a Espejo, 2015) a zejména na fuzzy produkční pravidla (Venturelli et al., 2017; Moreno a Espejo, 2015). Uvedené postupy a reprezentace znalostí jsou dány do kontextu expertních systémů vybraných autorů (Calderwood et al., 2017; Venturelli et al., 2017; Moreno a Espejo, 2015; Chen a Pollino, 2012). Zejména je kladen důraz na způsoby konstrukce a reprezentace znalostí v expertních

systémech. Podrobněji jsou popisovány metody fuzzy logiky, z nichž následně čerpá originální část této práce.

Záměr vlastní výzkumné práce, originální části disertační práce bude navázán na výsledky výzkumů pracovníků z katedry systémového inženýrství ČZU PEF na téma reprezentace znalostí a zejména znalostních jednotek. Autoři, kteří se věnují znalostním jednotkám jsou např. Brožová a Houška, 2011; Rauchová a Houška, 2013; Houška a Beránková, 2013. Znalostní jednotka tvoří základní východisko, ze kterého se odvozují a definují nové vlastnosti a pravidla v této práci. Nutným předpokladem je definice - navržení postupu inference znalostních jednotek.

Výhodou inference znalostních jednotek neboli dopředného a zpětného řetězení je formalizace inferenční sítě a znalostí jako takových. Inference znalostních jednotek je postavena na produkčních pravidlech, a proto, jak tvrdí Oliinyk et al. (2017), jsou vhodným nástrojem k popisu cesty k rozhodnutí. K dosažení výsledků inference jsou tak na rozdíl od klasických pravidel (Mařík et al., 2004) využívány znalostní jednotky a inference znalostních jednotek. Znalostní jednotky se využívají také k popisu dosažených výsledků.

K řešení neurčitosti nastalého jevu v inferenci a řešení rozhodovacích situací při inferenci se znalostními jednotkami je použita fuzzy logika. Je vybrána z toho důvodu, že umožňuje modelování jevu jako takového a stupně proveditelnosti tohoto jevu jsou intuitivní pro uživatele na rozdíl od pravděpodobnosti, která je závislá na provedeném pokusu a na tom, jestli definovaná situace nastane nebo ne. Dempster Shaferova teorie umožňuje kombinovat názory dvou funkcí domnění, domnění dvou expertů a modelovat situace i s částí nevědomosti, neurčitosti. Přesto se, obdobně jako v pravděpodobnosti, jedná o ohodnocení nastalého jevu jeho možností a nikoli o modelování jevu samotného.

Očekávaným výsledkem práce je definování fuzzy znalostní jednotky. U jednotlivých elementů znalostních jednotek jsou analyzovány možnosti jejich fuzzifikace a navrženy vhodné elementy k fuzzifikaci. Navržené elementy jsou fuzzifikovány pomocí lingvistických-jazykových proměnných. Lingvistická proměnná a fuzzy množina s funkcemi příslušnosti je promítnuta do elementu znalostní jednotky a tímto způsobem fuzzifikuje znalostní jednotku. Fuzzifikovaný element je poté začleněn do znalostní jednotky a vzniká fuzzy znalostní jednotka.

Vyústěním práce je návrh simulace rozhodovacího procesu s fuzzy znalostními jednotkami. Tato simulace bude obsahovat kroky a činnosti, které je potřeba provést obecně a také konkrétně pro simulace v prostředí MATLAB Simulink. Navržená simulace bude obsahovat nové a již definované možnosti formalizace fuzzy znalostních jednotek. V prostředí MATLAB Simulink budou také navrženy způsoby a doporučeny prostředky pro analýzu a ladění jednotlivých komponent simulace. Ladění simulace a její úpravy budou prováděny na úrovni jednotlivých komponent - uzlů fuzzy znalostních jednotek, nebo celého modelu. K obecnému popisu bude vytvořena konkrétní případová studie pro ilustraci funkce simulace a architektury řešení problémové situace.

Ve výsledku případové studie budou formalizovány odpovědi simulace do podoby fuzzy znalostní jednotky. Jedná se o vyhodnocení a doporučení výsledků simulace. Simulace modelu fuzzy znalostních jednotek probíhá s nástroji, které jsou jednak standardem MATLAB Simulink (MathWorks online Documentation, 2018), ale také s nástroji speciálně vyvinutými a nebo upravenými v tomto prostředí. Některé expertní systémy mají tu výhodu, že jejich výstupy lze graficky vykreslit pomocí vrstvy v GIS aplikaci (např. Semeraro, et al., 2016, navrhli expertní systém pro hodnocení zranitelnosti středomořské chráněné oblasti vůči požáru). U každého expertního systému tento způsob využít nelze. Znalostní jednotky představují nejen přehledný způsob formalizace znalostí, ale umožňují i snadnější práci se znalostmi, jejich kombinaci, využívání apod.

2 Cíl práce

Cílem disertační práce je navrhnout způsob implementace znalostních jednotek jako specifické formy reprezentace procedurálních znalostí do prostředí znalostních systémů, způsob zobrazení neurčitosti ve znalostních jednotkách pomocí fuzzy znalostních jednotek, a to včetně operací se znalostními jednotkami a inferenčních metod.

Dílčí cíle:

- Zjistit možnosti modifikací klasické inference se znalostní jednotkou.
- Vymezit pojem fuzzy znalostní jednotka a vymezit jednotlivé typy fuzzy znalostních jednotek.
- Odvodit inferenci fuzzy respektive znalostních jednotek.
- Vytvořit modelové řešení ve vhodné softwarové aplikaci a potvrdit předpoklad klasické inference fuzzy respektive znalostních jednotek.

3 Metodika

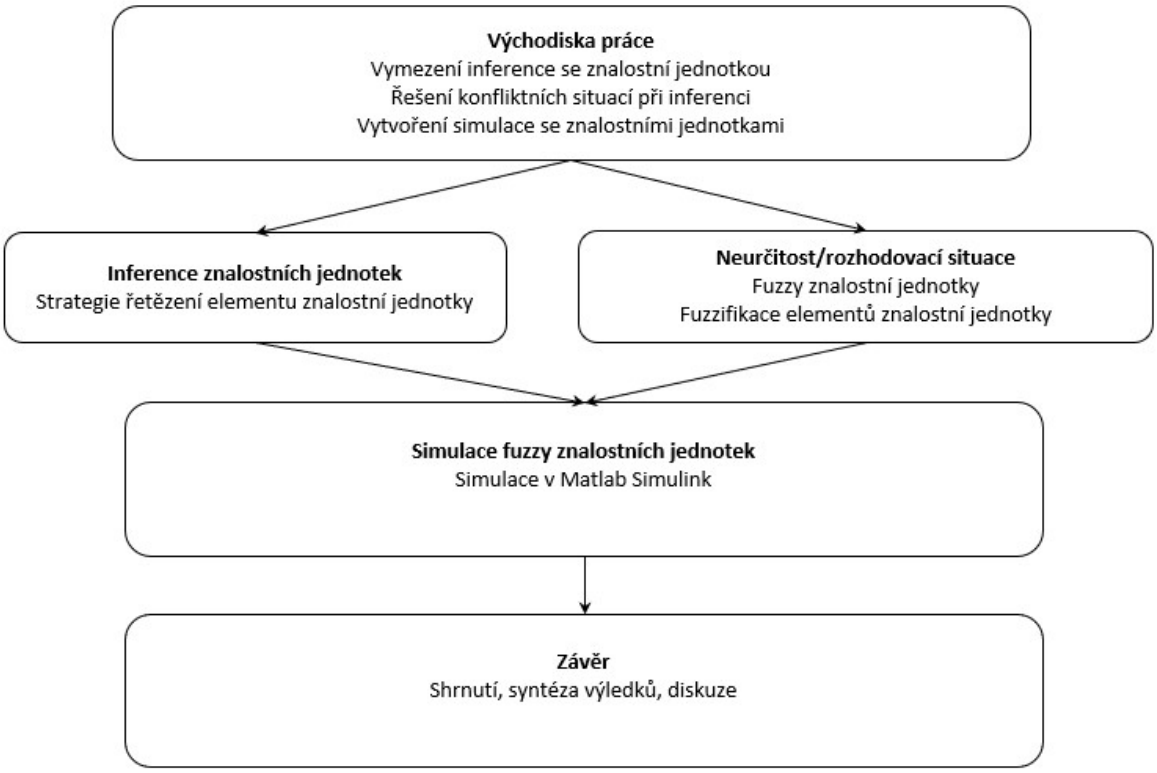
Cíle práce bude dosaženo pomocí následujícího metodického postupu:

1. Literární přehled

Literární přehled bude zaměřen na reprezentaci znalostí a postupy inferenčních mechanismů. Bude sumarizovat sekundární data získaná z literárních zdrojů, které se zabývají danou problematikou.

Přehled bude zaměřen na zjištění aktuálního stavu znalostí ve zkoumané oblasti reprezentací znalostí, inferenčních metod a expertních systémů. Konkrétně bude orientován na systémový pohled na reprezentace procedurálních znalostí a operace s těmito znalostmi v prostředí expertních systémů. Zaměření na využití znalostí v expertních systémech předpokládá snadnou uchopitelnost i pro automatizaci. Proto budou detailněji rozpracována témata produkčních pravidel a znalostních jednotek. Na produkční pravidla se práce bude zaměřovat z pohledu jejich modifikací při využití v expertních systémech na fuzzy znalostní jednotky a fuzzy produkční pravidla.

Schéma výzkumné práce a dílčí řešená témata jsou zachyceny na obrázku 1.



Obrázek 1 Schéma výzkumné práce a dílčí řešená témata; zdroj: autor

2. Modifikace klasické inference

Pro výzkum a zjištění potenciálu inference se znalostní jednotkou bude využita případová studie v podobě modelu ve vybrané softwarové aplikaci. V této aplikaci budou modelovány různé situace z případové studie s cílem zjištění, jestli je možné provádět klasickou inferenci, dopředné a zpětné řetězení se znalostní jednotkou a jakým způsobem. Bude vybírána vhodná metoda pro práci s neurčitostí při inferenci.

2.1 Zobrazení neurčitosti, fuzzy znalostní jednotka

Pro práci s neurčitostí bude rozpracována metoda, která vychází z fuzzy logiky, fuzzy Mamdani a lingvistických, jazykových proměnných. Z uvedených metod bude vytvořen postup fuzzifikace znalostních jednotek a definována fuzzy znalostní jednotka jako taková. Pro ověření předpokladů řešení neurčitosti fuzzy znalostních jednotek bude vytvořena případová studie a simulace. Ověření funkčnosti nově vytvořených modelů a algoritmů bude probíhat na modelových datech z vybrané znalostní domény.

2.2 Vytvoření simulace s fuzzy znalostními jednotkami

Cílem této simulace je ověření chování fuzzy znalostních jednotek při rozhodování za neurčitosti a při inferenci samotné. Aplikační doména simulace bude definována experty dané aplikační domény a znalostním inženýrem. Dále bude práce pokračovat vývojem nebo případnou modifikací kódu ve vhodném simulačním a modelovacím systému. Následným krokem bude otestování znalostní jednotky z hlediska použití při zpracování neurčitosti a řešení rozhodovacích situací jako fuzzy znalostní jednotky. V případové studii budou simulovány jednotlivé modelové situace a budou odvozovány odpovědi pro zadané situace – scénáře. Vyhodnocení a přínosy inference fuzzy znalostních jednotek oproti klasickým produkčním pravidlům budou demonstrovány na vytvořené simulaci.

3. Vyhodnocení a diskuse výsledků

Výsledky modelu vytvořeného v případové studii budou porovnány s výsledky expertů, které byly dosaženy jinými postupy. Dílčí výsledky, to znamená mezikroky při nastavování modelu a před jeho odpovědí, budou diskutovány s názory expertů na zvolenou aplikační doménu. V rámci případové studie bude podle odpovědí modelu navržena úprava postupů v tzv. Gap-Fit analýze. Je tedy očekáváno i praktické využití výsledků této práce.

Přínosy práce budou diskutovány s pracemi a výsledky jiných relevantních autorů. Těžiště diskuse bude spočívat ve shrnutí výhod postupů vytvořených v této práci oproti dosud zavedeným způsobům řešení. Základem budou znalostní jednotky a postupně budou prezentovány návrhy na úpravu stávajících metod pro práci s nimi nebo uplatnění znalostních jednotek v metodách, kde dosud používány nebyly.

V závěru disertační práce bude vyhodnoceno naplnění jejích cílů a jejího celkového kontextu.

4 Literární rešerše

Cílem literární rešerše je vytvořit teoretický podklad zaměřený na aktuální postupy inferenčních mechanismů a reprezentací znalostí (znalostní jednotky) v expertních systémech a vytvořit tak základ pro další výzkum v oblasti reprezentací procedurálních znalostí a operací s těmito znalostmi v prostředí expertních systémů. Pro dosažení výše uvedeného cíle bude literární rešerše zaměřena na aktuální stav znalostí ve zkoumané oblasti expertních systémů a inferenčních metod. Konkrétně bude orientována na systémový pohled na reprezentaci procedurálních znalostí a operace s těmito znalostmi v prostředí expertních systémů. Na konkrétních studiích autorů budou uvedeny postupy konstrukce expertních systémů z hlediska použitých inferenčních metod v různých vědních oborech. Dále bude uvedena studie, která srovnává inferenční metody při řešení problémové situace. Dalším logickým předpokladem bude přehled reprezentací znalostí. Tento přehled bude zaměřen na využití znalostí v expertních systémech. Snadná použitelnost znalosti je klíčová pro automatizaci. Proto budou detailněji rozpracována témata produkčních pravidel a znalostních jednotek. Na produkční pravidla se práce bude zaměřovat z pohledu jejich modifikací při využití v expertních systémech na fuzzy a Bayesovská produkční pravidla.

Literární rešerše je také zaměřena na identifikaci výzkumné příležitosti. Pro určení a vymezení výzkumné příležitosti budou využity aktuální zdroje vědeckých článků a trendů, kterými se vývoj expertních systémů ubírá. Pro identifikaci výzkumné příležitosti budou kritickým pohledem dané do souvislostí přístupy k reprezentaci znalostí a použité inferenční algoritmy v expertních systémech jednotlivých autorů. V oblasti inferenčních metod se bude práce věnovat také způsobům řešení neurčitostí rozhodovacích situací při inferenci.

4.1 Znalosti a jejich reprezentace

Úvodní kapitola definuje pojem znalost a předkládá možnosti zobrazení, zachycení a užívání znalostí. Vychází a je logicky uspořádána z pohledu na znalost bližšímu člověku k pohledu bližšímu strojovému zpracování. Znalost neexistuje jen sama o sobě. Je definována a vzniká vždy v kontextu dat, informací a za pomoci jiných znalostí. Vznik znalostí se váže k znalostnímu řetězci, který vysvětluje Truneček (2004) hierarchickým schématem Data - Informace - Znalosti. Znalost je na konci tohoto řetězce jako výsledek činnosti kombinace těchto prvků. Znalost je definována obecně podle Trunečka (2004) jako:

„Znalosti jsou možnosti účinného jednání."

Širší definici, zejména v informačních a komunikačních technologiích, udává Katuščák et al. (1998) jako:

"Schopnost člověka nebo jakéhokoli jiného inteligentního systému uchovávat, komunikovat a zpracovávat informace do systematicky a hierarchicky uspořádaných znalostních struktur. Znalost je charakterizována schopností abstrakce a generalizace dat a informací."

Z pohledu přenositelnosti znalostí vychází klasifikace autorů (Nonaka a Takeuchi, 1995). V této klasifikaci znalosti rozdělují na:

- Explicitní znalosti, které lze vyjádřit formálním jazykem, to znamená, že je můžeme napsat, nakreslit nebo jinak znázornit. Tato znalost je nejsnadněji, oproti ostatním, přenositelná a lze ji také ukládat a skladovat.
- Tacitní znalosti jsou vytvářeny interakcí explicitních znalostí a zkušeností, dovedností, intuice, představ, mentálních modelů. Tyto znalosti mají subjektivní charakter a jsou vázány k osobnosti člověka. Z hlediska přenositelnosti jsou velmi těžké pro přenos.
- Implicitní znalosti jsou tiché znalosti, které nelze formalizovat. Jsou vázány na podvědomí lidí. Někteří autoři zastávají názor, že při pokusu o přenos se taková znalost zničí. (Truneček, 2004)

Schéma uvedených znalostí v kontextu možnosti jejich předávání je podle Nonaka a Takeuchi (1995) na obrázku 2. Tito autoři dále uvádí čtyřfázový model procesu transformace znalostí SECI model (Socializace, Externalizace, Combination - kombinace, Internalizace),

který dává modelu dynamiku v podobě spirály učení mezi uvedenými prvky modelu transformace znalostí.

	Do	
Od	Tacitní	Explicitní
Tacitní	**Socializace**	**Externalizace**
Explicitní	**Internalizace**	**Kombinace**

Obrázek 2 Čtyři způsoby konverze znalostí; zdroj: Nonaka a Takeuchi, 1995

Znalosti lze členit na procedurální a deklarativní. K reprezentaci procedurálních a deklarativních znalostí se využívá více způsobů, do jisté míry také rozhoduje účel použití. Výchozí rozdíl procedurálních a deklarativních znalostí uvádí Pirttimaa et al. (2017). Rozdíl mezi procedurální = „knowing how“ a deklarativní = „knowing that“ znalostí je, že procesní poznání spočívá v tvrzení: “Že věděl jak” a deklarativní je: “Že věděl jakou formou”. Tato práce čerpá od autora Ryle (1949).

Znalosti současně s rozvojem informačních technologií a zavedením termínu umělá inteligence (Artificial Intelligence = AI) někteří autoři spojují s Alanem Turingem (od 1900 do 1956), který předložil návrhy blízké tomu, co bylo poté popsáno jako AI (del Águila et al., 2014). Buchanan a Duda (1983) v kapitole „*Representation of Knowledge*“ uvádějí reprezentaci znalosti pro využití v umělé inteligenci a expertních systémech. Tyto historické prameny uvádějí, že reprezentace znalosti je soubor konvencí pro popis světa a dávají ho do souvislosti s umělou inteligencí. Reprezentaci znalostí považují za slovník datových struktur a programů, který umožňuje popsat znalost z určité znalostní domény. Dále pak uvádí pravidlově orientované, rámcově orientované a logicky orientované systémy, z čehož se zaměřují na pravidlově orientované systémy a produkční pravidla. Také odkazují na souvislost s oblastí umělé inteligence (Groner et al., 1983; Barr et al., 1989; Brachman a Smith, 1980).

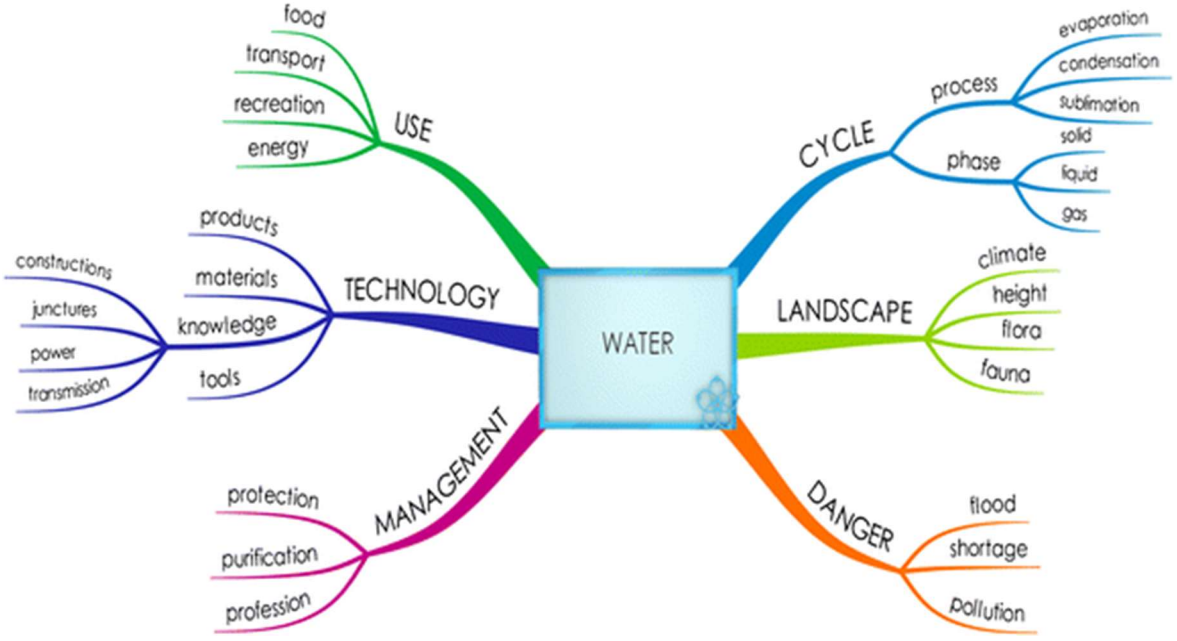
V některých publikacích je zmiňován pojem znalostních jednotek, nicméně není definován. Lytras a Naeve (2006) uvádí softwarové řešení pro učení zaměstnanců „*Dynamics Knowledge Network System (DKNS)*“ znalostní jednotku jako součást dynamického znalostního systému. Procedurální znalost reprezentuje grafem a hierarchickou strukturou

v systému. Znalostní jednotku včleňuje do tohoto řešení jako uzel v znalostní síti a navazuje na něj také odkazy, které vedou ke konkrétním příkladům k deklarativní znalosti. Zhang et al. (2014) uvádí, že není systémová a vědecká metoda pro čištění - úpravu vztahu mezi znalostními jednotkami respektive identifikaci vztahu mezi znalostními jednotkami. V této studii předkládá konkrétní příklad řešení, pomocí agregací uzlů grafu do znalostních jednotek. Čili tento postup se liší od autorů Lytras a Naeve (2006), který za znalostní jednotky považuje uzly grafu a vytváří hierarchii.

Jednou z možností popisu znalostní domény nabízí Gavrilov et al. (2015), kteří uvádějí postup při konstrukci domény projektového řízení včetně výběru ontologie pro popis domény v následujících krocích:

- Identifikace cíle, strategie a omezení
- Vytvoření slovníku domény a konceptu
- Specifikace řešení a kategorizace
- Čištění „*Orchestration*“ což znamená zpřehledňování, očišťování řešení

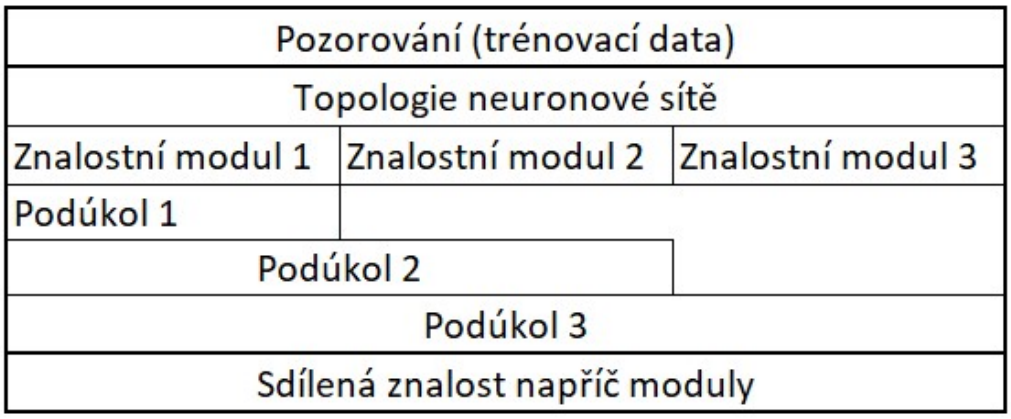
V tomto postupu se autoři zaměřují na reprezentaci znalostí o projektovém řízení a zejména na krok tzv. „*Orchestration*“ a to z pohledu způsobu popisu domény. V tomto příkladu prostřednictvím myšlenkových map navržených v 60. letech 20. století Tony Buzanem (Buzan a Buzan, 2011). Myšlenkové mapy mají volnou konstrukci, například Stokhof et al. (2018) využili myšlenkovou mapu pro zvýšení efektivity učení a upevňování znalostí (obrázek 3).



Obrázek 3 Myšlenková mapa "Voda"; zdroj: Stokhof et al., 2018

S ohledem na myšlenkové mapy a reprezentaci znalostí je vhodné zmínit, že není snadné porozumět cizí myšlenkové mapě a navíc nelze některá témata zobrazit hierarchicky (Buzan, 2007). Vzhledem k volnosti jejich tvorby nelze myšlenkové mapy považovat za znalostní jednotky, ale jak říká sám autor (Buzan, 2007) za *„nejlepší pomůcku pro přemýšlení"*, a to například pro tvorbu expertních systémů. Dále uvádí, že vizualizace konstrukce znalostí pomocí myšlenkové mapy pomáhá upřesnit strukturu znalostí.

V ohledu na nástroje myšlení a konstrukci expertních systémů uvádí Jakeman et al. (2006) doporučení „*Good practice*" deseti opakujících se kroků při tvorbě Bayesovské sítě, která je využívána expertním systémem. Tento postup deseti kroků také použili Chen a Pollino (2012) pro expertní systém na určování vhodných biotopů. Jeden z kroků tohoto postupu je řešení konceptuálního modelu, který je možné brát jako předstupeň k vytvoření znalostních jednotek pro expertní systémy. Další z možností reprezentace znalostí jsou neuronové sítě. Prostřednictvím neuronových sítí, jak uvádí Chandra et al. (2017), bylo již řešeno mnoho reálných problémových situací. Ve své práci Chandra et al. (2017) rozšiřuje neuronové sítě o víceúčelové učení prostřednictvím modulárních síťových topologií, ve které je uchována znalost (Evolutionary Multi-Task Learning). V navrhované metodě je každý úkol definován vybranou oblastí v topologii sítě (modulem). Tato technologie umožňuje uchovat znalosti v modulech. Na obrázku 4 je na obecné úrovni uvedeno schéma víceúčelového učení pro modulární reprezentaci znalostí.



Obrázek 4 Evoluční víceúčelové učení (Evolutionary Multi-Task Learning); zdroj: Chandra et al., 2017

Uvedené postupy v této kapitole uvádí některé možné reprezentace znalostí. Tyto postupy jsou poplatné účelu řešení. Vzhledem k tomu, že konečným uživatelem znalosti může být i expertní systém budou v této práci znalosti reprezentovány exaktními metodami, které budou popsány v dalších kapitolách.

4.1.1 **Produkční pravidla, řetězení**

Produkční pravidla jsou jedna z možností, jak zaznamenat informace o řešení problému a jejich prostřednictvím zachytit znalosti. V (Production rules, 2012), je uvedeno, že termín produkční pravidlo pochází z termínu používaného pro přepsání pravidel z Chomského hierarchie typů gramatiky, kde například gramatická pravidla bez kontextu jsou označována jako bezkontextová produkce. Produkční pravidla jsou také vhodnou formou zápisu pro automatizované zpracování v expertních systémech (Mařík et al., 2004). Produkční pravidla jsou často vyjadřována jako (IF – THEN rules), jsou složena ze dvou částí a to antecedent (důkaz, situace, problém) a konsekvent (hypotéza, akce, řešení).

Formalizované produkční pravidlo má dle Maříka et al. (2004) podobu

$$E \rightarrow H. \qquad (1)$$

kde E – je Pozorování (evidence), H – je Hypotéza (hypothesis)

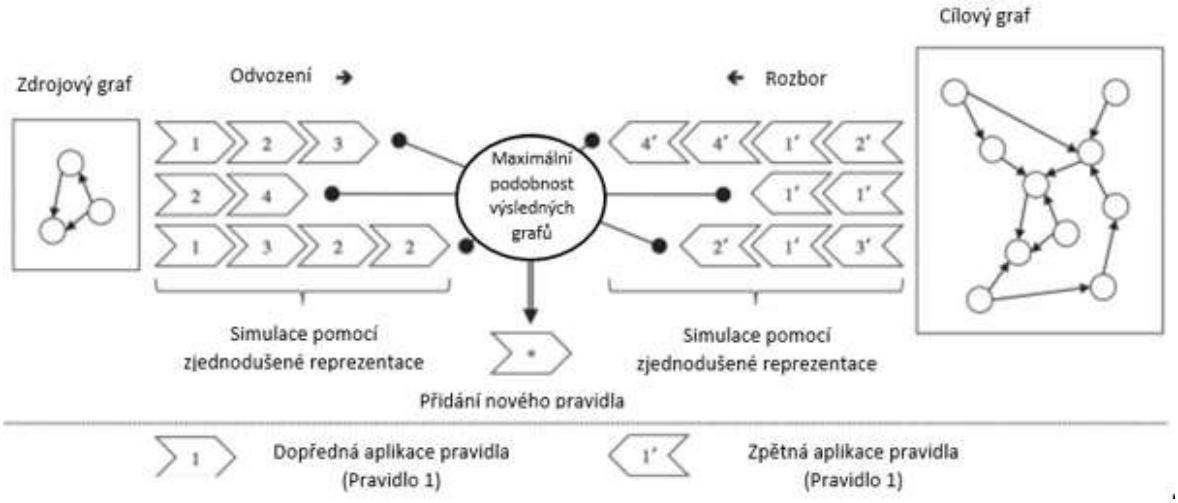
Konkrétní zápis produkčních pravidel lze uvést tímto způsobem (Production rules, 2012):

$$\begin{gathered} \text{když } \langle\text{podmínka}\rangle \text{ pak } \langle\text{závěr}\rangle \\ \text{nebo} \\ \text{když } \langle\text{podmínka}\rangle \text{ pak } \langle\text{akce}\rangle \end{gathered} \tag{2}$$

Takto definovaná produkční pravidla se využívají pro modelování rozhodovacích situací. Model má podobu buď dopředného řetězení = "modus ponens", nebo zpětného řetězení = "modus tollens". Při řešení složitých rozhodovacích situací se většinou rozhodovatel nesetkává s deterministickou povahou řešené problémové situace (Gass a Harris, 2001). Moreno a Espejo (2015) ve svém článku porovnávají na jednom příkladu – Diagnostiky lomů kovu striktně pravidlově orientovaný model s modely, které pracují s neurčitostí a dochází k závěru, že striktně pravidlově orientovaný model je hodnocen jako nejhorší k poměru přesnosti odpovědí s ostatními modely.

Produkční pravidla lze využít jako jeden ze základních kamenů pro tvorbu modelů s větší vypovídací hodnotou. Model vytvořený z produkčních pravidel má výhodu v tom, že je bližší pro lidské chápání. Z tohoto plyne, že je také dobře uchopitelný pro další analýzy (Sammut a Webb, 2016). Oliinyk et al. (2017) zmiňují, že při řešení problémů diagnostiky a automatické klasifikace je často nutné nejen odvodit přesné rozhodnutí, ale také vysvětlit, jaká cesta k rozhodnutí vede. Produkční pravidla ve smyslu popisu cesty k rozhodnutí jsou vhodným nástrojem. Oliinyk et al. (2017) uvádějí metodu extrakce produkčních pravidel založenou na paralelním principu konstrukce inteligentních modelů, tyto inteligentní modely jsou rozhodovací stromy, asociační pravidla a negativní výběr. Produkční pravidla, jak uvádí Eichhoff et al. (2016), lze uplatnit v softwarových aplikacích zvaných CAD (computer-aided design), založených na principech grafů. Tato studie je zaměřena na transformaci grafů a metody parse/derive v níž jsou prostřednictvím produkčních pravidel transformovány sady zdrojových grafů do výsledného grafu či grafů.

Tato metoda umožňuje automatické vytvoření nových produkčních pravidel, grafu z existujícího či existujících schémat. Na obrázku 5 je zobrazen proces transformace pomocí této metody.



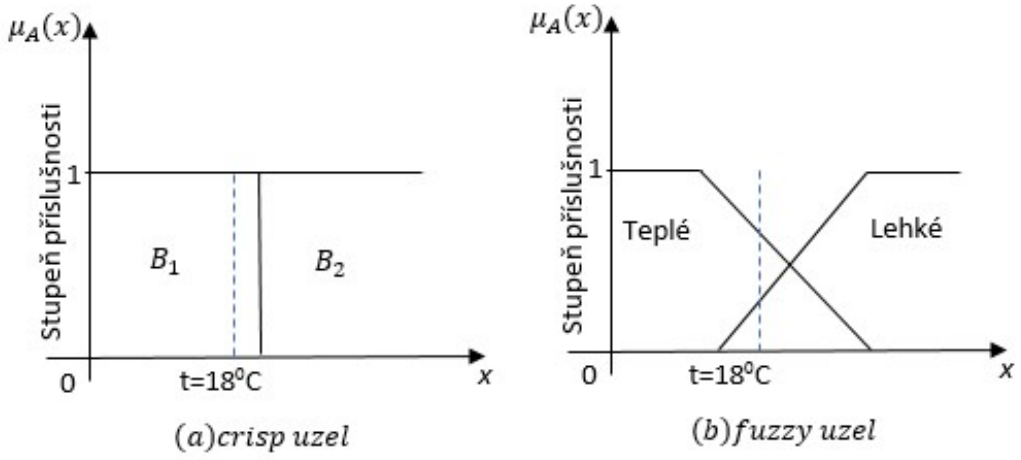
Obrázek 5 Proces transformace grafu metodou parse/derive; zdroj: Eichhoff et al., 2016

4.1.2 **Rozhodovací stromy**

Rozhodovací stromy umožňují větvení a popis rozhodovací situace. Konkrétně Li et al. (2015) uvádějí, že rozhodovací stromy jsou široce používány v data miningu a strojovém učení jako srozumitelné reprezentace znalostí. Ve své práci využívají rozhodovacích stromů také Oliinyk et al. (2017) a to v kombinaci s produkčními pravidly. Rozhodovací stromy dokáží zachytit větvení rozhodovací situace a využívají se k řešení klasifikačních problémů například v data miningu. Mirzamomen a Kangavari (2017) uvádí, že algoritmy učících se rozhodovacích stromů jsou nejčastěji využívány v klasifikačních úkolech. Kumar et al. (2016) ve své studii rozdělují pole atributů do dvou nebo více bloků. Při rozdělení do bloků zároveň upozorňují na možnost chyby při tomto „ostrém“ dělení, a konkrétně uvádí, že takové dělení může vést k velké citlivosti řešení.

Mirzamomen a Kangavari (2017) v tomto smyslu uvádí, že rozhodovací stromy se stávají nestabilní při malé změně tréninkových dat a tím pádem dochází k velmi odlišným výsledkům. Vzhledem k tomuto faktu navrhují Kumar et al. (2016) řešení v podobě fuzzifikace rozhodovacího stromu v části rozhodovacích uzlů a řešení nazývají: „*soft decision trees”*. Rozhodování v uzlu crisp (ostrého, přesného) rozhodovacího stromu a fuzzy rozhodovacího stromu jsou ukázány na příkladu volby oblečení při teplotě 18^0 C (obrázek 6). Crisp rozhodování znamená volbu buď teplého oblečení = B_1, nebo lehkého oblečení = B_2, tento uzel rozděluje atributy striktně na bloky B_1, B_2. Fuzzy uzel je definován funkcemi pro lingvisticky definované proměnné například lingvistické proměnné „Teplé“ a „Lehké“ oblečení. Při teplotě t = 18^0 C je výsledek podle crisp uzlu jednoznačně B_1 = teplé oblečení.

Výsledek podle fuzzy uzlu je dán vztahem fuzzy funkcí příslušnosti a pravidly vyhodnocení lingvistických proměnných. Při teplotě t = 18^0 C podle fuzzy uzlu by mohl být výsledek interpretován jako „Spíše teplé oblečení“.



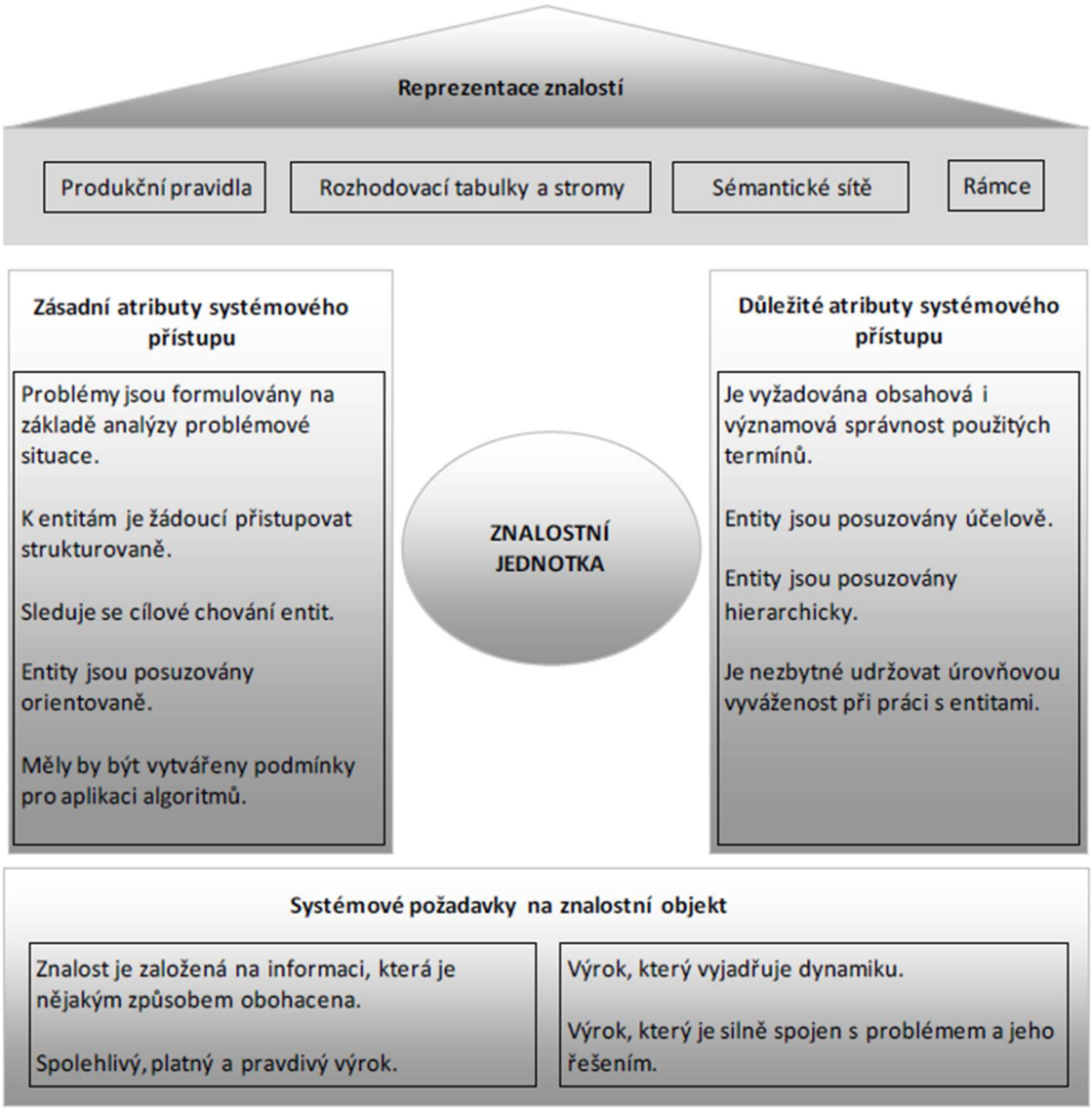
Obrázek 6 Crisp uzel a Fuzzy uzel; zdroj: Kumar et al., 2016

Rozhodovací stromy nejsou vhodné pro velké datové soubory. Tento fakt potvrzuje Kumar et al. (2016) tvrzením, že přepočtem všech uzlů narůstá výpočetní složitost. Li et al. (2015) ve svém článku uvádějí řešení této výpočetní náročnosti v podobě učících se algoritmů na základě tréninkových dat například algoritmem CS-C4.5 (Cost sensitive C4.5). Tento algoritmus je rozšířením tradičního indukčního učení s cílem minimalizovat celkové náklady na klasifikaci dat pomocí rozhodovacích stromů. Tento typ učení a současně učení z testu nesprávné klasifikace považují Li et al. (2015) za zajímavé. Autoři Mirzamomen a Kangavari (2017) ve své studii vytvořili rámec pro dosažení stabilnějších rozhodovacích stromů. Pro potvrzení funkčnosti vytvořené metody jsou ve studii uvedena porovnání s jinými metodami například prohledávání do hloubky a šířky rozhodovacího stromu.

4.1.3 **Znalostní jednotky**

Znalostní jednotky, jak uvádějí ve své práci Brožová a Houška (2011), vytvářejí systémový přístup k modelování znalostí. Nejprve jsou identifikovány atributy systémového přístupu k modelování znalostí a poté jsou ohodnoceny čtyřstupňovou stupnicí významnosti. Stupnice významnosti, vychází z ohodnocení důležitosti atributu a její hodnoty jsou: zásadní, důležitý, relevantní, irelevantní.

Po tomto ohodnocení atributů je odvozen model znalostní jednotky a autoři Brožová a Houška (2011) ji zobrazují jako dům (obrázek 7) ze kterého přiřazením významu k jednotlivým komponentám lze identifikovat znalostní jednotku a vhodně ji reprezentovat.



Obrázek 7 Postup konstrukce znalostní jednotky; zdroj: Brožová a Houška, 2011

Formulovaná znalostní jednotka musí obsahovat následující atributy:

1. Znalostní jednotka je uskupení (balíček) pravdivých a platných informací, které vzaty vcelku vyjadřují dynamiku a jsou spojeny s problémem a jeho řešením.

2. Aplikací systémového přístupu je možné identifikovat strukturu znalostní jednotky. Skládá se ze čtyř komponent:

• problémové situace;

• problému;

• cíle řešení problému;

• řešení problému.

3. Každá takto definovaná znalostní jednotka musí explicitně obsahovat všechny čtyři uvedené komponenty.

Dále je navrhnut způsob této reprezentace a konkrétně využití rozšířených produkčních pravidel. Analytická forma znalostní jednotky je (Dömeová et al., 2008; Brožová a Houška, 2011) vyjádřena následujícím způsobem:

$$\mathrm{ZJ} = \{\mathrm{X}, \mathrm{Y}, \mathrm{Z}, \mathrm{Q}\}, \mathrm{kde} \tag{3}$$

X = „problémová situace“,

Y = „elementární problém“,

Z = „cíl řešení elementárního problému“,

Q = „řešení elementárního problému“.

Přestože je dle Kendala a Creena (2007) problematické vytvářet z produkčních pravidel ucelené texty v přirozeném jazyce, znalostní jednotky textově reprezentovat lze. Textová podoba znalostní jednotky podle Brožové a Houšky (2011) je uvedena následovně:

"*Když je třeba v rámci problémové situace X řešit elementární problém Y, aby bylo dosaženo cíle Z, potom je třeba aplikovat řešení Q.*"

Text složený pouze ze znalostních jednotek by byl jen těžko čitelný, proto Rauchová a Houška (2013) navrhli metodiku, jak znalostní jednotky doplňovat informacemi a daty, aby se daly použít například jako základ vzdělávacích textů.

Analytická forma znalostní jednotky musí splňovat výše zmíněné atributy a požadavek na atomičnost znalostní jednotky. Atomičností je myšlena jednokriteriálnost, která připouští v pojetí znalostí jednotky jeden cíl „Z“. V souvislosti s takto definovanou znalostní jednotkou Brožová a Houška (2011) dále uvádějí vztahy mezi komponentami znalostní jednotky. Znalostní jednotky lze uspořádat do hierarchické struktury. Systémový přístup

otevírá možnosti k provádění operací se znalostními jednotkami, Houška a Beránková (2013) uvádí unární a binární operace se znalostními jednotkami. Definují předpoklady pro základní unární operace, kterými jsou drill-down, roll-up a pro binární operace, pro které jsou předpokladem vstupu dvě znalostní jednotky, prostý součet a odečítání. Teoretickou definici poté doplňují konkrétními příklady.

4.2 Interoperabilita znalostí

Pojem interoperability je využíván zejména v oblasti informačních a komunikačních technologií (ICT). Z hlediska ICT je interoperabilita vlastnost systémů navzájem kooperovat a vyměňovat si informace s co nejmenšími ztrátami a bez zásahu uživatele. Adel et al. (2017) vysvětlují interoperabilitu jako efektivní výměnu obsahu mezi systémy, která spoléhá na metadata popisující přenášený obsah. Přenesený obsah je poté pomocí metadat správně interpretován v cílovém systému. Interoperabilita je tak spjata s metadaty, která interpretují přenášený obsah (Riley, 2017). Interoperabilitě se věnují standardizační instituty, například NISO[2] a je také skloňována v souvislosti s internetem věcí (The Internet of Things = IoT). Jak uvádí Ganzha et al. (2017), idea IoT je zkoumána napříč kontinenty a přináší důležitou otázku, která zní: Jak dosáhnout interoperability mezi více existujícími (a neustále vytvářenými) platformami. V kontextu IoT je interoperabilita otevřené rozhraní komunikující napříč platformami a přidružená metodologie k zajištění této komunikace mezi heterogenními platformami. Definice interoperability podle French speaking Libre Software Users' Association (2015):

"Interoperabilita je vlastnost produktu nebo systému - jehož rozhraní jsou plně zveřejněna - komunikovat a pracovat s dalšími produkty či systémy bez jakéhokoliv omezení v přístupu a implementaci."

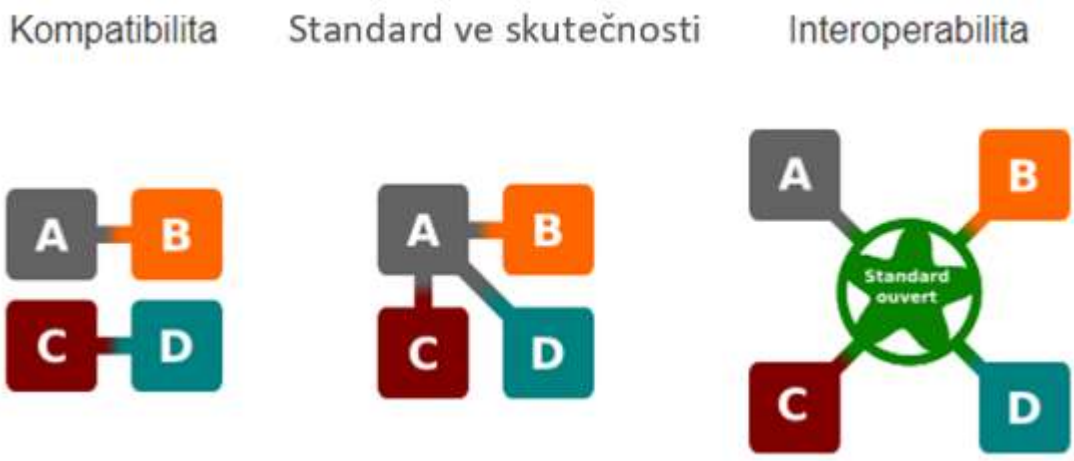
K této definici jsou dále uvedeny stupně provozuschopnosti:

- Kompatibilita, tj. schopnost vzájemného propojení dvou různých typů systémů.

[2] NISO = National Information Standards Organization

- Standard, tj. jeden dominantní systém a ostatní komunikují přes něj (nejsou kompatibilní mezi sebou, ale s ním). Výhodou je, že systémy mohou vzájemně komunikovat. Nevýhodou je, že dominantní systém určitým způsobem tuto komunikaci ovládá.
- Interoperabilita je schopnost různých systémů vzájemně komunikovat bez závislosti na konkrétním činiteli. Je založena na přítomnosti otevřeného standardu.

Na obrázku 8 jsou uvedeny na schématech stupně provozuschopnosti.

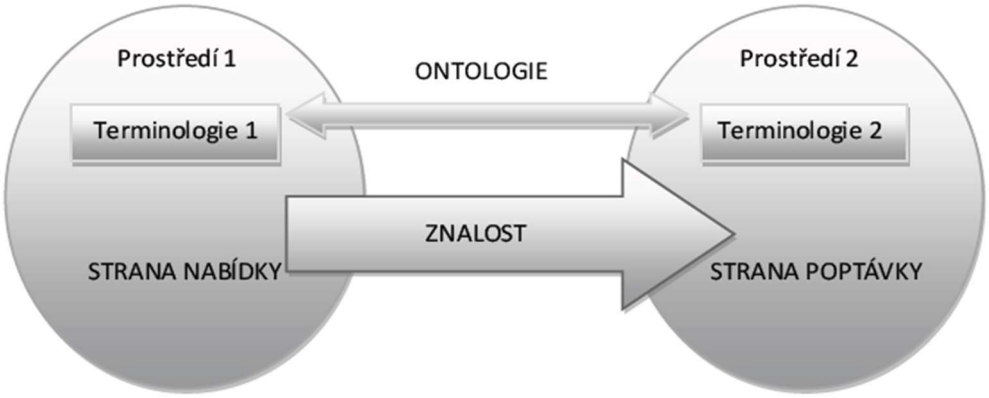


Obrázek 8 Stupně provozuschopnosti; zdroj: French speaking Libre Software Users' Association, 2015

Interoperabilita v kontextu znalostí je klíčem k vytvoření znalostního ekosystému, který prostřednictvím interoperability umožňuje přenos a sdílení znalostí. Interoperabilitu znalostí definovali Brožová a Houška (2011) jako:

"Proces, při kterém dochází k transferu znalostí, který probíhá mezi dvěma objekty z navzájem heterogenních prostředí. Klasický transfer znalostí probíhá v prostředí (prostředích), které je (která jsou) homogenní".

Heterogenní prostředí ve smyslu interoperability znamená obdobně jako v prostředí ICT, že terminologie a její význam mohou být různé v různých prostředích. To znamená, že při přenosu znalosti, znalostního obsahu je důležitý formát, který znalostní obsah vysvětluje. Jedná se o záležitost, o které se zmiňují také Sammut a Webb (2016) a Oliinyk et al. (2017), kteří kladou důraz na odůvodňování výsledků - rozhodnutí v expertních systémech pomocí produkčních pravidel. Obrázek 9 znázorňuje interoperabilitu znalosti mezi dvěma prostředími.



Obrázek 9 Interoperabilita znalostí; zdroj: Brožová a Houška, 2011

Pro popis správné terminologie slouží ontologie. Definice a význam ontologie z původního textu vysvětluje Gideon Harvey v 17. století jako:

"Metaphysical science or study of being."

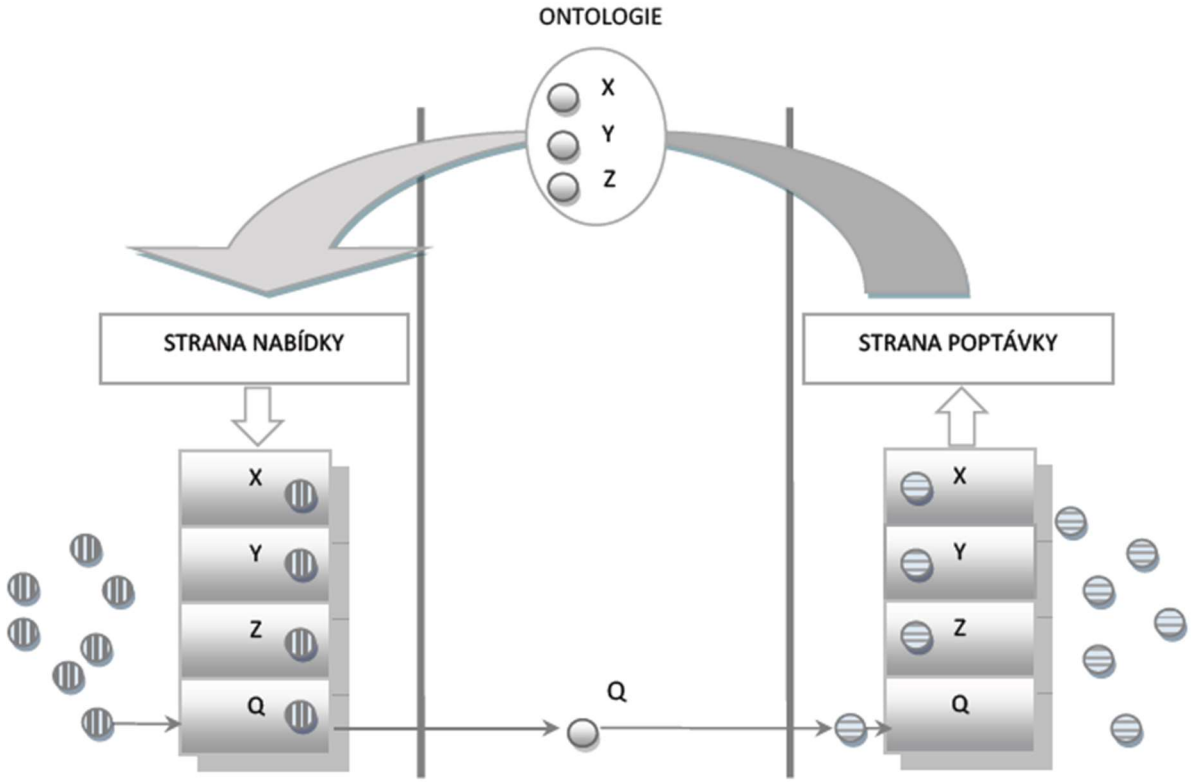
Například Ali et al. (2017) vysvětlují pojem ontologie jako sdílenou znalost (obsah) určité domény mezi lidmi a systémy (sémantika), která je napsána v určitém jazyce včetně programovacího (syntax).

Ontologie v kontextu interoperability znalostí je důležitým prvkem, který stanovuje obsah přenosu znalosti tak, aby byl srozumitelný. Podle Brožové a Houšky (2011) má interoperabilita znalostí dvě hlavní strany. Poptávku po znalostech, která požaduje znalost k vyřešení problému (znalostní požadavek) a druhou nabídkovou stranu znalostí, což je strana, která zná řešení problému (řešení). Do problematiky interoperability znalostí byla systematicky včleněna znalostní jednotka. Vzhledem ke své konstrukci je znalostní jednotka v podobě rozšířeného produkčního pravidla vhodným subjektem (syntaxí) pro modelování interoperability znalostí. Z hlediska znalostních jednotek byly problémy podle Brožové a Houšky (2011) v kontextu Simonovy hierarchie rozčleněny na:

- Nestrukturované znalostní jednotky (např. “X, ?, ?, ?”)
- Částečně strukturované (např. “X, Y, ?, ?”)
- Dobře strukturované (např. “X, Y, Z, ?”)

Možné vytvoření ontologie z poptávkové strany, která nezná řešení, je uvedeno na obrázku 9. Proto, aby se zkompletovala znalostní jednotka v případě, kdy strana poptávky nemá řešení Q, je samotný postup redukován na získání jedné konkrétní informace (Q). Pro získání Q ze znalostní jednotky je nezbytné sestavit obecnou ontologii a nalézt k ní analogii v cizích

prostředích, poté se pokusit převzít dané řešení. Naopak na stranu nabídky je doručena ontologie a na jejímž základě nabízí poptávce (prostřednictvím ontologie) možná řešení. Tento příklad je z uvedené hierarchie nejjednodušší, pro obtížnější případy lze například využít měkkých metodologií, obrázek 10 (Brožová a Houška, 2011).



Obrázek 10 Ontologie znalostní jednotky, poptávková strana nezná řešení; zdroj: Brožová a Houška, 2011

4.3 Expertní systémy

Expertní systémy (ES) se prolínají do mnoha různých vědních oborů. Adaptace lidského přemýšlení a uvažování do strojových kódů doznává v dnešní informační době velkého rozmachu. Cílem těchto snah automatizace myšlení pomocí expertních systémů je přenést čím dál více sofistikovanější činnosti na stroje a uvolnit tak kapacitu lidským expertům. Jedna z definic expertního systému podle Feigenbauma (Gevarter, 1984) je následující:

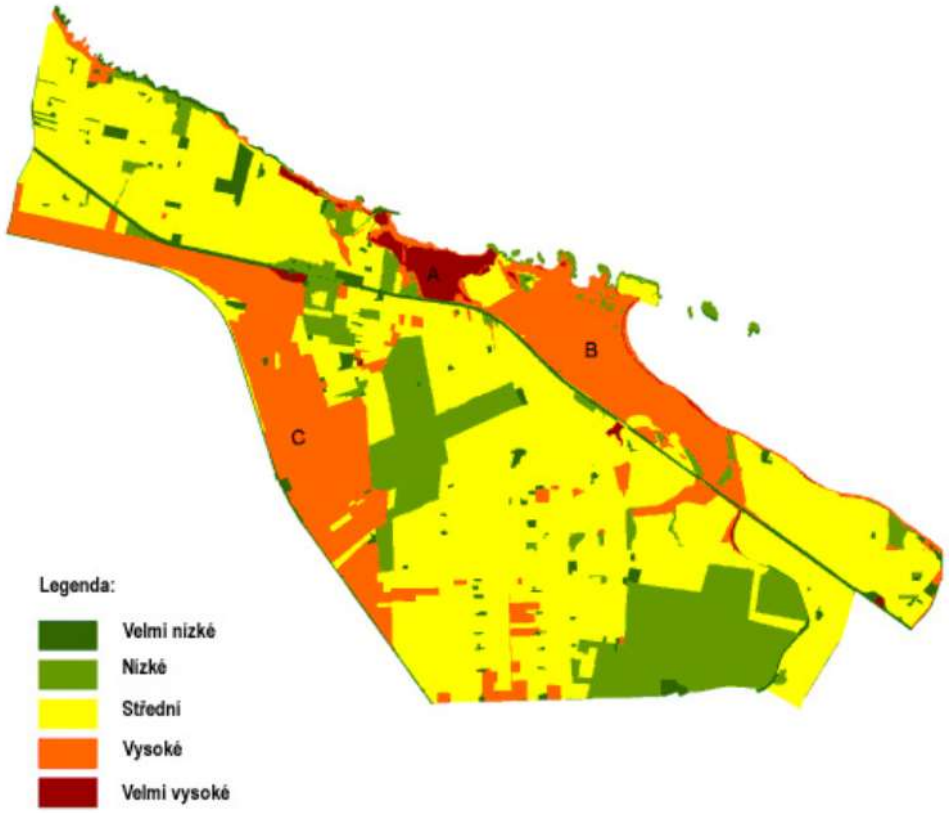
"Expertní systém je počítačový program, simulující rozhodovací činnost experta při řešení složitých úloh a využívající vhodně zakódovaných, explicitně vyjádřených znalostí,

převzatých od experta, s cílem dosáhnout ve zvolené problémové oblasti kvality rozhodování na úrovni experta."

Encyclopædia Britannica (2016) definuje expertní systém jako:

"Počítačový program, který využívá metody umělé inteligence pro řešení problémů ve specifické znalostní doméně, které normálně potřebují lidskou analýzu."

Obě definice se vyjadřují o expertním systému jako o počítačovém programu. Je otázkou, jestli lze jednoduše říci, že expertní systém je počítačový program, pokud inteligence ES je uložena mimo programový kód a řídící mechanismus ES je programový kód, nebo spíše softwarově hardwarový systém. Jednoznačně lze pomocí počítačových programů a počítačové paměti expertní systém vytvořit. Z definic je možné odvodit náměty na řešení různých problémových situací pomocí expertních systémů. Konkrétním příkladem je expertní systém pro hodnocení zranitelnosti středomořské chráněné oblasti vůči požáru (Semeraro et al., 2016). Systém byl zkonstruován, aby bylo možné účinně provádět předběžná opatření proti možným požárům. Systém je založen na fuzzy vícekriteriální analýze, která je integrovaná do geografického informačního systému (GIS). Výsledky tohoto systému jsou převedeny do aplikační vrstvy softwaru GIS, ve kterýchjsou zvýrazněny kritické oblasti (obrázek 11).



Obrázek 11 Kritické oblasti požáru; zdroj: Semeraro et al., 2016

Rozvoj informačních technologií a jejich miniaturizace tomuto trendu jdou naproti. Současně jsou obdobnou měrou precizovány postupy a metodiky vytváření expertních systémů. Jsou uváděny metodiky, jakým způsobem vytvářet a automatizovat expertní systémy, aby byly dynamické, snadno konstruovatelné, upravitelné, s intuitivním ovládáním a samozřejmě, aby dávaly přesné výsledky. Wagner (2017) publikoval přehledovou studii, kde zachytil rozsáhlé znalosti obsažené ve velkém korpusu případových studií s cílem vyzkoušet a poskytnout lepší pokyny budoucím vývojářům expertních systémů.

Zdrojové publikace byly publikovány v nejrůznějších časopisech a knihách a odrážely širokou škálu průmyslových odvětví a různých problémových oblastí. Expertní systémy jsou rozvíjeny s cílem přibližovat se lidským expertům v různých oblastech rozhodování. Výhodou expertních systémů je jejich využití v situacích, ve kterých účast lidského experta není možná, nicméně lidští experti stále mají výhodu intuice. K trendu využívání expertních systémů uvádějí Venturelli et al. (2017):

"Expertních systémů lze využít všude tam, kde je potřeba simulovat proces lidského rozhodování v složitých rozhodovacích situacích".

4.3.1 **Vznik a vývoj expertních systémů**

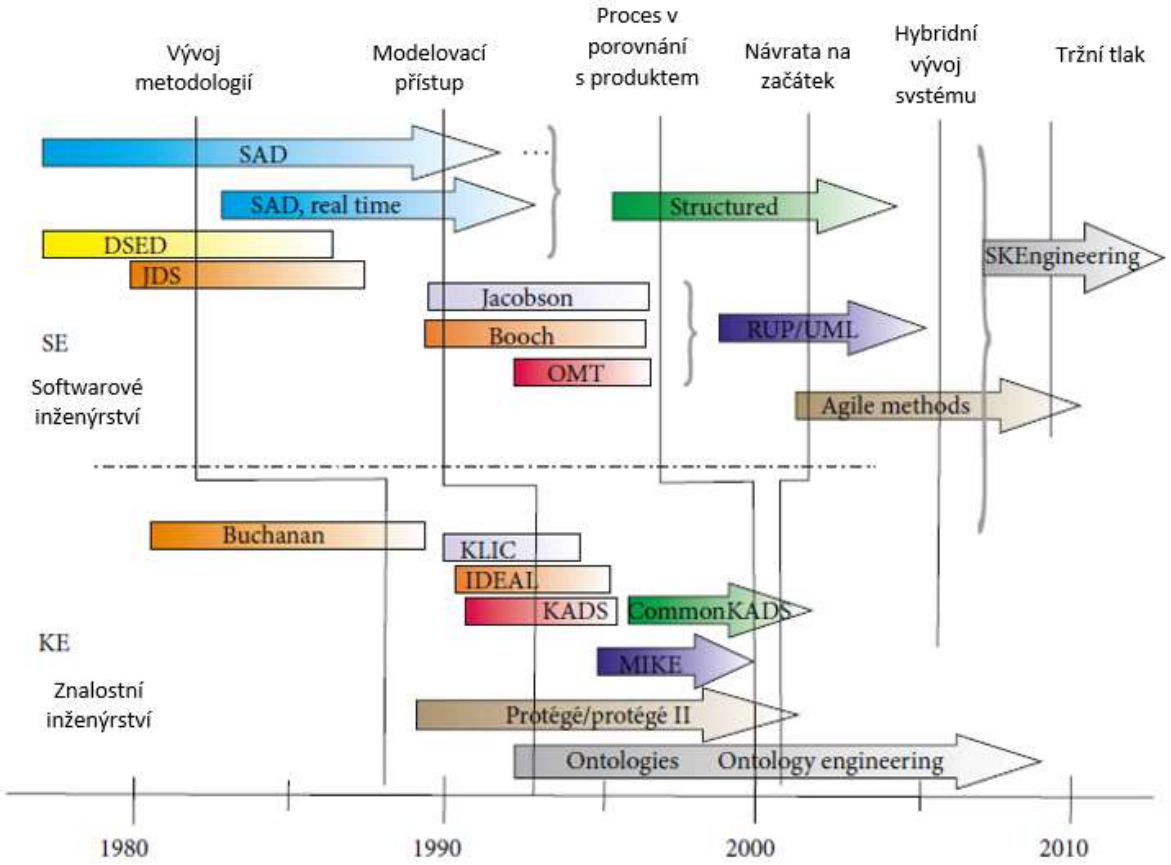
Datování vzniku prvního expertního systému se váže k systému s názvem „DENDRAL" (Lindsay et all., 1980). Tento expertní systém byl zaměřen na organickou chemii a pomáhal rozeznávat neznámé organické molekuly. Vývoj expertních systémů byl a je závislý na řadě odborností a vědních disciplín, ale stěžejními obory se stalo znalostní (KE) a softwarové (SE) inženýrství. Přehledová studie (del Águila et al., 2014) mapuje vývoj znalostního a softwarového inženýrství. Počátek softwarového inženýrství se datuje k roku 1956, kdy byl vytvořen první operační systém v automobilce General Motors. Počátek znalostního inženýrství je vázán k rokům 1956 až 1978. Za prvního průkopníka s vizí, že stroje mohou "myslet" jako lidští experti, je považován Alan Turing (1900 - 1956). V tomto období byly publikovány studie směřující k vývoji obecných systémů pro řešení problému jako například STRIPS (Stanford Reaserch Institut Problem Solver), Plánovací systém (Fikes a Nilson, 1971), nebo obecný řešitel problémů GPS (Newell a Simon, 1961). V přehledové studii del Águila et al. (2014) je popsáno šest hlavních období ve vývoji expertních systémů:

- Vývoj metodologií, počátek softwarového a znalostního inženýrství.
- Modelovací přístup, vývoj modelů, které podporují konstrukci softwaru.
- Proces v porovnání s produktem, samotný proces dosahuje stejného významu jako generované výstupy/artefakty během projektu.
- Návrat na začátek[3] byl způsoben nedostatečným úspěchem komerčního softwaru vyvinutého v rámci disciplíny znalostního inženýrství. Tato skutečnost spolu se zavedením a úspěšným rozvojem ontologií a sémantického webu a také agilní metody softwarového inženýrství vedly znalostní inženýrství zpět ke kořenům. Agilní metody na rozdíl od jiných metod vývoje softwaru jsou zaměřené na rychlou adaptaci softwaru na změny v realitě.

[3] Přeloženo z originálu – „*called second childhood"*

- Hybridní vývoj systému, spojení dvou pohledů znalostního a softwarového inženýrství prostřednictvím integrace algoritmů a heuristických metod. Z tohoto hybridního vývoje vzniká nová oblast Softwarově znalostní inženýrství[4].
- Tržní tlak, rozvoj softwarového inženýrství s cílem definovat nové postupy (metodiky), jako je například vývoj Cloud computingu. To je výzvou pro softwarové inženýrství, ale znalostní inženýrství je dosud neznámé. (del Águila et al., 2014)

Schéma na obrázku 12 zobrazuje vývoj znalostního a softwarového inženýrství a jejich hlavní fáze.



Obrázek 12 Metodologie a milníky; zdroj: del Águila et al., 2014

Aguilera et al. (2011) sledují vývoj v absolutních počtech vědeckých publikací na téma expertní systémy využívající Bayesovské sítě od roku 1990 do roku 2010. V tomto zmíněném období je z celkového počtu 1375 článků vyvozen nárůst počtu článků zabývajících se expertními systémy v roce 2002. Tento trend se drží i nadále, např. roku 2010 bylo publikováno 226 článků. Z výše deklarovaného vzorku je největší zastoupení v

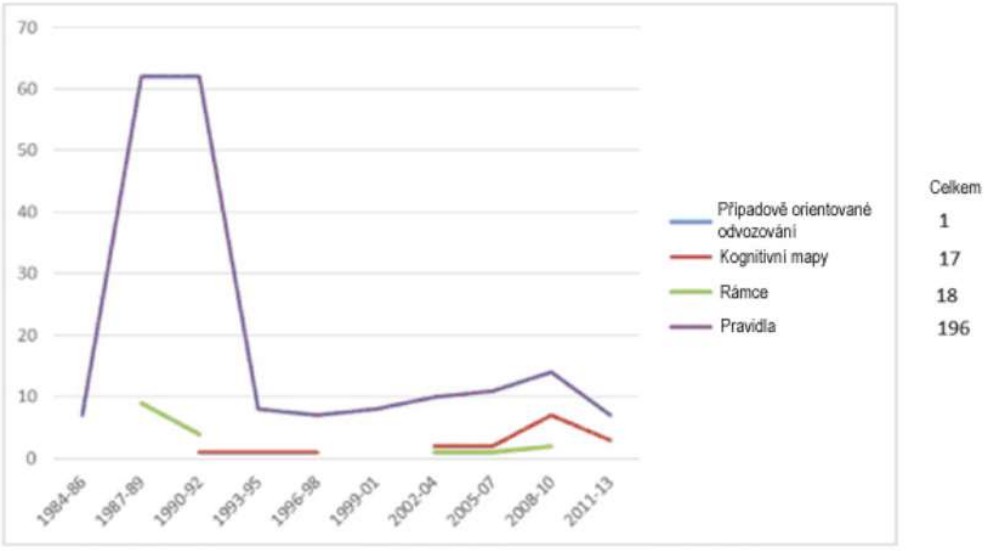
[4] SKEngineering = Software knowledge engeneering

oborech výpočetních a to 27,3 % a v matematice, naopak malé v sociologii, vzdělávání a v oblasti environmentální je zastoupení nejmenší - 4,2 %. Svým vývojem expertní systémy směřují k co nejlepší simulaci lidského rozhodování. Aby se této simulaci co nejvíce přiblížily, se musí vyrovnat s neurčitostí. Velmi slibným východiskem je orientace na fuzzy expertní systémy a na takzvané hybridní expertní systémy. Například Moreno a Espejo (2015) uvádí fuzzy expertní systém pro diagnostiku lomu kovů, kde podle popisu testovaného materiálu a lomu systém určuje, jakým způsobem k lomu došlo. Venturelli et al. (2017) jsou autoři expertního systému zaměřeného na posuzování firmy z hlediska CSR[5] indexů „*Social Responsibility index*“. Další ukázkou je expertní systém od Ghanei et al. (2015), který vyhodnocuje stav materiálů pomocí nedestruktivních kontrol a využívá ANFIS[6], což je postup, který kombinuje neuronové sítě a fuzzy logiku.

4.3.2 **Struktura expertních systémů**

Základem expertních systémů jsou strategie, s kterými expertní systém pracuje při odpovědi na dotazy. Obecně porovnává fakta z báze faktů s pravidly. Pokud nalezne více než jedno vyhovující pravidlo, rozhoduje, které vybrat. V tomto případě postupuje dál řešením konfliktů - neurčitostí. Tyto strategie pravidly orientovaných expertních systémů se obecně dají rozdělit do dvou skupin, a to na expertní systémy, které pracují pomocí porovnávání se vzorem a na expertní systémy, které vyhodnocují dotazy pomocí inferenční sítě. Wagner (2017) uvádí rozdělení expertních systémů podle primární reprezentace znalostí v expertním systému. Přehled je uveden v grafu na obrázku 13.

[5] CSR = Corporate Social Reponsibility
[6] ANFIS = Adaptive neuro-fuzzy inference system)



Obrázek 13 Primární reprezentace znalostí; zdroj: Wagner, 2017

Největší část expertních systémů používá produkční pravidla (196 z 232 zkoumaných). Wagner (2017) však poukazuje na fakt, že formu reprezentace znalostí v expertních systémech je obtížné určit. Důvodem je akvizice znalostí od expertů a na druhé straně různé platformy expertních systémů.

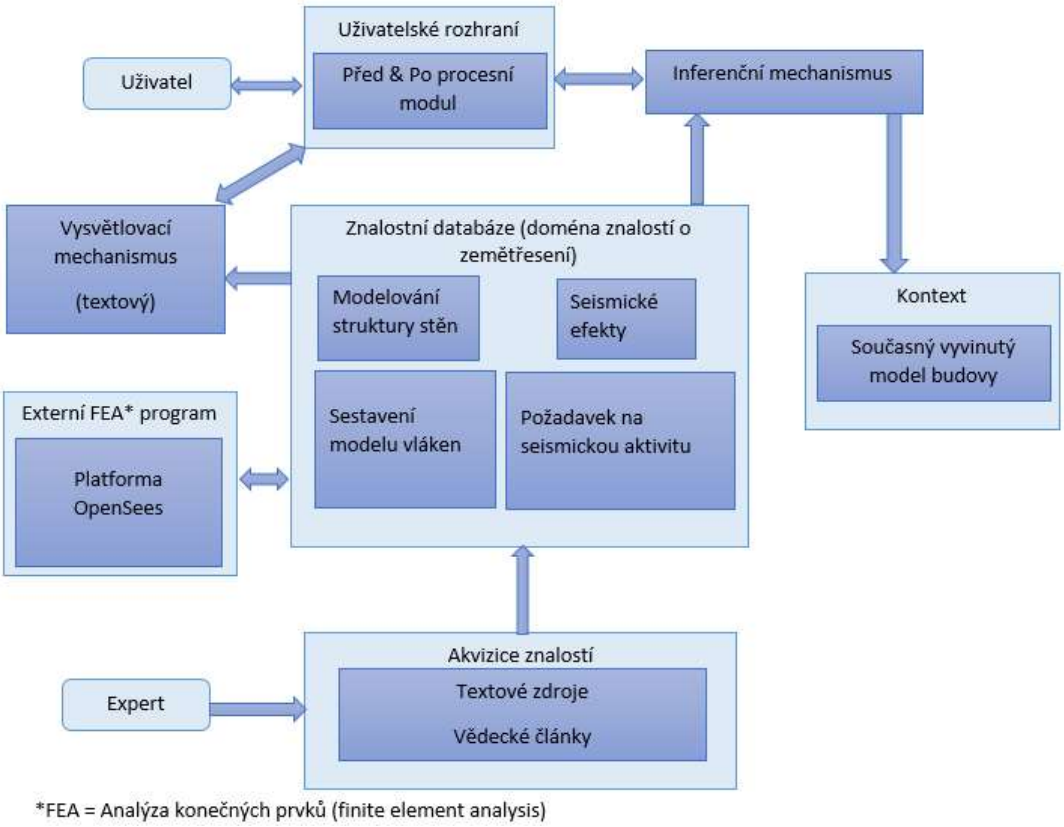
Prvky expertního systému jsou z části determinovány typem expertního systému. Základní prvky expertního systému dělí Garibaldi (1997) na:

- báze znalostí,
- inferenční mechanismus,
- uživatelské rozhraní.

Do většího detailu v rozdělení jde například Dvořák (2004) a škáluje obecné prvky expertního systému následujícím způsobem:

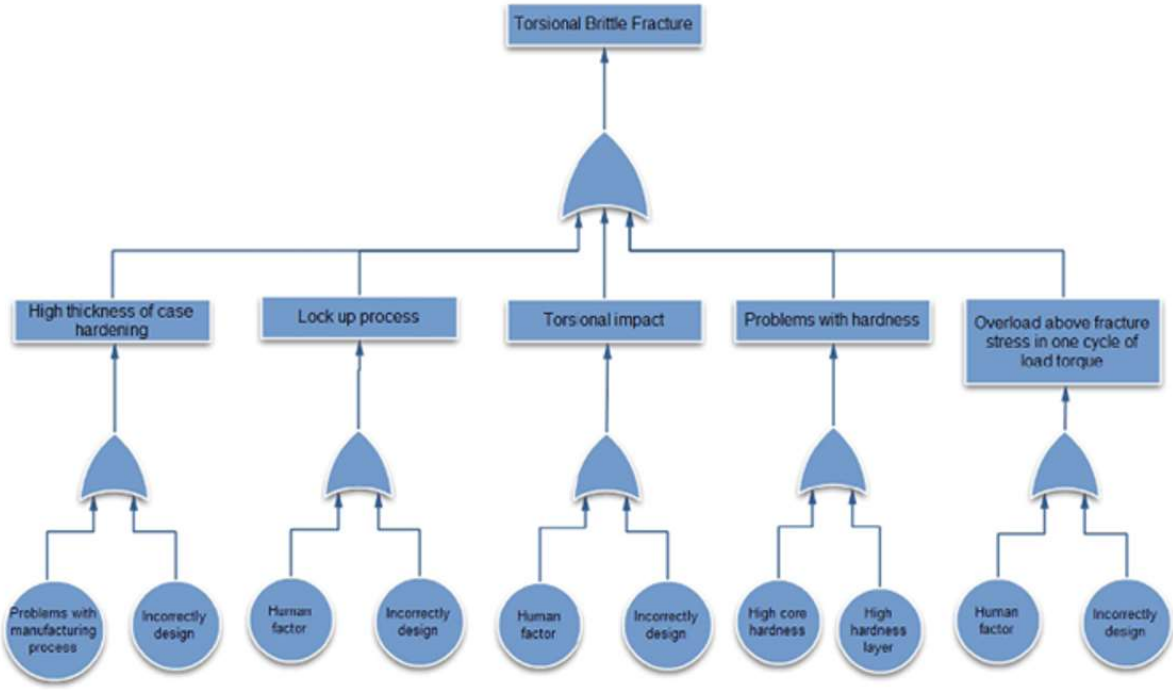
- báze znalostí,
- inferenční mechanismus,
- I/O rozhraní (uživatelské, vývojové, vazby na jiné systémy),
- vysvětlovací modul,
- modul pro akvizici (získávání) znalostí.

Tyto prvky spolu navzájem komunikují a tvoří obecné schéma expertního systému jako například v uvedeném schématu (obrázek 14).



Obrázek 14 Architektura expertního systému; zdroj: Psyrras a Sextos, 2018

Databáze znalostí je složena z uložených faktů a sad pravidel vytvořených pro danou znalostní doménu (Psyrras a Sextos, 2018). Sady pravidel jsou explicitně definovány experty na danou problematiku, nebo mohou být vytvářeny učícími se algoritmy. K tomuto učení dochází v modulu pro akvizici znalostí. Pro tvorbu databáze pravidel často využívá *if-then* pravidla (Seipel et al., 2018). Takto definovaná pravidla jsou využívána v inferenčním mechanismu při dopředném a zpětném řetězení. Databáze znalostí (Garibaldi, 1997) obsahuje znalosti nezbytné pro pochopení, formulaci a řešení problémů a uvádí její dva základní prvky, kterými jsou fakta a zvláštní heuristika nebo pravidla, která řídí využití znalostí. Nové znalosti jsou získávány pomocí lidských expertů a mechanismu akvizice znalostí (Venturelli et al., 2017). Kimball (2008) popisuje databázi faktů podle způsobu měření dat. Fakta v databázi faktů obsahují popis dané domény, pro kterou se expertní systémy konstruují. Například v práci autorů Moreno a Espejo (2015) jsou zachycena fakta představující možné příčiny lomu kovu, obrázek 15.



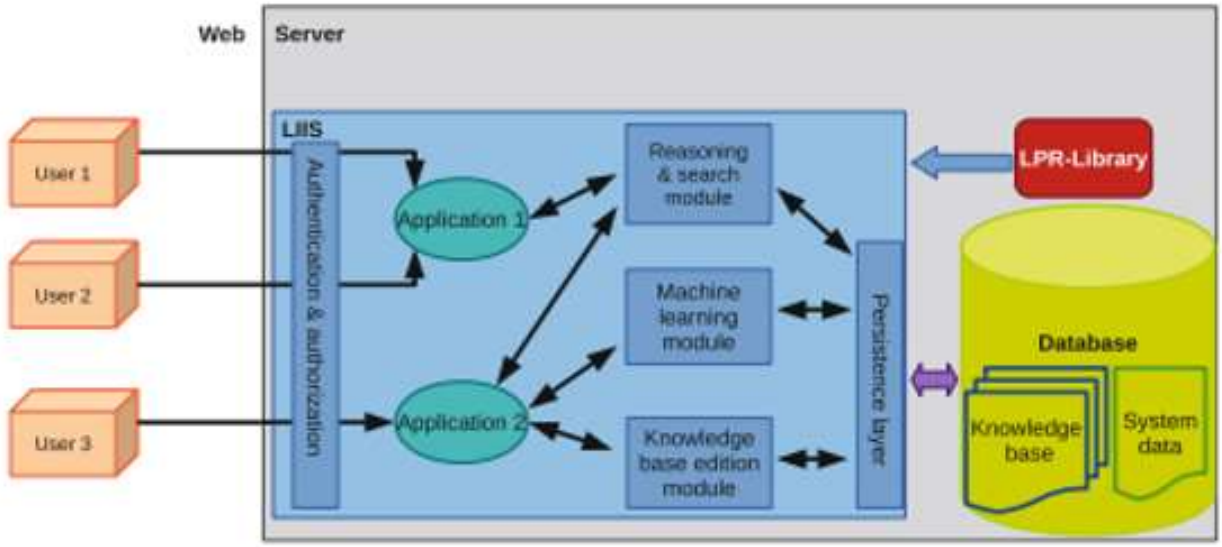
Obrázek 15 Příklad analýzy lomu kovu; zdroj: Moreno a Espejo, 2015

Znalostní databáze je tedy využívána inferenčním mechanismem, který odpovídá na uživatelské dotazy. Inferenční mechanismus odvozuje odpovědi a plní tak dynamickou funkci expertního systému, je mozkem expertního systému, řídící strukturou a poskytuje metodiku pro vyvozování výsledků (Gass a Harris, 2001).

Interface expertního systému zprostředkovává komunikaci mezi uživatelem a systémem (Psyrras a Sextos, 2018). Uživatelské rozhraní expertního systému slouží pro uživatelsky přívětivou komunikaci při řešení problémů. Z nejčastěji řešených problémů expertními systémy jsou například vývoj, konzultace problémů a vylepšování systému (Niveditha a Basavaraj, 2017). Uživatelské rozhraní může být vytvořeno různými způsoby. Například Lee a Evens (1998) uvádí rozhraní, které je tvořeno přirozeným jazykem uživatele pro komunikaci s analyzátorem (inferenčním mechanismem) expertního systému. Tento způsob komunikace je obecně zobrazen schématem na obrázku 16.

Obrázek 16 Rozhraní pro přirozený jazyk uživatele; zdroj: Lee a Evens, 1998

Rozhraní expertních systémů mohou mít různou podobu. Například rozhraní pro hlasovou komunikaci se systémem, které vychází z patentu AT&T Corp. (New York, N.Y., US09059912). Proces komunikace přes hlasové rozhraní přijímá hlasové příkazy, dokud nezíská dostatek informací, aby sestavilo gramatiku přirozeného jazyka pro použití rozhraní. Poté rozhraní využívá gramatiku přirozeného jazyka k analýze příkazů daných uživatelem pro budoucí komunikační spojení. Takto vytvořené rozhraní (Legień et al., 2015) je webová aplikace, která umožňuje přístup „on line“ přes internet více uživatelům zároveň. Architektura rozhraní je na obrázku 17.



Obrázek 17 Architektura rozhraní; zdroj: Legień et al., 2015

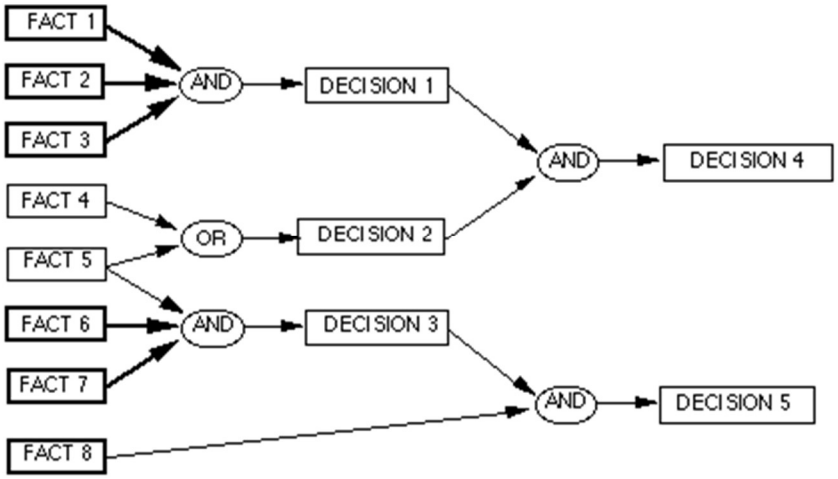
4.3.3 **Inference znalostí v expertních systémech**

Klíčovým prvkem v konstrukci expertních systémů je inferenční mechanismus a pravidla, s nimiž expertní systém pracuje. Metod inference je, jak uvádí Giarratano a Riley (1989), několik, z nichž se jen některé hodí k automatickému zpracování. Inferenční metody podle Giarratano a Riley (1989) jsou následující:

- Dedukce je usuzování, ve kterém závěry musejí vyplývat z předpokladů.
- Indukce je obecně postup od zdola nahoru tzn. od specifického k obecnému.
- Abdukce vychází ze správného závěru k předpokladům, které mohly být logickou příčinou.
- Heuristiky jsou pravidla, která jsou založena na zkušenostech.
- Generování a testování.
- Analogie ve smyslu odvozování závěru na základě podobnosti s jinou situací.

Dalšími typy jsou defaultní inference, nemonotónní inference a intuice. Intuice je obtížně vysvětlitelný zdroj znalostí a informací, v expertním systému těžko implementovatelný. Podle Dvořáka (2004) intuice nebyla v expertních systémech implementována a snad by se k ní mohlo blížit usuzování neuronových sítí. Inferenční mechanismus v pravidlových expertních systémech pracuje na základě dopředného řetězení „Forward chaining“ nebo zpětného řetězení „Backward chaining“.

Dopředné řetězení je založeno na syntéze neboli na sběru dat, na jejichž základě se rozhoduje. Dopředné řetězení je konstruováno za účelem odvodit ze shromážděných skutečností možná vysvětlení. Schematicky je inferenční síť dopředného řetězení uvedena na obrázku 18.

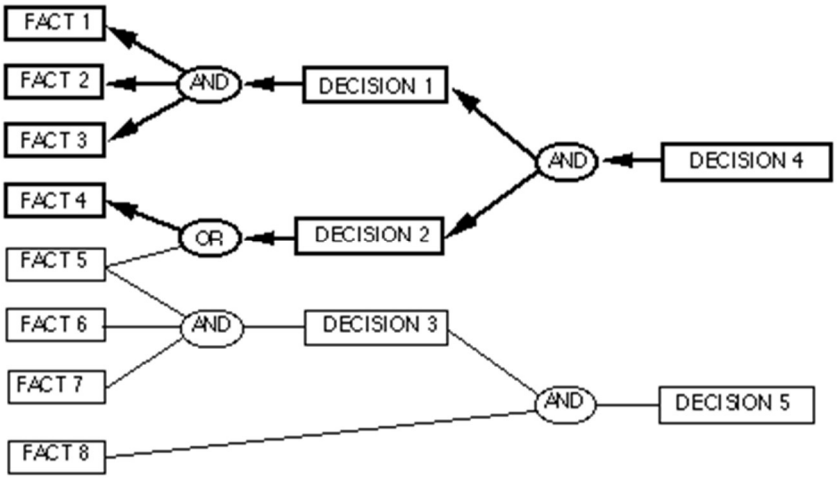


Obrázek 18 Dopředné řetězení; zdroj: Lecture notes Production rules, 2012

	Databáze pravidel pro dopředné řetězení
Pravidlo 1	Jestliže „FACT 1“ A „DECISION 1“ Potom „DECISION 4“
Pravidlo 2	Jestliže „FACT 2“ A „DECISION 1“ Potom „DECISION 4“
Pravidlo 3	….

Tabulka 1 Databáze pravidel pro dopředné řetězení; zdroj: Lecture notes Production rules, 2012

Zpětné řetězení oproti dopřednému plní účel rádce, který ze shromážděných skutečností navrhne závěry a ty poté odůvodňuje. Produkční pravidla při zpětném řetězení jsou řetězena za účelem získat podklady, důvody pro rozhodnutí. Schematicky je inferenční síť zpětného řetězení uvedena na obrázku 19.



Obrázek 19 Zpětné řetězení; zdroj: Lecture notes Production rules, 2012

	Databáze pravidel pro zpětné řetězení
Pravidlo 1	Jestliže „DECISION 4“ A „DECISION 1“ Potom „FACT 1“
Pravidlo 2	Jestliže „DECISION 4“ A „DECISION 1“ Potom „FACT 2“
Pravidlo 3	….

Tabulka 2 Databáze pravidel pro zpětné řetězen; zdroj: Lecture notes Production rules, 2012

Strategie inference jsou doplňovány různými metodami. Například Fakhrahmad et al. (2015) se zaměřil na inferenční sítě a také na hybridní expertní systémy. V tomto článku je využito dopředné řetězení a datamining k snižování nejednoznačnosti při automatickém překladu. Následně Ghanei et al. (2015) uvádí expertní systém na zjišťování stavu materiálů pomocí vyhodnocování nedestruktivních kontrol, kde používá postupu ANFIS, který kombinuje neuronové sítě a fuzzy logiku.

Z pohledu použití produkčních pravidel v inferenci dosahují nejhorších výsledků “crisp” pravidlově orientované systémy, které provádějí inferenci pouze s ostře nadefinovanými pravidly “IF - THEN” a hodí se pro deterministické situace, jak uvádí ve své porovnávací studii Moreno a Espejo, (2015). Produkční pravidla lze dále modifikovat dalšími sofistikovanějšími metodami, jakými jsou Bayesovské inferenční sítě (Jakeman et al., 2006; Chen a Pollino, 2012; Moreno a Espejo, 2015). Dále je možno využít Dempster Shaferovu (DS) teorii, která dává alternativu k pravděpodobnostním přístupům (Bayesovským sítím).

Umožňuje reprezentaci nevědomosti a součet měr domnění nemusí být roven jedné. Tomuto přístupu k zpracování neurčitostí v inferenci dal základ Dempster (1967).

4.4 Znalosti a expertní systémy v prostředí neurčitosti

Aby došlo k zpřesnění modelovaného reálného problému, který expertní systém pomáhá řešit, doplňují se produkční pravidla různými formami strategií pro řešení problémů způsobených neurčitostí. Například Venturelli et al. (2017) využívá fuzzifikace pravidel. Další možností jsou Bayesovké sítě a doplnění podmíněných pravděpodobností. Dalším způsobem zpracování neurčitosti je Dempster Shaferova teorie (Shafer, 1976) a Dempsterovo kombinační pravidlo (Dempster's rule of combination). Tuto teorii využívá Calderwood et al. (2017) pro zpřesňování podávaných informací, které poskytují sensory v SCADA[7] systému pomocí kombinačního pravidla (Shafer, 2016). Toto pravidlo obecně říká, že pokud existuje více měr důvěry, například hodnocení jedné události více experty, pak lze tyto jednotlivé míry důvěry sloučit a dostat tak celkovou míru důvěry. Faktory jistoty představují také možné zobrazení neurčitostí. Tato metoda (Garibaldi, 1997) je založena na přidruženém faktoru jistoty s hodnotami v rozmezí od -1 do +1, kde hodnota -1 reprezentuje přesvědčení, že hypotéza je zcela falešná a naopak +1 reprezentuje přesvědčení, že hypotéza je zcela pravdivá. Faktor jistoty je vypočítán jako rozdíl dvou ohodnocení aktuální míry přesvědčení (MB) a aktuální míry nepřesvědčení (MD). Přehled způsobů pro zpracování neurčitosti je podle Garibaldiho (1997) a Sonala et al. (2014) následující:

- Dempster-Shafer Theory of Evidence (modelování nejistoty pomocí domněnkových funkcí, tato teorie umožňuje, aby některé z pravděpodobností byly nepřiřazené.
- Certainty Factors (faktory jistoty) byly navrženy pro překonání obtíží s Bayesovskou teorií, která byla poprvé využita ES MYCIN (ES pro diagnostiku v medicíně).
- Bayesian Reasoning (Bayesovská teorie) je způsob použití teorie pravděpodobnosti v rozhodování expertního systému (Bayes, 1763).
- Possibility Theory (teorie možnosti), kterou definoval Zadeh (1978):

[7] SCADA systém. Kontrolní informační systém pro řízení a monitorování procesů na vysoké hierarchické úrovni. Například zde dostupné 8.1.2017 https://www.reliance.cz/cs/main.

"Intuitivně, možnost se týká našeho vnímání a stupně proveditelnosti nebo snadnosti dosažení, zatímco pravděpodobnost je spojena se stupněm pravděpodobnosti, víry, frekvence nebo poměru."

- Probability Theory (teorie pravděpodobnosti). Ta byla poprvé vytvořena v sedmnáctém století Pascalem a Fermatem, kteří studovali hazardní hry.
- Fuzzy Sets (fuzzy množiny), kvantový filozof Max Black (1937), který poprvé popsal klíčové pojmy pro zachycení neurčitého souboru. Fuzzy množiny byly v jejich současné podobě poprvé definovány Lotfi Zadehem (1965).

Složitost řešených situací expertním systémem vyžaduje kombinaci různých metod pro podporu inferenčního mechanismu s dalšími metodami, čímž vznikají hybridní expertní systémy, které se liší v architektuře řešení. Fakhrahmad et al. (2015) například využívají dopředné řetězení a datamining k snižování nejednoznačnosti při automatickém překladu. Dalším příkladem hybridního expertního systému je expertní systém Nilashiho et al. (2015), tento znalostně orientovaný expertní systém kombinuje metodu fuzzy inference a metodu AHP (analytický hierarchický proces) nad databází faktů a pravidel. Tento expertní systém je navržen pro posuzování úrovně energetických náročností ekologických budov takzvaných „green buildings“ na základě hodnocení definovaných faktorů.

4.4.1 Inference znalostí v expertních systémech

Bayesovský princip inference využívá teorii pravděpodobnosti pro nalezení podmíněné pravděpodobnosti úspěchu na základě k úspěšných a (n - k) neúspěšných pokusů při pozorování n pokusů. Tento přístup je rozšířením problému Jacoba Bernoulliho, který přidělil pravděpodobnost úspěchu v každém pokusu (Debenath a Basu, 2014). Bayes (1763) představil základní představu podmíněné pravděpodobnosti $P(A|B)$ události A, vzhledem k tomu, že událost B nastala, následovně:

$$P\,(A|B) = \frac{P\,(A \cap B)}{P\,(B)}, P\,(B) \neq 0, \tag{4}$$

nebo

$$P\,(A \cap B) = P(A|B)\,P(B). \tag{5}$$

Tato definice je známa jako zákon pro určení pravděpodobnosti, kdy nastane každá ze dvou událostí *A* a *B*. Použití tohoto multiplikačního zákona však neurčuje pouze pravděpodobnost, že se vyskytnou obě události. Protože $P\ (A \cap B) = P\ (B \cap A)$, z tohoto vyplývá následující tvrzení:

$$P(B \cap A) = \ P(B|A)\ P(A) = P(A|B)\ P(B), \tag{6}$$

a to vede k Bayesově teorému:

$$P(A|B) = \frac{P(B|A)\ P(A)}{P\ (B)}. \tag{7}$$

Z tohoto vyplývá vztah podmíněné (neboli inverzní) pravděpodobnosti. Ve skutečnosti také poskytuje přímý vztah mezi a-posteriori (pozdější) pravděpodobností $P\ (A|B)$ a a-priori (dřívější, předešlou) pravděpodobností $P\ (A)$ přes pravděpodobnostní funkci $P\ (B|A)$. Jinými slovy, a-priori pravděpodobnost $P\ (A)$ je kombinována s pravděpodobnostní funkcí pro určení a-posteriori pravděpodobnosti $P\ (A|B)$ (Debenath a Basu, 2014). Tato teorie pravděpodobnosti vytvořila podklad pro řešení neurčitosti v novodobých expertních systémech.

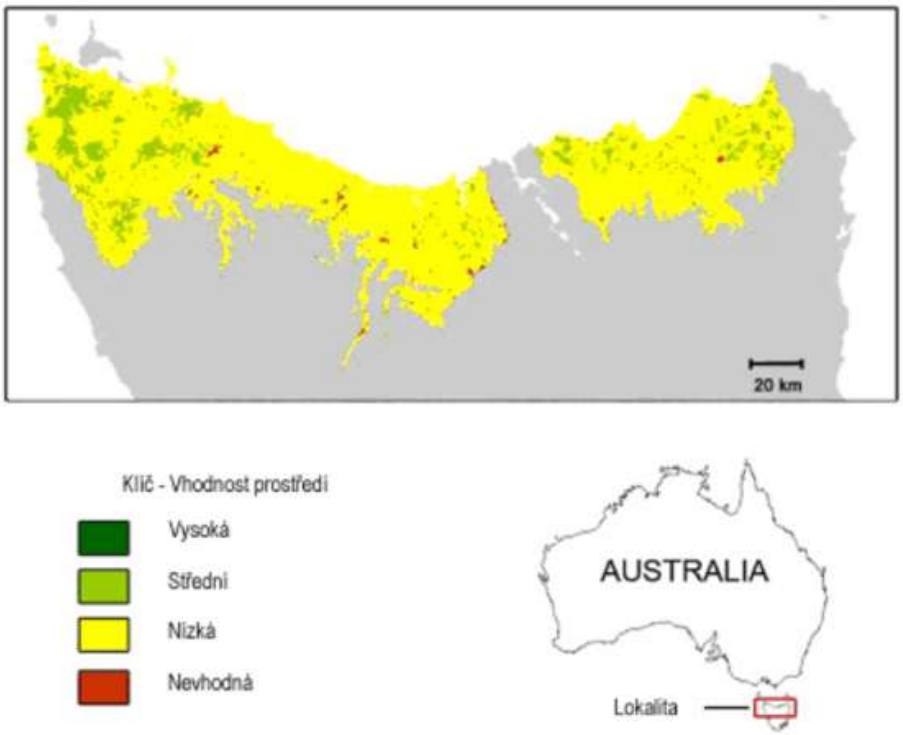
Výkonnost expertních systémů na bázi Bayesovské inference porovnávali Moreno a Espejo (2015), porovnávali výsledky tří typů inference v expertním systému na jednom praktickém příkladu – Diagnostiky lomů kovu. Inferenční mechanismy byly založeny na následujících základech:

- Crisp pravidlech,
- Fuzzy pravidlech,
- Bayesovské inferenci.

Toto srovnání bylo vyhodnoceno podle indexů pro srovnání inferenčních mechanismů, které navrhli (Moreno a Espejo, 2015). Podle těchto indexů lze vyhodnocovat vhodnost typu inferenčního mechanismu pro daný expertní systém. Z tohoto vyhodnocení vychází nejlépe Bayesovský princip inference, jen těsně za ním je fuzzy inference. Autoři proto pravidly orientovaný inferenční mechanismus nedoporučují vzhledem k nedeterministické povaze

problému. V tomto případě je vhodnější Bayesova a fuzzy inference. Např. také software HUGIN využívá Bayesovu inferenci a (Bromley et al., 2005).

Konkrétní uplatnění uvádí Chen a Pollino (2012) na příkladu expertního systému, ve kterém je inference zajišťována Bayesovskou inferenční sítí a výsledky této inference jsou poté zobrazovány pomocí prostorového GIS softwaru. V tomto případě je pak výsledkem mapa vhodného biotopu pro Juvenile Astacopsis gouldi[8]. Mapa, která je výsledkem celého procesu, je na obrázku 20.



Obrázek 20 Zobrazení výsledků; zdroj: Chen a Pollino, 2012

4.4.2 **Dempster Shaferova teorie**

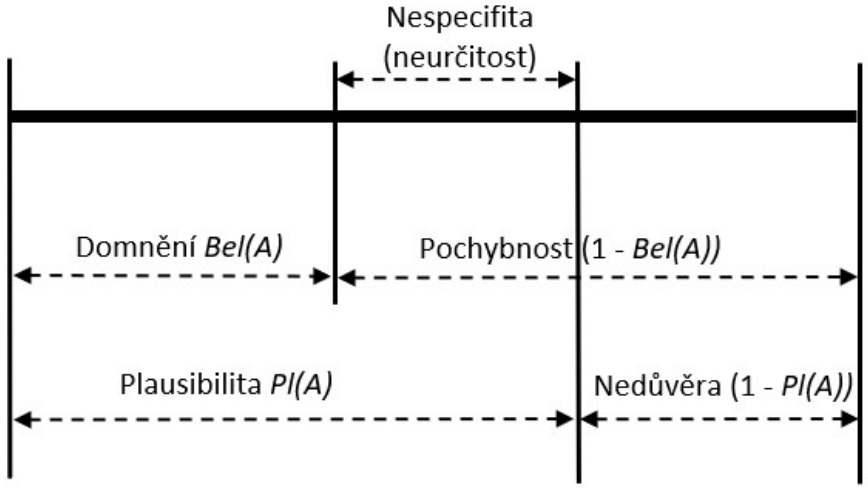
Alternativou k Bayesovu a fuzzy přístupu je Dempster-Shaferova teorie (Dempster, 1967, Shafer, 2016). Tato teorie pracuje s měrami domnění (believe) a důvěry (věrohodnosti - measure of plausibility) hypotéz a zároveň nevyžaduje, aby součet pravděpodobnosti hypotézy a její negace byl roven 1. Tím umožňuje reprezentaci nevědomosti. Tomuto přístupu k zpracování neurčitostí v rámci expertních systémů dal základ Dempster (1967).

[8] Juvenile Astacopsis gouldi - giant freshwater crayfish of Tasmania (obří sladkovodní rak)

Shafer (2016) uvádí spolu s definicí a důkazem popis Dempsterova kombinačního pravidla takto:

„Dempsterovo pravidlo kombinace je pravidlem kombinace dvou funkcí domnění f1 a f2 (belief functions), které jsou definovány na stejné množině hypotéz Θ, abychom získali novou funkci domnění Pjako ortogonální součet $f1 \oplus f2$*.“*

Míra domnění vyjadřuje míru přesvědčení o hypotéze A, její negací je míra pochybnosti. Míra domnění o nepravdivosti hypotézy A touto negací není. Míra věrohodnosti vyjadřuje, nakolik je důvěryhodná hypotéza A, jestliže i všechna neznámá fakta by hypotézu A podporovala (Beránek, 2010), schematicky jsou tyto míry zachyceny na obrázku 21.



Obrázek 21 Intervaly domnění; zdroj: Beránek, 2010

Tato teorie umožňuje kombinovat základní míry domnění pomocí operátoru slučování (kombinace). Operátor slučování je ortogonální suma, která se značí $\oplus$ a funguje následujícím způsobem:

Předpokládejme, že m_1 a m_2 jsou domněnkové funkce na množině hypotéz Θ. Pak lze tyto obě domněnkové funkce m_1 a m_2 na 2^{Θ} sloučit (kombinovat). Výsledná domněnková funkce $m_{12}(C) = m_1 \oplus m_2$ je definována vzorcem Dempsterova kombinačního pravidla (Beránek, 2010)

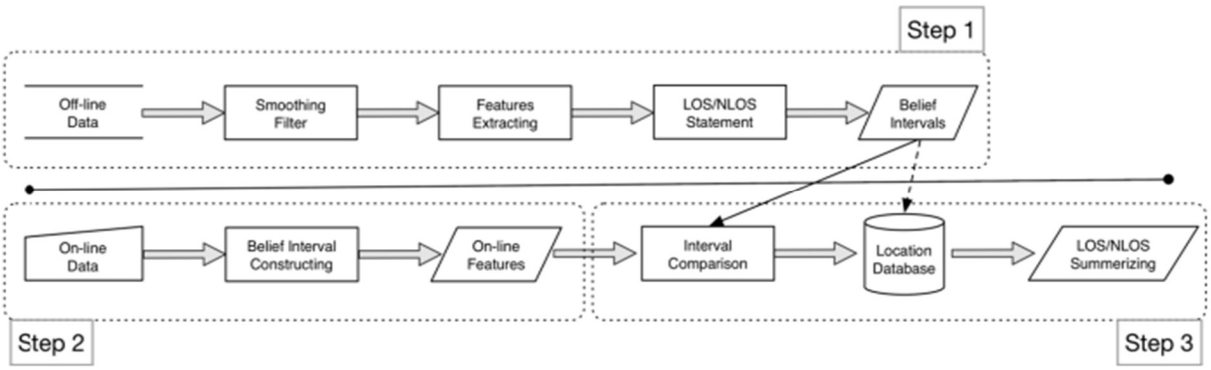
$$m_1(C) \oplus m_2(C) = m_{12}(C) = \frac{H}{K} \tag{8}$$

kde

$$H = \sum_{A,B:A\cap B=C} m_1(A).m_2(B)$$
$$K = 1 - \sum_{A,B:A\cap B=\emptyset} m_1(A)\cdot m_2(B) \quad (9)$$

Tento způsob zpracování neurčitosti například využívá Calderwood et al. (2017) a popisuje způsob zpřesňování podávaných informací, které poskytují sensory v SCADA systému pomocí Dempsterova kombinačního pravidla. Dempsterovo pravidlo (Shafer, 2016) obecně říká, že pokud existuje více měr důvěry, například více hodnocení expertů jedné události, pak lze tyto jednotlivé míry důvěry sloučit a dostat tak celkovou míru důvěry. Pokud hodnoty funkcí důvěry byly získány z nespolehlivého zdrojepomocí kombinačního pravidla zpracování nespolehlivých, neúplných informacílze získat informaci spolehlivou.

Dempster Shaferovu teorii lze využít v softwarových aplikacích, které pracují se signálem z různých sensorů, například pro identifikaci polohy uvnitř objektu. Wu et al. (2017) využívá DS teorie kvůli její schopnosti odvodit pravděpodobnost množiny hypotéz, zatímco klasická teorie pravděpodobnosti je stanovena pro jedinou hypotézu. Dále ji vyžívá i proto, že dovoluje systému pracovat s nepřesností a nejistotou. Jako největší výhodu autoři pro navrhovaný systém uvádějí, že teorie DS se dokáže vypořádat s chybějícími informacemi. To znamená, že je schopna popsat nejistou podmínku v určitém prostředí a je flexibilnější než tradiční metody pravděpodobnosti. Architektura systému pro zjištění polohy je na obrázku 22 (Wu et al., 2017). Na schématu jsou v prvním kroku procedury, které se následně aplikují na data snímané ze senzorů (on-line data) v dalších krocích. DS teorie je použita na spřensnění dat ze senzorů.



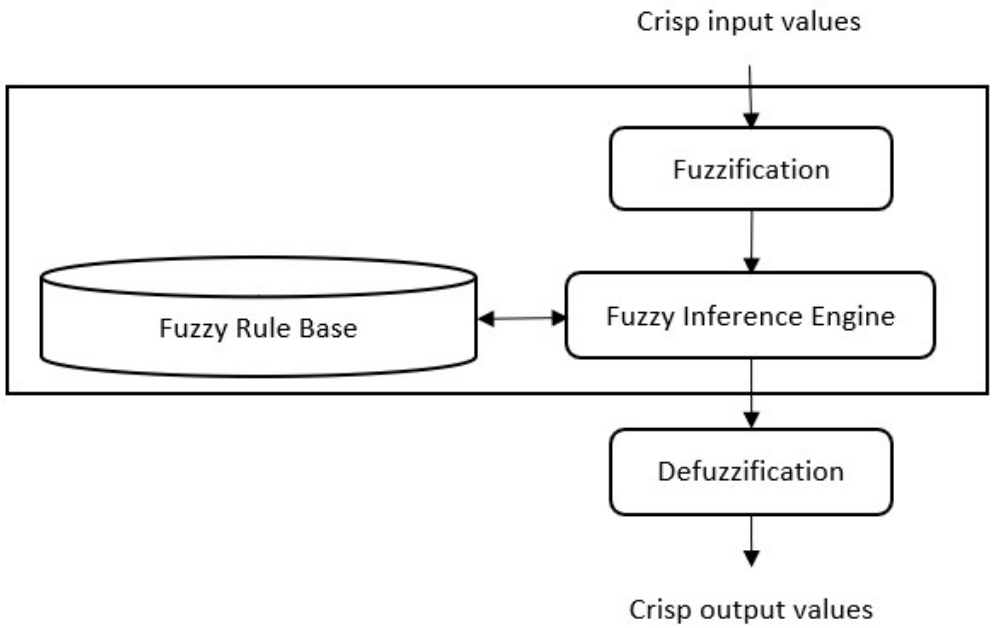
Obrázek 22 Architektura systému; zdroj: Wu et al., 2017

4.5 Fuzzy expertní systémy

Fuzzy nástroje jsou v kombinaci s dalšími metodami základem velké množiny expertních systémů, protože mají vysokou vypovídací hodnotu. Fuzzy množiny, jak uvádí Kaur et al. (2016), jsou velmi vhodnou metodou pro reprezentování určité formy nejistoty. Z hlediska tvorby FES pro konkrétní aplikační doménu je podstatné vhodně zvolit architekturu celého řešení a implementační strategie. Podle Garibaldiho (1997) jsou hlavními atributy, které je nutné zvolit:

- fuzzy lingvistické proměnné a fuzzy termy,
- funkce příslušnosti,
- množina pravidel fuzzy expertního systému,
- fuzzy operátory,
- inferenční metodologie.

Architektura FES má základ ve fuzzy pravidlech a je zobrazena schématem na obrázku 23. Definovaná fuzzy pravidla jsou poté vyhodnocována fuzzy inferenčním mechanismem a následně defuzzifikována.



Obrázek 23 Fuzzy expertní systém; zdroj: Kaur et al., 2016

Podle Kaura et al. (2016) je možné také zvyšováním počtu pravidel ve fuzzy logice dosáhnout přesnějších výsledků, a tím pádem docílit i efektivnějšího rozhodování. Pro ilustraci bude uvedeno jedno pravidlo zpracované rozdílným přístupem a to fuzzy, prostým pravidlem a pomocí pravděpodobnosti.

Moreno a Espejo (2015) uvádí příklad fuzzy pravidla pro fuzzy inferenční mechanismus při sestavení pravidel pro únavový lom kovu následovně: definují „únavová skluzová pásma (beach marks)“ a ty mohou mít následující vyhodnocení:

“absence, mírná přítomnost nebo jsou evidentně přítomna”,

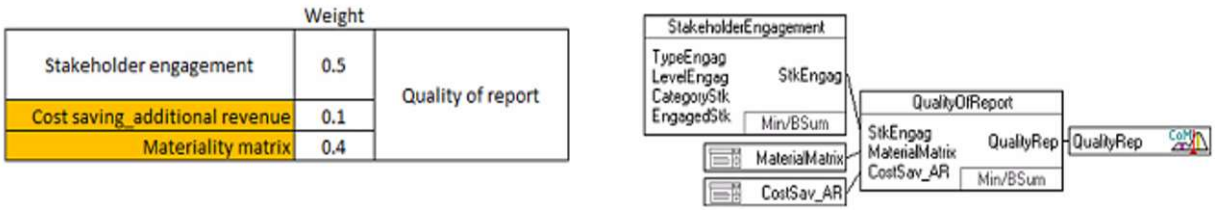
což znamená, že výskyt daného jevu může být expertem definován vágně, „neurčitě“ s přiřazením funkce příslušnosti. Naproti tomu pravidlově orientovaný expertní systém je vyhraněný na prosté If-Then:

“If únavová skluzová pásma Then únavový lom”.

Pro Bayesovské sítě jsou definovány pravděpodobnosti daného jevu a případně podmíněné pravděpodobnosti jevu:

“Pokud jsou naplněny dvě podmínky, a to plynulý a konečný lom, pravděpodobnost ohýbání a únavového lomu je 75% a pokud jsou na povrchu také únavová skluzová pásma, zvedá se pravděpodobnost na 85%”.

Venturelli et al. (2017) uvádí v prvním kroku tvorby FES definici bloků fuzzy pravidel a klade velký důraz na kvalitu expertů v dané problematice a také na kvalitu zachycení fuzzy pravidel v podobě funkcí příslušnosti. Na tomto příkladu uvádí tabulku fuzzy produkčních pravidel pro vyhodnocení kvality reportu, kde probíhá fuzzifikace v předpokladové (IF) „Antecedent“ části podle vah určených odborníky a v důsledkové části (THEN) „Konsekvent“. Svůj postup uvádí v softwaru „FuzzyTech“. Tento postup je zachycen schématem na obrázku 24.



Obrázek 24: Fuzzy pravidla; zdroj: Venturelli et al., 2017

V tabulce potom doplňuje k tomuto příkladu blok fuzzy pravidel ze software „FuzzyTech“ pro jeden uzel s hodnotou DoS, která představuje individuální významnost pravidla. Kvalita reportu = Quality of report, viz tabulka 3.

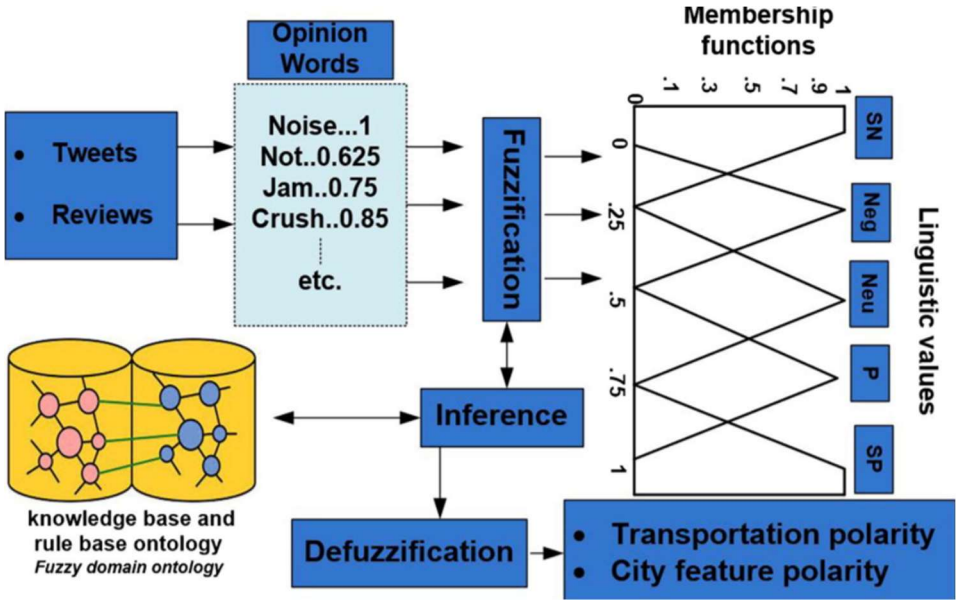
	IF			Then	
	StkEngang	MaterilMatrix	CostSav_AR	DoS	QualityRep
1	very_low	False	False	1.00	very_low
2	very_low	False	True	1.00	low
3	very_low	True	False	1.00	medium_low

Tabulka 3 Blok pravidel pro pomocné proměnné; zdroj: Venturelli et al., 2017

FES a fuzzy pravidla využívají Venturelli et al. (2017) ve svém expertním systému na posuzování firmy z hlediska CSR indexů „*Social Responsibility index*“. Venturelli et al. (2017) se obdobně jako Moreno a Espejo (2015) přiklání k fuzzy a Bayesově inferenci, a to kvůli nedeterministické povaze problému a možnosti modelování neurčitosti pomocí pravidel. Nilashi et al. (2015) zdůvodňují výběr a použití fuzzy inference z důvodu zachycení neurčitost lidského rozhodování a opírají se i o další autory (Facchinetti et al., 2001; Magni et al., 2006; Marchi et al., 2014).

Ali et al. (2017) vytvořili framework[9] fuzzy inferenční vrstvy modelu pro hodnocení dopravy ve městě, která je založena na znalostně pravidlové ontologii, (obrázku 25). Fuzzy inferenční vrstva vyhledává fráze hodnotící dopravu a integruje je za účelem stanovení relevance frází pro vyhodnocení bezpečnosti dopravy ve městě.

[9] Framework = konstrukce pro vytvoření programu http://www.vyznam-slova.com/framework



SP, P, Neu, Neg and SN stand for strong positive, positive, neutral, negative and strong negative

Obrázek 25 Framework fuzzy inferenční vrstvy; zdroj: Ali et al., 2017

4.5.1 **Fuzzy logika**

Fuzzy logika byla představena společně s teorií fuzzy množin v roce 1965 Lotfi Zadehem (Zadeh, 1965). Navazuje na vícehodnotovou logiku[10], která byla definována Janem Lukasiewiczem ve dvacátých letech devatenáctého století (Simons, 2014).

Fuzzy logiku popisuje Zadeh (1975) následovně:

" Fuzzy logika (FL) se v používá k popisu nepřesného logického systému, ve kterém pravdivostní hodnoty jsou fuzzy podmnožiny jednotkového intervalu s jazykovými proměnnými jako pravdivé, falešné, nepravdivé, velmi pravdivé, zcela pravdivé, ne moc pravdivé a ne příliš falešné atd. FL předpokládá, že pravdivostní hodnoty množiny T jsou generovány bezkontextovou gramatikou se sémantickými pravidly, která poskytují prostředky pro výpočet významu každé jazykové hodnoty v T jako fuzzy podmnožiny [0, 1]."

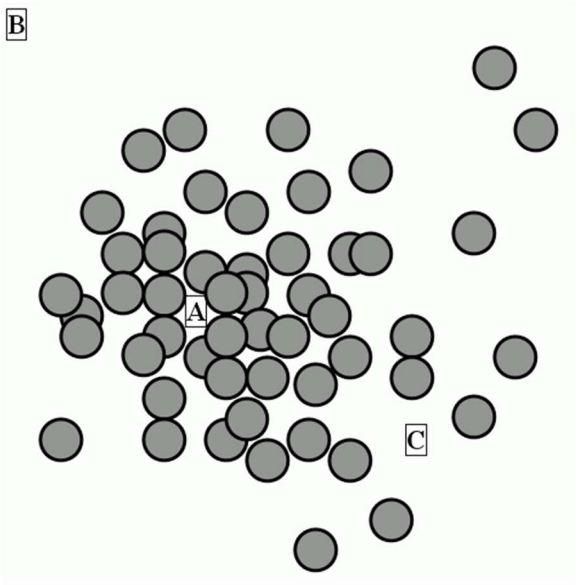
Novák et al. (1999) definuje fuzzy logiku následovně:

"Fuzzy logika je forma vícehodnotové logiky, ve které pravdivostní hodnoty proměnných mohou být jakékoli reálné číslo mezi 0 a 1. Používá se k vyrovnávání konceptu

[10] More than Three Values logic = Vícehodnotová logika. In proposing logics with infinitely many values, Łukasiewicz was thus the inventor of what was much later (43 years later, to be exact) to be called 'fuzzy logic'.

částečné pravdy, kde se hodnota pravdy může pohybovat mezi zcela pravdivý a zcela nepravdivý."

V kontextu binární logiky (0 - 1) mohou být pravdivostní hodnoty proměnných jen celá čísla, tedy 0 nebo 1. Fuzzy logika tedy umožňuje využívat neurčité výroky, jak je například zachyceno na obrázku 26. Jedná se o ptačí pohled na les s otázkou : Kde je hranice lesa? Která lokalita označená písmenem je v lese a která mimo? (Mareš, 2006) Lokalita A je jistě v lese, lokalita B jistě není v lese a lokalita C možná je, možná není v lese.



Obrázek 26 Ptačí pohled; zdroj: Mareš, 2006

V současné době je fuzzy logika aplikována v mnoha oborech, od teorie řízení po umělou inteligenci. Základními pojmy ve fuzzy logice jsou:

- fuzzy množiny,
- funkce příslušnosti,
- lingvistické proměnné.

Fuzzy množiny

Definice fuzzy množiny pochází od Lotfiho Zadeha.

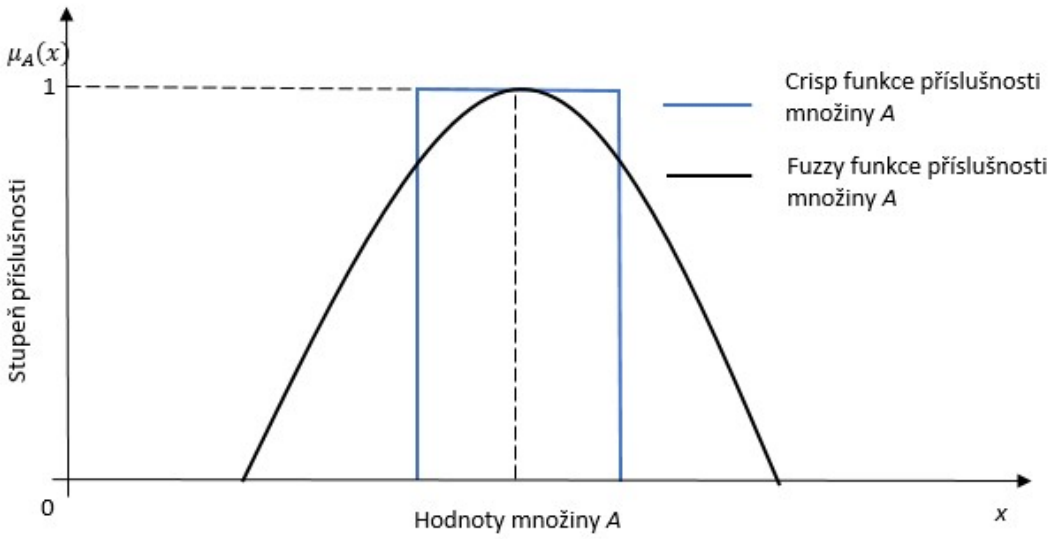
Fuzzy nmnožinu a funkci příslušnosti definoval Zadeh (1965) následovně:

"Funkce příslušnosti pro fuzzy množinu A v universu X je definována jako

$$\mu_A: X \rightarrow \langle 0; 1 \rangle, \tag{10}$$

která každému prvku univerza X přiřazuje hodnotu mezi 0 a 1. Tato hodnota je nazývaná hodnotou příslušnosti nebo stupněm příslušnosti, kvantifikuje stupeň členství prvku univerza X k fuzzy množině A."

Pojem fuzzy množiny je zobecněním pojmu klasické množiny, neboť klasickou množinu *A* lze definovat stejným způsobem s tím, že $\mu_A: U \rightarrow \{0; 1\}$. V grafu na obrázku 27 je uveden rozdíl mezi funkcí příslušnosti klasické množiny a funkcí příslušnosti fuzzy množiny: Množina A obsahuje všechny hodnoty z intervalu $\langle a_1; a_2 \rangle$, funkce příslušnosti těchto čísel je rovna 1, fuzzy množina A může být popsána otázkou: Jak moc číslo x může být viděno jako číslo z intervalu $\langle a_1; a_2 \rangle$? Čím je číslo x větší nebo menší, tím spíše ho do fuzzy intervalu nezařadíme, hodnoty funkce příslušnocti tedy klesají. Funkce příslušnosti vhodně zobrazuje neurčitost odpovědi na danou otázku.



Obrázek 27 Klasická a fuzzy funkce příslušnosti; zdroj: autor

Funkce příslušnosti

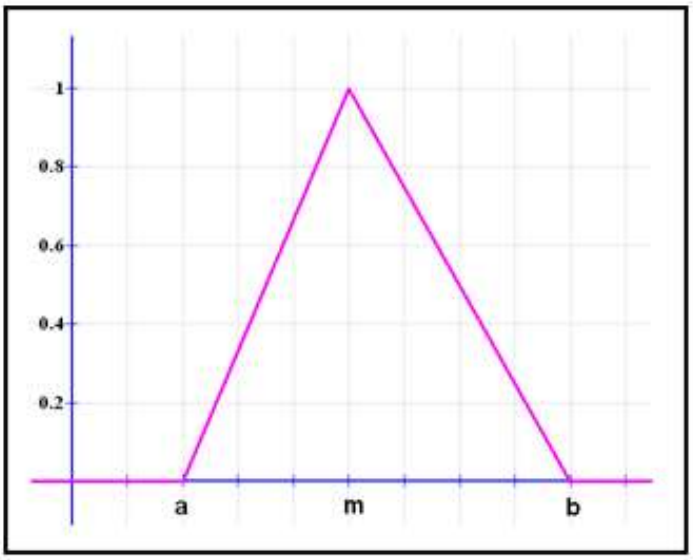
Pro odhad nebo návrh funkce příslušnosti k fuzzy množině existují různé metody, jedná se především o metody založené na heuristice, pravděpodobnosti možných transformací, histogramech, metody nejbližších sousedů, neuronových sítí, shlukování (clustering) a

dekompoziční metody (Medasani et al., 1998). Návrh funkce příslušnosti pro konkrétní příklad z určité znalostní domény může být založen na využití obecných algoritmů, které hlavně slouží k optimalizaci funkcí příslušnosti, nebo funkce příslušnosti může být vytvořena experty ve spolupráci se znalostními inženýry. Návrh funkce příslušnosti z trénovacích dat je jedním ze základních problémů spojených s aplikací teorie fuzzy množin (Medasani et al., 1998). Obecně nejsou k dispozici žádné pokyny nebo pravidla, techniky generování, která mohou být použita k výběru funkce příslušnosti. Dalším problémem, který činí úkol generování funkcí příslušností netriviálním, je nedostatek konsenzu ohledně definice a interpretace funkcí příslušnosti.

Při výběru vhodného typu funkce příslušnosti je dobré brát v úvahu několik hledisek, například výpočetní složitost funkce v kontextu přidané hodnoty v řešení, podpora výpočtu v software a jiné. Vybrané typy funkcí příslušnosti jsou uvedeny také v kapitole „4.5.3 Fuzzy expertní systém v MATLABu“.

Velmi často užívané typy funkcí příslušnosti jsou trojúhelníková a trapezoidní funkce (eMathTeacher, 1999). Trojúhelníková funkce příslušnosti je definovaná třemi hodnotami, dolní hranicí *a,* horní hranicí *b* a hodnotou *m*, kde $a < m < b$ (viz obrázek 28).

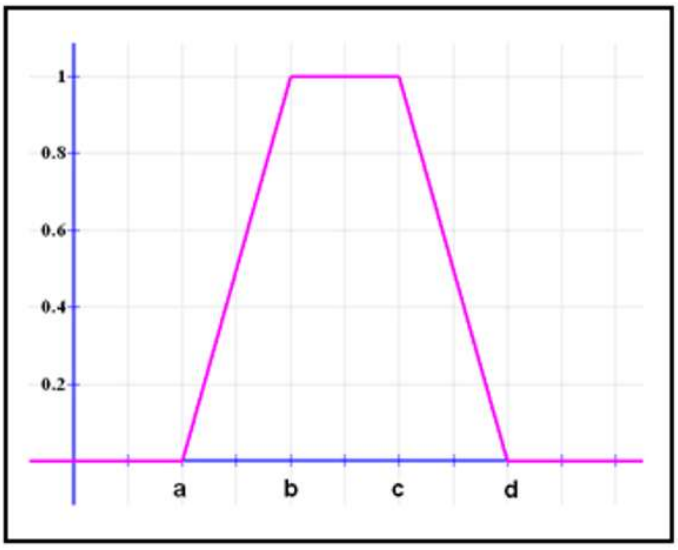
$$\mu_A(x) = \begin{cases} 0, & x \leq a \\ \dfrac{x-a}{m-a}, & a \leq x \leq m \\ \dfrac{b-x}{b-m}, & m \leq x \leq b \\ 0, & x \geq b \end{cases} \tag{11}$$



Obrázek 28 Trojúhelníková funkce; zdroj: eMathTeacher, 1999

Trapezoidní funkce příslušnosti je definována čtyřmi hodnotami (a, b, c, d), kde a je dolní hranice, d horní hranice, a b a c jsou mezemi jádra fuzzy množiny, $a < b < c < d$ obrázek 29.

$$\mu_A(x) = \begin{cases} 0, (x < a) or\ (x > d) \\ \frac{x-a}{b-a}, a \leq x \leq b \\ 1, b \leq x \leq c \\ \frac{d-x}{d-c}, c \leq x \leq d \end{cases} \quad (12)$$

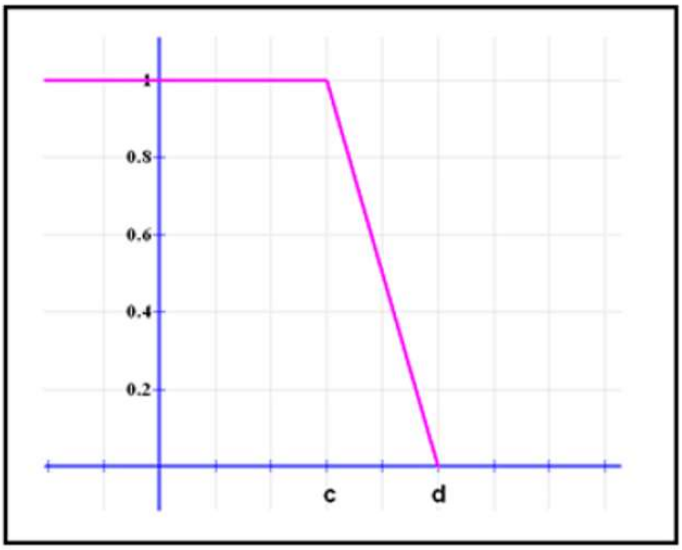


Obrázek 29 Trapezoidní funkce; zdroj: eMathTeacher, 1999

K této funkci existují ještě dva speciální podtypy, které se nazývají trapezoidní funkce L a R, jsou uvedeny v grafu na obrázku 30 a 31.

Trapezoidní funkce R je definovaná parametry $a = b = -\infty$.

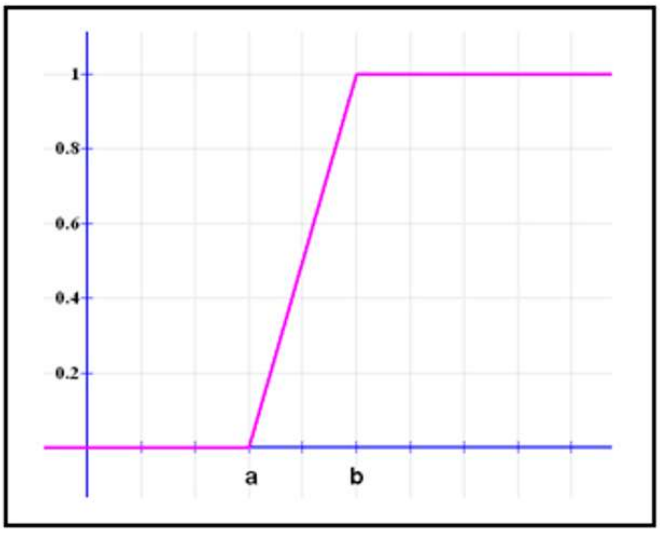
$$\mu_A(x) = \begin{cases} 0, x > d \\ \frac{d-x}{d-c}, c \leq x \leq d \\ 1, x < c \end{cases} \quad (13)$$



Obrázek 30 Trapezoidní funkce R; zdroj: eMathTeacher, 1999

Trapezoidní funkce L je definovaná parametry $c = d = +\infty$.

$$\mu_A(x) = \begin{cases} 0, x < a \\ \frac{x-a}{b-a}, a \leq x \leq b \\ 1, x > b \end{cases} \quad (14)$$



Obrázek 31 Trapezoidní funkce L; zdroj: eMathTeacher, 1999

Dále jsou uvedeny vybrané charakteristiky fuzzy množin podle Nováka et al. (2016) a Dvořáka (2004):

Nosič (support) fuzzy množiny *A* je klasická množina

$$supp(A) = \{x|\mu_A(x) > 0\}, \tag{15}$$

Jádro (kernel, core) fuzzy množiny *A* je klasická množina

$$ker(A) = \{x|\mu_A(x) = 1\}, \tag{16}$$

Výška (height) fuzzy množiny *A* je definována takto

$$hgt(A) = sup\{\mu_A(x)|x \in X\}, \tag{17}$$

Řez (cut) fuzzy množiny *A* je definován takto, nechť $\alpha \in [0,1]$ potom

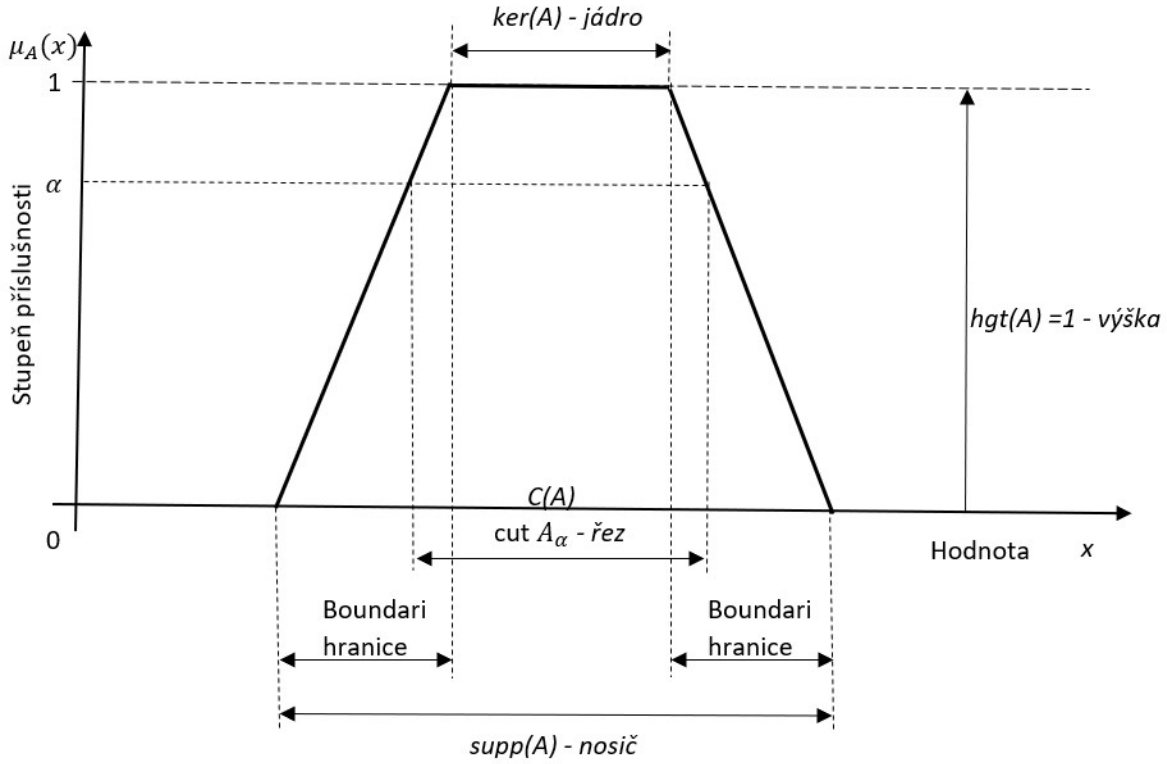
$$A_\alpha = \{x|x \in X, \mu_A(x) \geq \alpha\}, \tag{18}$$

Hranice (boundary) fuzzy množiny *A* jsou definovány takto (Ross, 2010)

$$boundary(A) = 0 < \mu_A < 1. \tag{19}$$

Hranice fuzzy množiny *A* jsou části univerza, které obsahují prvky s hodnotou funkce příslušnosti jsou menší než 1 (Ross, 2010).

Tyto charakteristiky jsou uvedeny v grafu na obrázku 32 v kontextu fuzzy funkce příslušnosti typu trapezoid.



Obrázek 32 Vybrané charakteristiky fuzzy množin; zdroj: autor

Pro využití fuzzy množin, fuzzy čísel a fuzzy logiky ve FES je nutné definovat vhodné operace s těmito prvky a fuzzy inferenci podle Mamdani a Assilian (1975).

- Sjednocení fuzzy množin

  Sjednocení fuzzy podmnožin A a B, které označili $A + B$ definovali následovně:

$$A + B = \sum_{i=1}^{n} \mu_A(u_i) \vee \mu_B(u_i), \tag{20}$$

  kde $\vee$ je maximum. Sjednocení odpovídá logické spojce „nebo“.

- Průnik fuzzy množin

  Průnik fuzzy podmnožin A a B, které označili $A \cdot B$ a definovali následovně:

$$A \cdot B = \sum_{i=1}^{n} \mu_A(u_i) \wedge \mu_B(u_i), \tag{21}$$

  kde $\wedge$ je minimum. Průnik odpovídá logické spojce „a“.

- Doplněk množin

Doplněk množiny *A*, který je označen ¬*A* a definován následovně:

$$\neg A = \sum_{i=1} 1 - \mu_A(u_i), \tag{22}$$

kde ¬ je negace. Negace odpovídá logickému NOT.

4.5.2 Nástroje fuzzy expertního systému

Lingvistické proměnné a fuzzifikace

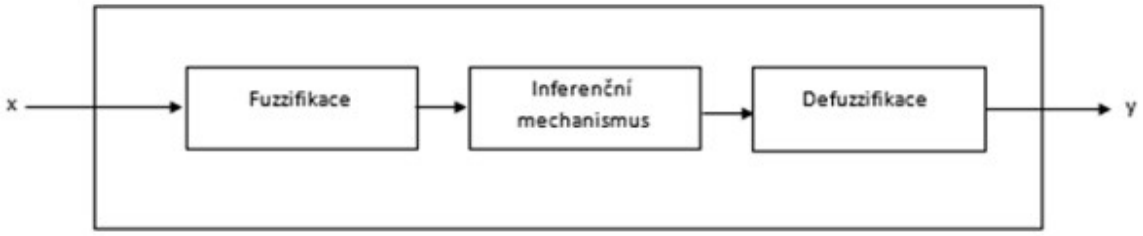
Lingvistické proměnné nabývají slovních hodnot na rozdíl od matematických proměnných, které jsou číselné (Zadeh et al., 1996; Zadeh 1999). Je tak umožněno vyjádření neurčitosti. V aplikacích fuzzy logiky se často používají nečíselné hodnoty pro usnadnění vyjádření pravidel a faktů. Fuzzy lingvistické proměnné jsou obecně definovány jako uspořádaná pětice (Dvořák, 2004):

$$X = (M, G, U, T, X), \tag{23}$$

kde *X* je název proměnné, *M* je množina sémantických pravidel, která interpretuje hodnoty z *T* jako fuzzy množiny na univerzu *U* (neprázdná klasická množina), *G* je množina syntaktických pravidel, pomocí nichž jsou generovány hodnoty z *T*. *T* je množina termů (tj. slovních hodnot, které může lingvistická proměnná nabývat).

Fuzzy pravidlově orientovaný systém

Základní strukturu fuzzy systému na vysokém stupni abstrakce uvádí Lilly (2010) ve schématu na obrázku 33. V tomto schématu jsou vstupy *x* crisp hodnoty a výstupy *y* jsou také crisp hodnoty. Vstupy a výstupy nemusí mít pouze podobu crisp hodnot, mohou být fuzzy. Například vstupy i výstupy v podobě fuzzy lingvistických proměnných (jazykově orientovaný fuzzy expertní systém), nebo v podobě vektoru fuzzy čísel $C_1', C_2', \dots C_m'$ jejímž hodnotám fuzzy vstupů je na základě fuzzy modelu při dané bázi pravidel generována výstupní fuzzy množina D^M případně po defuzzifikaci D^S (Talašová, 2003). Výstupy stejně jako vstupy mohou být fuzzy proměnné zpracované inferenčním mechanismem.



Obrázek 33 Struktura fuzzy systému; zdroj: Lilly, 2010

Ze základní struktury fuzzy systému vychází fuzzy pravidlově orientovaný systém.

- Fuzzy (pravidlově orientovaný) systém.

Nejspíš nejpoužívanější způsob reprezentace lidských znalostí je formulace do přirozených jazykových výrazů – pravidel typu (Ross, 2010):

„Když předpoklad (antecendent), potom důsledek (konsekvent)“.

Kanonická forma zápisu fuzzy pravidlově orientovaného systému (Ross, 2010) je následující.

$$\begin{aligned} &Rule\ 1: IF\ condition\ C^1, THEN\ restriction\ R^1 \\ &Rule\ 2: IF\ condition\ C^2, THEN\ restriction\ R^2 \\ &\dots \\ &Rule\ r: IF\ condition\ C^r, THEN\ restriction\ R^r \end{aligned} \tag{24}$$

Pravidly je myšlen výběr výstupu (omezení – restriction) na základě daného vstupu (podmínka – condition).

Fuzzy pravidlově orientovaný systém je velmi vhodný pro modelování komplexních systémů. Jedním z důvodů je využívání lingvistických proměnných, které jsou bližší lidskému uvažování. Tyto lingvistické proměnné mohou být reprezentovány fuzzy množinami s fuzzy funkcí příslušnosti (Ross, 2010).

Pro fuzzifikaci produkčních pravidel lze využít metodu odvozování, kterou uvádí Tsang a Yeung (1997) a navazují na Buchanana a Dudu (1983). Tsang a Yeung (1997) rozšiřují produkční pravidla na fuzzy pravidla a uvádějí způsob fuzzifikace pravidel. V pravidlech jsou využívány operátory, které jsou známé jako Zadehovy operátory:

Boolean	Fuzzy
AND(x,y)	MIN(x,y)
OR(x,y)	MAX(x,y)
NOT(x)	$1-x$

Tabulka 4 Zadehovy operátory; zdroj: Zadeh, 1973

Omezení v pravidlech jsou obecně modelována více fuzzy množinami a jejich relacemi. Omezení jsou obvykle interpretována jako výrazy nad lingvistickými proměnnými se základními spojkami „nebo“ a „a“. Definice základních typů omezení v pravidlech jsou uvedeny níže (Ross, 2010).

- Konjunkce (Conjunction) označována spojkou „a“ je definována standardní fuzzy operací průnik (Intersection) v předpokladové části pravidla a má následující tvar:

$$Když\ x\ je\ A^1\ a\ A^2\ \dots a\ A^L\ potom\ y\ je\ B^S. \tag{25}$$

Za předpokladu nové fuzzy podmnožiny A^S

$$A^S = A^1 \cap A^2 \cap \dots \cap A^L, \tag{26}$$

vyjádřené ve tvaru funkce příslušnosti

$$\mu_{A^S}(x) = min[\mu_{A^1}(x), \mu_{A^2}(x), \dots, \mu_{A^L}(x)], \tag{27}$$

podle definice standardního fuzzy průniku, může být pravidlo přepsáno jako

$$Když\ A^S potom\ \ B^S. \tag{28}$$

- Disjunkce (Disjunction) označována spojkou „nebo“ je definována standardní fuzzy operací sjednocení (Union) v předpokladové části pravidla a má následující tvar:

$$Když\ x\ je\ A^1\ nebo\ A^2\ \dots nebo\ A^L\ potom\ y\ je\ B^S. \tag{29}$$

Což lze přepsat jako

$$Když\ x\ je\ A^S\ potom\ y\ je\ B^S, \tag{30}$$

kde fuzzy množina A^S je definována jako

$$A^S = A^1 \cup A^2 \cup \dots \cup A^L, \tag{31}$$

$$\mu_{A^S}(x) = max[\mu_{A^1}(x), \mu_{A^2}(x), \dots, \mu_{A^L}(x)], \tag{32}$$

což odpovídá definici standardního sjednocení fuzzy množiny (Ross, 2010).

Většina fuzzy systémů pracuje s více než jedním pravidlem. Proto je potřeba definovat proces pro agregaci fuzzy pravidel. Proces agregace fuzzy pravidel stanovuje z dílčích důsledků (konsekventů) pravidel finální důsledek (konsekvent). Obdobně jako pro modelování základních pravidel jsou pro proces agregace pravidel dva základní způsoby a to následující.

- Konjunkce (Conjunction) označována spojkou „a“. Je definována standardní fuzzy operací průnik (Intersection) důsledkové části (konsekvent), *y* je vytvořen všemi jednotlivými důsledky (konsekventy) pravidel y^i, kde i = 1, 2, … *r*, jako

$$y = y^1 a\, y^2 a \ldots a\, y^r \tag{33}$$

  nebo

$$y = y^1 \cap y^2 \cap \ldots \cap y^r, \tag{34}$$

  který je definován funkcí příslušnosti

$$\mu_y(y) = \min\left(\mu_{y^1}(y), \mu_{y^2}(y), \ldots, \mu_{y^r}(y)\right), pro\ y \in Y. \tag{35}$$

- Disjunkce (Disjunction) označován spojkou „nebo“. Je definována standardní fuzzy operací sjednocení (Union) důsledkové části (konsekvent), *y* je vytvořen všemi možnými jednotlivými důsledky (konsekventy) pravidel y^i, kde i = 1, 2, … *r*, jako

$$y = y^1 nebo\ y^2 nebo \ldots nebo\ y^r \tag{36}$$

  nebo

$$y = y^1 \cup y^2 \cup \ldots \cup y^r, \tag{37}$$

  který je definován funkcí příslušnosti

$$\mu_y(y) = \max\left(\mu_{y^1}(y), \mu_{y^2}(y), \ldots, \mu_{y^r}(y)\right), pro\ y \in Y.$$

$$\mu_y(y) = \max\left(\mu_{y^1}(y), \mu_{y^2}(y), \ldots, \mu_{y^r}(y)\right), pro\ y \in Y. \tag{38}$$

Fuzzy inference

Dalším nástrojem v postupu tvorby fuzzy expertního systému je mechanismus fuzzy inference. Nejčastěji je využívána Mamdaniho inferenční mechanismus.

- Mamdaniho fuzzy inference

Mamdaniho fuzzy inference[11] je grafickou technikou fuzzy inference (Mamdani a Assilian, 1975). Algoritmus Mamdaniho fuzzy inference je popsán následujícím způsobem (Talašová, 2003):

Pravidlo 1: Jestliže X_1 je $A_{1,1}$ a … a X_m je $A_{1,m}$, pak Y je D_1

Pravidlo 2: Jestliže X_1 je $A_{2,1}$ a … a X_m je $A_{2,m}$, pak Y je D_2

……..……..

Pravidlo n: Jestliže X_1 je $A_{n,1}$ a … a X_m je $A_{n,m}$, pak Y je D_n

Pozorování: Jestliže X_1 je A'_1 a … a X_m je A'_m,

Závěr: Y je D' (39)

kde D' je jazyková aproximace fuzzy množiny

$$D' = \bigcup_{i=1}^{n} D'_i, \tag{40}$$

kde pro každé i=1, 2…, n je funkce příslušnosti fuzzy množiny D_i^M definována vztahem

$$\forall y \in \vee: D'_i(y) = min\{h_i, D_i(y)\}, \tag{41}$$

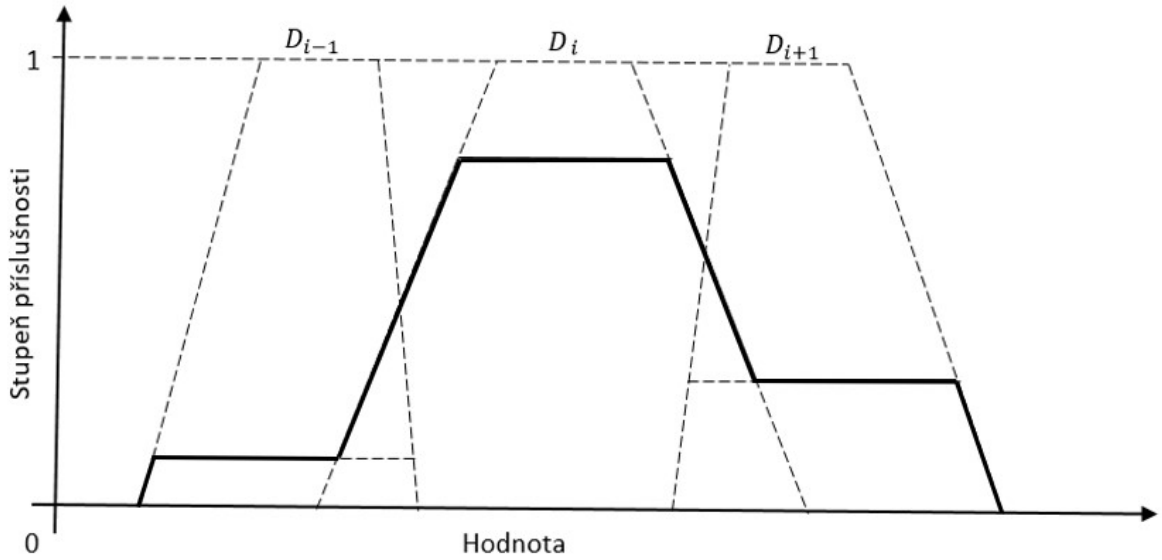
kde

$$\begin{aligned} h_i = hgt\left((A'_1 \times \ldots \times A'_m) \cap \left(A_{i,1} \times \ldots \times A_{i,m}\right)\right) = \\ = min\left\{hgt\left(A'_1 \cap A_{i,1}\right), \ldots, hgt\left(A'_m \cap A_{i,m}\right)\right\}. \end{aligned} \tag{42}$$

(Talašová, 2003)

[11] Mamdani fuzzy inference

Příklad grafického výstupu fuzzy Mamdaniho inference je na obrázku 34.



Obrázek 34 Typický tvar výstupu Mamdaniho fuzzy inference; zdroj: Talašová, 2003

Mamdani využil fuzzy logiku k převodu nepřesných intuitivních pravidel řízení pro nahrazení lidského operátora. Nahrazuje tak funkci lidského operátora strojově automatizovaným řízením. Fuzzy Mamdani inferenční mechanismus poskytuje prostředky pro vyjádření lingvistického pravidla v takové formě, aby mohlo být využito pro konkrétní řídící strategii.

Výstupní fuzzy množina při využití fuzzy Mamdani inference nemusí mít vždy výšku rovnu 1 a nemusí být nutně konvexní a není tedy zaručeno, že výstup je fuzzy číslem (Talašová, 2003).

Defuzzifikace

Podle Nováka et al. (2016) je defuzzifikace speciální operace, která transformuje fuzzy množinu na jeden jednotlivý prvek na takzvanou Crisp hodnotu. Defuzzifikace je operace, která přiřazuje neprázdné fuzzy množině prvek z jejího nosiče (Support), je to funkce:

$$DEF\ (A) \in supp(A) \tag{43}$$

platí pro všechny neprázdné fuzzy množiny (Novák et al., 2016).

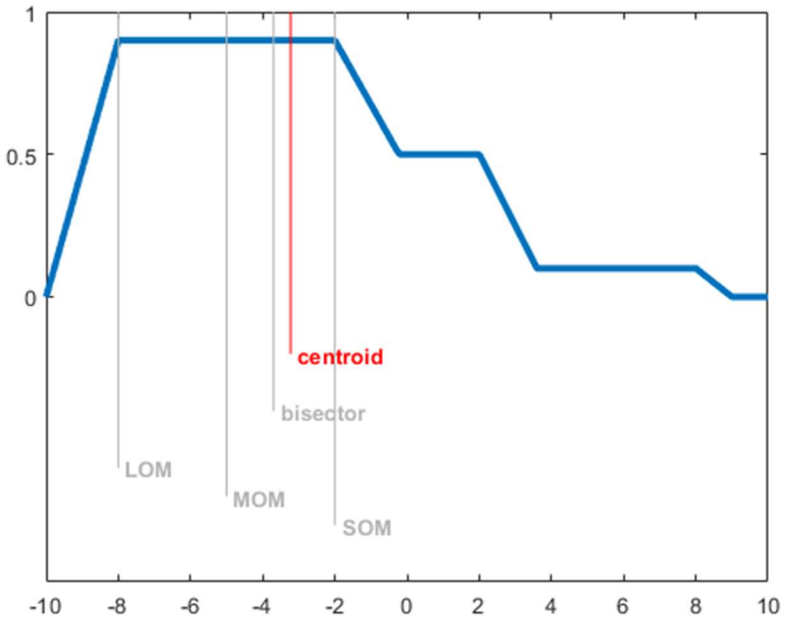
Existuje mnoho způsobů, jak lze tuto operaci provést. Pro defuzzifikace se například se využívá metoda Centroid Metoda Centroid vychází z plochy určené funkcí příslušnosti řešení (Area-based princip) a určuje těžiště této plochy (Keshwani et al., 2007).

Aplikace MATLAB Fuzzy Logic Toolbox nabízí více metod defuzzifikace jako je již zmiňovaná metodu Centroid, která vrací výsledek v podobě těžiště pod výslednou agregovanou funkcí příslušnosti. To znamená, že pokud je oblast pod křivkou deska o stejné hustotě, Centroid je bod *c*, ve kterém je těžiště.

Dalšími metodami jsou například:

- Metoda Bisector hledá bod, který rozdělí plochu pod výslednou agregovanou funkcí příslušnosti na dvě podoblasti se stejnou plochou.
- Metody Middle, Smallest, and Largest of Maximum (MOM, SOM a LOM) ukazují rozsah maximální hodnoty ve výsledné agregované funkci příslušnosti.
    - Pokud má funkce příslušnosti jedinečné maximum, potom jsou MOM, SOM, a LOM v jediném bodě.

Výše popsané způsoby defuzzifikace (Centroid, Bisector, MOM, SOM a LOM) jsou zobrazeny v grafu na obrázku 35 (MATLAB online Documentation, 2018b).



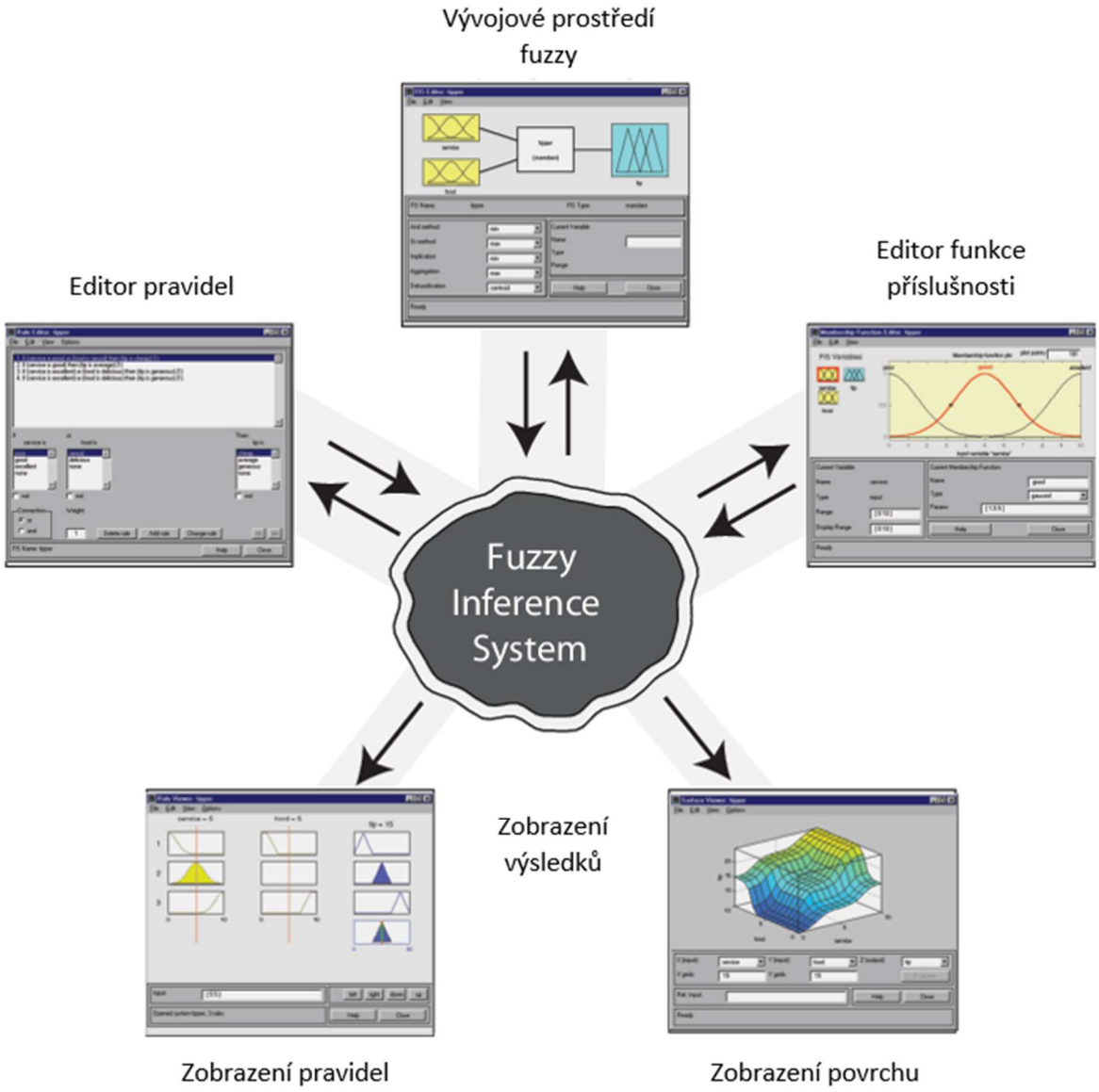
Obrázek 35 Metody defuzzifikace; zdroj: MATLAB online Documentation, 2018b

4.5.3 **Fuzzy expertní systém v MATLABu**

Software MATLAB umožňuje provádět složité matematické výpočty a matematické simulace. MATLAB je využíván pro vědecké a výzkumné účely, a to jak v soukromém sektoru, tak i v akademické sféře. Hlavní oblastí využití jsou technické obory a ekonomie (Zaplatílek a Doňar, 2005). Aplikace využívá maticové struktury dat při výpočtech a programovací jazyk MATLABu vychází z původního programovacího jazyka FORTRAN (Moler, 2004). Pro matematickou simulaci, modelování a vytváření inferenčních sítí je tato aplikace využívána i pro vědecké účely.

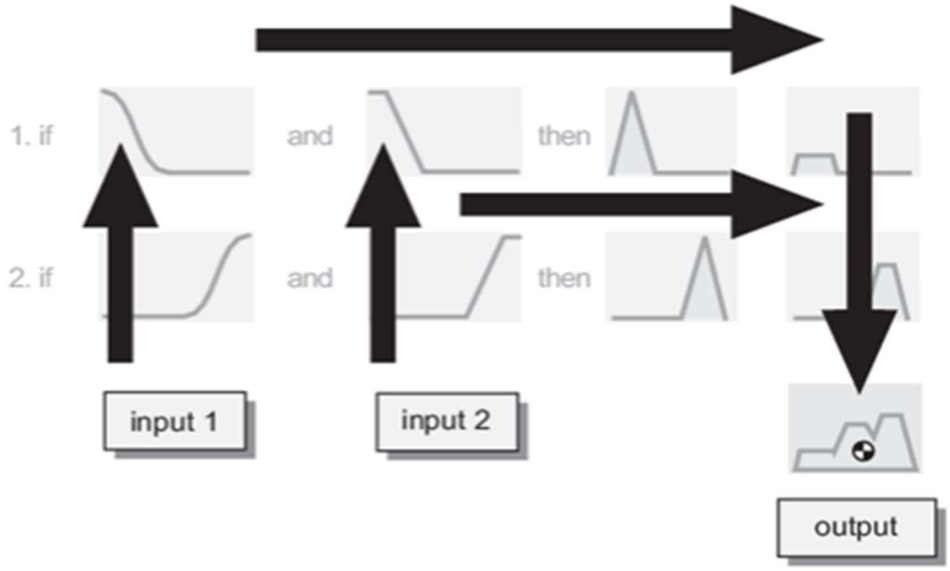
Například Khamis (2017) využil aplikaci pro vytvoření fuzzy kontrolního systému na řízení výkonu větrných elektráren. Sharma a Vashistha (2017) využili aplikaci k fuzzy zpřesnění dat z nasbíraných vzorků při analýze jódového deficitu v různých zeměpisných oblastech. Výhodou je, že tyto simulace lze exportovat do zdrojového kódu a dále využít v jiných aplikacích. Aplikace obsahuje doplňky - Toolboxy (knihovny funkcí) pro modelování a simulaci s využitím různých nástrojů.

Jeden z toolboxů je navržený pro využití a simulaci problémů pomocí fuzzy logiky. Tento doplněk se jmenuje Fuzzy Logic Toolbox. V tomto prostředí lze mimo jiné vytvořit expertní systém typu fuzzy Mamdani anebo Sugeno. Na následujícím schématu (obrázek 36) je zobrazeno vývojové prostředí a uživatelské možnosti Fuzzy Logic Toolboxu.



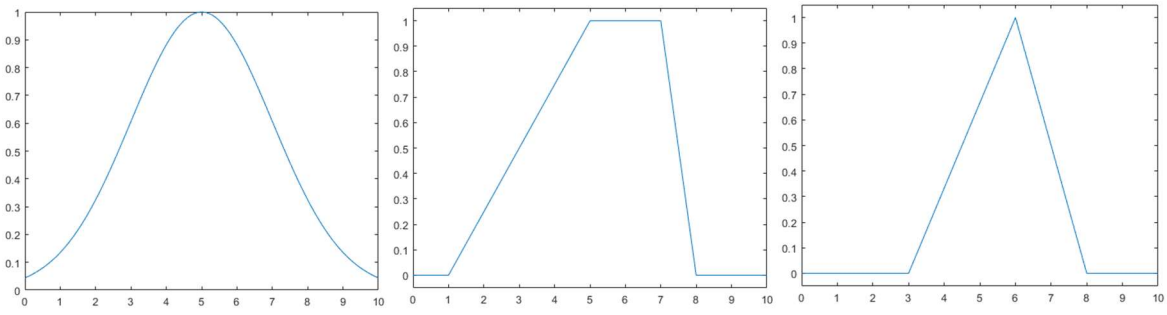
Obrázek 36 Vývojové prostředí Fuzzy Logic Toolbox; zdroj: MATLAB online Documentation, 2018a

V aplikaci lze pracovat s Mamdaniho typem inferenčního mechanismu, který využívali ve své vědecké práci například Olesiak (2017), Akgun et al. (2012) a jehož tvůrcem je Mamdani a Assilian (1975). Celý postup fuzzifikace vstupů a defuzzifikace výstupů pro dvě pravidla je zobrazen ve schématu na obrázku 37.



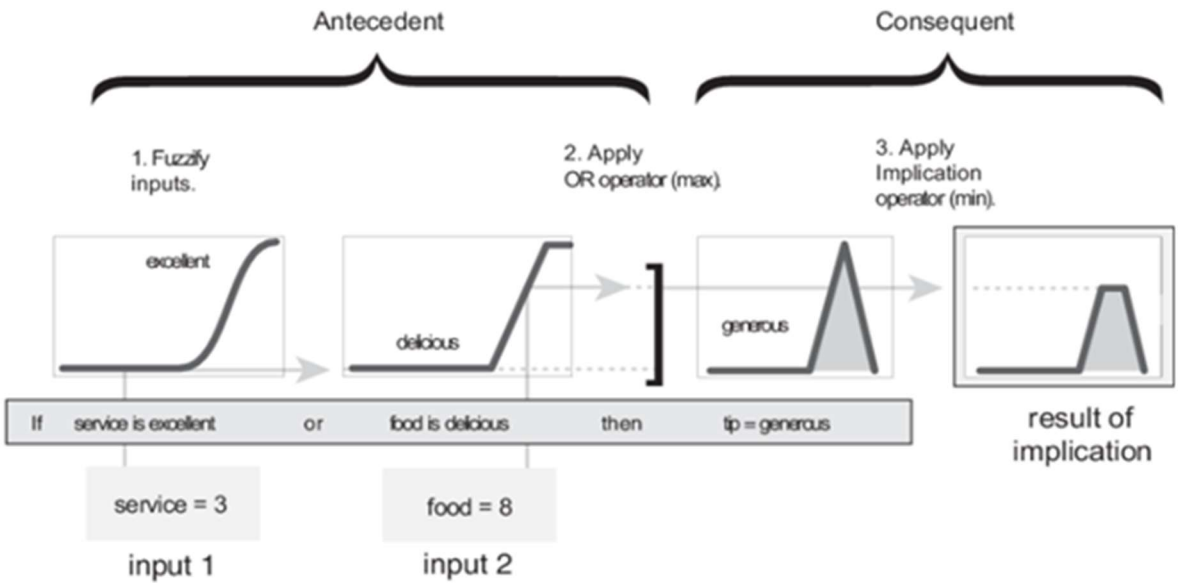
Obrázek 37 Schéma fuzzy Mamdani; zdroj: MATLAB online Documentation, 2017

V Matlabu je možné vstupní crisp hodnoty fuzzifikovat pomocí široké škály dostupných funkcí příslušnosti v aplikaci. Z funkcí lze využít například Gaussovu křivku funkce příslušnosti (gaussmf), trapezoidní funkci příslušnosti (trapmf), trojúhelníkovou funkci příslušnosti (trimf) obrázek 38.



Obrázek 38 Funkce příslušnosti; zdroj: MATLAB online Documentation, 2018c

Dále je k dispozici editor pravidel, ve kterém je možné zadat a kombinovat pravidla pro průběh inference. Průběh samotné inference fuzzy Mamdani je zachycen na následujícím schématu (obrázek 39).



Obrázek 39 Příklad fuzzifikace a inference; zdroj: MATLAB online Documentation, 2017

Literární rešerše je významně zaměřena na konstrukci a strukturu expertních systémů s fuzzy logikou. Důvodem je ověření chování modelu, který bude vytvořen v softwarové aplikaci MATLAB Simulink. Navržený model bude založen na fuzzy logice, fuzzy znalostních jednotkách a jejich inferenci.

5 Návrh modelu inference fuzzy znalostních jednotek

Výzkumný záměr disertační práce vychází a teorií z oblastí modelování znalostí, řešení neurčitosti a expertních systémů. Vlastní práce je uspořádána do kapitol, logicky podle vývoje daných předpokladů a jejich potvrzení. Jednotlivé vlastní přidané postupy k postupům uvedených v literární rešerši jsou popsány v příslušných kapitolách a demonstrovány v případových studiích vytvořených autorem. Synergický efekt plynoucí z dílčích výsledků je uveden v závěrečné části práce. Část průběžných výsledků již byla publikována na mezinárodních i tuzemských vědeckých konferencích. Část výzkumu byla podpořena dvouletým grantem IGA PEF ČZU v Praze s názvem „Data, informace a znalosti v expertních systémech“.

5.1 Východiska

Identifikovaná výzkumná příležitost je popsána následující otázkou:

Je možné provádět klasickou inferenci, dopředné a zpětné řetězení se znalostní jednotkou?

Pokud ano, jakým způsobem to provádět?

Prvním východiskem je problematika produkčních pravidel a metody inference dopředného a zpětného řetězení v kontextu znalostní jednotky. Tato problematika je vymezena v kapitole „5.2 Inference znalostních jednotek“. Jednotlivé postupy zobrazení neurčitosti a rozhodovacích situací jsou demonstrovány v případové studii. Pro tuto případovou studii byl využit problém z oblasti systémů ERP (Enterprise resource planning). Důvodem je dlouholetá praxe autora práce v pozici konzultanta ERP systémů, navíc má k dispozici zázemí odborníků na tuto problematiku z různých firem zabývající se ERP systémy. Na základě výsledků této případové studie bylo rozhodnuto navrhnout postup fuzzifikace znalostních jednotek, která by umožnila pracovat s neurčitostí.

Dalším východiskem je oblast fuzzy logiky a práce s neurčitostí v kapitole „5.3 Případová studie inference znalostních jednotek“, jejíž výsledky umožňují navrhnout fuzzy znalostní jednotku. To je obsahem kapitolty „5.4 Zobrazení neurčitosti fuzzy znalostní jednotkou“, která opět obsahuje i případovou studii využití fuzzy znalostní jednotky.

Navržený způsob modelového řešení inference znalostních jednotek je rozšířen i na fuzzy znalostní jednotky. Tento krok je popsán v kapitole „5.5 Případová studie fuzzifikace znalostní jednotky“ a je opět doplněn příkladem.

Na základě těchto dílčích kroků je vytvořen návrh obecného modelu inference fuzzy znalostních jednotek v kapitole „5.6 Model fuzzy znalostních jednotek“. V následující případové studii je tento přístup prakticky využit a jsou ukázány jeho výhody. Výsledky modelu jsou též porovnány s klasickým přístupem expertů k rozhodování.

5.2 Inference znalostních jednotek

Základem pro řešení neurčitosti/rozhodovacích situací se staly lingvistické proměnné a fuzzy přístupy, o kterých je detailněji pojednáno v kapitole „4.5 Fuzzy expertní systémy“. Spojení těchto vybraných metod a znalostních jednotek poté vyústilo k definování fuzzy znalostní jednotek (*angl. Fuzzy Knowledge Unit),* viz kapitola „5.4.1 Fuzzifikace znalostní jednotky“ a byly publikovány v článku Peták et al. (2020) a na konferenci DIVAI 2018 článkem Peták a Houška (2018).

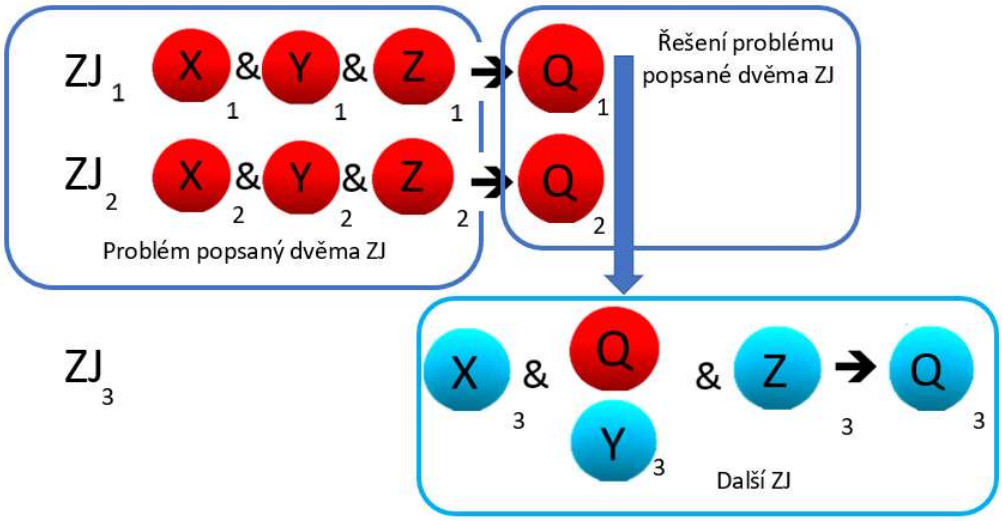
Inference v kontextu znalostních jednotek musí umožňovat dopředné a zpětné řetězení se znalostmi, které jsou jako znalostní jednotky definovány. Pro vytvoření potenciálu k inferenci je využita základní vlastnost znalostní jednotky, kterou je možnost definice znalostní jednotky v podobě rozšířeného produkční pravidla. Rozšířené produkční pravidlo znalostní jednotky je složeno z jednotlivých elementárních prvků, které mají mezi sebou vztahy. Vzhledem k tomu, že jsou elementy znalostních jednotek v analytickém tvaru označovány velkými písmeny (Dömeová et al., 2008; Brožová a Houška, 2011), budou tak označovány i v následujících částech práce. Vztahy elementů znalostních jednotek jsou definovány následovně,

$$\text{IF (X and Y and Z) THEN Q.} \tag{44}$$

Vztah (47) je rozšířeným produkčním pravidlem vyjadřujícím znalostní jednotku. Standardní produkční pravidla IF – THEN jsou složena ze dvou částí a to antecedent (důkaz, situace, problém) a konsekvent (hypotéza, akce, řešení). V kontextu znalostní jednotky jsou antecedentem elementy znalostní jednotky $\{X, Y, Z\}$ a konsekventem je $\{Q\}$. To znamená, že popis problémové situace (X) s konkrétním problémem (Y) a cílem řešení (Z) je antecedent

a způsob řešení konkrétního problému (Q) je konsekvent. Uvedená vlastnost znalostní jednotky je využita ve strategii dopředné a zpětné inference se znalostní jednotkou. Strategii dopředného a zpětného řetězení lze aplikovat na znalostní jednotky a dále na jejich elementy následujícím způsobem (Peták a Houška, 2017).

Znalostní jednotka je tedy složena z elementárních částí, které mají mezi sebou pevnou vazbu. Průběh inference mezi elementy napříč jednotlivými znalostními jednotkami lze odvozovat na základě vlastností znalostních jednotek a jejich elementů. Vlastnosti a atributy znalostních jednotek jsou detailně popsány v kapitole „4.1.3 Znalostní jednotky“. Předpokladem této inference je společná aplikační doména a strategie řetězení, tj. dopředné a zpětné řetězení, které jsou detailně popsány v kapitolách „4.1.1 Produkční pravidla, řetězení“ a „4.3.3 Inference znalostí v expertních systémech“. Základním předpokladem inference znalostních jednotek je, že řešení problémů (konsekventů) popsaných znalostní jednotkou (antecedent) může být více a nikoli jedno (viz obrázek 40).



Obrázek 40 Výchozí schéma inference se znalostními jednotkami; zdroj: autor

Nyní bude zformulován proces inference dvou znalostních jednotek.

Jsou dány dvě znalostní jednotky (ZJ) z jedné aplikační domény. Znalostní jednotky jsou následující.

$$\begin{aligned} ZJ^1 &= \{X^1, Y^1, Z^1, Q^1\}, \\ ZJ^2 &= \{X^2, Y^2, Z^2, Q^2\}. \end{aligned} \tag{45}$$

Mezi těmito znalostními jednotkami lze provést inferenci, a to jednak dopředné řetězení a také zpětné řetězení. Pro inferenci jsou využity elementy Q (konsekvent) a Y (elementární část antecedentu) právě proto, jak jsou definovány v analytickém tvaru. Jelikož Q je řešení elementárního problému a Y je elementární problém. Podrobný popis znalostních jednotek je součástí kapitoly „4.1 Znalosti a jejich reprezentace“ V kontextu výše uvedených dvou znalostních jednotek lze v textové formě uvést inferenci takto:

Zpětné řetězení

„Když nejsme schopni aplikovat řešení Q^1, stane se (Q^1) elementárním problémem Y^2, který řeší problémovou situaci X^2 pro dosažení cíle Z^2, potom použijeme řešení Q^2.“

Dopředné řetězení

„Když potřebujeme nalézt řešení Q^2, stane se (Q^2) elementárním problémem Y^1, který řeší problémovou situaci X^1 pro dosažení cíle Z^1, potom použijeme řešení Q^1.“

Inference se znalostními jednotkami lze formalizovat prostřednictvím elementů znalostní jednotky:

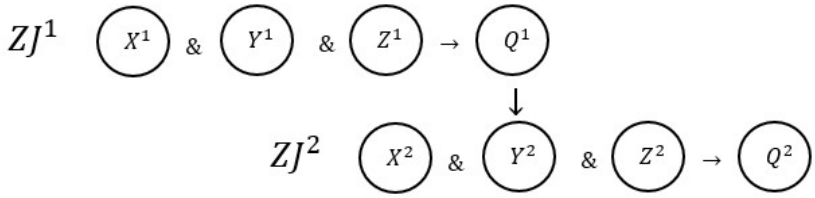
zpětné řetězení

$$\mathrm{ZJ}^{\mathrm{INF}} = \{\mathrm{Q}^1\} \rightarrow \{\mathrm{Y}^2\}, \tag{46}$$

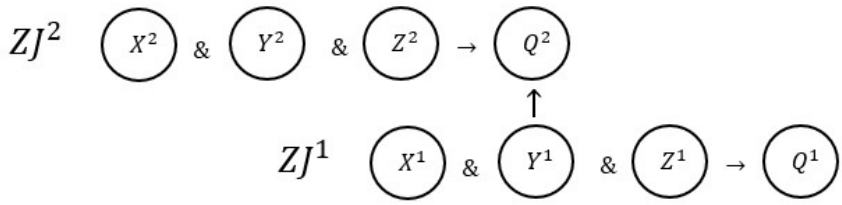
dopředné řetězení

$$\mathrm{ZJ}^{\mathrm{INF}} = \{\mathrm{Y}^1\} \rightarrow \{\mathrm{Q}^2\}. \tag{47}$$

Analytické schéma strategie řetězení je uvedeno na obrázku 41 a 42.



Obrázek 41 Zpětné řetězení; zdroj: autor



Obrázek 42 Dopředné řetězení; zdroj: autor

Uvedená strategie inference elementárních členů znalostních jednotek umožňuje řetězení, u kterého je již dopředu známé logické uspořádání znalosti. Uspořádání má tvar rozšířeného produkčního pravidla znalostní jednotky na rozdíl od standardní inference dopředného a zpětného řetězení, kdy je výsledkem dotazu inferenční síť faktů, pravidel a konečného konsekventu.

Při inferenci více znalostních jednotek je výsledkem inferenční síť složená ze znalostních jednotek, jejich elementů a konečný konsekvent je opět znalostní jednotka.

Podstatný rozdíl mezi oběma strategiemi inference je v tom, že strategie inference elementárních členů znalostních jednotek sdružuje, "řetězí" jednotlivé znalostní jednotky a z toho odvozuje konsekvent v podobě znalostní jednotky. Tato inference oproti klasické využívá fakta, která již mají strukturu znalostní jednotky.

5.3 Případová studie inference znalostních jednotek

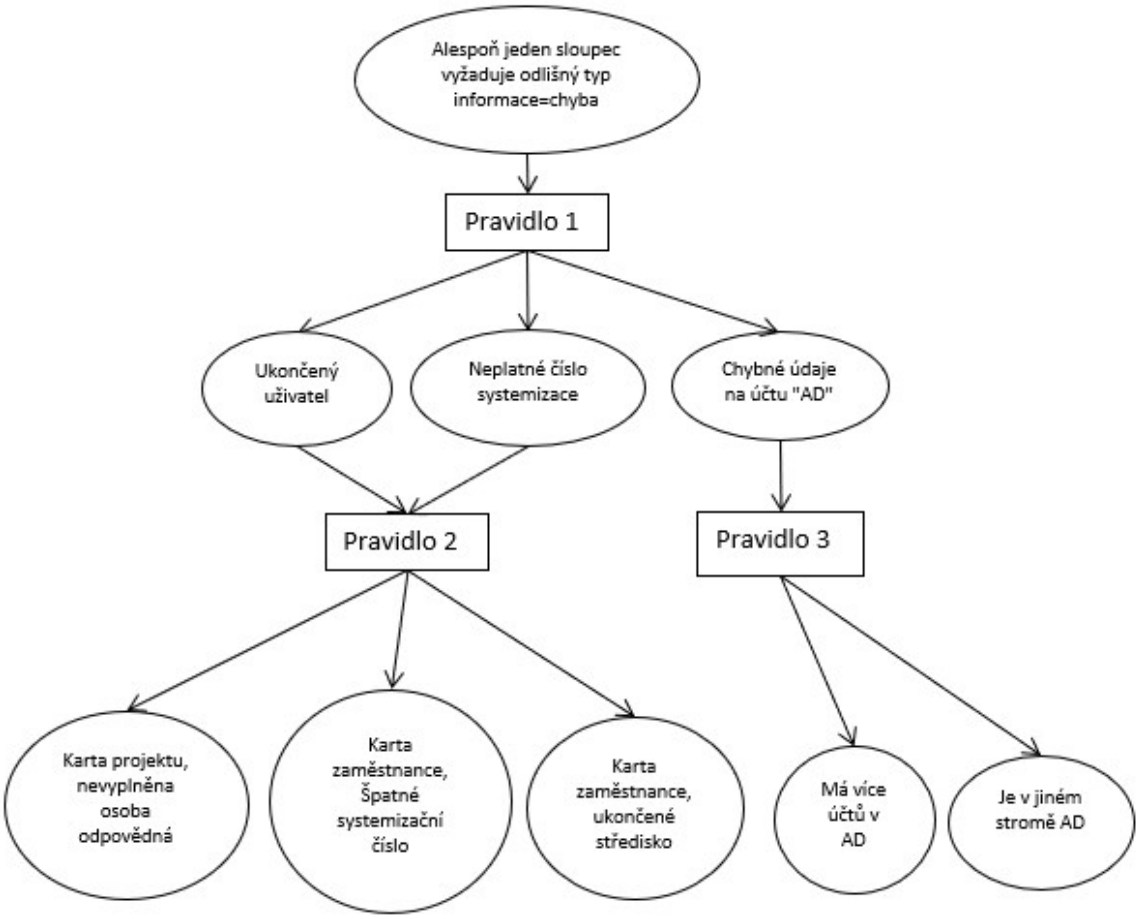
Standardní strategie zpětného řetězení vyžaduje pro průběh inference bázi faktů a bázi pravidel z příslušné aplikační domény. Případová studie ukazuje jednak inferenci znalostních jednotek s určitostí a také analýzy tří základních přístupů k zahrnutí neurčitosti. Případová studie se zabývá rozhodovací situací vznikající při jednoduché diagnostice chyby v informačním systému. Databáze pravidel obsahuje tři pravidla (viz tabulka 5) a to následující.

Pravidla	Obsah pravidla
1	Jestliže "je ukončený uživatel" NEBO "neplatné číslo systemizace" NEBO "Chybné údaje na účtu "AD"" POTOM "Alespoň jeden sloupec vyžaduje odlišný typ informace=chyba"
2	Jestliže "Karta projektu, nevyplněna osoba odpovědná" NEBO "Karta zaměstnance, Špatné systemizační číslo" NEBO "Karta zaměstnance, ukončené středisko" POTOM "Ukončený uživatel" NEBO "Neplatné číslo systemizace"
3	Jestliže "Má více účtů v AD" NEBO "Je v jiném stromě AD" POTOM "Chybné údaje na účtu "AD""

Tabulka 5 Databáze pravidel; zdroj: autor

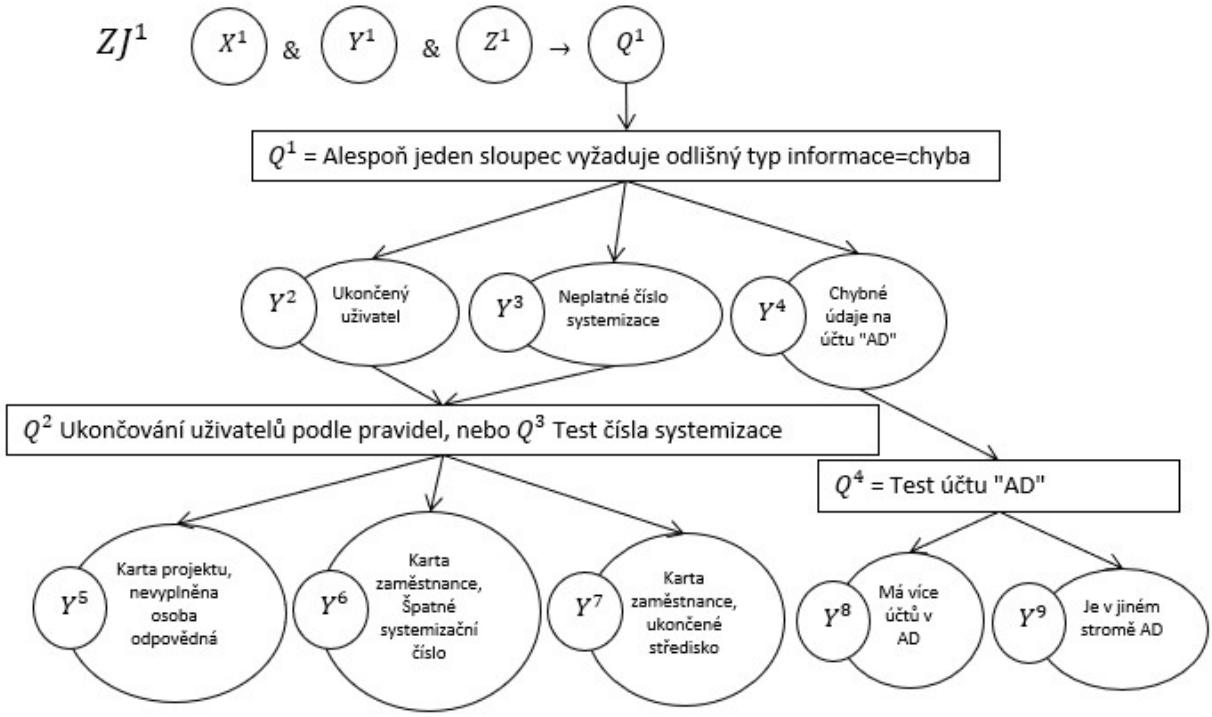
5.3.1 **Inference s určitostí**

Databáze faktů je obsažena v diagramu řešení v jednotlivých oválech. Odvozená Inferenční síť má následující tvar (obrázek 43).



Obrázek 43 Standardní zpětné řetězení u diagnostiky chyby; zdroj: autor

Strategie řetězení znalostních jednotek a její odvozená inferenční síť je uvedena na obrázku 44. Liší se od standardní strategie inference v pojetí řetězených faktů a pravidel. Z každého uzlu v inferenční síti lze interpretovat již definovanými způsoby zobrazení znalostní jednotku.



Obrázek 44 Zpětné řetězení znalostních jednotek u diagnostiky chyby; zdroj: autor

Uvedená inference znalostních jednotek obsahuje devět znalostních jednotek. Například znalostní jednotky ZJ^1 a ZJ^2, které jsou definovány pro aplikační doménu diagnostiky chyby informačního systému.

ZJ^1 "Pokud potřebujeme Vyřešit chybu a Nalézt, kde je chyba, abychom Identifikovali chybu v aplikaci, musíme Spustit analytický test aplikace výskytu chyby."

X^1 = Řešení chyby

Y^1 = Nalézt, kde je chyba

Z^1 = Identifikace chyby v aplikaci

Q^1 = Spustit analytický test aplikace výskytu chyby

ZJ^2 "Pokud potřebujeme stanovit Pravidla ukončování uživatelů v systémech a Ukončit uživatele, abychom Aktualizovali uživatele, musíme Provést ukončování uživatelů podle pravidel."

X^2 = Pravidla ukončení uživatelů v systémech

Y^2 = Ukončování uživatelů

Z^2 = Aktualizace uživatelů

Q^2 = Ukončování uživatelů podle pravidel

Výsledkem jsou znalostní jednotky, které explicitně vyjadřují znalost, jak opravit chybu v propojených systémech.

5.3.2 **Zobrazení neurčitosti v rozhodovacích situací**

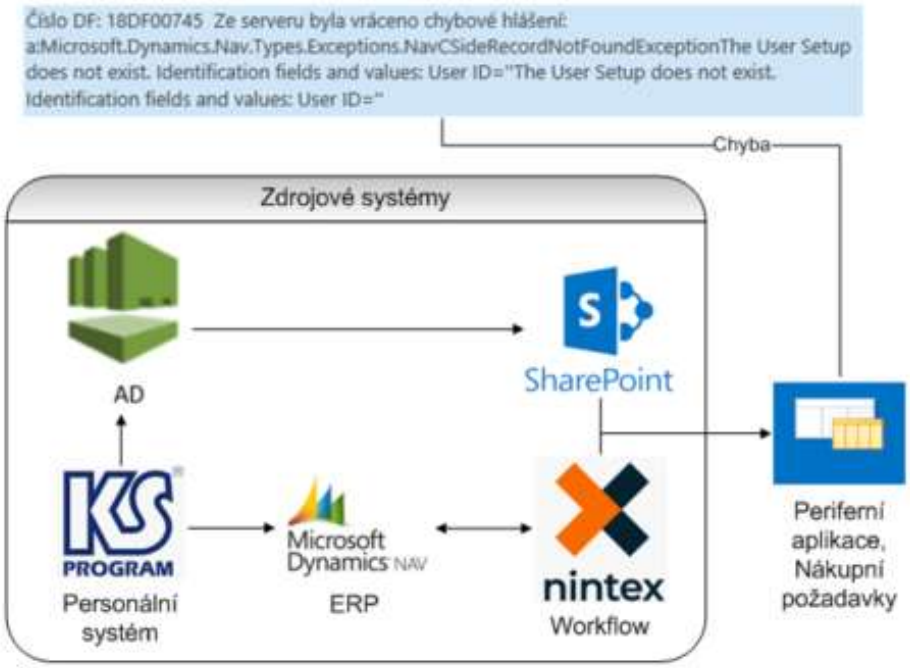
Po potvrzení možnosti inference znalostních jednotek je navázáno výběrem metody pro řešení neurčitosti a rozhodovacích situací při inferenci znalostních jednotek. Základem pro výběr metody je literární rešerše a vhodnost využití metody ve spojení se znalostní jednotkou. Teoretickými východisky jsou kapitoly „4.1.3 Znalostní jednotky“, „4.4 Znalosti a expertní systémy v prostředí neurčitosti“ a „4.5 Fuzzy expertní systémy“. Pro účel výběru metody řešení neurčitosti a rozhodovacích situcí byla vytvořena případová studie. Případová studie je zpracována v rámci vlastního výzkumu, je přidanou hodnotou této práce a vychází z reálné řešené situace. Východiskem jsou jednak uvedené části rešerše a dále výsledky jednotlivých řešení případové studie provedené v praxi.

Případová studie vychází z prostředí informačních systémů konkrétně systémů ERP (Enterprise Resource Planning) a přidružených – periferních aplikací. Případová studie byla zpracována v rámci vlastní práce. Popisovaný příklad se váže k produkčně nasazeným systémům (periferní aplikace) ve firmě Ústav jaderného výzkumu - ÚJV Řež, a. s., které urychlují některé firemní procesy.

Studie se týka aplikace „Workflow nákupních požadavků“, která řeší proces nákupu ve firmě i dceřiných společnostech. Pro další periferní aplikace a jejich správnou funkčnost je systém ERP klíčový. ERP je zdroj informací, dat a metadat, v některých případech je také spotřebitelem dat a prostředníkem v procesu přenosu dat mezi aplikacemi. K těmto zdrojům v ERP přistupují různými způsoby periferní aplikace. Tyto periferní aplikace jednak data zapisují, ale také poskytují a tím urychlují chod firemních procesů.

Problémová situace nastává ve chvíli, kdy uživatel některé z aplikací narazí na chybu. Pokud se nejedná o chybu uživatelského charakteru, kterou uživatel sám opraví/upraví, je podstatné zjistit zdroj této chyby. V této chvíli uživatel nahlásí chybu na helpdesk a operátor heldesku by měl nasměrovat chybové hlášení na správného experta, přestože chybová hlášení nebývají přesná.

U některých chyb nemusí stačit k řešení expert na aplikaci, ve které se chyba odehrála. Do řešení problémové situace tak musí vstoupit expert, který zná problematiku propojení dotčených aplikací, a do jisté míry i aplikací samotných. Analýzou a diagnostikou chyby v dotčené aplikaci musí nalézt zdroj, příčinu této chyby, případně i to, že se jedná o chybu v jiných aplikacích. Schéma celkové situace je na obrázku 45.



Obrázek 45 Diagnostika chyby; zdroj: autor

V praxi bývá náročné nahlášenou chybu úspěšně vyhodnotit a předat odpovědnému řešiteli. Správné vyhodnocení chyby operátorem je podmínkou úspěšně provedené diagnostiky chyby, která je v praxi časově náročná a dochází v ní k vysoké chybovosti. Řešení chyby často provádí jeden nebo více expertů, probíhá na úrovni vytvořených rozhraní mezi aplikacemi a dat či metadat v provázaných aplikacích.

Všechny tyto atributy mohou být ve zdrojových systémech a rozhraních neúplné, neaktuální, chybí nebo se jedná o systémové chyby či aktualizace, nebo nekompatibilní verze aplikací i

prostředí, ve kterém je aplikace provozována. Případová studie je vytvořená pro diagnostiku jedné z konkrétních chyb aplikace, která může mít více různých příčin ve více aplikacích.

Na základě aktuální dostupné odborné literatury (viz kapitola „4.4 Znalosti a expertní systémy v prostředí neurčitosti“) je navrhnut přístup k řešení rozhodovacích situací při inferenci znalostních jednotek, jsou to Bayesova síť/inference, Dempster Shaferova teorie, a Fuzzy logika/Fuzzy množiny. Tyto metody jsou v této kapitole aplikovány na definovanou případovou studii. V následujících podkapitolách jsou uvedena navržená řešení podle jednotlivých metod. Tato řešení slouží jako východiska pro odvození způsobu práce s neurčitostí v rozhodovacích situacích.

5.3.2.1 Bayesova síť/inference

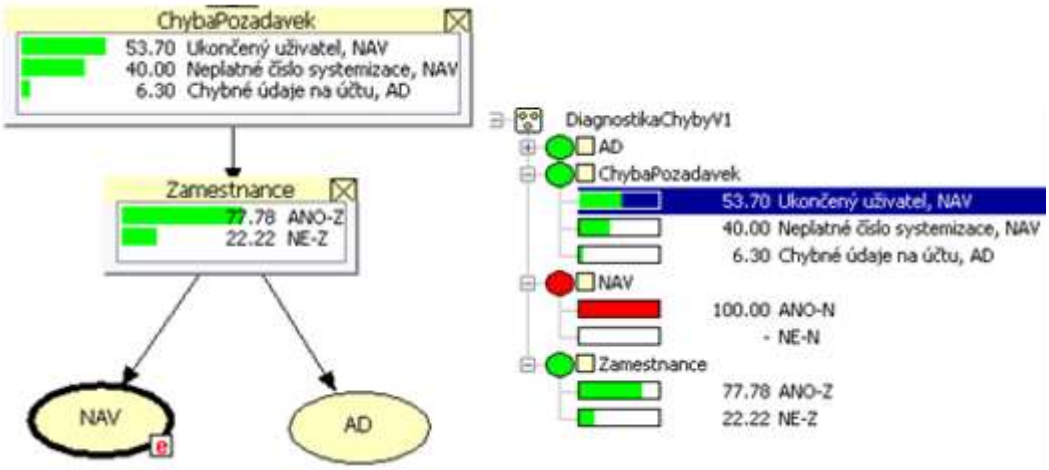
Bayesovská teorie, a z ní vycházející podmíněné pravděpodobnosti, je rozšířená metoda pro řešení neurčitostí a rozhodovacích situací při inferenci. Na základě této metody bylo vyvinuto více aplikací umožňující vytvářet inferenční síť. Jedna ze známých aplikací je program podporující vytváření expertních systémů HUGIN. Tento program slouží jako nástroj pro vytvoření vlastní analytické situace. V tomto programu je autorem práce zpracována problémová situace diagnostiky chyby periferní aplikace s názvem „Nákupní požadavky“.

Pro tento příklad byly pravděpodobnosti zdrojů chyb naplněny expertním odhadem. Oslovení experti byli tvůrci aplikace „Nákupní požadavky“ a experti řešící tyto chyby. Na obrázku 46 je zobrazena síť pro tento příklad diagnostiky chyby a výsledek. Pravděpodobnosti byly zadány do programu HUGIN jako konsensus expertů v dané problémové situaci v tabulce 6.

Chyba Nákupního požadavku	Pravděpodobnost výskytu
Ukončený uživatel, NAV	53,70
Neplatné číslo systemizace, NAV	40,00
Chybné údaje na účtu, AD	6,30

Tabulka 6 Odhad pravděpodobnosti chyb; zdroj: autor

V Bayesovské teorii se pracuje s pravděpodobností jevu a v tomto případě lze popsat zadání pravděpodobnosti následovně. Pokud se objeví uživateli chyba v aplikaci „Nákupní požadavky“, je s pravděpodobností 53,70 % chyba na straně systému ERP NAV (Microsoft Dynamics Navision) a v tomto případě je přesný popis chyby „Ukončený uživatel, NAV“, což je chybový status uživatele, viz tabulka 6.



Obrázek 46 HUGIN síť a diagnostika chyby; zdroj: autor

V simulovaném případě se objeví chyba v aplikaci Nákupní požadavky, konkrétně je chyba znázorněna na obrázku 45. Tato chyba zjednodušeně odpovídá popisu „Ukončený uživatel“. Na obrázku 46 je zobrazena v Bayesovské inferenční síti diagnostika chyby. Zelené sloupce a hodnoty u nich uvedené zobrazují pravděpodobnost nastalého jevu. Bayesovská síť se po zadání vstupní informace v případě této chyby upraví podle vytvořené struktury sítě (pravidel) a přepočte podmíněné pravděpodobnosti. Tímto způsobem navrhne možné řešení a zobrazí v pravděpodonosti jevů pro zadanou vstupní informaci. Inference prochází objektem zaměstnanec, ve kterém je definováno více parametrů, ve kterých může být chyba, konkrétně v NAV nebo AD (Active directory).

Objekt zaměstnanec je mezikrok v inferenci a na tomto stupni je již více patrné, v jakém systému, aplikaci zdroj chyby může být. V jakém systému je pravděpodobně chyba odpovídá až další krok, viz obrázek 46 - větev NAV. Vzhledem k definovaným pravděpodobnostem a nastalé chybě je zdroj chyby v systému NAV. Řešitel chybového hlášení by tedy měl podle pravděpodobnosti zkontrolovat nejprve stav a platnost systemizačního čísla zaměstnance v NAV (s pravděpodobností 77,78 % je chyba tam) a poté atributy zaměstnance v AD (pravděpodobnost chyby 22,22 %).

5.3.2.2 Dempster Shaferova teorie

Alternativou k uvažovaným metodám řešení rozhodovacích situací je Dempster Shaferova torie (Dempster, 1967). V této metodě lze reprezentovat nevědomost s tím, že součet měr domnění nemusí být roven jedné. Vztaženo na uvedenou problémovou situaci a chybu není třeba, aby experti definovali všechny míry domnění tak, aby byl součet roven jedné. Dopočet do jedné je totiž nevědomost a zároveň také možnost nedefinované chyby v problémové

situaci. Dalším atributem této metody je možnost kombinování jednotlivých měr domnění. To znamená, že jednotlivé názory všech expertů na danou problémovou situaci zachycené prostřednictvím míry domnění se dají kombinovat do jedné výsledné míry domnění.

Kombinační pravidlo je definováno přesně v kapitole „4.4 Znalosti a expertní systémy v prostředí neurčitosti“. Základní přiřazení „pravděpodobnosti“ vzniku událostí jednotlivých expertů z intervalu <0;1> jsou uvedena v následujících tabulkách 7 a 8. Jedná se o tytéž oslovené experty, kteří byli osloveni pro definici pravděpodobnosti chyb v kapitole „5.3.2.1 Bayesova síť/inference“.

Expert A			
Událost	Odhad *m*	Domnění Bel	Plausibilita Pl
Ukončený uživatel, NAV	0,6	0,6	0,7
Neplatné číslo systemizace, NAV	0,3	0,3	0,4
Ω-Chybné údaje na účtu, AD, jiné	0,1	1	1

Tabulka 7 Základní charakteristiky Dempster Shaferovy teorie experta A; zdroj: autor

Expert B			
Událost	Odhad *m*	Domnění Bel	Plausibilita Pl
Ukončený uživatel, NAV	0,5	0,5	0,7
Neplatné číslo systemizace, NAV	0,3	0,3	0,5
Ω-Chybné údaje na účtu, AD, jiné	0,2	1	1

Tabulka 8 Základní charakteristiky Dempster Shaferovy teorie experta B; zdroj: autor

K základním přiřazením „pravděpodobností“ vzniku událostí jsou také přidány základní charakteristiky Dempster Shaferovy teorie. Je zde zobrazena domněnková funkce Bel, která přiřazuje hodnotu z <0, 1> každé neprázdné podmnožině. Dalším ukazatelem je míra věrohodnosti Pl, která uvádí nakolik lze věřit události, jestliže i všechna dosud neznámá fakta by danou událost podporovala. Jedná se tak o maximální množství důvěry, které může být přiřazeno k události.

Dempster Shaferova torie umožňuje kombinančním pravidlem sjednotit názory expertů. V této případové studii je výsledná kombinace základních přiřazení v tabulce 9. Pokud nastane chyba, tato metoda prostřednictvím kombinačního pravidla dokáže specifikovat, kde chyba nastala.

Události	Ukončený uživatel, NAV	Neplatné číslo systemizace, NAV	Ω - Chybné údaje na účtu, AD, jiné
Sloučení preferencí expertů	0,70149	0,26866	0,02985

Tabulka 9 Kombinace základních přiřazení expertů A a B; zdroj: autor

Z kombinace názorů expertů vyplývá, že pokud nastane chyba v aplikaci Nákupní požadavky, je to kvůli tomu, že je ukončený uživatel v systému ERP NAV. Řešitel této chyby v aplikaci by nejprve měl zkontrolovat atributy zaměstnance v ERP NAV, potom zkontrolovat platnost čísla systemizace a nepřiřazený zbytek může být jiná nespecifikovaná chyba a chybné údaje na účtu AD.

5.3.2.3 Fuzzy logika

Příkladem je již uvedená případová studie diagnostiky chyby v aplikaci „Nákupní požadavky“. Pro účel jejího vytvoření je použit MATLAB a jeho komponenta Fuzzy Logic Toolbox. Podstatnou částí případové studie je volba funkcí příslušností jednotlivých chybových stavů. Chybový jev je modelován fuzzy funkcemi příslušností podle expertního odhadu již oslovených expertů. Tímto způsobem je pro určitý chybový stav aplikace „Nákupní požadavky“ definována možná příčina. Lingvistická proměnná je definovaná jako Chyba nákupního požadavku a její termy, chybové stavy jsou definovány v tabulce 10.

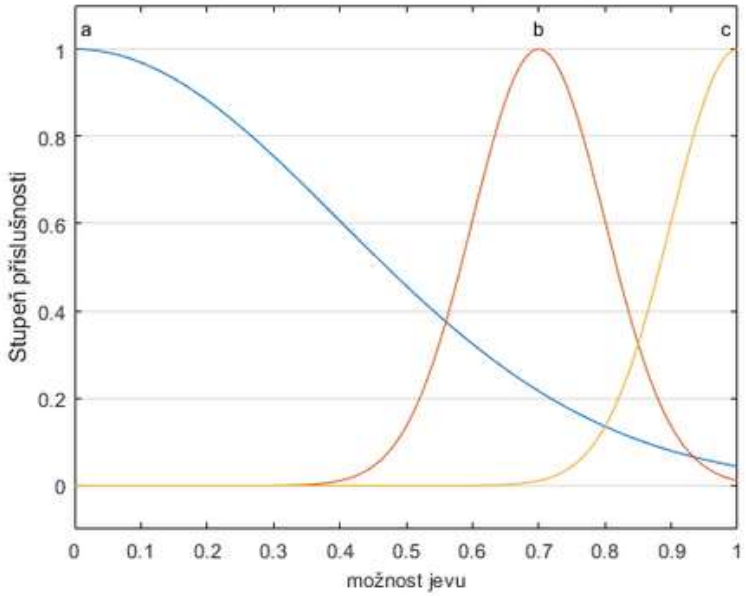
Chyba nákupního požadavku	Proměnná
Ukončený uživatel, NAV	a
Neplatné číslo systemizace, NAV	b
Chybné údaje na účtu, AD	c

Tabulka 10 Chybové stavy a proměnné; zdroj: autor

Pro tento chybový stav byly navrženy funkce příslušnosti pomocí MATLABu, kde byl vytvořen zdrojový kód pro vytvoření funkcí:

```
a = newfis('Chyba nákupního požadavku');
a = addvar(a,'input','možnost jevu',[0 1]);
a = addmf(a,'input',1,'a','gaussmf', [0.4 0]);
a = addmf(a,'input',1,'b',' gaussmf', [0.1 0.7]);
a = addmf(a,'input',1,'c','gaussmf', [0.1 1]);
plotmf(a,'input',1)
set(gca,'XGrid','off','YGrid','on')
ylabel('Stupeň příslušnosti')
```

Zobrazení chybového jevu je na obrázku 47.

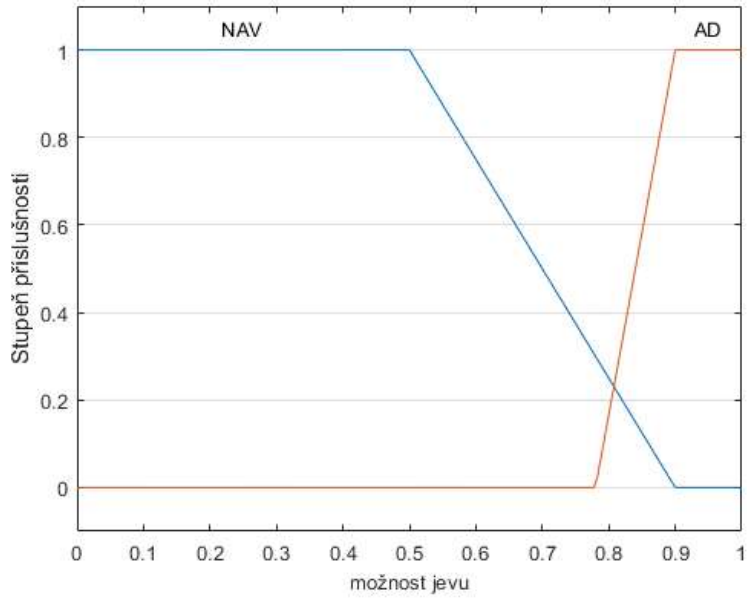


Obrázek 47 Chyba nákupního požadavku; zdroj: autor

Obdobným způsobem je vytvořeno řešení chybové situace a odpověď na diagnostiku chyby.

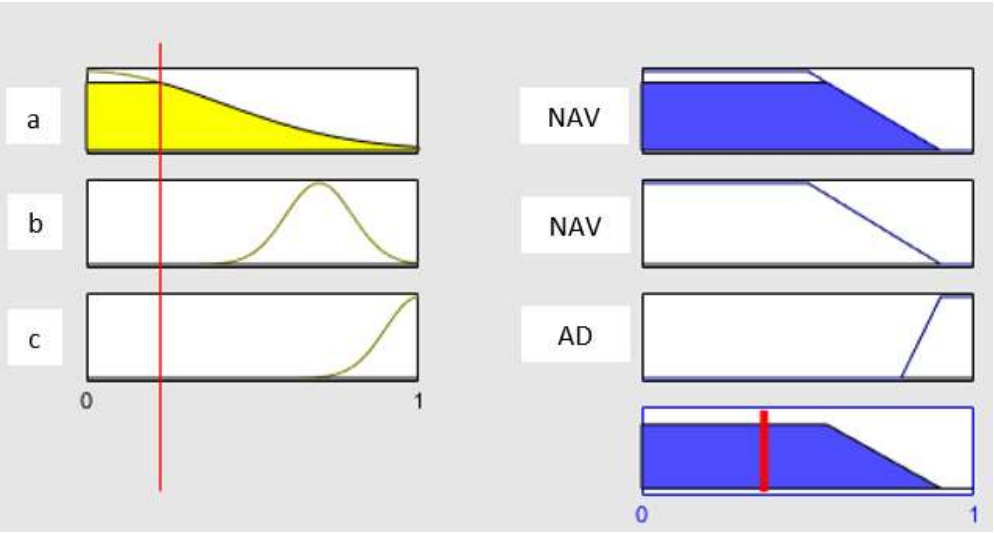
```
a = newfis('Chyba nákupního požadavku');
a = addvar(a,'input','možnost jevu',[0 1]);
a = addmf(a,'input',1,'NAV','trapmf',  [-0.9 -0.5 0.5 0.9]);
a = addmf(a,'input',1,'AD','trapmf',  [0.78 0.9 1.1 1.9]);
plotmf(a,'input',1)
set(gca,'XGrid','off','YGrid','on')
ylabel('Stupeň příslušnosti')
```

Zobrazení řešení chybové situace pomocí funkcí příslušností je na obrázku 48.



Obrázek 48 Funkce příslušností řešení chybové situace; zdroj: autor

Po spuštění simulace následuje vyhodnocení s účelem zjistit v jakém systému je chyba a dále upřesnění této chyby v systému. Vyhodnocení pomocí Mamdaniho grafické inference je na obrázku 49.



Obrázek 49 Vyhodnocení diagnostiky chyby; zdroj: autor

Z popisu chybové situace a z průběhu funkcí příslušností je Mamdaniho inferencí odvozeno, že diagnostikovaná chyba je v aplikaci NAV a „nejspíše" je to příčina „a" neboli „Ukončený uživatel, NAV" v systému NAV. Konkrétní chybový stav „a" má v popisu řešení chyby nejobsáhlejší funkci příslušnosti. Tudíž pokud nastane chyba v aplikaci, je nanejvýše možné, že se jedná právě o chybu „a" v systému NAV.

5.4 Zobrazení neurčitosti fuzzy znalostní jednotkou

Základem pro fuzzy znalostní jednotky (FZJ) jsou znalostní jednotky (ZJ). Znalostní jednotky vychází ze systémového pojetí znalosti a lze jimi formalizovat procedurální znalosti. Složení znalostní jednotky lze přirovnat k rozšířenému produkčnímu pravidlu, které obsahuje pevné pořadí a vztahy čtyř elementů znalostní jednotky. Výhodou této formalizace jsou možné unární (drill down, roll up) a binární operace (jednoduché skládání, rozklad, integrace, desintegrace a ekvivalence).

Další výhodou jsou také různé možnosti zobrazení, a to analytické, textové, a v této práci navržené grafické. Na základě definice znalostních jednotek bude navržen postup fuzzifikace znalostní jednotky. Znalostní jednotka tak bude rozšířena o řešení rozhodovacích

situací s neurčitostí pomocí fuzzy lingvistických proměnných. Grafická interpretace částí znalostní jednotky vychází z potřeb fuzzy Mamdaniho grafické inference (Mamdani a Assilian, 1975; Talašová, 2003) a lingvistických proměnných (Novák et al., 2016).

5.4.1 **Fuzzifikace znalostní jednotky**

Fuzzifikace znalostní jednotky a výběr zobrazení neurčitosti v rozhodovacích situací vychází z rešerše a východisek této práce. Odráží jeden z hlavních směrů v oblasti řešení rozhodovacích situací. Důvodem výběru fuzzy logiky je zejména zobrazení a popis jevu jako takového, a ne toho jestli nastane – Ano/Ne. Hlavní přidanou hodnotou této kapitoly je právě fuzzifikace znalostní jednotky a potřebné kroky k této fuzzifikaci. Fuzzy logika a lingvistické proměnné napomáhají přesnější formulaci znalostní jednotky a umožňují zachytit neurčitost při formalizaci a řešení problémových situací.

Znalost expertů je formalizována pomocí znalostních jednotek, lingvistických proměnných a fuzzy funkcí příslušnosti. Lingvistické proměnné se standardně značí velkými písmeny kurzívou (Zadeh et al., 1996; Zadeh, 1999; Dvořák, 2004), zde budou značeny také kurzívou. Fuzzifikaci znalostních jednotek je možné provést v těchto krocích:

- Sestavení znalostní jednotky
- Lingvistická proměnná a znalostní jednotka
- Fuzzifikace elementu Q
- Fuzzifikace elementu Y
- Formalizace fuzzy znalostní jednotky

Popis obsahu těchto kroků následuje.

5.4.1.1 Sestavení znalostní jednotky

Znalostní jednotka v části antecedent (předpokladové) vysvětluje - popisuje problémovou situaci a v druhé části konsekvent (důsledkové) navrhuje řešení. Při sestavování znalostní jednotky musí být definována aplikační doména a současně vhodná abstrakce popisované problematiky. Vhodným postupem při sestavování znalostní jednotky je začít analytickou podobou a poté například textovou pro ověření logické správnosti znalostní jednotky. Analytická podoba znalostní jednotky je v tabulce 11.

Část pravidla	Element	Popis elementu znalostní jednotky
Antecedent	*X*	Problémová situace
	Y	Elementární problém
	Z	Cíl řešení elementárního problému
Konsekvent	*Q*	Řešení elementárního problému

Tabulka 11 Analytická podoba znalostní jednotky; zdroj: autor

Textová forma:

"*Když je třeba v rámci problémové situace X řešit elementární problém Y, aby bylo dosaženo cíle Z, potom je třeba aplikovat řešení Q."*

Tato činnost by měla být prováděna expertem aplikační domény a znalostním inženýrem.

5.4.1.2 Lingvistické proměnné a znalostní jednotka

Sestavená znalostní jednotka obsahuje elementární části X, Y, Z, Q. V podstatě každá z těchto jednotlivých částí by mohla být označena jako lingvistická proměnná. K této lingvistické proměnné by mohla být také přiřazena množina termů a dále definovány jejich významy. Avšak jen u některých elementů to dává logický smysl a neodporuje to definici znalostní jednotky.

Element X – je formální popis problémové situace a vychází z analýzy dané situace. Popis problémové situace musí být jednoznačný. Popisuje oblast, prostor, ve kterém se nachází více dílčích problémů/elementárních problémů. Z tohoto důvodu je vytvoření lingvistické proměnné a následná fuzzifikace tohoto elementu teoreticky možná, avšak v této práci nebude řešena.

Element Y – je formální popis elementárního problému, který lze smysluplně řešit, na rozdíl od problémové situace, kterou lze z podstaty věci jen vylepšit. Vylepšit ji lze v případě znalostní jednotky aplikací elementu Q. Elementárních problémů může obsahovat znalostní jednotka více. Z tohoto důvodu lze uvažovat o vytvoření lingvistické proměnné a následné fuzzifikaci.

Element Z – cíl řešení elementárního problému musí být jedinečný, aby splňoval podmínku atomičnosti. Vzhledem k tomuto faktu nelze uvažovat o vytvoření lingvistické proměnné a následné fuzzifikaci.

Element Q - je z hlediska znalostní jednotky vhodný pro vytvoření lingvistické proměnné označované kurzívou *Q*. Element Q je konsekvent znalostní jednotky a v obecném popisu znamená řešení elementárního problému. Řešení reálných elementárních problémů může být jedno ale i více, které směřují k jednomu cíli. Současně definice lingvistického členu znalostní jednotky *Q* vyhovuje všem předpokladům a definici znalostní jednotky, viz kapitola „4.1.3 Znalostní jednotky“, i požadavku na atomičnost, čímž je myšlena jednokriteriálnost, která připouští v pojetí znalostí jednotky jeden cíl Z.

Z analýzy jednotlivých elementů vyplývá možnost vytvoření lingvistické proměnné ke dvěma elementům. Jedná se o element Y a element Q. U elementu znalostní jednotky Q lze postupovat vytvořením lingvistické proměnné. Tento stejný způsob lze zvolit i pro element znalostní jednotky Y. To znamená, že *Q, resp. Y* je název lingvistické proměnné a dále *T(Q), resp. T*(Y*)* je množina termů (proměnné *Q, resp. Y*), kde jednotlivé termy z množiny *T(Q) resp. T*(*Y)* jsou $\mathcal{A}$*,* $\mathcal{B}$*,* …, které jsou generovány syntaktickými pravidly *G*. V případě znalostních jednotek vychází generování *G* ze znalostní jednotky. To znamená, že je generováno zbývajícími elementy znalostní jednotky, v případě lingvistické proměnné *Q* jsou to X, Y, Z a v případě *Y* jsou to X, Z, Q. Toto generování vychází z formulace znalostní jednotky ve formě rozšířeného produkčního pravidla. Tato forma má vždy stejnou strukturu a je návodná pro definici lingvistické proměnné *Q* a také jejich významu neboli termů. Pro interpretaci významové části *Q, resp. Y*, slouží sémantické pravidlo *M*, pomocí kterého je každému termu (jazykovému výrazu) $\mathcal{A} \in T(Q)$*, resp.* $\mathcal{A} \in T(Y)$ přiřazen jeho význam, fuzzy množina *A* na universu *U*:

$$M: \mathcal{A} \rightarrow A \tag{48}$$

Schéma znalostní jednotky v analytické formě s rozšířením o množinu lingvistické proměnné *T(Q)* a *T(Y):*

Znalostní jednotka s lingvistickými proměnnými *Q a Y*

X = problémová situace,

Y = elementární problém z množiny termů *T(Y)* lingvistické proměnné *Y*,

Z = cíl řešení elementárního problému,

Q = řešení elementárního problému z množiny termů *T(Q)* lingvistické proměnné *Q*.

5.4.1.3 Fuzzifikace Q

Fuzzifikace znalostní jednotky - elementu Q. Tento element je charakterizován jako konsekvent znalostní jednotky a určuje řešení elementárního problému Q. Ostatní elementy znalostní jednotky spolu s elementem Y upřesňují popis celé problémové situace a jsou antecedentem znalostní jednotky. Lingvistická proměnná *Q* je na místě elementu Q a platí že Q = *Q*. Lingvistická proměnná *Q* je obdobně jako standardní lingvistická proměnná modelována fuzzy množinou *A*.

Fuzzifikace lingvistické proměnné *Q,* resp. definice funkcí příslušnosti jsou dány znalostmi expertů aplikační domény. Znalosti těchto expertů prochází akvizicí přes znalostní inženýry, kteří koordinují a formalizují znalosti a informace. Jedná se zejména o sestavení znalostních jednotek a určení fuzzy funkcí příslušností lingvistické proměnné *Q.* Fuzzy funkce příslušnosti se určují pro množinu termů *T(Q)*, lingvistické proměnné *Q*. Pro každý term $\mathcal{A}$, $\mathcal{B}$, … z množiny *T(Q)* je určena, přiřazena funkce příslušnosti $\mathcal{A} \in T(Q)$.

Přiřazení funkce příslušnosti je při modelování znalostní jednotky a elementu Q podstatným bodem. Jednotlivým termům lingvistické proměnné určuje jejich význam v kontextu fuzzy množiny. Výběr tvaru funkce příslušnosti je možný teoreticky z jakékoli známé funkce.

Tvar funkce závisí na expertech aplikační domény a vychází z popisu problému. Znalostní inženýr explicitně formuluje tvar/průběh funkce ve vhodném prostředí, například v MATLABu. Pokud je více termů, je také více funkcí příslušností, vztah mezi termy a funkcemi příslušností je 1:1.

Fuzzifikace konsekventu znalostní jednotky elementu Q je tedy dána prostřednictvím lingvistické proměnné *Q* , konečným výčtem jejích termů *T(Q)* označovanými $\mathcal{A}$, $\mathcal{B}$, …. a fuzzy číslem definovaným právě pro každý term.

5.4.1.4 Fuzzifikace Y

Fuzzifikace znalostní jednotky - elementu Y. Tento element je součástí popisu problémové situace neboli antecedentu znalostní jednotky a formalizuje elementární problém Y. Lingvistická proměnná *Y* je na místě elementu Y a platí že Y = *Y*. Lingvistická proměnná *Y* je obdobně jako standardní lingvistická proměnná modelována fuzzy množinou *A*. Shodně, jako při fuzzifikaci lingvistické proměnné *Q,* se jedná o definice funkcí příslušnosti, které jsou dány znalostmi expertů aplikační domény. Při této formulaci platí stejná pravidla jako

pro element Q. Fuzzy funkce příslušnosti se určují pro množinu termů T(Y), lingvistické proměnné Y. Pro každý term $\mathcal{A}, \mathcal{B}, \ldots$ z množiny $T(Y)$ je určena, přiřazena funkce příslušnosti $A \in T(Y)$. Jednotlivým termům lingvistické proměnné určuje jejich význam v kontextu fuzzy množiny.

Výběr tvaru funkce příslušnosti je možný teoreticky z jakékoli známé funkce. Tvar funkce závisí na expertech aplikační domény a vychází z popisu problému. Znalostní inženýr explicitně formuluje tvar/průběh funkce ve vhodném prostředí, například v MATLABu. Pokud je více termů, je také více funkcí příslušností, vztah mezi termy a funkcemi příslušností je 1:1.

Fuzzifikace elementárního problému znalostní jednotky elementu Y je tedy dána prostřednictvím lingvistické proměnné Y, konečným výčtem termů $T(Y)$ s jednotlivými termy označovanými $\mathcal{A}, \mathcal{B}, \ldots.$, fuzzy číslem definovaným právě pro každý term.

5.4.1.5 Formalizace fuzzy znalostní jednotky

Formalizace fuzzifikované znalostní jednotky vychází z již definovaných způsobů (Brožová a Houška, 2011). Analytická forma znalostní jednotky je rozšířena o fuzzifikovaný element lingvistickou proměnnou Q a Y.

Charakter elementu	**Část pravidla**	**Element**	**Popis elementu znalostní jednotky**
Crisp X, Z Fuzzy Y	Antecedent	X	Problémová situace $X \neq \emptyset$ je klasická množina
		Y	Elementární problém definovaný linguistickou proměnnou Y,
		Z	Cíl řešení elementárního problému $Z \neq \emptyset$ je klasická množina
Fuzzy Q	Konsekvent	Q	Řešení elementárního problému lingvistickou proměnou Q

Tabulka 12 Znalostní jednotka s fuzzy lingvistickým členem Q; zdroj: autor

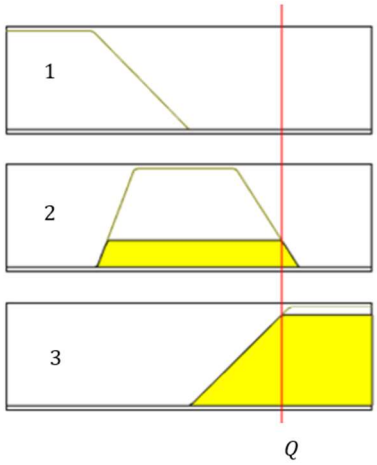
"*Když je třeba v rámci problémové situace X řešit elementární problém Y* z *T(Y), aby bylo dosaženo cíle Z, potom je třeba aplikovat řešení Q z T(Q).*"

Zápis znalostní jednotky ve formě rozšířeného produkčního pravidla s fuzzy lingvistickou proměnnou na místě elementu znalostní jednotky Q= Q a Y = Y:

Když X a *Y* a Z, potom *Q* , (49)

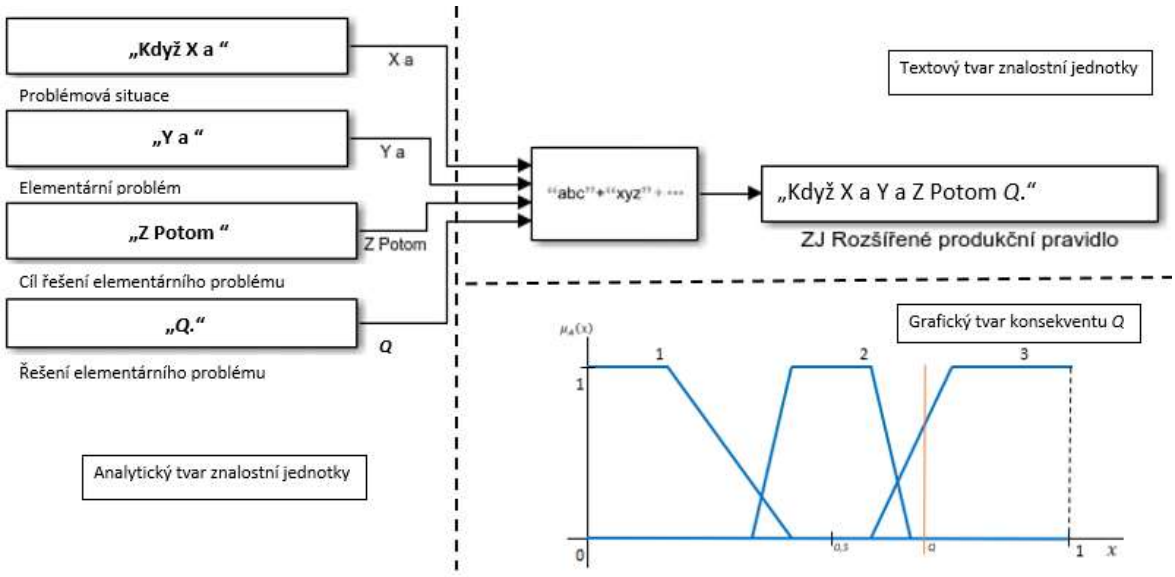
kde X, *Y*, Z je antecedent a *Q* je konsekvent.

Nová možnost grafického zobrazení konsekventu fuzzy znalostní jednotky je inspirována fuzzy Mamdaniho grafickým zobrazením, viz obrázek 50. Jednotlivé funkce příslušnosti na obrázku 50 jsou označeny čísly 1, 2 a 3. K této formě zobrazení je třeba doplnit antecedent znalostní jednotky ve výše uvedených formách. Stejným způsobem lze postupovat pro element znalostní jednotky Y.



Obrázek 50 Grafické zobrazení konsekventu Q fuzzy znalostní jednotky, MATLAB; zdroj: autor

Celkové zobrazení fuzzy znalostní jednotky s fuzzifikovaným elementem Q je na obrázku 51. Ve schématu je naznačeno možné vyhodnocení, hodnota *Q* pro danou fuzzy znalostní jednotku.



Obrázek 51 Schéma fuzzy znalostní jednotky grafický tvar, MATLAB; zdroj: autor

Grafické zobrazení je vytvořeno spojením analytické a textové reprezentace znalostní jednotky k níž je přidána grafická část konsekventu. Ke grafickému zobrazení elementu je vždy nutné doplnit ve vhodném tvaru zbylé elementy znalostní jednotky, tak aby byla zachována její struktura.

5.5 Případová studie fuzzifikace znalostní jednotky

5.5.1 První krok fuzzifikace

Pro příklad bude fuzzifikován element Q. Prvním krokem fuzzifikace znalostní jednotky je sestavení znalostní jednotky jako takové. Jako východisko bude použit příklad z aplikační domény znalostí informačních technologií a konkrétně problematiky aktualizací podnikových informačních systémů ERP, která je uvedena v kapitole „5.7 Případová studie“. Vytvořená znalostní jednotka bude nazvána “Customizace procesu” (úprava standardu systému). Jednotlivé elementy znalostní jednotky jsou v následující tabulce 13.

Charakter elementu	Část pravidla	Element	Popis elementu znalostní jednotky
Crisp X, Y, Z	Antecedent	X	Problémová situace Upgrade ERP
		Y	Elementární problém Zda provést customizaci
		Z	Cíl řešení elementárního problému Uchování firemní logiky a standardu systému
Fuzzy Q	Konsekvent	Q	Řešení elementárního problému Vyhodnotit proveditelnost, uživatelské a nákladové hledisko

Tabulka 13 Analytický tvar fuzzy znalostní jednotky „Customizace procesu"; zdroj: autor

V textové podobě:

„Když je třeba při upgrade ERP vyhodnotit, zda provést customizaci procesu v ERP s cílem uchovat firemní logiku a standard systému, potom je nutné vyhodnotit proveditelnost a uživatelské a nákladové hledisko."

5.5.2 **Druhý krok fuzzifikace**

Dalším krokem je vytvoření lingvistické proměnné *Q.* Lingvistická proměnná *Q*, která je součástí znalostní jednotky, bude nazvána "Customizace". Tato lingvistická proměnná *Q* má následující množinu termů s konečným výčtem hodnot *T(Q)*:

a. $\mathcal{A}$ = nenákladné

b. $\mathcal{B}$ = středně nákladné

c. $\mathcal{C}$ = nákladné

Právě každému uvedenému termu $\mathcal{A} \in T(Q)$ bude přiřazeno fuzzy číslo A.

Universum *U* v tomto případě budou představovat možné velikosti nákladů customizace, které v krajním případě budou znamenat proveditelnost řešení.

5.5.3 **Třetí krok fuzzifikace**

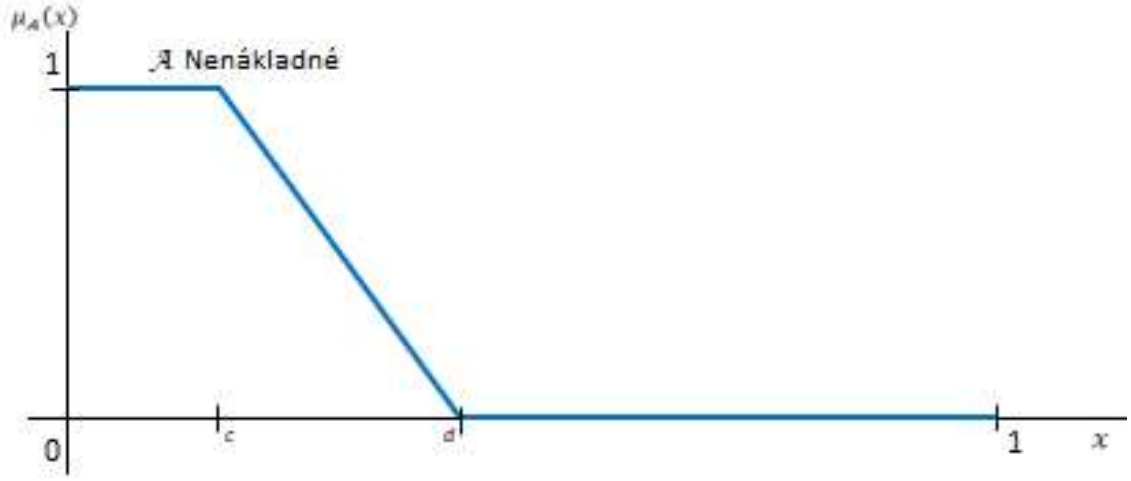
Třetím krokem fuzzifikace je přiřazení funkcí příslušnosti jednotlivým termům. V tomto kroku budou přiřazovány jednotlivým termům ve spolupráci s experty dané problematiky funkce příslušnosti. V tomto kroku byli osloveni experti, kteří pomohli vybrat tvar a průběh funkce příslušnosti.

Pro výběr funkce příslušnosti byli k dispozici tři experti s víceletou praxí a zkušenostmi s upgrade ERP ze dvou různých firem. Tito experti/konzultanti systému ERP z možného výběru typů funkcí příslušnosti zvolili trapezoidní funkci příslušnosti. Výběr probíhal řízenou diskuzí a vyvozením konsensu o podobě tvaru a průběhu funkce. Jednotlivé tvary těchto funkcí jsou uvedeny v následujících bodech:

a) Přiřazení funkce příslušnosti pro term $\mathcal{A}$ = nenákladná customizace je podmíněno názorem experta a antecedentem znalostní jednotky. Tvar funkce příslušnosti může být téměř jakýkoliv známý tvar funkce. Pro příklad byla expertem zvolena trapezoidní funkce R, která je definována parametry $a = b = -\infty$ a následujícím fuzzy číslem (vzorec 50):

$$\mu_A(x) = \begin{cases} 0, x > d \\ \frac{d-x}{d-c}, c \leq x \leq d \\ 1, x < c \end{cases} \tag{50}$$

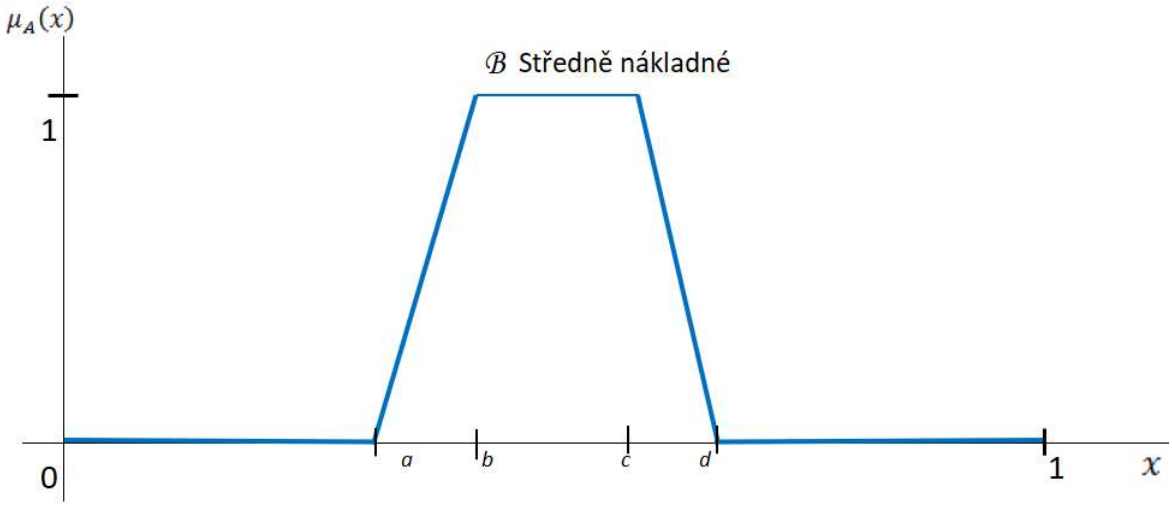
Potom má trapezoidní funkce příslušnosti termu $\mathcal{A}$ lingvistické proměnné Q „Customizace“ následující tvar (obrázek 52):



Obrázek 52 Trapezoidní funkce R; zdroj: autor

b) Funkce příslušnosti termu $\mathcal{B}$ má trapezoidní tvar a je definována čtyřmi parametry (a, b, c, d), kde a je dolní hranice, d horní hranice, a b a c jsou mezemi jádra fuzzy množiny, $a < b < c < d$, je definována následujícím fuzzy číslem (vzorec 51) a zobrazením průběhu na obrázku 53.

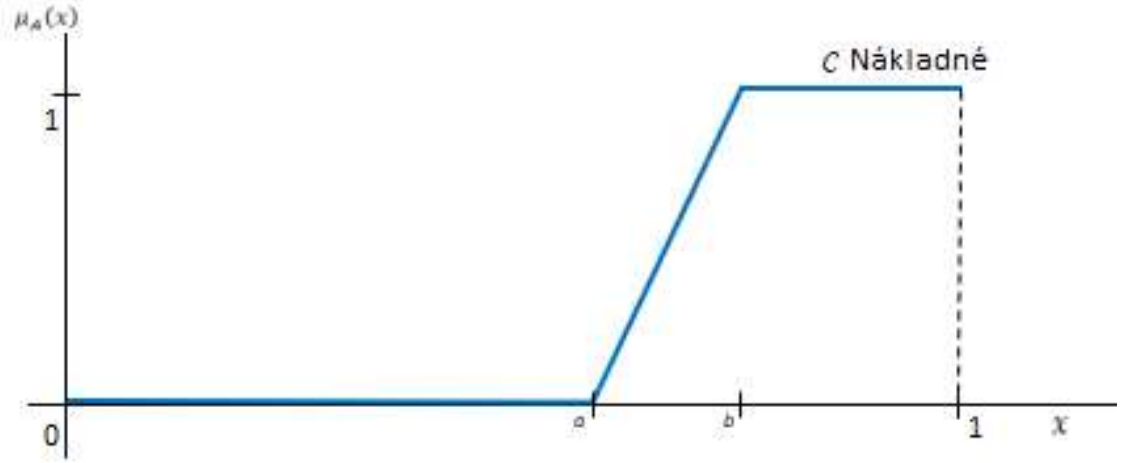
$$\mu_A(x) = \begin{cases} 0, (x < a) or\ (x > d) \\ \dfrac{x-a}{b-a}, a \le x \le b \\ 1, b \le x \le c \\ \dfrac{d-x}{d-c}, c \le x \le d \end{cases} \tag{51}$$



Obrázek 53 Trapezoidní funkce; zdroj: autor

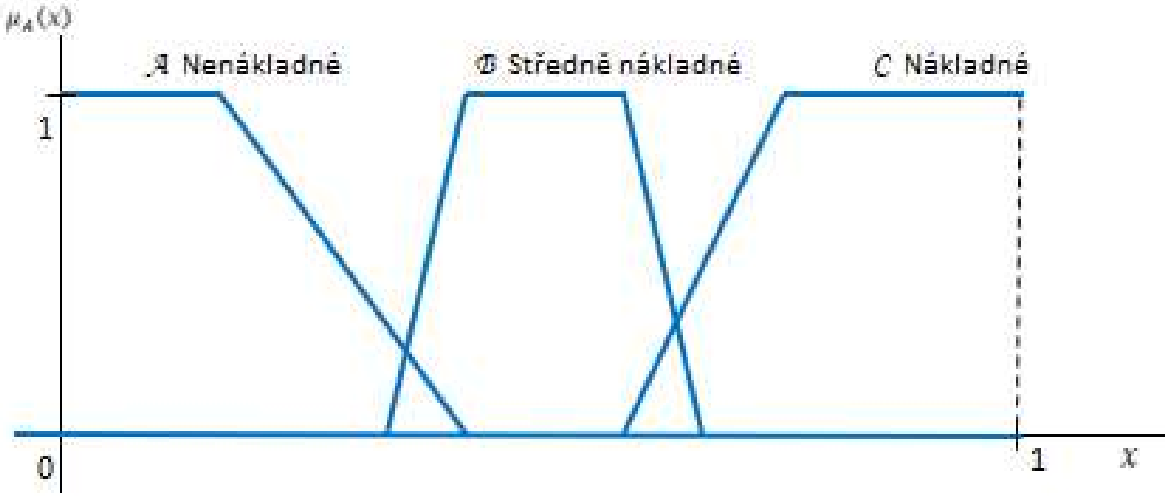
c) Funkce příslušnosti termu C má tvar trapezoidní funkce L, která je definována parametry $c = d = +\infty$, následujícím fuzzy číslem (vzorec 52) a zobrazením průběhu na obrázku 54.

$$\mu_A(x) = \begin{cases} 0, x < a \\ \frac{x-a}{b-a}, a \leq x \leq b \\ 1, x > b \end{cases} \quad (52)$$



Obrázek 54 Trapezoidní funkce L; zdroj: autor

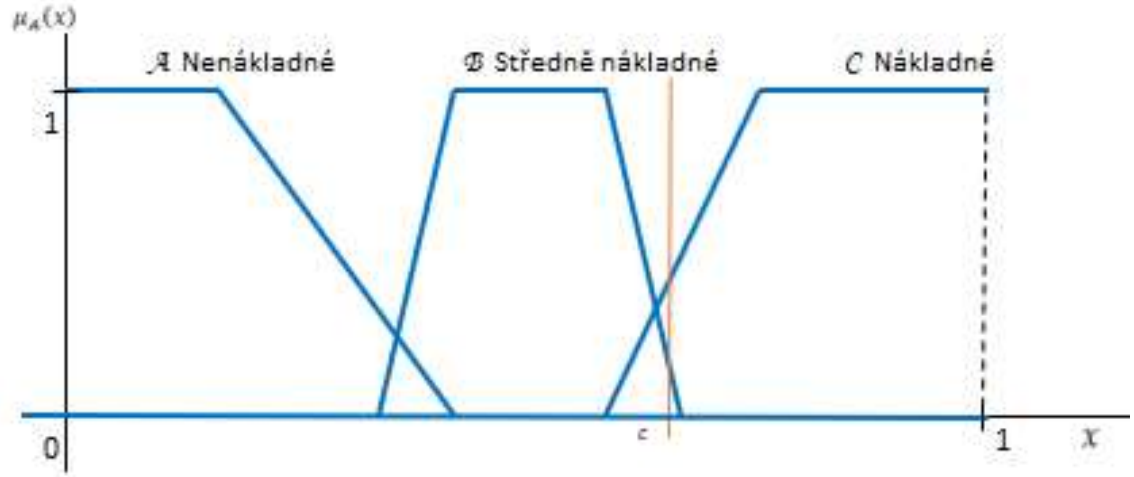
Výsledné zobrazení lingvistické proměnné Q "Customizace" s přiřazenými funkcemi příslušnosti je na obrázku 55. Při aplikaci znalostní jednotky na reálnou situaci lze posuzovat do jaké míry by bylo nákladné/výhodné případně proveditelné provádět customizaci. V této podobě je konsekvent neboli řešení elementárního problému připravený na modelování různých rozhodovacích situací.



Obrázek 55 Konsekvent Q; zdroj: autor

5.5.4 **Čtvrtý krok fuzzifikace**

Čtvrtým a posledním krokem je formalizace znalostní jednotky a její vyhodnocení. Vyhodnocení je možné provést v základní analytické podobě v grafickém schématu nebo v textové podobě. Pro příklad vyhodnocení stávající znalostní jednotky bude uvažována konkrétní customizace s názvem "Pracovní výkazy" s výší nákladů této customizace $c = 0{,}6$. Základní zobrazení řešení elementárního problému Q (konsekventu znalostní jednotky) je pomocí grafu (obrázek 56), kde je zobrazena hodnota c.



Obrázek 56 Grafické vyhodnocení Q; zdroj: autor

V grafickém zobrazení Q lze odečíst pro zadanou hodnotu c slovní hodnocení customizace. V tomto případě je doporučení customizaci pracovních výkazů "spíše neprovádět". Vyhodnocenou znalostní jednotku lze uvést také v analytické podobě.

Charakter elementu	Část pravidla	Element	Popis elementu znalostní jednotky
Crisp X, Y, Z	Antecedent	X	Upgrade ERP
		Y	Zda provést customizaci (Pracovní výkazy)
		Z	Uchování firemní logiky a standardu systému
Fuzzifikované Q	Konsekvent	*Q*	Vyhodnotit proveditelnost, uživatelské a nákladové hledisko, *Q* = "spíše neprovádět"

Tabulka 14 Customizace (Pracovní výkazy) analytická forma; zdroj: autor

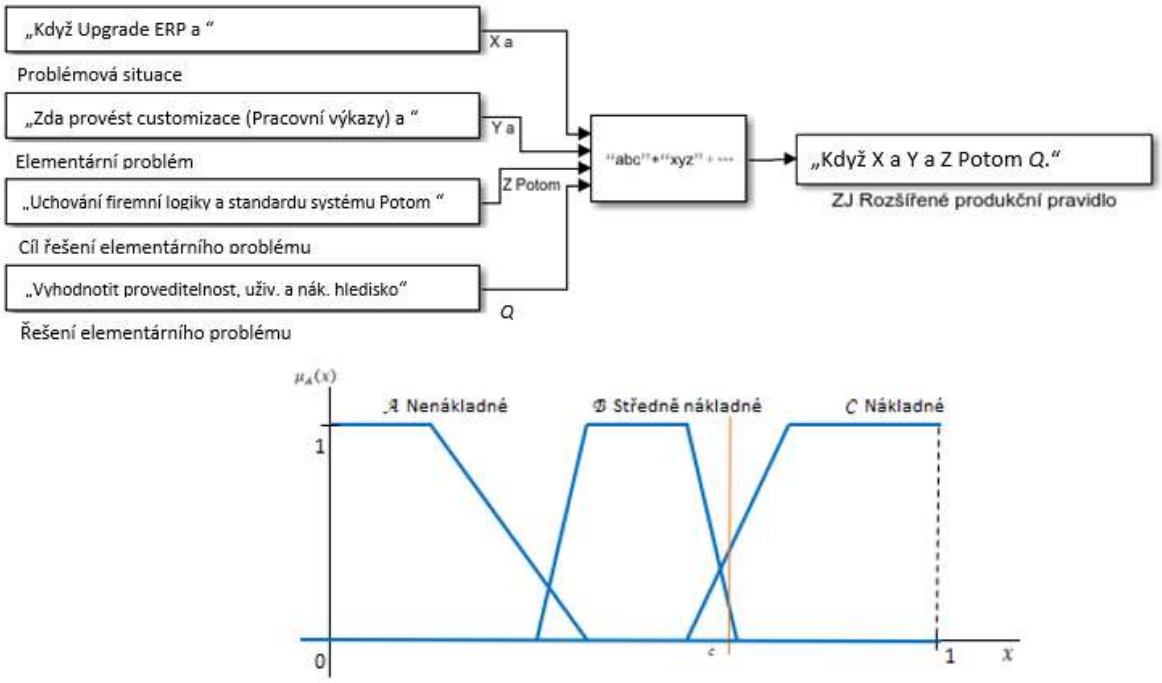
Customizace pracovních výkazů v podobě rozšířeného produkčního pravidla:

Když *X = „Upgrade ERP“* ***a*** *Y = „Zda provést customizaci (Pracovní výkazy)“ a Z = „Uchování firemní logiky a standardu systému“* ***potom*** *Q =”Customizace (Pracovní výkazy) Spíše neprovádět”.*

Customizace pracovních výkazů v textové podobě:

“*Když je třeba v rámci problémové situace* ***upgrade ERP*** *řešit elementární problém,* ***zda provést customizaci (Pracovní výkazy)****, aby bylo dosaženo cíle* ***uchování firemní logiky a standardu systému****, potom je třeba aplikovat fuzzy řešení* ***vyhodnotit proveditelnost, uživatelské a nákladové hledisko, Q = “spíše neprovádět”****.”*

Fuzzy znalostní jednotka v kompletním rozsahu včetně grafického tvaru konsekventu je uvedena na obrázku 57.



Obrázek 57 Schéma fuzzy znalostní jednotky a její grafický tvar, Customizace pracovních výkazů, MATLAB; zdroj: autor

Fuzzifikovaná znalostní jednotka sdružuje dohromady v tomto zobrazení analytickou a textovou formu a přidává dále zobrazení neurčitosti. Zobrazení neurčitosti je vyjádřeno fuzzy množinou a fuzzy funkcemi příslušnosti (viz kapitola „5.4.1 Fuzzifikace znalostní jednotky"). Na grafickém schématu konsekventu *Q* lze pro konkrétní daný elementární problém hodnotu konsekventu odečíst nebo nastavit. Celkovým řešením je po naplnění všech kroků obecný formát pro vytvoření fuzzy znalostní jednotky.

5.6 Model fuzzy znalostních jednotek

Fuzzy logika, fuzzy expertní systémy a fuzzy regulátory jsou vhodným prostředkem k modelování neurčitosti. Fuzzy znalostní jednotky tvoří znalostní kostru, tj. popis problémové situace a napomáhají orientaci a přehlednosti v problémové situaci a tím pádem i v expertním systému. Model fuzzy znalostních jednotek vychází ze základu znalostních jednotek a je modifikován pomocí fuzzy lingvistických proměnných a Mamdaniho grafického způsobu inference.

Cílem této kapitoly je vytvořit model a funkční simulaci fuzzy znalostních jednotek pomocí softwaru MATLAB Simulink. Celý koncept inference a fuzzy znalostní jednotek je převeden

do simulačního prostředí Simulink, ve kterém je vytvořena případová studie. Aplikační doménou je systém ERP (Enterprise Resource Planning), v případě této kapitoly se jedná o problematiku upgrade (aktualizace stávající verze systému ERP), konkrétně hledá odpověď na otázku:

Máme se při aktualizaci systému ERP držet standardu tohoto systému nebo provést vlastní firemní úpravu?

Význam otázky pro pro firmu je následující. Když se přikloní ke standardu, tak budou některé procesy (například vyplňování pracovních výkazů) velmi náročné na realizaci a bude třeba si na to najmout další pracovní sílu. Pokud ale provedou firemní úpravu systému (customizaci), tak se může jednat o dražší řešení, které ale ušetří výdaje na dodatečnou pracovní sílu. Dále se zohledňují faktory, které se odvíjí od druhu používaného softwaru, požadavků konkrétní firmy a dalších, nyní nespecifikovaných, například že customizace není přenositelná do další verze systému a musí se při upgradu opět vyvinout.

5.6.1 Model a simulace fuzzy znalostních jednotek v MATLAB Simulink

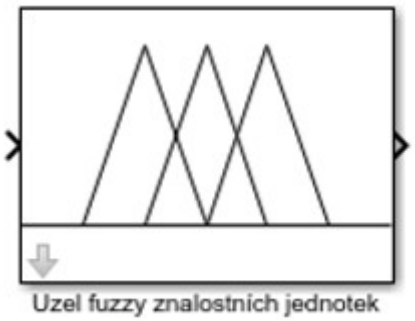
Pro simulaci fuzzy znalostních jednotek byly zvažovány softwary MATLAB (viz kapitola „4.5.3 Fuzzy expertní systém v MATLABu“) a fuzzyTECH. Vzhledem k tomu, že fuzzyTECH je určen pro produkční prostředí a konkrétní případy užití, nikoli pro vlastní vývoj, bylo vybráno softwarové prostředí MATLAB Simulink. Nezanedbatelnou výhodou je také velké množství příruček pro jeho využití. MATLAB používá velké množství komponent neboli toolboxů, které lze využívat v simulacích v Simulinku. Toto prostředí je dostatečně obecné pro simulaci fuzzy inference a lze v něm také vyvíjet vlastní modifikace. V tomto prostředí jsou definovány a vytvořeny struktury pro simulování inference s fuzzy znalostní jednotkou. Výzkumná část práce využívá dílčí výsledky popsané ve východiscích a navazuje tvorbou simulce fuzzy znalostních jednotek při inferenci. Jednotlivé využívané komponenty, upravené ze standardu, a jejich propojení jsou uvedeny v následujících bodech.

- Textové bloky - Pomocí textových bloků jsou vytvořeny znalostní jednotky s lingvistickými proměnnými.
- Subsystém – Subsystém umožňuje seskupit textové bloky a hodnoty pro iniciaci simulace. Subsystém v této práci je znalostní jednotka s nastavitelným vstupním

parametrem. Tento parametr je dán do kontextu fuzzy logiky (viz obrázek 58). Detailně je uveden Subsystém znalostní jednotky v kapitole „5.6.2 Krok 1. Znalostní jednotka v Simulinku“.

Obrázek 58 Symbol subsystému Znalostní jednotka; zdroj: autor

- Fuzzy Logic Toolbox – Ve Fuzzy Logic Toolboxu lze provádět podstatnou část simulace a tou je Mamdaniho grafická inference. Formální popis nástroje Fuzzy Logic Toolbox je v kapitole „4.5.3 Fuzzy expertní systém v MATLABu“. Výsledkem tohoto spojení je možnost fuzzifikace znalostní jednotky a export do simulačního prostředí Simulink.
- Fuzzy Logic Toolbox v Simulinku - V této komponentě exportované do Simulinku jsou definovány funkce příslušnosti, a pokud obsahuje více znalostních jednotek na stejné úrovni, také vztahy mezi těmito fuzzy znalostními jednotkami. Dále v této komponentě dochází k fuzzifikaci znalostních jednotek a následně při simulaci i k jejich inferenci. Tento proces je detailně popsán v kapitole „5.6.3 Krok 2. Znalostní jednotky a Fuzzy Logic Toolbox“. Celá tato komponenta je přenesena do Simulinku jako jeden za stavebních bloků celé simulace. Tomuto bloku je dán pracovní název Uzel fuzzy znalostních jednotek. Tento pojem je vysvětlen v kapitole „5.6.3.1 Uzel fuzzy znalostních jednotek“. Celá interaktivní simulace se odehrává v prostředí MATLAB Simulink a je v modelu znázorněna symbolem na obrázku 59.



Obrázek 59 Interaktivní prvek ve Fuzzy Logic Toolbox; zdroj: autor

V prostředí Simulink jsou doplněny standardní bloky pro simulaci inference fuzzy znalostních jednotek vycházející z kapitoly „4.5.3 Fuzzy expertní systém v MATLABu“.

V prostředí aplikace MATLAB Fuzzy Logic Toolbox je modelována inference znalostních jednotek. Model fuzzy znalostních jednotek je modelován v nadstavbě systému MATLAB Simulink. V Simulinku jsou řešena propojení mezi znalostními jednotkami a fuzzy inferencí, kterou zprostředkovává Fuzzy Logic Toolbox. Výsledkem tohoto propojení je interaktivní model s možností simulace fuzzy znalostních jednotek v MATLAB Simulink. V tomto prostředí je možné simulovat pomocí nastavitelných vstupních parametrů různé scénáře a ladit tak odpovědi systému. Postup vytvoření modelu lze popsat v následujících krocích:

- Krok 1. Vytvoření modelu obecné znalostní jednotky tak, aby byl použitelný do simulace včetně nastavení vstupních hodnot pro simulaci.
- Krok 2. Vložení znalostních jednotek do Fuzzy Logic Toolboxu. Tento krok zahrnuje fuzzifikaci znalostní jednotky a definici vztahů mezi jednotlivými fuzzy znalostními jednotkami. Definice vztahů mezi fuzzy znalostními jednotkami probíhá ve Fuzzy Logic Toolboxu a výsledkem je „Uzel fuzzy znalostních jednotek“.
- Krok 3. Propojení znalostních jednotek a uzlů fuzzy znalostních jednotek. Tento krok je proveden v bloku „Fuzzy Logic Controller with Ruleviewer“.
- Krok 4. Propojení mezi uzly fuzzy znalostních jednotek, případně lze mezi tato propojení přidat zobrazovací blok pro zachycení průběžných stavů při simulaci. Blok pro zobrazení je „Display“. V tomto kroku je podstatné sestavení hierarchie propojení podle problémové situace a charakteristiky uzlů (vstupně/výstupní). Pokud více uzlů ústí do jednoho, je nutné použít blok pro sloučení s názvem „Mux“.
- Krok 5. Iniciace simulace a vložení hodnot vstupních dat. Pro tento krok je použit blok „Constant“/“Slider“. V tomto kroku jsou hodnoty doplněny podle kontextu vytvořené vstupní znalostní jednotky, případně fuzzy znalostní jednotky. Tento prvek simulace je vytvořen v bloku „Subsystem“.

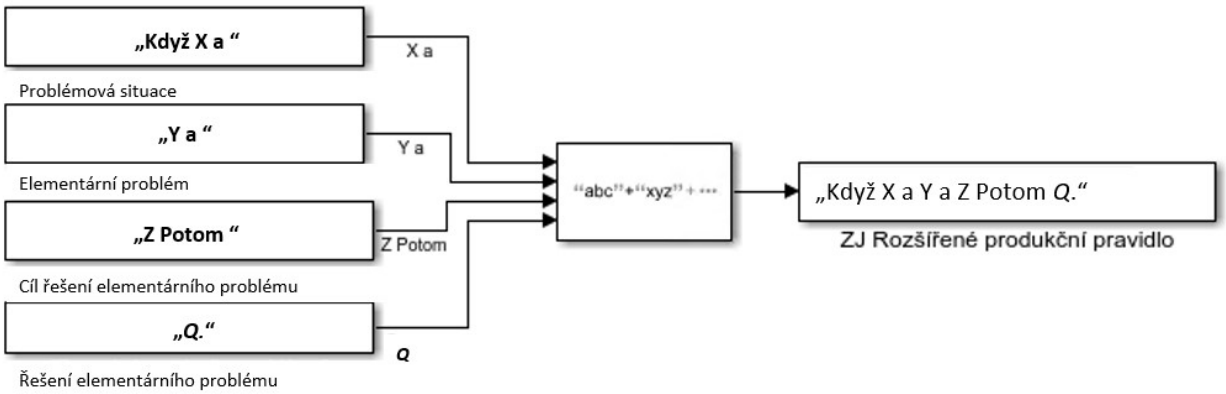
Po dokončení předcházejících kroků lze provést spuštění a simulaci modelu uživatelem. Vlastní nastavení simulovaného modelu probíhá při spuštění simulace. Po startu se objeví dialogová okna pro každý uzel. V dialogovém oknu, uzlu lze interaktivně nastavovat hodnoty vstupních fuzzy znalostních jednotek. Podle výsledků dílčích uzlů lze upravovat

například pravidla mezi vstupními a výstupními jednotkami, nebo jejich váhy a dále analyzovat výsledky modelu a porovnávat je s názory expertů.

5.6.2 **Krok 1. Znalostní jednotka v Simulinku**

Simulink disponuje standardně nástroji k provádění dynamických simulací. V tomto simulačním prostředí lze propojovat toolboxy, které MATLAB nabízí a vytvořit kompaktní simulaci.

Prvky simulace v Simulinku jsou knihovny bloků, toolboxy a jejich spojení. Znalostní jednotka je v simulaci definována a zobrazena pomocí textových konstant (String constant) a spojením těchto konstant v bloku spojení textových konstant (String concatenate) – viz obrázek 60. Zobrazení znalostní jednotky umožňuje blok zobrazení (Display). Obecný model je využit pro jakoukoli znalostní jednotku v analytické podobě a simulace provede spojení do rozšířeného produkčního pravidla a případně také do textové podoby, pokud by to bylo pro simulaci vhodné. Jelikož *Q* je lingvistickou proměnnou, která má množinu termů a dále v simulaci těmto termům bude přiřazena funkce příslušnosti, je pro simulaci vhodné formalizovat znalostní jednotky, které vytváří vstupní parametry simulace a také výstupní znalostní jednotky.

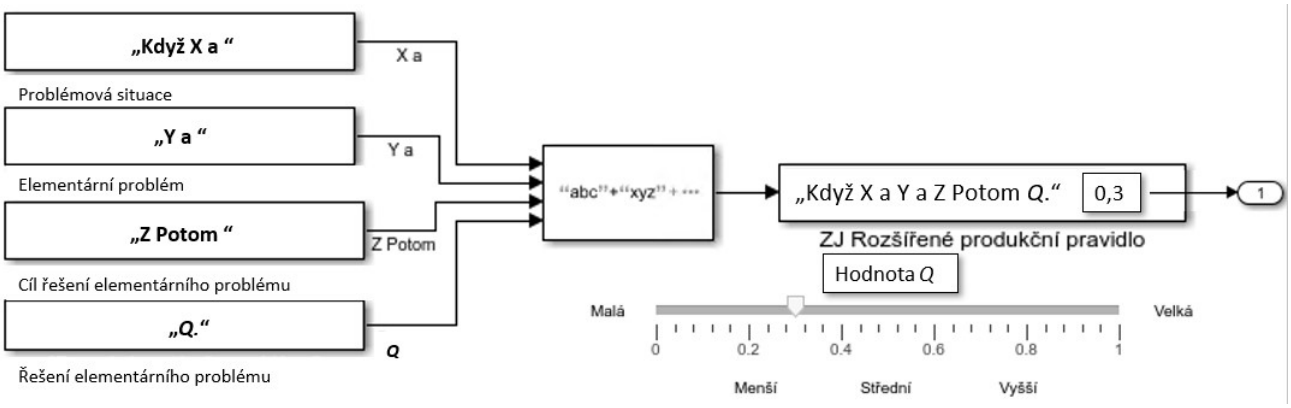


Obrázek 60 Znalostní jednotka v Simulinku; zdroj: autor

Pro vytvoření dynamické simulace je nutné doplnit „hodnotu“ lingvistické proměnné elementu *Q* a definovat její termy. Hodnotu nastavuje uživatel řešící danou problémovou situaci X, z univerza definovaného pro lingvistickou proměnnou a její termy. Tato hodnota

lingvistické proměnné *Q* musí zohledňovat antecedent znalostní jednotky neboli ostatní elementy a zvolené rozpětí univerza *U*.

Nastavením hodnoty se nejprve iniciuje simulace a poté je možné vstupní parametry simulace měnit. Nastavení hodnoty *Q* lingvistické proměnné zprostředkovává blok konstanta (Constant). Zejména pro termy lingvistické proměnné *Q* a uživatelské rozhraní je využit blok posuvník (Slider), který slouží pro přesné nastavení hodnoty *Q* v kontextu termů. Na schématu (obrázek 61) je zobrazena znalostní jednotka v obecném tvaru pro simulaci.



Obrázek 61 Znalostní jednotka a Slider v Simulinku; zdroj: autor

Znalostní jednotka v obecném tvaru pro simulaci je vytvořena v bloku subsystém (Subsystem), který tvoří samostatný zapouzdřený objekt. Tento objekt předává do dalšího chodu simulace definovanou hodnotu elementu *Q*. Hodnota *Q* je vstupním parametrem pro Fuzzy Logic Toolbox, kde probíhá fuzzy inference.

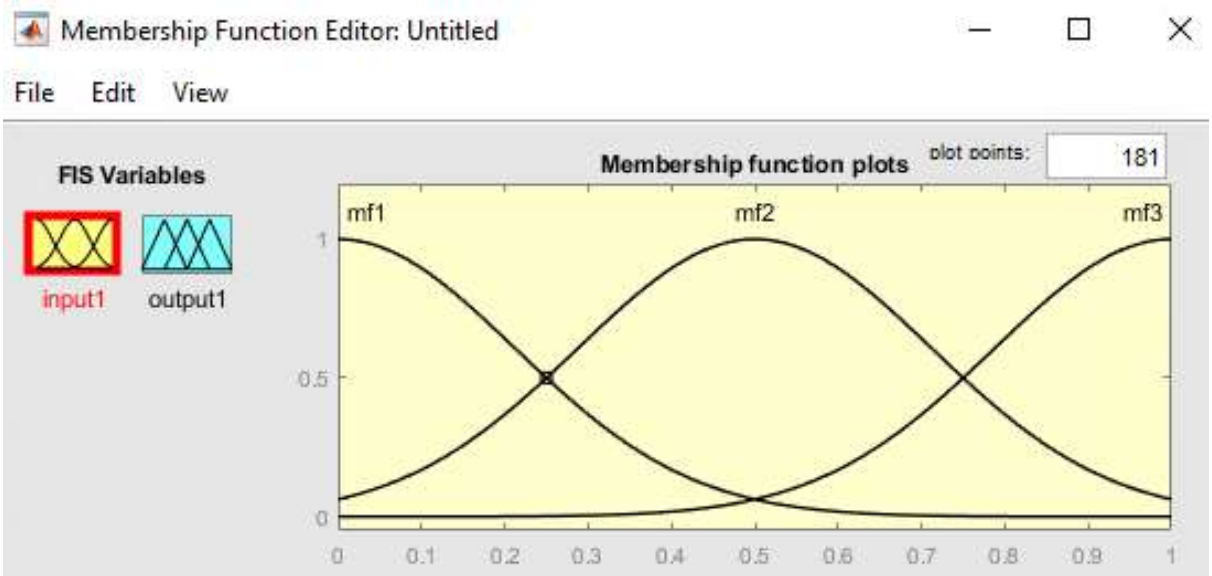
Vývojové prostředí umožňuje kompilovat zdrojový kód do programovacího jazyka C. Celý kód modelu případové studie Customizace je v příloze „Příloha 1. Kód C vytvořeného modelu“. Zdrojový kód pro definici a načtení znalostní jednotky, obdobný jako v modelu, ale obecného charakteru, má následující podobu:

```
% Znalostní jednotka
x = input('Problemova situace: ','s');
y = input('Elementární problém: ','s');
z = input('Cíl řešení elementárního problému: ','s');
q = input('Řesení elemntárního problému: ','s');
eval x;
eval y;
eval z;
eval q;
```

5.6.3 **Krok 2. Znalostní jednotky a Fuzzy Logic Toolbox**

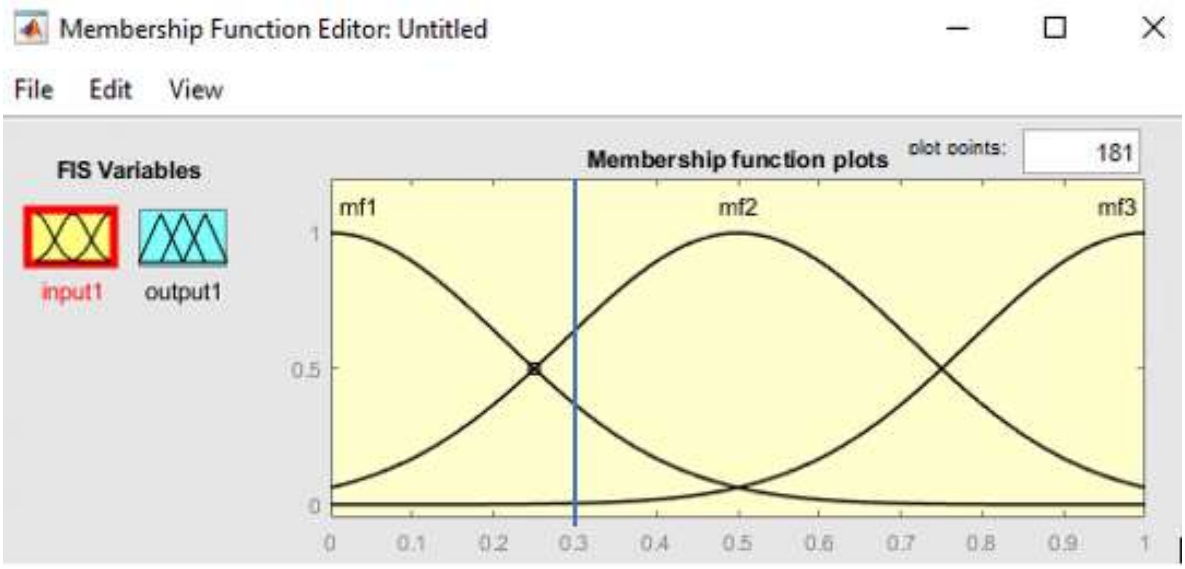
Fuzzy Logic Toolbox zprostředkovává fuzzifikaci lingvistické proměnné *Q*, která vyjadřuje množinu řešení neboli jednotlivých termů $\mathcal{A}$, $\mathcal{B}$, ... z množiny *T(Q)*. Tímto způsobem fuzzifikace znalostní jednotky lze říci, že problémová situace X s konkrétním definovaným problémem Y může mít více řešení s určitou mírou příslušnosti *Q*, které mají jeden cíl Z. Vytvoření fuzzy znalostních jednotek ve Fuzzy Logic Toolboxu pro následný průběh Mamdaniho inference je proveden následovně.

- Jsou použity znalostní jednotky v obecném tvaru určené pro simulaci (viz předcházející kapitola „5.6.2 Krok 1. Znalostní jednotka v Simulinku“). Do těchto obecných subsystémů jsou zadány znalostní jednotky, respektive jejich elementy. Znalostní jednotky jsou zadány v analytické podobě a jejich textová podoba se automaticky generuje. Tímto způsobem je využita analytická i textová forma zápisu znalostní jednotky a vyplývá z ní také kontext řešené situace. Tento fakt je vhodný pro následné nastavení hodnoty *Q,* více o zadání hodnot je uvedeno v kapitole „5.6.4 Krok 3. Fuzzy Logic Toolbox a Simulink“.
- Dále jsou pro element *Q* vytvořeny funkce příslušnosti ty jsou ale pro termy v rozhraní Fuzzy Logic Toolboxu (obrázek 62) pomocí editoru pro zadávání funkcí příslušností. V tomto rozhraní jsou zadány typy funkcí příslušnosti a jejich průběh v definovaném univerzu pro každý term konsekventu *Q* neboli množiny termů *T(Q)*.



Obrázek 62 Rozhraní pro zadávání funkcí příslušností termů z množiny T(Q); zdroj: autor

- Zadáním hodnoty *Q* do fuzzy znalostní jednotky pro iniciaci modelu (viz kapitola „5.6.2 Krok 1. Znalostní jednotka v Simulinku“) se zprostředkovává nastavení vstupních hodnot simulace (obrázek 63). Vstupní hodnoty jsou přebírány Fuzzy Logic Toolboxem, kde jsou interpretovány jako vstup Mamdaniho inference a je s nimi dále počítáno v simulaci. Takové zobrazení pro vstupní hodnotu Q = 0,3 je uvedeno na obrázku 63. Zároveň je zobrazení na obrázku 63 vyhodnocením konsekventu Fuzzy znalostní jednotky. Fuzzy Logic Toolbox nabízí obdobné zobrazení, ale až po proběhnutí simulace v dialogovém oknu (viz kapitola 5.4.1.5 Formalizace fuzzy znalostní jednotky).



Obrázek 63 Zobrazení vstupní hodnoty (Q = 0,3); zdroj: autor

5.6.3.1 Uzel fuzzy znalostních jednotek

Jak je popsáno níže v kapitole, slouží objekt „Uzel fuzzy znalostních jednotek“ k určení vztahu mezi znalostními jednotkami. Je tvořen ze vstupních a výstupních fuzzy znalostních jednotek a vztahů mezi nimi.

V uzlu fuzzy znalostních jednotek jsou definována pravidla podle vzájemného vztahu znalostních jednotek a kontextu aplikační domény. Uzel fuzzy znalostních jednotek je složen ze vstupních fuzzy znalostních jednotek nebo znalostních jednotek a výstupních fuzzy znalostních jednotek.

Spojením fuzzy znalostních jednotek a jejich vzájemných pravidel (vztahů) vzniká uzel fuzzy znalostních jednotek. Jejím vstupem jsou zadané hodnoty *Q* a výstupem je hodnota

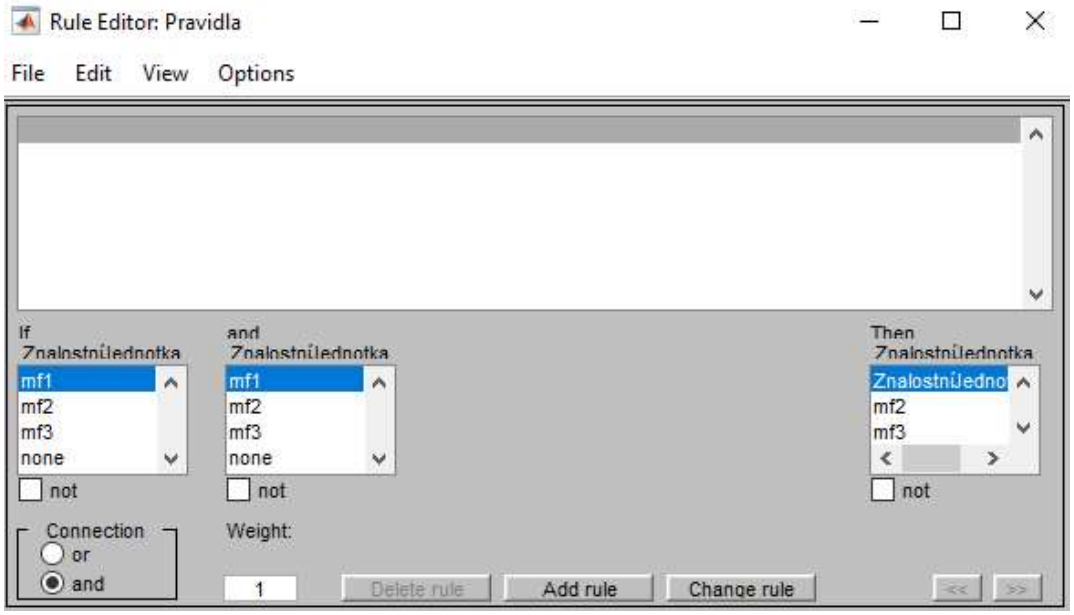
inference (viz kapitola 5.2 „Inference znalostních jednotek“) $Q^1 \rightarrow Y^2$, respektive pro pokračující inferenci hodnota Q^2 výstupní znalostní jednotky. Uzel fuzzy znalostních jednotek se dvěma vstupními jednotkami a jednou výstupní má strukturu, která je uvedena na obrázku 66. Při tvorbě znalostního modelu v simulaci fuzzy znalostních jednotek se postupuje následujícím způsobem:

- Vytvoření fuzzy znalostních jednotek nebo znalostních jednotek.
    - Určení funkcí příslušností k jednotlivým termům.
- Sestavení hierarchie – vstupní/výstupní (fuzzy) znalostní jednotky.
- Určení pravidel pro inferenci a vztahy mezi znalostními jednotkami.

Jednotlivé znalostní jednotky jsou definovány s fuzzifikovaným členem Q (viz kapitola „5.4.1 Fuzzifikace znalostní jednotky“), tj. lingvistickou proměnnou, a to jednak pro vstupní proměnné, tak i pro výstupní. Dále je nutné definovat pravidla pro inferenci a vztahy mezi znalostními jednotkami v uzlu. Mamdaniho inference v uzlu fuzzy znalostních jednotek zprostředkovává přechod do nové znalostní jednotky. Postup definice pravidel se dá srovnat s postupem vytváření pravidel pro databázi znalostí v expertním systému. Rozdíl plyne z definice znalostních jednotek, které celému řešení dávají kostru a kontext znalostí v jednotlivých fuzzy znalostních jednotkách a následně uzlech.

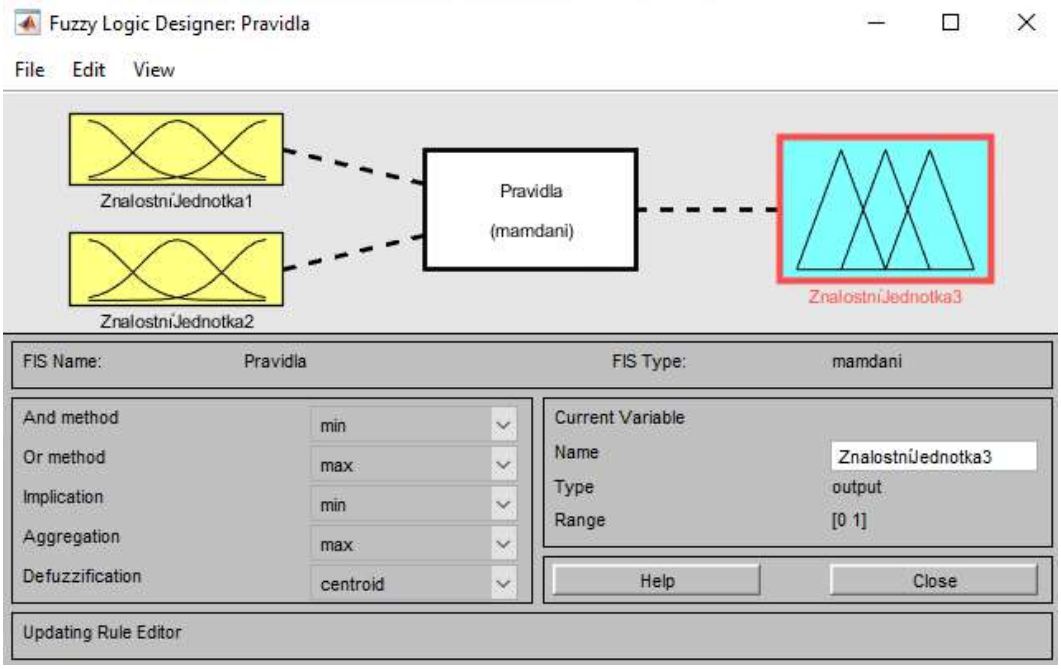
Dále je nutné v toolboxu definovat pravidla pro řešení situace, kdy je více fuzzy znalostních jednotek na vstupu a případně i na výstupu (obrázek 64). Pro řešení této situace jsou využita klasická logická produkční pravidla používaná ve strategiích řetězení (viz kapitola „4.1.1 Produkční pravidla, řetězení“). Pravidla jsou definována expertem dané aplikační domény a znalostním inženýrem.

Při definici těchto pravidel slouží fuzzy znalostní jednotky k explicitnímu vyjádření znalosti a přiblížení situace. Tato pravidla určují vztah mezi fuzzy znalostními jednotkami, respektive konsekventy Q fuzzy znalostních jednotek. Tyto vztahy a fuzzy znalostní jednotky vytvářejí uzly fuzzy znalostních jednotek.



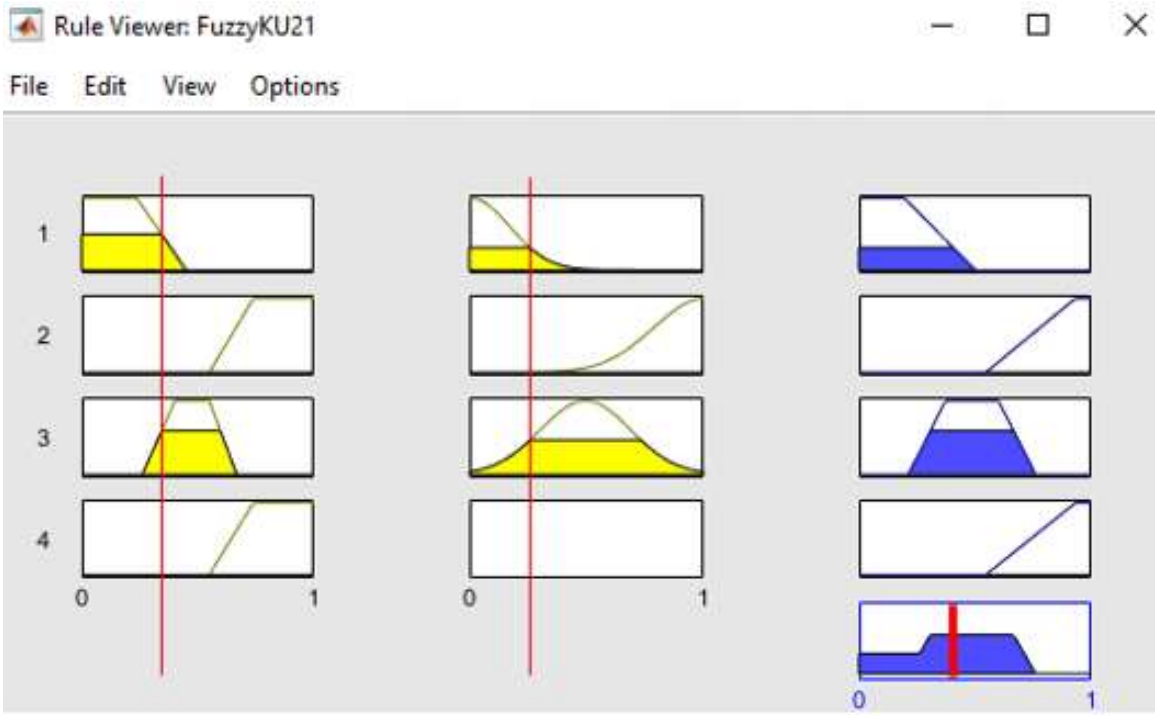
Obrázek 64 Rozhraní pro zadávání pravidel; zdroj: autor

Po aplikaci těchto kroků ve Fuzzy Logic Toolboxu vznikne uzel fuzzy znalostních jednotek, se kterým lze po uložení dále pracovat v Simulinku na úrovni objektu/bloku systému. Uzel fuzzy znalostních jednotek tvoří další stavební blok modelu fuzzy znalostních jednotek. Jeho zdrojový kód zakomponovaný do příkladu je v příloze „Příloha 1. Kód C vytvořené simulace“ a je uvozen textem „Fuzzy Logic Controller“. Tuto komponentu lze také využít například v expertním systému nebo jako fuzzy regulátor (obrázek 65).



Obrázek 65 Uzel simulačního modelu fuzzy znalostních jednotek se dvěma vstupními jednotkami a jednou výstupní; zdroj: autor

Vytvořené uzly fuzzy znalostních jednotek lze samostatně vyhodnotit. Dílčím způsobem lze optimalizovat výsledky jednotlivých částí celé simulace. Vyhodnocení vztahů fuzzy znalostních jednotek a inference neboli uzlů probíhá pomocí Fuzzy Logic Toolboxu. Zobrazení vyhodnocení Mamdaniho grafické inference je na obrázku 66.



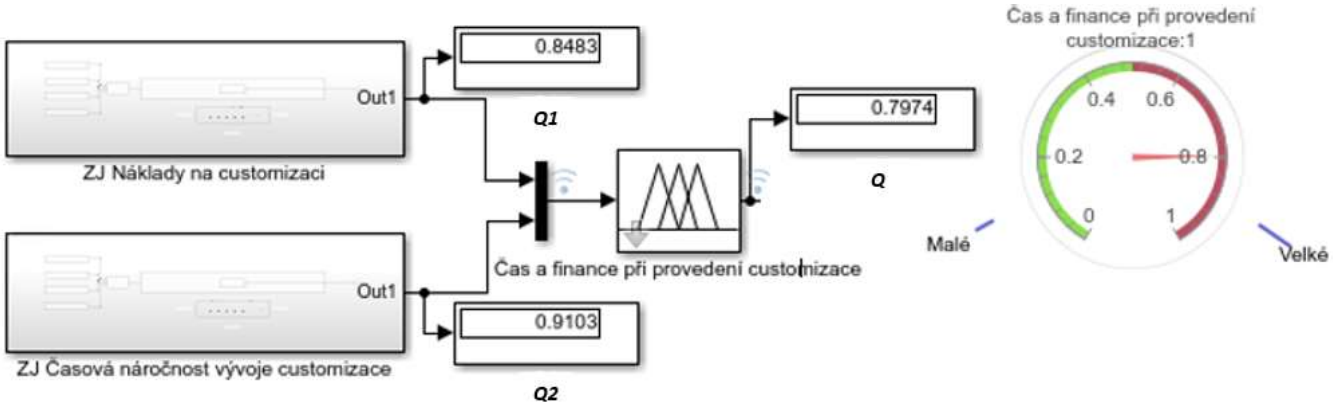
Obrázek 66 Vyhodnocení uzlu fuzzy znalostních jednotek se dvěma vstupními jednotkami a jednou výstupní; zdroj: autor

Jednotlivé řádky na obrázku představují definovaná pravidla, sloupce fuzzy znalostní jednotky, respektive jejich konsekvent *Q*. Sloupec na pravé straně představuje vyhodnocení pravidel výstupní fuzzy znalostní jednotkou. Pod tímto sloupcem je defuzzifikovaný výsledek metodou centroid. Po vytvoření uzlů je potřeba tyto uzly mezi sebou logicky podle problémové situace propojit a případně ohodnotit váhy těchto spojení. Váhy lze také přiřadit jednotlivým pravidlům mezi vstupními a výstupními znalostními jednotkami. Metoda defuzzifikace je zvolena s ohledem na řešený elementární problém. Ve většině případů je to metoda centroid.

5.6.4 **Krok 3. Fuzzy Logic Toolbox a Simulink**

Jednotlivé stavební bloky v podobě uzlů fuzzy znalostních jednotek vytvořené ve Fuzzy Logic Toolboxu jsou přesunuty do simulačního prostředí. Import do Simulinku je proveden

prostřednictvím bloku s názvem „Fuzzy Logic Controller with Ruleviewer“. Každý fuzzy uzel znalostních jednotek je při tomto typu inference vždy iniciován hodnotou elementu *Q*. Hodnota elementu *Q* může být zadána uživatelem, to znamená faktickou hodnotou, kterou nastaví podle kontextu znalostní jednotky. Druhou možností je, že hodnota elementu *Q* vychází z předešlého uzlu fuzzy znalostních jednotek. Vstupních hodnot elementu *Q* může být libovolný počet.



Obrázek 67 Uzel fuzzy znalostních jednotek v prostředí Simulink; zdroj: autor

5.6.5 **Krok 4. Simulace modelu v Simulinku**

Kompaktní vytvoření simulace v Simulinku následuje po vytvoření všech jednotlivých předešlých kroků. Jednotlivé funkční celky modelu pro simulaci jsou:

A. Objekt (zapouzdřená) (fuzzy) znalostní jednotka do bloku subsystém „Subsystem“

B. Uzel systému fuzzy znalostních jednotek na úrovni bloku systému „Fuzzy Logic Controller with Ruleviewer“.

Oba funkční celky modelu jsou také převedeny do zdrojového kódu C, který je uveden v příloze „Příloha 1. Kód C vytvořeného modelu“. Tyto bloky jsou podle cíle řešení dané problematiky hierarchicky seskupeny do výsledné architektury řešení. Strategie řetězení (dopředné nebo zpětné) musí být zvolena jako předpoklad modelu pro simulaci podle charakteru řešeného problému (viz kapitola „4.3.3 Inference znalostí v expertních systémech“). Subsystém znalostní jednotky je využit pro vstup modelu. Počátek/první úroveň modelu tvoří subsystémy znalostních jednotek a směřují do uzlu fuzzy inference.

Obecně platí, že pokud je více subsystémů znalostních jednotek nebo uzlů na stejné úrovni a ústí do jednoho uzlu, je použit blok „Mux“, sloučení. Pro zobrazení stavů hodnot v průběhu

simulace je využíván blok „Display“, zobrazení. Výsledný blok modelu je typu uzel fuzzy znalostních jednotek s definovanou výstupní znalostní jednotkou typu subsystém. Výstup je dále rozšířen o grafické zobrazení výsledku inference v podobě budíku „Gauge“, s přiřazenými termy výsledné lingvistické proměnné *Q*.

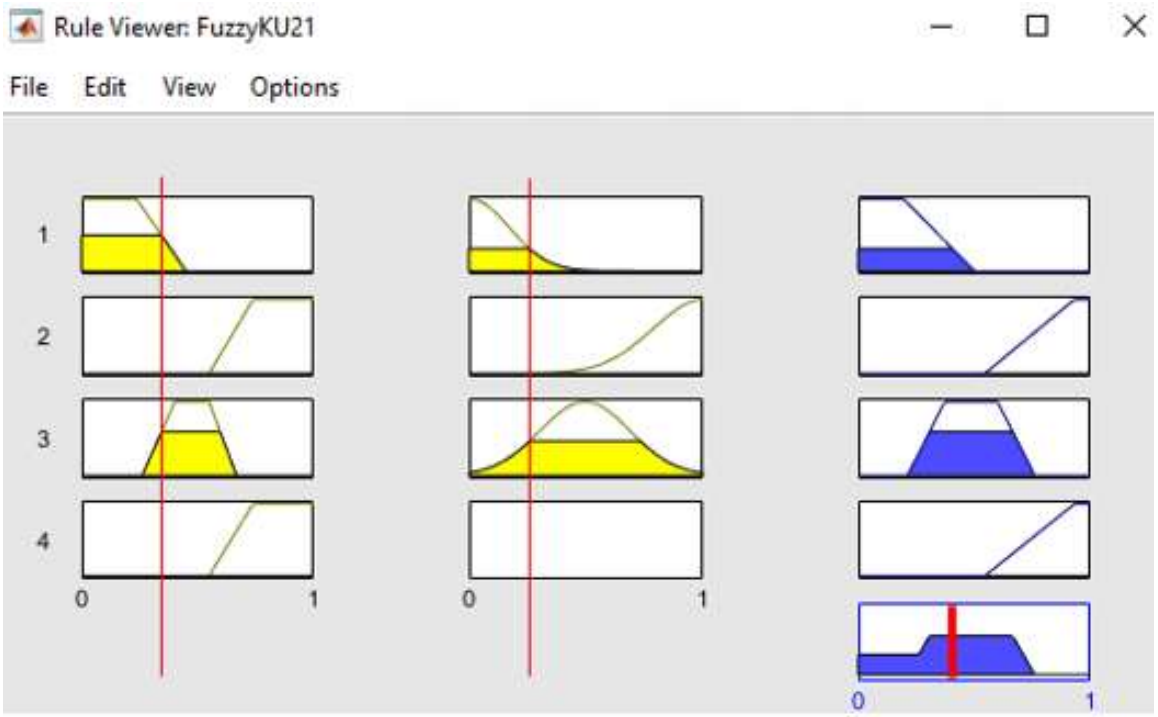
5.6.6 **Krok 5. Práce s modelem**

Po sestavení a vytvoření modelu v prostředí Simulink lze provádět simulaci jeho chování, zejména jeho odpovědí a chování vytvořené inference fuzzy znalostních jednotek. Před iniciací modelu a spuštěním simulace se musí provést nastavení vstupních hodnot v jednotlivých fuzzy znalostních jednotkách, subsystémech. Pro tento krok je použit blok „Slider“, posuvník. Po tomto nastavení lze spustit simulaci modelu.

Při spuštění simulace proběhne nejprve kompilace a poté proběhne inference se zadanými parametry. Model při spuštění zobrazí dialogová okna inference lingvistických proměnných (obrázek 68) všech jednotlivých uzlů fuzzy znalostních jednotek včetně výsledného. V dialogu je graficky zobrazen průběh inference včetně funkcí příslušností, pravidel a vyhodnocení jednotlivých konsekventů.

Takto zobrazené jednotlivé konsekventy lze vztáhnout na výstupní fuzzy znalostní jednotky. Zakomponování konsekventu do výstupní fuzzy znalostní jednotky není podmínkou inference a je možné ji provést pro všechny fuzzy znalostní jednotky v simulaci. Tento proces probíhá doplněním konsekventu na místo elementu fuzzy znalostní jednotky, podle strategie inference například Y^1 neboli Q^2 pro pokračující inferenci. Do subsystému fuzzy znalostní jednotky je zavedena hodnota konsekventu na místo elementu a je přiřazeno slovní hodnocení podle definovaných termů fuzzy znalostní jednotky.

Celkový výsledek, konsekvent, je zobrazen dialogovým oknem Mamdaniho grafické inference (obrázek 68). Výsledek je dále formalizován do textové podoby fuzzy znalostní jednotky, která je zprostředkována blokem subsystém fuzzy znalostní jednotky. Pro okamžité vyhodnocení slouží grafické zobrazení s přiřazenými patřičnými termy k celkovému výslednému konsekventu fuzzy znalostní jednotky.



Obrázek 68 Dialogové okno inference lingvistických proměnných v prostředí Simulink; zdroj: autor

5.6.6.1 Možnosti zobrazení výsledků modelu

Podle odpovědi modelu na zadané iniciační parametry lze testovat a verifikovat, zda model odpovídá adekvátně. Odpovědi modelu lze porovnávat s výsledky odpovědí lidských expertů a zjišťovat chování při zadání krajních hodnot. V případě rozporů a neadekvátních odpovědí lze provádět interaktivně změny a přenastavení vstupních parametrů v modelu a opětovně model spouštět a simulovat tak jeho chování.

Pro případ úprav uzlů ve Fuzzy Logic Toolboxu lze provádět úpravy libovolného charakteru a ihned zobrazit změnu tohoto uzlu. Typicky se jedná o úpravu pravidel, vah a tvarů funkcí příslušnosti. Po ukončení úprav a ladění jednotlivých uzlů se provede uložení. Po uložení do pracovního prostoru MATLABu a opětovném spuštění v Simulinku dojde k vyhodnocení celkového výsledku úprav. Tímto způsobem lze posoudit celkový dopad funkčních úprav na model. V následující případové studii bude model porovnán s odhady expertů v dílčích i celkovém výsledku.

5.7 Případová studie

V případové studii je čerpáno ze složité rozhodovací situace, zda provádět „Customizace“ (vlastní úpravy) v podnikovém systému nebo se držet standardu garantovaného výrobcem. Rozhodovacím bodem pro zvážení customizací, nebo držení se standardu systému je „Upgrade“ (Aktualizace verze) nebo zavádění nového procesu. V těchto situacích je nutné se rozhodnout, jak procesy nastavit. Zda jít cestou standardu zaručeném výrobcem, nebo si procesy v systému přizpůsobit. Počet bodů zájmu neboli očekávaných rozhodnutí je přímo úměrný počtu procesů v stávajícím systému.

Stávající procesy se definují a popisují na základě Gap-Fit analýzy. Část této Gap-Fit analýzy je součástí scénáře v kapitole „5.7.2 Testování scénářů na modelu“. V této části analýzy jsou popsány procesy v nové verzi systému s informací, zda procesy lze realizovat ve standardu systému nebo ne. Rozhodovací situace je již definována ve fuzzy znalostních jednotkách, tj. způsobem popsaným v kapitole „5.4.1 Fuzzifikace znalostní jednotky“, které jsou uvedeny níže v tabulkách (15 až 21). Tento model fuzzy znalostních jednotek má název „Customizace“.

Předpokladem vytvoření modelu je spolupráce s experty na ERP systémy. Musí být definovány potenciální fuzzy znalostní, které budou zadány do komponenty v Simulinku. Model fuzzy znalostních jednotek s názvem „Customizace“ je sestaven ze sedmi fuzzy znalostních jednotek. Jedná se o čtyři fuzzy znalostní jednotky vstupní, dvě vstupně/výstupní a jednu fuzzy znalostní jednotku výstupní.

X	Customizace z hlediska nákladů
Y	Zjistit finanční nákladnost
Z	Posoudit nákladnost customizace
Q	Provést odhad nákladů customizace (Termy: Malé náklady, Střední náklady, Velké náklady)

Tabulka 15: Fuzzy znalostní jednotka Finance/vstupní; zdroj: autor

V textové podobě má znalostní jednotka tento tvar:

Když je třeba v rámci problémové situace Customizace z hlediska nákladů řešit elementární problém Zjistit finanční nákladnost, aby bylo dosaženo cíle Posoudit nákladnost customizace, potom je třeba aplikovat řešení Provést odhad nákladů customizace.

X	Vytvoření customizace
Y	Časová náročnost vývoje
Z	Zachovat firemní logiku
Q	Provést odhad časové náročnosti vývoje (: Krátká, Střední, Dlouhá)

Tabulka 16: Fuzzy znalostní jednotka Čas/vstupní; zdroj: autor

X	Co umožňuje standard systému ERP
Y	Standard definovaného procesu firmou (Př: Zadávání pracovních výkazů)
Z	Zachování standardu systému (bez customizací)
Q	Vyhodnotit Gap-Fit analýzu (: Standard, Firemní logika)

Tabulka 17: Fuzzy znalostní jednotka Standard vs. Firemní logika/vstupní; zdroj: autor

X	Nároky na klíčové uživatele při vývoji a při provozu
Y	Časové vytížení klíčových uživatelů
Z	Eliminovat zatížení klíčových uživatelů
Q	Provést odhad časového vytížení (Malé, Střední, Velké)

Tabulka 18: Fuzzy znalostní jednotka Klíčoví uživatelé/vstupní; zdroj: autor

X	Časové a finanční náklady po upgrade
Y	Časové a finanční náklady při použití řešení customizace
Z	Minimalizace pracovních hodin
Q	Analyzovat časové a finanční náklady (: Malé, Střední, Velké)

Tabulka 19: Fuzzy znalostní jednotka Čas a finance při provedení customizace/vstupně-výstupní; zdroj: autor

X	Jak se změní systém ERP uživatelům
Y	Co budou dělat jinak
Z	Naplnění očekávání uživatelů
Q	Vyhodnotit změny (: Malé, Střední, Velké)

Tabulka 20: Fuzzy znalostní jednotka Změny systému ERP/vstupně-výstupní; zdroj: autor

X	Customizace ERP
Y	Zda provést customizaci
Z	Aby byl proveden Upgrade
Q	Vyhodnotit řešení customizace (Necustomizovat, Customizace?, Customizovat)

Tabulka 21: Fuzzy znalostní jednotka Customizace/výstupní; zdroj: autor

5.7.1 **Aplikace kroků pro vytvoření případové studie**

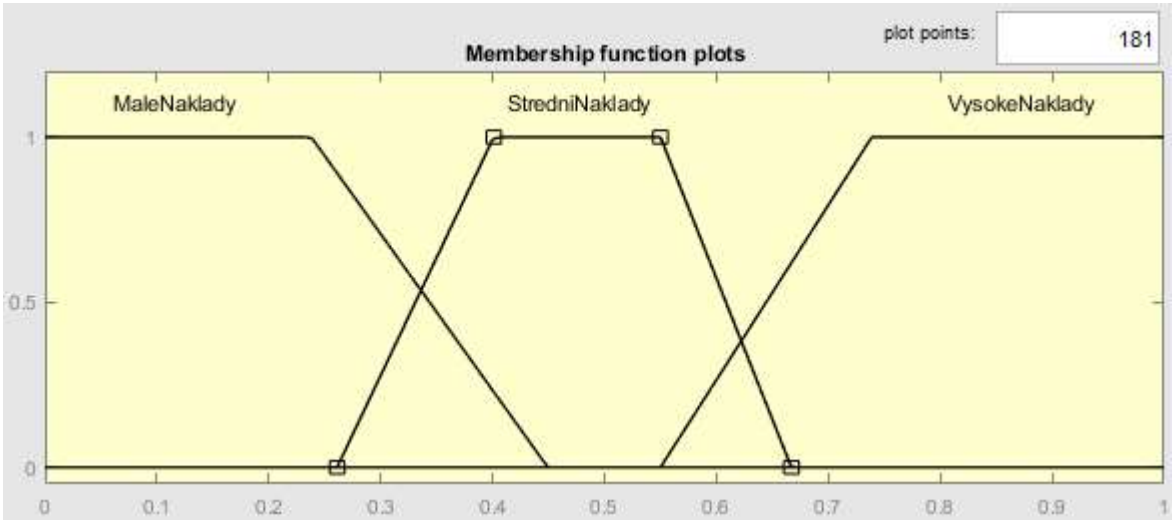
Obecný model je popsán v předchozí kapitole „5.6 Model fuzzy znalostních jednotek“, zde je ukázka naplnění struktur modelu konkrétním případem, problémovou situací popsanou v kapitole „5.7 Případová studie“.

Krok 1. Znalostní jednotka v Simulinku

Prvním krokem sestavení modelu je vytvoření komponent pro zadání fuzzy znalostních jednotek případně znalostních jednotek. Vytvoření komponenty znalostní jednotky je detailně popsáno v kapitole „5.6.2 Krok 1. Znalostní jednotka v Simulinku“.

Krok 2. Znalostní jednotky a Fuzzy Logic Toolbox

Analytické formy fuzzy znalostních jednotek jsou zadány do vytvořených komponent. Z těchto komponent tak vzniknou konkrétní potenciální fuzzy znalostní jednotky již v rámci modelu. Aby se z potenciálních fuzzy znalostních jednotek staly fuzzy znalostní jednotky, je nutné fuzzifikovat element Q. V tomto ohledu je nutná spolupráce experta při odhadu funkcí příslušností pro každou fuzzy znalostní jednotku a její termy. Například pro fuzzy znalostní jednotku Finance/vstupní byly nastaveny následovně (obrázek 69).



Obrázek 69 Funkce příslušnosti; zdroj: autor

Stejným způsobem se postupuje pro všechny fuzzy znalostní jednotky v případové studii. Po definování všech fuzzy znalostních jednotek následuje určení jejich vzájemných vztahů. K tomuto je využit Fuzzy Logic Toolbox, kde probíhá inference fuzzy znalostních jednotek, které mezi s sebou v rámci celé problémové situace mají vztah. Tímto způsobem vzniká uzel fuzzy znalostních jednotek. V této případové studii to jsou tři fuzzy znalostní jednotky.

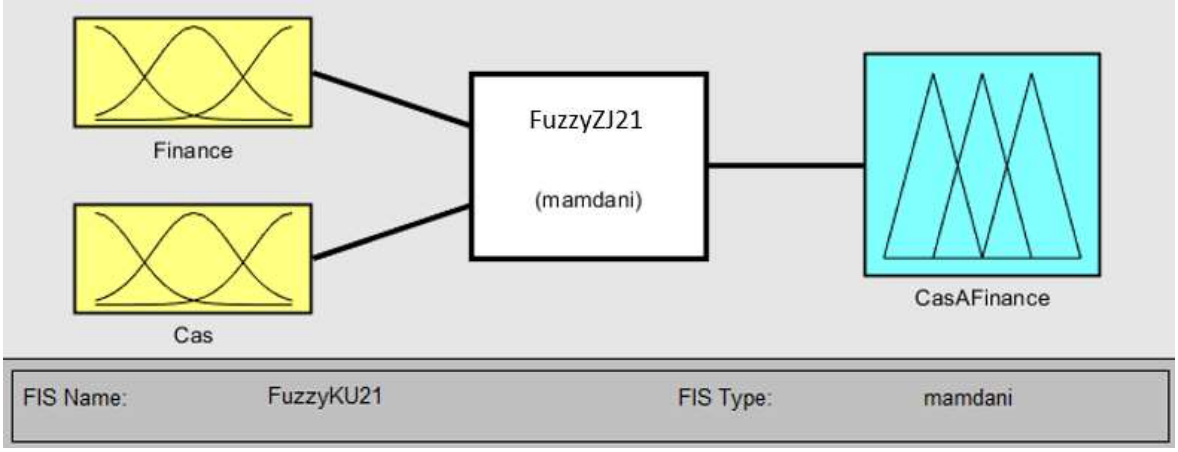
- Fuzzy znalostní jednotka Finance/vstupní
- Fuzzy znalostní jednotka Čas/vstupní
- Fuzzy znalostní jednotka Čas a finance při provedení customizace/vstupně-výstupní

Pro tyto tři jednotky jsou vytvořena pravidla, která umožňují průběh inference, uvedena v tabulce 22. Pravidla se nastavují právě pro každý uzel fuzzy znalostních jednotek. Samotné zadání probíhá v editoru pravidel, který je součástí Fuzzy Logic Toolboxu. Pro tento uzel byla nastavena pravidla v tabulce 22.

Pravidla
1. Když (Finance jsou MaleNaklady) a (Cas je Kratky) potom (CasAFinance jsou Male)
2. Když (Finance jsou VysokeNaklady) a (Cas je Dlouhy) potom (CasAFinance jsou Velke)
3. Když (Finance jsou StredniNaklady) nebo (Cas je Stredni) potom (CasAFinance jsou Stredni)
4. Když (Finance jsou VysokeNaklady) potom (CasAFinance jsou Velke)

Tabulka 22 Pravidla uzlu fuzzy systému znalostních jednotek Čas a finance; zdroj: autor

Na obrázku 70 je vytvořené schéma uzlu fuzzy znalostních jednotek, na tomto schématu jsou výše uvedené dvě vstupní fuzzy znalostní jednotky, které prostřednictvím pravidel/vztahů označených FuzzyZJ21 odvozují výstupní fuzzy znalostní jednotku.

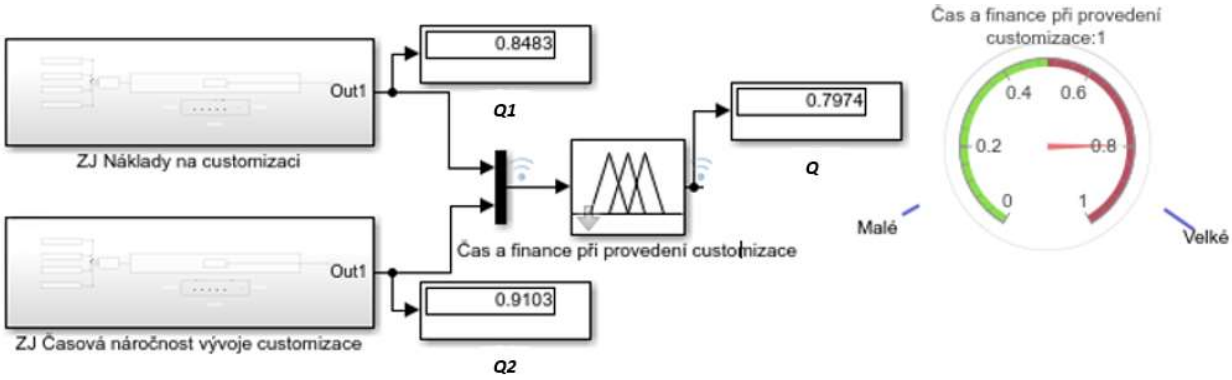


Obrázek 70 Uzel fuzzy systému znalostních jednotek Čas a Finance; zdroj: autor

Způsob inference elementu znalostní jednotky *Q* je Mamdaniho grafický způsob inference.

Krok 3. Fuzzy Logic Toolbox a Simulink

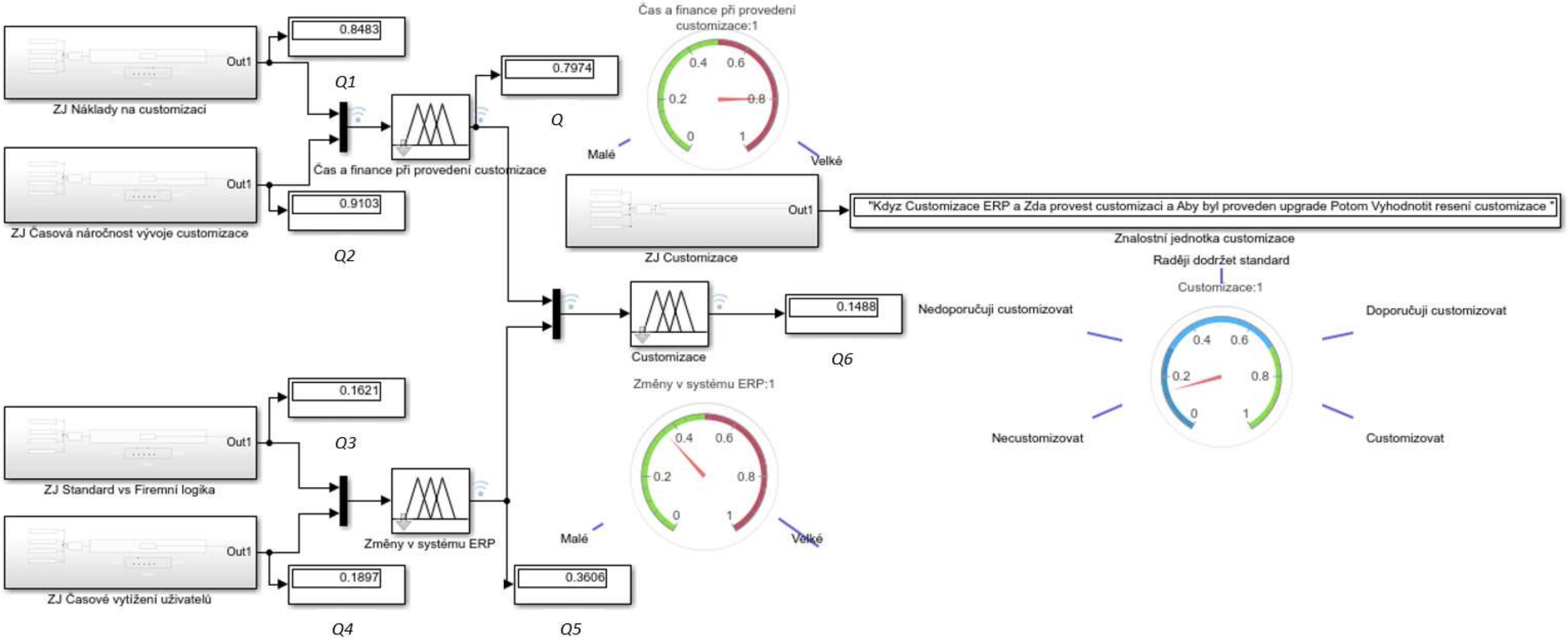
Znalostní jednotky jsou seskupeny do uzlů ve Fuzzy Logic Toolboxu podle jejich vzájemných vztahů a dále mohou být propojovány s jinými. Na příkladu uzlu fuzzy znalostních jednotek Čas a Finance je účelem odpovědět na otázku, jestli provést customizaci při zvážení časové náročnosti vývoje a financí. Architektura sestaveného uzlu Čas a Finance v rámci modelu v simulačním prostředí MATLAB Simulink je na obrázku 71.



Obrázek 71 Část modelu simulace fuzzy znalostních jednotek v simulačním prostředí Simulink; zdroj: autor

Krok 4. Simulace modelu v Simulinku

Ze znalostních jednotek jsou po předešlých krocích 1, 2 a 3 sestaveny uzly celkové problémové situace z kapitoly „5.7 Případová studie“. Je sestavena hierarchická struktura řešené problémové situace, v níž jsou určeny vstupní, vstupně/výstupní a výstupní znalostní jednotky a jejich vztahy. Jednotlivé uzly modelu jsou seskupeny a propojeny podle kontextu celé problémové situace a řešení elementárních problémů. U tohoto kroku je nutná spolupráce experta z dané znalostní domény a znalostního inženýra. Celý model simulace fuzzy znalostních jednotek „Customizace“ je na obrázku 72.

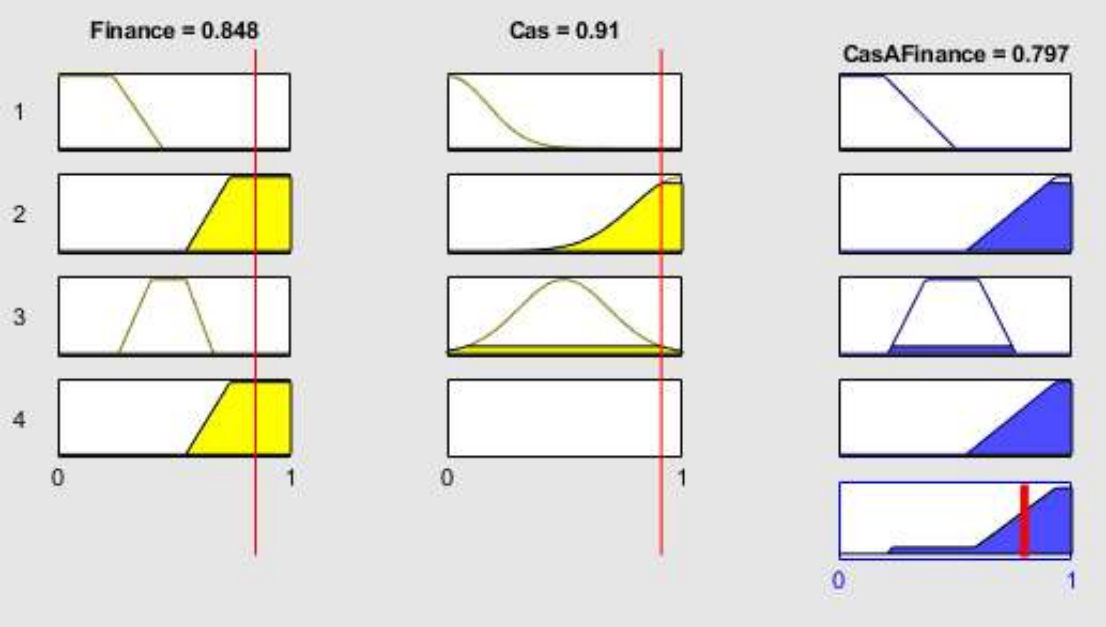


Obrázek 72 Model simulace fuzzy znalostních jednotek v simulačním prostředí Simulink; zdroj: autor

Tento model fuzzy znalostních jednotek „Customizace“ v simulačním prostředí Simulink je určen pro testování a simulaci scénářů.

Krok 5. Iniciace modelu

Pro iniciaci modelu a simulaci chování je potřeba zadat vstupní parametry pro hodnoty *Q* fuzzy znalostní jednotky, prvotní zadání probíhá přes „Subsystem“ a posuvník, případně zadáním hodnoty. Po tomto nastavení a spuštění simulace se vyvolá dialogové okno (obrázek 73) právě pro každý uzel modelu fuzzy znalostních jednotek a proběhne první odvození. V dialogovém okně lze interaktivně nastavit hodnotu *Q* při běžící simulaci již v kontextu funkcí příslušností a znalostní jednotky.



Obrázek 73 Nastavení uzlu simulace fuzzy znalostních jednotek v simulačním prostředí Simulink; zdroj: autor

V tomto uzlu jsou dvě fuzzy znalostní jednotky vstupní a jedna výstupní. Z nastavení parametrů vstupních fuzzy znalostních jednotek je odvozena hodnota výstupní fuzzy znalostní jednotky neboli výstupní hodnota uzlu modelu fuzzy znalostních jednotek. Po odhadu, případně analýze vstupních hodnot *Q,* se musí tyto hodnoty upravit ve vstupních parametrech a opětovně spustit simulace.

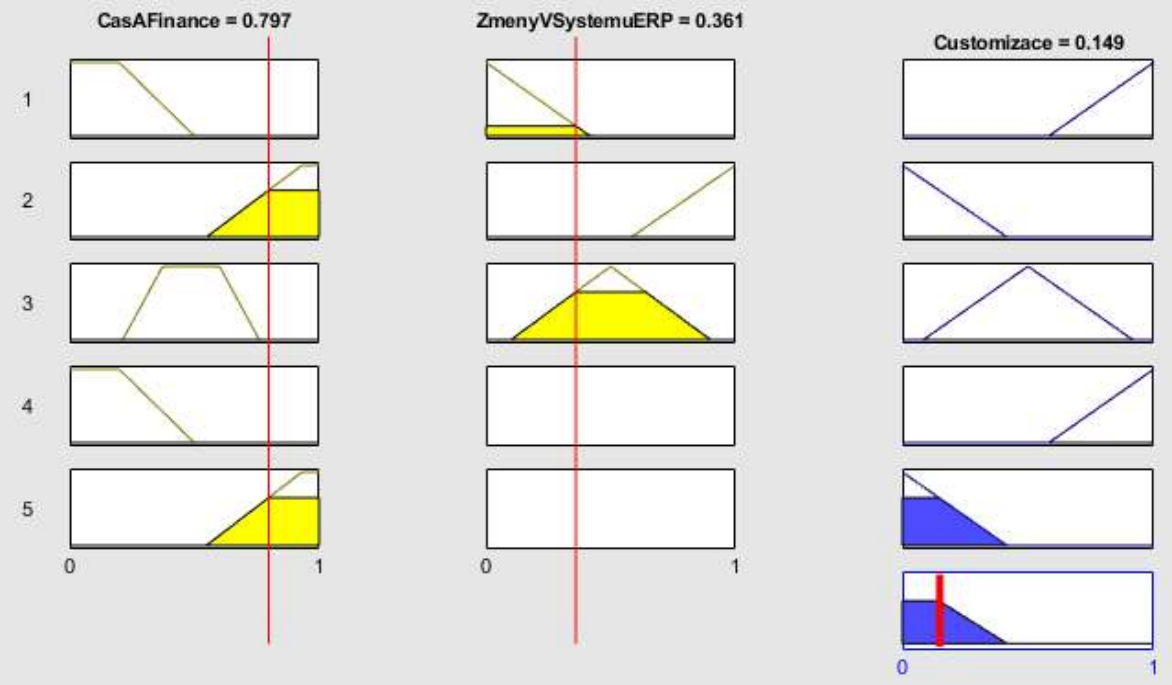
V této části procesu dostává důležitou roli expert, aby upravil funkce příslušnosti, nebo pravidla vztahů fuzzy znalostních jednotek. Každý jednotlivý uzel musí být nastaven tak, aby odpovídal reálné situaci zadání. Tento bod je významný, protože ovlivňuje odpovědi simulace. V případové studii byly využiti nezávisle dva experti k definici funkcí příslušnosti

a fuzzy pravidel. Jako nástroj bylo použito právě dialogové okno na obrázku 73. Pro takto nastavený model je vyhodnocení plynoucí z tohoto uzlu „Spíš vysoké náklady“ na customizaci. Výraz „Spíš vysoké náklady“ odpovídá slovnímu vyhodnocení výstupní fuzzy znalostní jednotky Čas a Finance a z grafického výstupu. Toto je vyhodnocení jednoho uzlu a v kontextu řešené problémové situace je uvedený dílčí konsekvent. Na základě další inference těchto dílčích konsekventů je vyvozeno konečné doporučení.

Po zadání vstupních parametrů, viz obrázek 74, je celkovou odpovědí „Raději necustomizovat“ (0,149) vzhledem k tomu, že odpovědi z jednotlivých uzlů jsou:

- Čas a finance = „velmi vysoké náklady“ (0,797)
- Změny systému ERP = „skoro významné“ (0,361)

Grafické zobrazení celkové odpovědi systému je na obrázku 74.



Obrázek 74 Odpověď fuzzy systému znalostních jednotek „Customizace“; zdroj: autor

Celkovou odpověď v textové podobě fuzzy znalostní jednotky lze vyjádřit takto:

Když je třeba v rámci problémové situace Customizace ERP řešit elementární problém Zda provést customizaci, aby byl Proveden upgrade, potom je třeba aplikovat řešení Vyhodnotit řešení customizace s výsledkem ***Raději neprovádět customizaci****”.*

Odpovědi lze také zobrazit dalšími formami, jak je popsáno v kapitole „5.4 Zobrazení neurčitosti fuzzy znalostní jednotkou“. V rámci defuzzifikace je užita metoda centroid, a to

u všech dílčích vyhodnocení i u celkového vyhodnocení. Kompletní zdrojový kód celého modelu je v příloze „Příloha 1. Kód C vytvořené simulace“.

5.7.2 Testování scénářů na modelu

Výchozí problémovou situací je případová studie s názvem „Customizace“. Pro provedení simulace modelu je podstatné vytvoření scénáře, ve kterém jsou obsaženy všechny předpoklady pro testování chování modelu. Tyto předpoklady z velké části vychází z Gap-Fit analýzy. Gap-Fit analýza posuzuje jednotlivé firemní procesy (požadavky) a standardy ERP systému. Na základě této analýzy jsou definovány předpoklady pro scénář. Pro scénář jsou vybrány body z celkové Gap-Fit analýzy firmy. Gap-Fit analýza je v plném rozsahu v příloze č. 2 Zdrojová Gap-Fit analýza. Scénář je základem pro jednotné zadání do modelu „Customizace“ a pro experty. Scénář v tabulce 23 obsahuje tři požadavky k rozhodnutí, jestli customizovat, jsou to požadavky na systém ve zkratkách POZ07, POZ12 a POZ20.

FZJ	**Požadavek**	POZ07.	POZ12.	POZ20.
FZJ	**Popis**	Vytvořit reporty Termínové hlášení, Přehled fakturací oddělení, Rozpracovanost útvaru a Externí a interní subdodávky v JetReports.	Možnost zaznamenávat nemoci v deníku zdrojů – neprůměrovat mzdu jako celek, ale navázat na mzdové složky.	Zachovat funkce pro zpracování měsíční závěrky (mimo funkčnosti Cenotvorby-Skladové režie). Řešení v jiném systému by vyžadovalo nového dodavatele a podstatné navýšení rozpočtu.
FZJ3	**Gap/Fit**	Gap	Gap	Gap
FZJ2	**Doba vývoje (hod)**	24	32	80
FZJ1	**Náklady na customizaci (CZK)**	10 000,00	15 000,00	30 000,00
FZJ3	**Priorita**	2	3	1
FZJ4	**Časové vytížení uživatelů, Nároky na klíčové uživatele při vývoji a při provozu**	Převod reportů NAV do nové verze, nutná kooperace při zadání a analýze 38 hod. V produkci nutná aktualizace nastavení, update verzí.	Před účtováním závěrky, nutná neustálá aktualizace mzdových složek na rozúčtování mezd, některé složky mají definovaný účetní předpis. Komunikace NAV a Personální systém.	Kooperace při vývoji 120 hod. Jedná se o funkce: - Rozúčtování mezd (R 92205) - Výpočet přímých a režijních nákladů (R 92206) - Přeúčtování režijních nákladů (R 90830) - Výpočet provozní režie (R 90815) - Výpočet správní režie (R 90826) - Přeúčtování mzdové složky (R 50013) - report Aktualizace zaměstnanců/prodejců/zdroj Hromadná rozúčtování sníží chybovost a čas v produkci úspora ca 100 hod měsíčně.
FZJ3	**Standard vs Firemní logika popis**	Nová verze NAV priorizuje řešení reportů v externích systémech.	Pomůže při vykazování práce na dotační projekty zejména ekonomům.	Z hlediska auditu a vykazování je závěrka klíčová. Know How rozúčtování by mělo být konsolidováno do jednoho systému.

Tabulka 23 Scénář tří požadavků na customizaci; zdroj: autor

V hlavičce řádků jsou uvedeny fuzzy znalostní jednotky, ze kterých bude sestaven model Customizace a také bude sloužit jako podklad pro rozhodování expertů. Nastavení modelu, respektive jeho vstupních hodnot ze scénáře není provedeno expertem na danou problematiku, i když to tak může být. Nastavení modelu může provádět osoba znalá Gap-Fit analýzy a je obeznámena s náležitostmi projektu Upgrade, zejména s rozpočtem a jeho časovou osou. Pokud by nastavení prováděl expert, v podstatě by mu model zprostředkoval názory jiných expertů.

5.7.3 **Porovnání scénáře v modelu Customizace s experty**

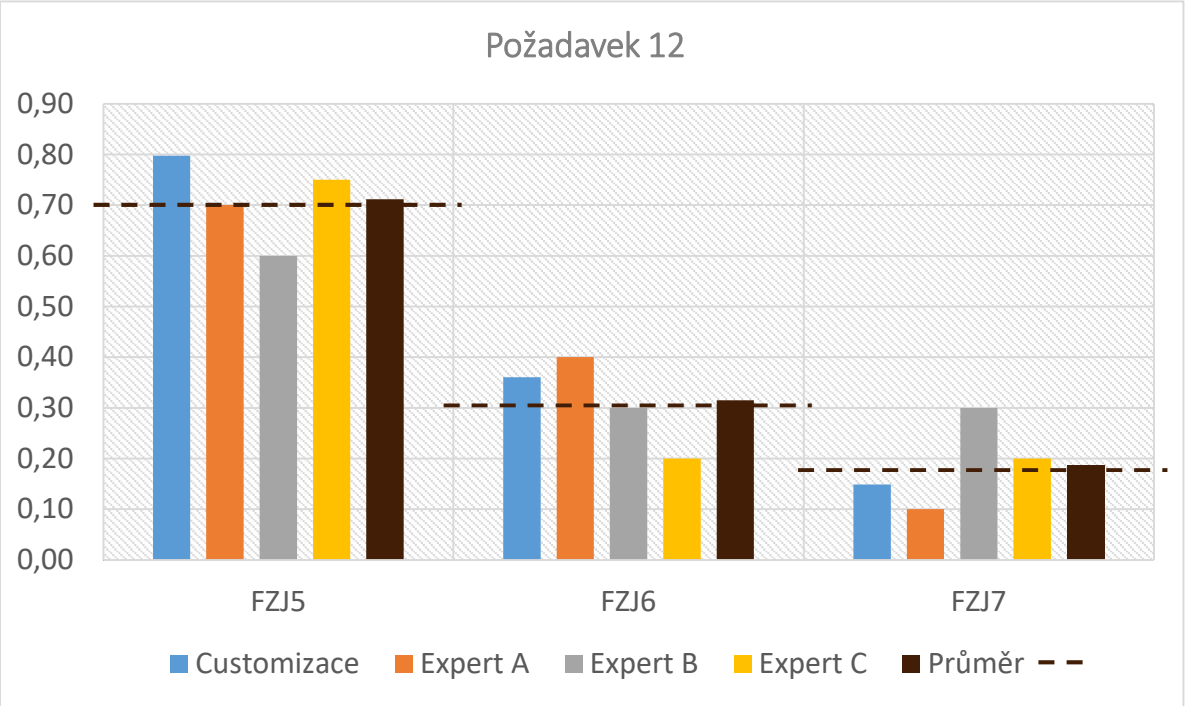
Připravený scénář je využit pro nastavení modelu Customizace. Podle situace uvedené ve scénáři jsou nastaveny vstupní parametry Customizace, jsou použity hodnoty z tabulky 23 k požadavku 12 (POZ12) a je spuštěna simulace. Výsledky této simulace jsou uvedeny v číselných hodnotách po defuzzifikaci metodou centroid v tabulce 24. Jedná se o výsledky simulace, které lze obecně odečíst (viz kapitola „5.6.6.1 Možnosti zobrazení výsledků modelu“) ze schématu, případně z dialogového okna, kde jsou hodnoty uvedeny v kontextu funkcí příslušností a fuzzy množin (obrázek 74).

POZ12	Nastavení simulace Customizace				Dílčí vyhodnocení		Výsledek
	FZJ1	FZJ2	FZJ3	FZJ4	FZJ5	FZJ6	FZJ7
Customizace	0,85	0,91	0,16	0,19	0,80	0,36	0,15
Expert A					0,70	0,40	0,10
Expert B					0,60	0,30	0,30
Expert C					0,75	0,20	0,20
Průměr					0,71	0,32	0,19

Tabulka 24 Výsledky pro požadavek 12 (hodnoty jsou zaokrouhleny); zdroj: autor

Tabulka 24 a obrázek 75 slouží k porovnání názoru expertů a výsledků provedené simulace na modelu Customizace. Pro své názory má expert k dispozici scénář a detaily projektu, tj. stejné informace jaké byly použity při nastavení modelu Customizace. Názory expertů A, B, C představují defuzzifikované dílčí vyhodnocení a konečný výsledek. Rozdíl mezi simulací a odhadem experta je v tom, že expert zadává svůj názor na dílčí vyhodnocení a výsledek do prázdného schématu (viz příloha č. 3 Schéma pro zachcycení názoru experta - prázdné schéma, funkce příslušnosti), kde má náhled i na zobrazení patřičných funkcí příslušností, ale simulace má nastaveny vstupní hodnoty a výsledky odvodí. Veškeré hodnoty dílčího vyhodnocení a výsledků uvedené tabulkách 24, 25, 26 lze prostřednictvím modelu zobrazit

ve fuzzifikované podobě. Celkový průměr je určen pro zachycení odchylek jednotlivých výsledků a také pro případné úpravy modelu.



Obrázek 75 Graf dílčích vyhodnocení a výsledku požadavku 12; zdroj: autor

POZ07	Nastavení simulace Customizace				Dílčí vyhodnocení		Výsledek
	FZJ1	**FZJ2**	**FZJ3**	**FZJ4**	**FZJ5**	**FZJ6**	**FZJ7**
Customizace	0,66	0,90	1,00	0,75	0,76	0,64	0,18
Expert A					0,72	0,77	0,13
Expert B					0,68	0,73	0,20
Expert C					0,73	0,80	0,05
Průměr					0,72	0,74	0,14

Tabulka 25 Výsledky pro požadavek 07 (hodnoty jsou zaokrouhleny); zdroj: autor

POZ20	Nastavení simulace Customizace				Dílčí vyhodnocení		Výsledek
	FZJ1	**FZJ2**	**FZJ3**	**FZJ4**	**FZJ5**	**FZJ6**	**FZJ7**
Customizace	0,22	0,12	0,20	0,26	0,24	0,23	0,71
Expert A					0,20	0,20	0,80
Expert B					0,10	0,15	0,85
Expert C					0,15	0,13	0,88
Průměr					0,17	0,18	0,81

Tabulka 26 Výsledky pro požadavek 20 (hodnoty jsou zaokrouhleny); zdroj: autor

- Závěry z porovnání scénáře modelu Customizace a experty

Na modelu byl testován scénář s náhodným výběrem požadavků z Gap-Fit analýzy, která byla vytvořena a použita ve firmě ÚJV. Názory expertů na dané požadavky ze scénářů a výsledky modelu Customizace jsou podle dosažených hodnot (viz tabulky 24, 25, 26) srovnatelné. Pokud by byly převedeny veškeré názory expertů do modelu, nezřídka by byly slovně ohodnocené stejně, nebo velmi podobně. Model Customizace dokáže objasnit důvod svého výsledku (názoru). V tomto ohledu je podstatné, že v modelu jsou obsaženy názory více expertů.

Dále lze zobrazit dílčí vyhodnocení a případně hodnoty nastavení. Jedním z dalších přínosů modelu jsou možnosti zobrazení fuzzy znalostních jednotek v pozici dílčích hodnocení a výsledků. Dostupné možnosti zobrazení byly popsány v kapitole „5.6.6 Práce s modelem“ a její podkapitole. Vyhodnocené absolutní srovnání scénáře experty vedlo k úpravám pravidel simulace.

Dalším závěrem je, že budoucí analýzy Gap-Fit (tyto analýzy jsou prováděny při upgrade systému) budou upraveny tak, aby lépe vyhovovaly nastavení simulace a byly využitelné při sestavení modelu. Úprava Gap-Fit analýzy bude spočívat ve strukturování požadavků v podobě znalostně orientovaných textů. Z těchto textů poté budou vycházet (fuzzy) znalostní jednotky. Srovnání názorů expertů a modelu podle scénáře vedlo k úpravám pravidel/vztahů mezi fuzzy znalostními jednotkami modelu (viz Příloha 3. Schéma pro zachcycení názoru experta - prázdné schéma, funkce příslušnosti). Tyto úpravy jsou již zapracovány do modelu.

6 Diskuze

Výzkumná činnost autora je dlouhodobě zaměřena na využití expertních systému při řešení složitých rozhodovacích situací. Výsledky disertační práce dokládají, že tento směr může přinést do praxe postupy, které zvýší kvalitu firemních rozhodnutí. V této kapitole jsou jednotlivé přínosy práce vyjmenovány a porovnány s výsledky jiných autorů.

- Rozšíření inference o kontext znalostních jednotek.

Prvním výsledkem je rozšíření inference na inferenci se znalostními jednotkami. Klasické strategie inference dopředného a zpětného řetězení produkčních pravidel jsou od dob Alana Turinga (od 1900 do 1956) víceméně v nezměněné podobě, i když je toto téma spojeno se vznikem AI a věnovala se mu řada odborníků, kteří formalizovali znalosti pro automatizované použití (Buchanan a Duda, 1983; Groner, Groner, Walter, 1983; Barr, Feigenbaum, Cohen, 1989; Brachman a Smith, 1980). Například v práci autorů Moreno a Espejo (2015) jsou konstruována produkční pravidla standardně, konkrétně jedno z pravidel: *"Když únavová skluzová pásma Potom únavový lom"*. Jednotlivé výroky v tomto a dalších použitých pravidlech nemají pevné pořadí a místo. Produkční pravidla tak vznikají bez ohledu na vlastní vnitřní strukturu.
K běžným / stávajícím strategiím inference je přiřazen element znalostní jednotky, přes který inference probíhá a řetězí tak znalostní jednotky. Inference znalostních jednotek vychází z toho, že jedna znalostní jednotka nemusí stačit k popisu řešení složité problémové situace a je potřeba znalostní jednotky mezi sebou propojovat. Ať už se jedná o strategii dopředného (Fakhrahmad et al., 2015) nebo zpětného řetězení (Ghanei et al., 2015), znalostní jednotky umožňují obojí. Výsledkem je nový typ inference prostřednictvím elementů znalostní jednotky (Peták a Houška, 2017).

- Nové využití pravidel a faktů při inferenci znalostní jednotky i fuzzy znalostní jednotky

Ve své studii Wagner (2017) uvádí, že nejvíce ze zkoumaného vzorku expertních systémů je v expertních systémech zastoupena primární reprezentace znalostí v podobě pravidel. Pohled na problematiku produkčních pravidel a řetězení je, že v klasické strategii řetězení jsou pravidla a fakta od sebe oddělena, a řetězí se v mezích logických operátorů do produkčních pravidel (Mařík et al., 2004, Moreno a Espejo, 2015, Semeraro et al., 2016, Venturelli et al., 2017, Psyrras a Sextos, 2018).

Klíčovým rozdílem oproti využití znalostních jednotek i fuzzy znalostních jednotek je to, že na produkčních pravidlech je stavěno a jsou z nich formalizovány znalostní jednotky i fuzzy znalostní jednotky. Znalostní jednotky i fuzzy znalostní jednotky jsou dále využívány a brány jako základ pro inferenci se znalostními jednotkami i fuzzy znalostními jednotkami.

- Využití kontextu znalostní jednotky při inferenci

Klasické strategie inference znalostních jednotek i fuzzy znalostních jednotek tak řetězí již rozšířené produkční pravidlo neboli znalostní jednotku, která obsahuje kontext. Výhoda spočívá v tom, že uživatel má vše potřebné uspořádané v jedné znalostní jednotce, není odděleno rozhodovací pravidlo od faktů, což je pro rozhodování přínosnější. Jazykové výroky neboli elementy (viz kapitola 4.1.3 Znalostní jednotky) dodávají znalostní jednotce kontext. Protože, jak uvádějí Oliinyk et al. (2017), je často nutné nejen odvodit přesné rozhodnutí, ale také vysvětlit, jak systém k rozhodnutí došel a tuto cestu prezentovat.
Znalostní jednotky mají z hlediska formalizace více možností a navíc obsahují kontext (Dömeová et al., 2008; Brožová a Houška, 2011). U klasické inference a produkčních pravidel tomu tak být nemusí, protože pravidla nemusí obsahovat kontext, který je součástí databáze faktů. Wagner (2017) poukazuje na to, že formu reprezentace znalostí v expertních systémech je obtížné určit. Znalostní jednotky mohou představovat základní stavební kámen expertního systému a díky kontextu je snadnější samotná konstrukce expertního systému.

- Vytvoření grafické podoby znalostní jednotky

Pro znalostní jednotku i pro fuzzy znalostní jednotku bylo vyvinuto grafické zobrazení. Jedná se o rozšíření analytického tvaru (fuzzy) znalostní jednotky, v níž je konsekvent zobrazen v grafu. Graf zobrazuje funkce příslušnosti a jejich možné nastavení. Vytvořením grafické podoby došlo k další možné formalizaci (fuzzy) znalostní jednotky. Pokud se jedná o vstupní (fuzzy) znalostní jednotku, grafické zobrazení ulehčuje její nastavení pro uživatele.

U výstupní znalostní jednotky spočívá přidaná hodnota v tom, že dává výsledku inference kontext v podobě grafu s funkcemi příslušností a výslednou hodnotou inference. Běžná fuzzy pravidla (Moreno a Espejo, 2015; Semeraro et al., 2016; Ali et al., 2017; Venturelli et al., 2017) zobrazí také graf s funkcemi příslušností a výslednou hodnotu, ale naproti (fuzzy) znalostním jednotkám již bez kontextu elementů (fuzzy) znalostní jednotky.

- Zobrazení neurčitosti prostřednictvím fuzzy znalostních jednotek

Ve vývoji znalostních jednotek se s neurčitostí doposud nepracovalo (Dömeová et al., 2008; Brožová a Houška, 2011). Jedna z výhod znalostních jednotek spočívá v možnostech formalizace v podobě produkčních pravidel, respektive rozšířených produkčních pravidlech. Pravidla jsou také hojně využívána, jak uvádí v přehledové studii Wagner (2017), např. největší část expertních systémů používá produkční pravidla (196 z 232 zkoumaných).

Pravidla také dovolují modifikace řadou metod pro popis a zobrazení neurčitosti, jako jsou fuzzy produkční pravidla (Ali et al., 2017; Venturelli et al., 2017), Dempster Schferova metoda (Calderwood et al., 2017), Bayesovská pravděpodobnost (Moreno a Espejo, 2015; Chen a Pollino, 2012). Pokud by byla využívána pouze ostrá „crisp" pravidla, jak dokládá Moreno a Espejo (2015), expertní systémy by nebyly dostatečně přesné ve svých výsledcích. Předložené možnosti práce neurčitosti se snaží zpřesnit výsledky expertních systémů.

Autor této disertační práce vytvořil případovou studii reálného problému, v níž byla vytvořena tři různá řešení třemi metodami. Z této případové studie byla vybrána metoda fuzzy produkčních pravidel, jelikož se autor přiklání k názoru (Nilash et al., 2015; Ali et al., 2017; Venturelli et al., 2017), který uvádí fuzzy logiku vzhledem k expertnímu systému následovně.

„Fuzzy expertní systém kombinuje schopnost expertního systému simulovat rozhodovací proces s nejistotou typickou pro lidské uvažování, která je přítomna ve fuzzy logice."

K tomuto výroku lze dodat, že fuzzy logika modeluje jev jako takový a neusuzuje z výsledku pokusu, jestli jev nastane, nebo ne.

- Vytvoření / formulace fuzzy znalostní jednotky

Zobrazení neurčitosti ve znalostních jednotkách je vytvořeno fuzzifikací vybraného elementu znalostní jednotky. Podstatným rozdílem oproti výsledkům fuzzifikace pravidel (například Venturelli et al., 2017) je, že fuzzifikací vybraných elementů znalostních jednotek vznikne fuzzy znalostní jednotka. Fuzzifikací pravidla vznikne fuzzy produkční pravidlo.

Zobrazení neurčitosti v jádru zůstává stejné, oboje vychází z fuzzy logiky. Nicméně změny jsou ve způsobu, jak se pravidla vytváří. Jak uvádějí Nilash et al. (2015), Ali et al. (2017), Venturelli et al. (2017), jsou pravidla vytvářena bez vnitřní, dopředu dané organizace a akvizice znalostí také nemusí probíhat standardní strukturovanou formou. Naproti tomu

fuzzy znalostní jednotky tím, že jsou vnitřně strukturované, slouží pro standardní definici řešené problémové situace. Svou vnitřní strukturou také napomáhají v definici funkcí příslušností. Akvizice znalostí od expertů tak probíhá uživatelsky přívětivějším způsobem než pro klasická fuzzy produkční pravidla (Nilash et al., 2015, Ali et al., 2017, Venturelli et al., 2017).

- Uplatnění lingvistické proměnné ve fuzzy znalostní jednotce

Vzhledem k textové definici jednotlivých elementů znalostních jednotek byla rozvinuta paralela slovních - lingvistických proměnných tak, jak je uvedli Novák, Perfilieva a Dvořák (2016). Definice lingvistických proměnných umožňuje slovní proměnné přiřadit fuzzy množiny a jednotlivé funkce příslušnosti odpovídající termům lingvistických proměnných. V tomto bodě je dána vlastnost lingvistické proměnné znalostní jednotce, respektive jejímu elementu.

Další potenciální výhody nabízejí objektově orientované metodologie (Houška et al., 2006). Rozšířená pravidla umožňují vytvořit objekt znalostní jednotky v MATLABu a dále ho kompilovat například do kódu C, jak je uvedeno v kapitole „5.6.2 Krok 1. Znalostní jednotka v Simulinku“ a s těmito jednotkami provádět inferenci. Fuzzy znalostní jednotka v kombinaci s objektově orientovanou metodologií a formalizovaná v kódu C tvoří základ pro simulaci a testování.

- Řešení rozhodovacích situací v rámci modelování znalostních jednotek

Proto, aby byl problém připraven pro simulaci, je nutné zvolit způsob řešení rozhodovacích situací. V tomto ohledu bylo využito fuzzifikace a případné defuzzifikace a Mamdaniho grafický způsob inference (Mamdani a Assilian, 1975; Talašová, 2003). Mamdaniho grafický způsob inference oproti metodě Takagi-Sugeno-Kang (Sugeno, 1985) vytváří možnost grafického zobrazení probíhající inference, zejména při odvozování konsekventu. Výsledné funkce příslušnosti získané metodou Sugeno jsou buď lineární nebo konstantní, nicméně systém MATLAB dovoluje konverzi mezi Sugeno a Mamdani způsoby inference (MATLAB online Documentation, 2018e).

Pro uživatele simulace je tak výstup zobrazen pomocí plochy pod funkcí příslušnosti řešení, detailní zobrazení je v kapitole „5.6.3 Krok 2. Znalostní jednotky a Fuzzy Logic Toolbox“. Grafické zobrazení v Simulinku umožňuje interaktivní úpravu vstupních parametrů a

okamžité přepočítání modelu a zejména je řešení zobrazeno v kontextu funkcí příslušností a vztahů (pravidel) (fuzzy) znalostních jednotek. Řešení konfliktních rozhodovacích situací zprostředkovává Mamdaniho grafický způsob inference ve Fuzzy Logic Toolboxu.

- Využití uzlů (fuzzy) znalostních jednotek

Pravidla se dají použít a jsou využita i pro řešení rozhodovacích situací modelovaných pomocí (fuzzy) znalostních jednotek. Je-li více (fuzzy) znalostních jednotek na stejné úrovni řešení problému a jejich výstupy ústí do jedné nebo více (fuzzy) znalostních jednotek, je nutné vytvořit vztahy - vazby mezi těmito (fuzzy) znalostními jednotkami. V tomto případě jsou využívána produkční pravidla, jak je uvádí Mařík et al. (2004). Tyto vztahy na rozdíl od produkčních pravidel jsou postaveny do jiné role. V této roli definují, jaké vztahy mají konsekventy (Q), respektive (fuzzy) znalostní jednotky mezi sebou.

Výsledkem vzájemného ovlivňování těmito vztahy při inferenci je nová hodnota elementu Q výstupní (fuzzy) znalostní jednotky. V důsledku je tímto způsobem řešena rozhodovací situace, která může nastat mezi (fuzzy) znalostními jednotkami. To se liší od inference řetězením faktů (Ali et al., 2017; Venturelli et al., 2017; Calderwood et al., 2017; Chen a Pollino, 2012; Moreno a Espejo, 2015).

- Ověření vybraného prostředí pro simulaci s fuzzy znalostními jednotkami

V této simulaci jde o propojení všech předešlých přínosů a postupů. Dále je ověřována řešitelnost a realizace předpokladů na datech z reálné situace. Pro simulaci byly uvažovány softwarové aplikace Fuzzytech (Products/fuzzyTECH Editions/Features Overview, 2018) a MATLAB s komponentou Simulink, který je detailně popsán v kapitole „4.5.3 Fuzzy expertní systém v MATLABu“. Software Fuzzytech, ve kterém vytvořil expertní systém Venturelli et al. (2017), je orientován na vytváření produkčních aplikací a má k tomu připravené struktury.

Z pohledu autora této disertační práce je vhodnější aplikací z hlediska variability vývoje a možností simulace MATLAB se Simulinkem, jak uvádějí ve svých pracích Khamis (2017) a Sharma a Vashistha (2017). MATLAB poskytuje větší svobodu v úpravách jednotlivých komponent (toolboxů) a návrhu simulace (Zaplatílek a Doňar, 2005).

V MATLABu byl vytvořen vlastní subsystém znalostní jednotky, který byl využit v simulačním modelu. Fuzzytech nemá tak vysoký potenciál pro úpravy a vývoj samotných

metod fuzzifikace a následné simulace. Vzhledem k těmto faktům byl zvolen MATLAB, obdobně jako v pracích Khamise (2017), Sharmy a Vashisthy (2017) a Zaplatílka a Doňara (2005). Pomocí standardních a upravených objektů (subsystému znalostní jednotka) je v Simulinku a Fuzzy Logic Toolboxu vytvořena simulace. V této simulaci poté probíhá vlastní testování reálných situací pomocí scénářů z praxe.

7 Závěr

Expertní systémy mají široké pole uplatnění ve vědních disciplínách a průmyslových aplikacích. Při vývoji těchto systémů jsou podstatné způsoby reprezentací znalostí, a to zejména s ohledem na možnosti strojového zpracování a interoperabilitu. Důležité jsou rovněž postupy řešení rozhodovacích situací, které mohou nastat v průběhu zpracování znalostí a v procesech inference. Dále je potřeba zvolit vhodný způsob práce s neurčitostí, které se v reálných aplikacích lze jen stěží vyhnout, a proto se značná část práce zabývá fuzzy logikou při řešení těchto situací.

Lze konstatovat, že cíle této disertační práce byly naplněny a odpovídají jednotlivým kapitolám v originální části této práce:

- **Zjistit možnosti modifikace klasické inference pro dopředné a zpětné řetězení se znalostními jednotkami.** Novou přidanou hodnotou této práce je odvození inference se znalostními jednotkami. Inference byla odvozena v kapitole „5.2 Inference znalostních jednotek“. Inference znalostních jednotek a také fuzzy znalostních jednotek je založena na klasické metodě dopředného a zpětného řetězení vybraných elementů znalostních jednotek. Na základě toho byla dále rozpracována problematika popisu neurčitosti a řešení rozhodovacích situací při inferenci.

- **Vymezit pojem fuzzy znalostní jednotka a vymezit jednotlivé typy fuzzy znalostních jednotek**. Problémy spojené s neurčitostí vyústily v řešení a definici fuzzy znalostních jednotek, které umožňují pracovat s neurčitostí. S tím je i spojen problém zapojení neurčitosti při inferenci. V tomto bodě byla využita simulace fuzzy znalostních jednotek a Fuzzy Logic Toolbox. Řešení rozhodovacích situací pak bylo vyvinuto na základě vlastního vývoje objektu fuzzy znalostní jednotka v MATLAB Subsystem (viz kapitola „5.6 Model fuzzy znalostních jednotek“) s propojením do Mamdaniho způsob grafické inference.

- **Vytvořit modelové situace ve vhodné softwarové aplikaci a potvrdit předpoklad klasické inference, dopředné a zpětné řetězení se znalostní jednotkou**. Celkový výsledek práce je návrh simulace fuzzy znalostních jednotek, a to jednak v obecném popisu v krocích, jak provádět tuto simulaci v aplikaci MATLAB a Simulink, tak v uvedené v případové studii v konkrétní firmě s konkrétním výsledkem. Celá simulace je koncipovaná jako základní model, pro který byl vyvinut postup **inference s fuzzy znalostními jednotkami.** Tento

postup byl aplikován na případovou studii, ve které byly simulovány scénáře a výsledek simulace byl porovnán s názory expertů. Na základě této simulace byly testovány a vyvíjeny další úpravy pro zpřehlednění celého řešení, formální úpravy objektů a ladění pravidel.

- **Navržení způsobu implementace znalostních jednotek jako specifické formy reprezentace procedurálních znalostí do prostředí znalostních systémů, a to včetně operací se znalostními jednotkami a inferenčních metod.** Jedním z očekávaných přínosů práce je celkové zpřehlednění konstrukce expertních systémů tím, že pro uchovávání pravidel - znalostí budou využívány znalostní jednotky. Fuzzy znalostní jednotky doplňují systémový pohled na reprezentaci procedurálních znalostí. V kontextu expertních systémů mohou být jedním ze stavebních kamenů těchto systémů. Průnik znalostních jednotek s fuzzy logikou do podoby fuzzy znalostních jednotek doplňuje koncept znalostních jednotek s neurčitostí pomocí fuzzy přístupů a konkrétně fuzzy znalostně lingvistických pravidel. Naskýtá se tím tak možnost simulace a porovnání s jinými metodami konstrukce expertních systémů, což bude i předmětem dalšího výzkumu.

Synergický efekt plynoucí z jednotlivých postupů vytvořených v originální části této práce je jednoznačným naplněním cíle této disertační práce.

Výhled na pokračování výzkumu je v zakomponování teoretických poznatků uvedených v této práci v reálných systémech a aplikacích nad rámec již zapracovaných postupů, i když většina systémů má vlastní jedinečná řešení (Moreno a Espejo, 2015; Venturelli, Caputo, et al., 2017; Zhang et al., 2018). Ambicí je pokračovat ve vývoji vlastní aplikace, případně MATLAB Toolboxu (např. Knowledge Unit Toolbox), nebo dalšího doplňku v Simulink na základě vytvořeného modelu. Konkrétně se bude jednat o začlenění fuzzy znalostních jednotek do kódu expertních systémů, například při sestavování dotazu do expertního systému a také do jeho odpovědí. Vhodná příležitost pro rozvoj fuzzy znalostních jednotek je také v rámci akvizičních technik expertních systémů. V akvizičním modulu znalostí v rámci expertního systému by fuzzy znalostní jednotka mohla sloužit k obohacení znalostní báze expertních systémů. Akvizice znalostí pro expertní systémy by poté podporovala interoperabilitu (Brožová a Houška, 2011) fuzzy znalostních jednotek v expertních systémech.

8 Seznam literatury

ADEL, E.; EL-SAPPAGH, S.; BARAKAT, S.; ELMOGY, M., 2017. Distributed electronic health record based on semantic interoperability using fuzzy ontology: a survey. *International Journal of Computers and Applications* [online]. 1-19. DOI: 10.1080/1206212X.2017.1418237. ISSN 1206-212X.

AGUILERA, P. A.; FERNÁNDEZ, A. R.; FERNÁNDEZ, R.; RUMÍ, A.; SALMERÓN, 2011. Bayesian networks in environmental modelling. *Environmental Modelling & Software*, 26.12: 1376-1388. DOI: 10.1016/j.envsoft.2011.06.004.

AKGUN, A.; SEZER, E. A.; NEFESLIOGLU, H. A.; GOKCEOGLU, C.; PRADHAN, B., 2011. An easy-to-use MATLAB program (MamLand) for the assessment of landslide susceptibility using a Mamdani fuzzy algorithm. *Computers & Geosciences* [online]. 38: 23-34. DOI: https://doi.org/10.1016/j.cageo.2011.04.012.

ALI, F. D.; KWAK, P.; KHAN, S. M.; ISLAM, R.; KIM, K. H.; KIM, K.; KWAK, S., 2017. Fuzzy ontology-based sentiment analysis of transportation and city feature reviews for safe traveling. *Transportation Research: Part C,* 77: 33-48. DOI: 10.1016/j.trc.2017.01.014.

AT&T Corp., New York, N. Y. Interface for a voice-activated connection system. Vynálezci: Dawn L. Dutton, Shrikanth Sambasivan Narayanan, Ilija Zeljkovic. United States. Patent Number: US006138100A. 24. 10. 2000.

BARR, A.; FEIGENBAUM, E. A.; COHEN, P. R., 1989. *The Handbook of artificial intelligence*. Stanford, Calif.: HeurisTech Press. ISBN 0-86576-004-7.

BAYES, T.; PRICE, R., 1763. An Essay towards Solving a Problem in the Doctrine of Chances. By the Late Rev. Mr. Bayes, F. R. S. Communicated by Mr. Price, in a Letter to John Canton, A. M. F. R. S. *Philosophical Transactions of the Royal Society of London*. 53: 370–418. DOI:10.1098/rstl.1763.0053.

BĚLOHLÁVEK, R.; DAUBEN, J. W.; KLIR, G. J., 2017. *Fuzzy logic and mathematics: a historical perspective*. New York, NY, United States of America: Oxford University Press, 2017. ISBN 9780190200015.

BERTALANFFY, L. von., 2006. *General System Theory, Foundation, Development, Application.* New York: George Braziller, inc.. ISBN-13: 978-0-8076-0453-3.

BERÁNEK, L., 2010. Základy dempster-shaferovy teorie a její aplikace pro modelování bezpečnosti a spolehlivosti (I.). *Chemagazin XX.* 6: 28-30.

BLACK, M., 1937. Vagueness. An Exercise in Logical Analysis. *Philosophy of Science*. [online]. 4.4: 427-455 [cit. 2018-04-18]. DOI: 10.1086/286476. ISSN 0031-8248. Dostupné z: https://www.journals.uchicago.edu/doi/10.1086/286476

BRACHMAN, R. J.; SMITH, B. C., 1980. Special issue on knowledge representation. *ACM SIGART Bulletin* [online]. 70: 1-138 [cit. 2018-04-18]. DOI: 10.1145/1056751.1056752. ISSN 0163-5719. Dostupné z: http://portal.acm.org/citation.cfm?doid=1056751.1056752

BROMLEY, J.; JACKSON, N. A.; CLYMER, O. J.; GIACOMELLO, A. M.; JENSEN, F. W., 2005. The use of Hugin® to develop Bayesian networks as an aid to integrated water resource planning. *Environmental Modelling & Software* [online]. 20.2: 231-242 [cit. 2018-04-18]. DOI: 10.1016/j.envsoft.2003.12.021. ISSN 1364-8152. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S1364815204000404

BROŽOVÁ, H.; HOUŠKA, M., 2011. *Modelování znalostí*. Praha: Professional Publishing. ISBN 978-80-7431-069-0.

BUCHANAN, B. G.; DUDA, R. D., 1983. Principles of Rule-Based Expert Systems. *Advances in Computers Volume 22* [online]. Elsevier, s. 163-216 [cit. 2018-04-18]. DOI: 10.1016/S0065-2458(08)60129-1. ISBN 9780120121229. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0065245808601291

BUZAN, T., 2007. *Mentální mapování*. Praha: Portál, 2007. ISBN 978-80-7367-200-3.

BUZAN, T.; BUZAN, B., 2011. *Myšlenkové mapy: probuďte svou kreativitu, zlepšete svou paměť, změňte svůj život*. Brno: Computer Press, 2011. ISBN 978-80-251-2910-4.

CALDERWOOD, S., MCAREAVEY, K.; LIU, W.; HONG, J., 2017. Context-dependent combination of sensor information in Dempster–Shafer theory for BDI. *Knowledge and Information Systems* [online]. 51.1: 259-285 [cit. 2018-04-21]. DOI: 10.1007/s10115-016-0978-0. ISSN 0219-1377. Dostupné z: http://link.springer.com/10.1007/s10115-016-0978-0

DEBNATH, L.; BASU, K., 2014. A short history of probability theory and its applications. *International Journal of Mathematical Education in Science and Technology* [online]. 46.1: 13-39 [cit. 2018-04-18]. DOI: 10.1080/0020739X.2014.936975.

DEL ÁGUILA, I., M.; PALMA, J., T.; TÚNEZ S., 2014. Milestones in Software Engineering and Knowledge Engineering History: A Comparative Review. *The Scientific World Journal* [online]. 1-10 [cit. 2018-04-18]. DOI: 10.1155/2014/692510.

DEMPSTER, A. P., 1967. Upper and lower probability inferences based on a sample from a finite univariate population. *Biometrika* [online]. 54.(3-4): 515-528 [cit. 2018-04-21]. DOI: 10.1093/biomet/54.3-4.515. ISSN 0006-3444.

DÖMEOVÁ, L.; HOUŠKA, M.; HOUŠKOVÁ BERÁNKOVÁ, M., 2008. *Systems Approach to Knowledge Modelling*. Hradec Králové: Graphical Studio Olga Čermáková.

DVOŘÁK, J., 2004. *Expertní systémy*. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky.

EICHHOFF, J.; SCHMIDT, R.; SCHMIDT, J.; ROLLER, D., 2016. Inducing production rules to extend existing design grammars: The parse/derive method. *Computer-Aided Design and Applications* [online]. 14.4: 535-548 [cit. 2018-04-18]. DOI: 10.1080/16864360.2016.1257195. ISSN 1686-4360.

eMathTeacher, 1999. *Mamdani' s Fuzzy Inference Method,* Membership functions [online]. [cit. 2018-04-15]. Dostupné z: http://www.dma.fi.upm.es/recursos/aplicaciones/logica_borrosa/web/fuzzy_inferencia/funpert_en.htm

Encyclopædia Britannica, 2016. Expert system [online]. [cit. 2018-04-15]. Dostupné z: https://www.britannica.com/technology/expert-system

FACCHINETTI, G.; MAGNI, C. A.; MASTROLEO, G.; VIGNOLA, M., 2001. Valuing strategic investments with a fuzzy expert system: an Italian case. In: *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference (Cat. No. 01TH8569)* [online]. 812-817 [cit. 2018-04-18]. DOI: 10.1109/NAFIPS.2001.944708. ISBN 0-7803-7078-3.

FAKHRAHMAD, S. M.; SADREDDINI, M. H.; ZOLGHADRI JAHROMI, M., 2015. A proposed expert system for word sense disambiguation: deductive ambiguity resolution based on data mining and forward chaining. *Expert Systems* [online]. 32.2: 178-191 [cit. 2018-04-18]. DOI: 10.1111/exsy.12075. ISSN 0266-4720.

FIKES, R. E.; NILSSON J. N., 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* [online]. 2.3-4: 189-208 [cit. 2018-04-18]. DOI: 10.1016/0004-3702(71)90010-5. ISSN 0004-3702.

French speaking Libre Software Users' Association, 2015. Définitions [online]. [cit. 2018-04-15]. Dostupné z: https://aful.org/gdt/interop

GASS, S. I.; HARRIS, C. M., 2001. *Encyclopedia of Operations Research and Management Science*, New York: Springer US. ISBN 9781441911537.

GANZHA, M.; PAPRZYCKI, M.; PAWŁOWSKI, W.; SZMEJA, P.; WASIELEWSKA, K., 2017. Semantic interoperability in the Internet of Things: An overview from the INTER-IoT perspective. *Journal of Network and Computer Applications* [online]. 81**:** 111-124 [cit. 2018-04-20]. DOI: 10.1016/j.jnca.2016.08.007. ISSN 1084-8045.

GARIBALDI, J. M., 1997. *Intelligent techniques for handling uncertainty in the assessment of neonatal outcome* [online]. [cit. 2018-04-14].

GAVRILOVA, T.; LESHCHEVA, I.; STRAKHOVICH, E., 2015. Gestalt principles of creating learning business ontologies for knowledge codification. *Knowledge Management Research & Practice* [online]. 13.4: 418-428 [cit. 2018-04-20]. DOI: 10.1057/kmrp.2013.60. ISSN 1477-8238.

GEVARTER, W. B., 1984. *Artificial intelligence, expert systems, computer vision, and natural language processing*. Park Ridge, N. J. USA: Noyes Publications. ISBN 0815509944.

GHANEI, S.; VAFAEENEZHAD, H.; KASHEFI, M.; EIVANI, A. R.; MAZINANI, M., 2015. Design of an expert system based on neuro-fuzzy inference analyzer for on-line microstructural characterization using magnetic NDT method. *Journal of Magnetism and Magnetic Materials* [online]. 379: 131-136 [cit. 2018-04-20]. DOI: 10.1016/j.jmmm.2014.12.028. ISSN 0304-8853.

GIARRATANO, J. C.; RILEY, G., 1989. *Expert systems: principles and programming*. Boston: Boyd & Fraser. ISBN 0878353356.

GRONER, R.; GRONER, M.; WALTER, F., 1983. *Methods of heuristics*. Hillsdale, N. J.: L. Erlbaum Associates. ISBN 978-0898592511.

HARVEY, G., 1663. *Archelogia Philosophica Nova* [online]. London: St. Pauls Church-yard. [cit. 2018-10-15]. Dostupné z: https://quod.lib.umich.edu/e/eebo/A43008.0001.001?rgn=main;view=fulltext

HOUŠKA, M.; BERÁNKOVÁ, M., 2013. Binary Operations with Knowledge Units. AWERProcedia Information Technology & Computer Science [online]. 3: 1716–1726.

HOUŠKA, M.; BERÁNKOVÁ, M.; BARTOŠKA, J.; MERUNKA, V. 2006. Reprezentace elementární znalosti pomocí objektově orientované metodologie. In: *Agrární perspektivy XV*. Prague: 626-629. ISBN: 80-213-1531-8.

CHANDRA, R.; GUPTA, A.; ONG, Y. S., GOH, CH. K., 2017. Evolutionary Multi-task Learning for Modular Knowledge Representation in Neural Networks. Neural Processing Letters [online]. [cit. 2018-04-21]. DOI: 10.1007/s11063-017-9718-z. ISSN 1370-4621.

CHEN, S.; CARMEL, H.; POLLINO, A., 2012. Good practice in Bayesian network modelling. *Environmental Modelling & Software* [online]. 37: 134-145 [cit. 2018-04-20]. DOI: 10.1016/j.envsoft.2012.03.012. ISSN 1364-8152.

JAKEMAN, A. J.; LETCHER, R. A.; NORTON, J. P., 2006. Ten iterative steps in development and evaluation of environmental models. *Environmental Modelling & Software* [online]. 21.5: 602-614 [cit. 2018-04-20]. DOI: 10.1016/j.envsoft.2006.01.004. ISSN 1364-8152.

KATUŠČÁK, D.; MATTHAEIDESOVÁ, M.; NOVÁKOVÁ, M.; PASTIER, J.; ĎURIČ, L.; HOTÁR, V. S. ed., 1998. *Informačná výchova*. Bratislava: Slovenské pedagogické nakladateľstvo. Terminologický a výkladový slovník. ISBN 80-08-02818-1.

KAUR, M.; CHOPRA, V.; SINGH, L., 2016. Fuzzy Membership Functions: Selection Techniques and Implementation in WSN. *International Journal of Engineering Research and development*. 12: 64-72.

KENDAL, S.; CREEN, M., 2007. An Introduction to Knowledge Engineering [online]. London: Springer London [cit. 2018-04-21]. ISBN 978-1-84628-475-5.

KESHWANI, D. R.; JONES, D. D.; MEYER, G. E.; BRAND, R. M., 2008. Rule-based Mamdani-type fuzzy modeling of skin permeability. *Applied Soft Computing* [online]. 8: 285-294. DOI: https://doi.org/10.1016/j.asoc.2007.01.007.

KHAMIS, A.; MOHD, R.; GHANI, A. B.; GAN, CH. K.; MOHD ARAS, M. S.; KHAMIS, M. F.; SUTIKNO, T.; ZANARIAH, J., 2017. Fuzzy Logic Implementation with MATLAB for PV-Wind Hybrid System. *TELKOMNIKA (Telecommunication Computing Electronics and Control)* [online]. 15.3: 1181- [cit. 2018-04-20]. DOI: 10.12928/telkomnika.v15i3.6099. ISSN 2302-9293.

KIMBALL, R., 2008. *The data warehouse lifecycle toolkit*. 2nd ed. Indianapolis, IN: Wiley Pub. ISBN 978-0-470-14977-5.

KUMAR, G.; KISHOR, P.; VISWANATH, A.; ANANDA, R., 2016. Ensemble of randomized soft decision trees for robust classification. *Sadhana* [online]. [cit. 2018-04-20]. DOI: 10.1007/s12046-016-0465-z. ISSN 0256-2499.

LEE, Y. H.; EVENS, M. W., 1998. Natural language interface for an expert system. *Expert Systems* [online]. 15.4: 233-239 [cit. 2018-04-13]. ISSN 0266-4720.

LEGIEŃ, G.; ŚNIEŻYŃSKI, B.; WILK-KOŁODZIEJCZYK, D.; KLUSKA-NAWARECKA, S.; NAWARECKI, E.; JAŚKOWIEC, K., 2015. Expert System with Web Interface Based on Logic of Plausible Reasoning. *Database and Expert Systems Applications* [online]. Cham: Springer International Publishing, 9262: 13-20 [cit. 2018-04-20]. DOI: 10.1007/978-3-319-22852-5_2. ISBN 978-3-319-22851-8.

LI, X.; ZHAO, H.; ZHU, W., 2015. A cost sensitive decision tree algorithm with two adaptive mechanisms. *Knowledge-Based Systems* [online]. 88: 24-33 [cit. 2018-04-20]. DOI: 10.1016/j.knosys.2015.08.012. ISSN 0950-7051.

LILLY, J. H., 2010. *Fuzzy Control and Identification* [online]. Hoboken, NJ, USA: John Wiley. [cit. 2018-04-21]. ISBN 9780470874240.

LINDSAY, R. K., 1980. *Applications of artificial intelligence for organic chemistry: the DENDRAL project*. New York: McGraw-Hill Book Co. ISBN 0-07-037895-9.

LYTRAS, M. D.; NAEVE, A., 2006. *Intelligent learning infrastructure for knowledge intensive organizations: a semantic web perspective*. Hershey PA: Information Science Pub. ISBN 159140505x.

MAGNI, C. A.; MALAGOLI, S.; MASTROLEO, G., 2006. An alternative approach to firms' evaluation: Expert systems and fuzzy logic. *International Journal of Information*

Technology & Decision Making [online]. 05.01: 195-225 [cit. 2018-04-20]. DOI: 10.1142/S0219622006001812. ISSN 0219-6220.

MAMDANI, E. H.; ASSILIAN, S., 1975. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies* [online]. 7.1: 1-13 [cit. 2018-04-21]. DOI: 10.1016/S0020-7373(75)80002-2. ISSN 0020-7373.

MAREŠ, M., 2006. Fuzzy sets. *Scholarpedia* [online]. 1.10: 2031- [cit. 2018-04-20]. DOI: 10.4249/scholarpedia.2031. ISSN 1941-6016. Dostupné z: http://www.scholarpedia.org/article/Fuzzy_sets

MARCHI, G.; VIGNOLA, M.; FACCHINETTI, G.; MASTROLEO, G., 2014. International market selection for small firms: a fuzzy-based decision process. *European Journal of Marketing* [online]. 48.(11/12): 2198-2212 [cit. 2018-04-20]. DOI: 10.1108/EJM-09-2012-0512. ISSN 0309-0566.

MAŘÍK, V.; ŠTĚPÁNKOVÁ O.; LAŽANSKÝ J., 2004. *Artificial Intelligence I – IV.* Prague: Academia Praha.

MATLAB online Documentation, 2017. Fuzzy Inference Process [online]. [cit. 2018-04-15]. Dostupné z: https://www.mathworks.com/help/fuzzy/fuzzy-inference-process.html

MATLAB online Documentation, 2018(a). Fuzzy Logic Toolbox Graphical User Interface Tools [online]. [cit. 2018-04-15]. Dostupné z: https://www.mathworks.com/help/fuzzy/building-systems-with-fuzzy-logic-toolbox-software.html#FP637DUP10

MATLAB online Documentation, 2018(b). Defuzzification Methods. [online]. [cit. 2018-04-15]. Dostupné z: https://www.mathworks.com/help/fuzzy/examples/defuzzification-methods.html#d119e2851

MATLAB online Documentation, 2018(c). Function trapmf gaussmf trapmf [online]. [cit. 2018-04-15]. Dostupné z: https://www.mathworks.com/help/fuzzy/trapmf.html

MATLAB online Documentation, 2018(d). What Is Sugeno-Type Fuzzy Inference. [cit. 2018-06-15]. Dostupné z: https://nl.mathworks.com/help/fuzzy/what-is-sugeno-type-fuzzy-inference.html#d117e2485

MATLAB online Documentation, 2018(e). Comparison of Sugeno and Mamdani Systems. [cit. 2018-06-15]. Dostupné z: https://nl.mathworks.com/help/fuzzy/what-is-sugeno-type-fuzzy-inference.html#d117e2485

MEDASANI, S.; KIM, J.; KRISHNAPURAM, R., 1998. An overview of membership function generation techniques for pattern recognition. *International Journal of Approximate Reasoning* [online]. 19.(3-4): 391-417 [cit. 2018-04-20]. DOI: 10.1016/S0888-613X(98)10017-8. ISSN 0888-613X.

MIRZAMOMEN, Z.; KANGAVARI, M. R., 2017. A framework to induce more stable decision trees for pattern classification. *Pattern Analysis and Applications* [online]. 20.4: 991-1004 [cit. 2018-04-20]. DOI: 10.1007/s10044-016-0542-2. ISSN 1433-7541.

MOLER, C., 2004. The Origins of MATLAB. *MathWorks* [online]. [cit. 2018-04-15]. Dostupné z: https://www.mathworks.com/company/newsletters/articles/the-origins-of-matlab.html

MORENO, C. J.; ESPEJO, E., 2015. A performance evaluation of three inference engines as expert systems for failure mode identification in shafts. *Engineering Failure Analysis* [online]. 53: 24-35 [cit. 2018-04-20]. DOI: 10.1016/j.engfailanal.2015.03.020. ISSN 1350-6307.

NEWELL, A.; SIMON, H. A., 1961. GPS, a program that simulates human thought, in Computers and Thought, E. A. Feigenbaum and J. Feldman, Eds., pp. 279–293, McGraw-Hill, New York, NY, USA.

NILASHI, M.; ZAKARIA, R.; IBRAHIM, O.; MAJID, M.; Z.; ZIN, R.; M.; CHUGTAI, M. W.; ABIDIN, N. I. Z.; SAHAMIR, S. R.; YAKUBU, D. A., 2015. A knowledge-based expert system for assessing the performance level of green buildings. *Knowledge-Based Systems* [online]. 86: 194-209 [cit. 2018-04-20]. DOI: 10.1016/j.knosys.2015.06.009. ISSN 0950-7051.

NIVEDITHA, R.; BASAVARAJ, T. V., 2017. Artifical inteligence and expert systems. *International Research Journal of Computer Science*. 4: 155-160.

NONAKA, I.; TAKEUCHI, H., 1995. *The knowledge-creating company: how Japanese companies create the dynamics of innovation*. New York: Oxford University Press, ISBN 0-19-509269-4.

NOVÁK, V.; PERFILIEVA, I.; MOČKOŘ, J., 1999. *Mathematical principles of fuzzy logic*. Boston: Kluwer Academic. ISBN 0-7923-8595-0.

NOVÁK, V.; PERFILIEVA, I.; DVOŘÁK, A., 2016. *Insight into fuzzy modeling*. Hoboken, New Jersey: John Wiley, ISBN 9781119193197.

OLESIAK, K., 2017. Application of Fuzzy Logic Toolbox for modeling fuzzy logic controllers. *Proceedings of the International Scientific Conference. Society. Integration. Education* [online]. 3: 539-546.

OLIINYK, A.; SKRUPSKY, S.; SUBBOTIN, S; KOROBIICHUK, I., 2017. Parallel method of production rules extraction based on computational intelligence. *Automatic Control and Computer Sciences* [online]. 51.4: 215-223 [cit. 2018-04-20]. DOI: 10.3103/S0146411617040058. ISSN 0146-4116.

PETÁK, M.; HOUŠKA, M., 2017. An Inference Strategy for Knowledge Units. *HAICTA 2017: 8th International Conference on Information & Communication Technologies in Agriculture, Food and Environment, September 21-24, Chania, Crete, Greece*. 2030: 304-313. http://ceur-ws.org/Vol-2030/.

PETÁK, M.; HOUŠKA, M., 2018. Fuzzy knowledge unit. *12th International Scientific Conference on Distance Learning in Applied Informatics*. Štúrovo, Slovakia: Wolters Kluwer, 491-502. ISBN 978-80-7598-059-5.

PIRTTIMAA, M.; HUSU, J.; METSÄRINNE, M., 2017. Uncovering procedural knowledge in craft, design, and technology education: a case of hands-on activities in electronics. *International Journal of Technology and Design Education* [online]. 27.2: 215-231 [cit. 2018-04-20]. DOI: 10.1007/s10798-015-9345-9. ISSN 0957-7572.

Production rules, 2012. Bratko ed., chapter 15, page 343. [online]. [cit. 2018-04-15]. Dostupné z: http://www.cse.unsw.edu.au/~billw/cs9414/notes/kr/rules/rules.html

Products/fuzzyTECH Editions/Features Overview, 2018. IEC 1131-7 standard on fuzzy logic development [online]. [cit. 2018-02-15]. Dostupné z: https://www.fuzzytech.com/

PSYRRAS, N. K.; SEXTOS, A. G., 2018. Build-X: Expert system for seismic analysis and assessment of 3D buildings using OpenSees. *Advances in Engineering Software* [online]. 116: 23-35 [cit. 2018-04-11]. DOI: 10.1016/j.advengsoft.2017.11.007. ISSN 0965-9978.

RAUCHOVÁ, T.; HOUŠKA, M., 2013. Efficiency of Knowledge Transfer through Knowledge Texts: Statistical Analysis. *Journal on Efficiency and Responsibility in Education and Science*, 6.1: 46-60. ISSN: 1803-1617.

RILEY, J., 2017. *Understanding Metadata what is metadata and what is it for*. Baltimore: National Information Standards Organization (NISO), ISBN: 978-1-937522-72-8.

ROSS, T. J., 2010. *Fuzzy logic with engineering applications*. 3rd ed. Chichester, U. K.: John Wiley, ISBN 978-0-470-74376-8.

RYLE, G., 1949. (edice 2009). *The concept of mind*. New York: Routledge. ISBN 0-203-87585-0.

SAMMUT, C.; WEBB, G., 2016. *Encyclopedia of machine learning and data mining*. New York, NY: Springer Berlin Heidelberg. ISBN 978-1-4899-7687-1.

SEIPEL, D.; NOGATZ, F.; ABREU, S., 2018. Domain-specific languages in Prolog for declarative expert knowledge in rules and ontologies. *Computer Languages, Systems & Structures* [online]. 51: 102-117 [cit. 2018-04-21]. DOI: 10.1016/j.cl.2017.06.006. ISSN 1477-8424.

SEMERARO, T.; MASTROLEO, G.; ARETANO, R.; FACCHINETTI, G.; ZURLINI, G.; PETROSILLO, I., 2016. GIS Fuzzy Expert System for the assessment of ecosystems vulnerability to fire in managing Mediterranean natural protected areas. *Journal of Environmental Management* [online]. 168: 94-103 [cit. 2018-04-21]. DOI: 10.1016/j.jenvman.2015.11.053. ISSN 0301-4797.

SHAFER, G., 1976. *A mathematical theory of evidence*. Princeton, N. J.: Princeton University Press. ISBN 0-691-10042-x.

SHAFER, G., 2016. Dempster's rule of combination. *International Journal of Approximate Reasoning* [online]. 79: 26-40 [cit. 2018-04-21]. DOI: 10.1016/j.ijar.2015.12.009. ISSN 0888-613X.

SHARMA, M. K.; VASHISTHA, S., 2017. Fuzzy Mathematical Approach to Detect Micronutrient Deficiency Disorders by using MATLAB Functions. *Bulletin of Pure & Applied Sciences- Mathematics and Statistics* [online]. 36e.1: 1- [cit. 2018-04-21]. DOI: 10.5958/2320-3226.2017.00001.7. ISSN 0970-6577.

SIMONS, P., 2014. Jan Łukasiewicz. The Stanford Encyclopedia of Philosophy (Spring 2017 Edition) [online]. [cit. 2018-04-15]. Dostupné z: https://plato.stanford.edu/archives/spr2017/entries/lukasiewicz/

SONAL, D.; PANDEY, R. K.; GAUTAM, S. S., 2014. Dealing with Uncertainty in Expert Systems. *International Journal of Soft Computing and Engineering (IJSCE)*. 4: 105-111.

STOKHOF, H.; DE VRIES, B.; BASTIAENS, T.; MARTENS, R., 2018. Using Mind Maps to Make Student Questioning Effective: Learning Outcomes of a Principle-Based Scenario for Teacher Guidance. *Research in Science Education* [online]. [cit. 2018-04-21]. DOI: 10.1007/s11165-017-9686-3. ISSN 0157-244X.

SUGENO, M., 1985. *Industrial Applications of Fuzzy Control*, Elsevier Science Ltd; First Edition edition (December 1, 1985). ISBN-13: 978-0444878298.

TALAŠOVÁ, J., 2003. *Fuzzy metody vícekriteriálního hodnocení a rozhodování*. Olomouc: Univerzita Palackého. ISBN 80-244-0614-4.

TRUNEČEK, J., 2004. *Management znalostí*. Praha: C. H. Beck pro praxi. ISBN 80-7179-884-3.

TSANG, E. C. C.; YEUNG, D. S., 1997. Modelling fuzzy production rules with fuzzy expert networks. *Expert Systems with Applications* [online]. 13:3: 169-178 [cit. 2018-04-21]. DOI: 10.1016/S0957-4174(97)00029-8. ISSN 0957-4174.

VENTURELLI, A.; CAPUTO, F.; LEOPIZZI, R.; MASTROLEO, G.; MIO, C., 2017. How can CSR identity be evaluated? A pilot study using a Fuzzy Expert System. *Journal of Cleaner Production* [online]. 141: 1000-1010 [cit. 2018-04-21]. DOI: 10.1016/j.jclepro.2016.09.172. ISSN 0959-6526.

WAGNER, W. P., 2017. Trends in expert system development: A longitudinal content analysis of over thirty years of expert system case studies. *Expert Systems with Applications* [online]. 76: 85-96 [cit. 2018-04-21]. DOI: 10.1016/j.eswa.2017.01.028. ISSN 0957-4174.

WU, J.; ZHAO, T.; LI, S.; OWN, C., M., 2017. Belief Interval of Dempster-Shafer Theory for Line-of-Sight Identification in Indoor Positioning Applications. *Sensors (14248220)* [online]. 17.6: 1-18 [cit. 2018-04-15]. DOI: 10.3390/s17061242. ISSN 1424-8220.

ZADEH, L., A., 1965. Fuzzy sets. *Information and Control* [online]. 8(3): 338-353 [cit. 2018-04-20]. DOI: 10.1016/S0019-9958(65)90241-X. ISSN 0019-9958. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S001999586590241X

ZADEH, L. A., 1973. Outline of a new approach to the analysis of complex systems and decision processes, *IEEE Transactions on Systems, Man, and Cybernetics.* 3: 28-44. DOI: 10.1109/TSMC.1973.5408575.

ZADEH, L. A., 1975. Fuzzy logic and approximate reasoning. Synthese [online]. 30.3-4: 407-428 [cit. 2018-05-07]. DOI: 10.1007/BF00485052. ISSN 0039-7857.

ZADEH, L. A., 1978. Fuzzy sets as a basis for a theory of possibility. International Journal Fuzzy Sets Systems, vol. 1. pp. 3-28.

ZADEH, L. A., 1999. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems* [online]. 100: 9-34 [cit. 2018-04-21]. DOI: 10.1016/S0165-0114(99)80004-9. ISSN 0165-0114.

ZADEH, L. A.; KLIR, G. J.; YUAN, B., 1996. *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers*. River Edge, N. J.: World Scientific. ISBN 981-02-2421-4.

ZAPLATÍLEK, K.; DOŇAR, B., 2005. *MATLAB pro začátečníky*. 2. vyd. Praha: BEN - technická literatura. ISBN 80-7300-175-6.

ZHANG, Y. H.; REN J.; Na YIN, G., 2014. Identification Method of Directed Digraph in Organization Knowledge Unit. Applied Mechanics and Materials [online]. 610: 673-679 [cit. 2018-04-21]. DOI: 10.4028/www.scientific.net/AMM.610.673. ISSN 1662-7482.

Dosud publikované práce:

PETÁK, M.; HOUŠKA, M., 2018. Fuzzy knowledge unit. In: 12th International Scientific Conference on Distance Learning in Applied Informatics. Štúrovo, Slovakia: Wolters Kluwer, 491-502. ISBN 978-80-7598-059-5.

PETÁK, M.; HOUŠKA, M., 2017. An Inference Strategy for Knowledge Units in: *HAICTA 2017: 8th International Conference on Information & Communication Technologies in Agriculture, Food and Environment, September 21-24, Chania, Crete, Greece*. 2030: 304-313. http://ceur-ws.org/Vol-2030/.

PETÁK, M.; BROŽOVÁ, H.; HOUŠKA, M., 2020. Modeling of knowledge via fuzzy knowledge unit in a cese of the ERP systems upgrade. *Automatic Control and Computer Sciences* [online]. ISSN 1558-108X. (v tisku)

9 Seznam obrázků

10 Seznam tabulek

11 Přílohy

11.1 Příloha 1. Kód C vytvořené simulace

```
/*
 * Academic License - for use in teaching, academic research, and meeting
 * course requirements at degree granting institutions only.  Not for
 * government, commercial, or other organizational use.
 *
 * File: ZJ.c
 *
 * Code generated for Simulink model 'ZJ'.
 *
 * Model version                  : 1.62
 * Simulink Coder version         : 8.14 (R2018a) 06-Feb-2018
 * C/C++ source code generated on : Wed May 22 19:56:44 2019
 *
 * Target selection: ert.tlc
 * Embedded hardware selection: Intel->x86-64 (Windows64)
 * Code generation objectives:
 *    1. Execution efficiency
 *    2. RAM efficiency
 * Validation result: Not run
 */

#include "ZJ.h"
#define NumBitsPerChar                 8U

/* Block signals and states (default storage) */
DW rtDW;

/* Real-time model */
RT_MODEL rtM_;
RT_MODEL *const rtM = &rtM_;

/* Forward declaration for local functions */
static real_T trapmf_k(real_T x, const real_T params[4]);
static void trapmf_n(const real_T x[101], const real_T params[4], real_T
y[101]);
```

```
static void trimf_g(const real_T x[101], const real_T params[3], real_T
y[101]);
extern real_T rtInf;
extern real_T rtMinusInf;
extern real_T rtNaN;
extern real32_T rtInfF;
extern real32_T rtMinusInfF;
extern real32_T rtNaNF;
extern void rt_InitInfAndNaN(size_t realSize);
extern boolean_T rtIsInf(real_T value);
extern boolean_T rtIsInfF(real32_T value);
extern boolean_T rtIsNaN(real_T value);
extern boolean_T rtIsNaNF(real32_T value);
typedef struct {
  struct {
    uint32_T wordH;
    uint32_T wordL;
  } words;
} BigEndianIEEEDouble;

typedef struct {
  struct {
    uint32_T wordL;
    uint32_T wordH;
  } words;
} LittleEndianIEEEDouble;

typedef struct {
  union {
    real32_T wordLreal;
    uint32_T wordLuint;
  } wordL;
} IEEESingle;

real_T rtInf;
real_T rtMinusInf;
real_T rtNaN;
real32_T rtInfF;
real32_T rtMinusInfF;
```

```
real32_T rtNaNF;
extern real_T rtGetInf(void);
extern real32_T rtGetInfF(void);
extern real_T rtGetMinusInf(void);
extern real32_T rtGetMinusInfF(void);
extern real_T rtGetNaN(void);
extern real32_T rtGetNaNF(void);

/*
 * Initialize the rtInf, rtMinusInf, and rtNaN needed by the
 * generated code. NaN is initialized as non-signaling. Assumes IEEE.
 */
void rt_InitInfAndNaN(size_t realSize)
{
  (void) (realSize);
  rtNaN = rtGetNaN();
  rtNaNF = rtGetNaNF();
  rtInf = rtGetInf();
  rtInfF = rtGetInfF();
  rtMinusInf = rtGetMinusInf();
  rtMinusInfF = rtGetMinusInfF();
}

/* Test if value is infinite */
boolean_T rtIsInf(real_T value)
{
  return (boolean_T)((value==rtInf || value==rtMinusInf) ? 1U : 0U);
}

/* Test if single-precision value is infinite */
boolean_T rtIsInfF(real32_T value)
{
  return (boolean_T)(((value)==rtInfF || (value)==rtMinusInfF) ? 1U : 0U);
}

/* Test if value is not a number */
boolean_T rtIsNaN(real_T value)
{
```

```
  return (boolean_T)((value!=value) ? 1U : 0U);
}

/* Test if single-precision value is not a number */
boolean_T rtIsNaNF(real32_T value)
{
  return (boolean_T)(((value!=value) ? 1U : 0U));
}

/*
 * Initialize rtInf needed by the generated code.
 * Inf is initialized as non-signaling. Assumes IEEE.
 */
real_T rtGetInf(void)
{
  size_t bitsPerReal = sizeof(real_T) * (NumBitsPerChar);
  real_T inf = 0.0;
  if (bitsPerReal == 32U) {
    inf = rtGetInfF();
  } else {
    union {
      LittleEndianIEEEDouble bitVal;
      real_T fltVal;
    } tmpVal;

    tmpVal.bitVal.words.wordH = 0x7FF00000U;
    tmpVal.bitVal.words.wordL = 0x00000000U;
    inf = tmpVal.fltVal;
  }

  return inf;
}

/*
 * Initialize rtInfF needed by the generated code.
 * Inf is initialized as non-signaling. Assumes IEEE.
 */
real32_T rtGetInfF(void)
```

```
{
  IEEESingle infF;
  infF.wordL.wordLuint = 0x7F800000U;
  return infF.wordL.wordLreal;
}

/*
 * Initialize rtMinusInf needed by the generated code.
 * Inf is initialized as non-signaling. Assumes IEEE.
 */
real_T rtGetMinusInf(void)
{
  size_t bitsPerReal = sizeof(real_T) * (NumBitsPerChar);
  real_T minf = 0.0;
  if (bitsPerReal == 32U) {
    minf = rtGetMinusInfF();
  } else {
    union {
      LittleEndianIEEEDouble bitVal;
      real_T fltVal;
    } tmpVal;

    tmpVal.bitVal.words.wordH = 0xFFF00000U;
    tmpVal.bitVal.words.wordL = 0x00000000U;
    minf = tmpVal.fltVal;
  }

  return minf;
}

/*
 * Initialize rtMinusInfF needed by the generated code.
 * Inf is initialized as non-signaling. Assumes IEEE.
 */
real32_T rtGetMinusInfF(void)
{
  IEEESingle minfF;
  minfF.wordL.wordLuint = 0xFF800000U;
```

```
  return minfF.wordL.wordLreal;
}

/*
 * Initialize rtNaN needed by the generated code.
 * NaN is initialized as non-signaling. Assumes IEEE.
 */
real_T rtGetNaN(void)
{
  size_t bitsPerReal = sizeof(real_T) * (NumBitsPerChar);
  real_T nan = 0.0;
  if (bitsPerReal == 32U) {
    nan = rtGetNaNF();
  } else {
    union {
      LittleEndianIEEEDouble bitVal;
      real_T fltVal;
    } tmpVal;

    tmpVal.bitVal.words.wordH = 0xFFF80000U;
    tmpVal.bitVal.words.wordL = 0x00000000U;
    nan = tmpVal.fltVal;
  }

  return nan;
}

/*
 * Initialize rtNaNF needed by the generated code.
 * NaN is initialized as non-signaling. Assumes IEEE.
 */
real32_T rtGetNaNF(void)
{
  IEEESingle nanF = { { 0 } };

  nanF.wordL.wordLuint = 0xFFC00000U;
  return nanF.wordL.wordLreal;
}
```

```
/* Function for MATLAB Function: '<S17>/Evaluate Rule Antecedents' */
static real_T trapmf_k(real_T x, const real_T params[4])
{
  real_T b_y1;
  real_T y2;
  b_y1 = 0.0;
  y2 = 0.0;
  if (x >= params[1]) {
    b_y1 = 1.0;
  }

  if (x < params[0]) {
    b_y1 = 0.0;
  }

  if ((params[0] <= x) && (x < params[1]) && (params[0] != params[1])) {
    b_y1 = 1.0 / (params[1] - params[0]) * (x - params[0]);
  }

  if (x <= params[2]) {
    y2 = 1.0;
  }

  if (x > params[3]) {
    y2 = 0.0;
  }

  if ((params[2] < x) && (x <= params[3]) && (params[2] != params[3])) {
    y2 = 1.0 / (params[3] - params[2]) * (params[3] - x);
  }

  return fmin(b_y1, y2);
}

/* Function for MATLAB Function: '<S17>/Evaluate Rule Consequents' */
static void trapmf_n(const real_T x[101], const real_T params[4], real_T
y[101])
```

```
{
  real_T a;
  real_T b;
  real_T c;
  real_T d;
  int32_T i;
  real_T b_y1;
  real_T y2;
  a = params[0];
  b = params[1];
  c = params[2];
  d = params[3];
  for (i = 0; i < 101; i++) {
    b_y1 = 0.0;
    y2 = 0.0;
    if (x[i] >= b) {
      b_y1 = 1.0;
    }

    if (x[i] < a) {
      b_y1 = 0.0;
    }

    if ((a <= x[i]) && (x[i] < b) && (a != b)) {
      b_y1 = 1.0 / (b - a) * (x[i] - a);
    }

    if (x[i] <= c) {
      y2 = 1.0;
    }

    if (x[i] > d) {
      y2 = 0.0;
    }

    if ((c < x[i]) && (x[i] <= d) && (c != d)) {
      y2 = 1.0 / (d - c) * (d - x[i]);
    }
```

```
    y[i] = fmin(b_y1, y2);
  }
}

/* Function for MATLAB Function: '<S13>/Evaluate Rule Consequents' */
static void trimf_g(const real_T x[101], const real_T params[3], real_T
y[101])
{
  real_T a;
  real_T b;
  real_T c;
  int32_T i;
  a = params[0];
  b = params[1];
  c = params[2];
  for (i = 0; i < 101; i++) {
    y[i] = 0.0;
    if ((a != b) && (a < x[i]) && (x[i] < b)) {
      y[i] = 1.0 / (b - a) * (x[i] - a);
    }

    if ((b != c) && (b < x[i]) && (x[i] < c)) {
      y[i] = 1.0 / (c - b) * (c - x[i]);
    }

    if (x[i] == b) {
      y[i] = 1.0;
    }
  }
}

/* Model step function for TID0 */
void ZJ_step0(void)                /* Sample time: [0.0017241379310344947s,
0.0s] */
{
  /* (no output/update code required) */
}
```

```
/* Model step function for TID1 */
void ZJ_step1(void)                    /* Sample time: [0.4948275862069s,
0.0s] */
{
  real_T inputMFCache[6];
  int32_T inputID;
  static const int8_T b[4] = { 1, 1, 2, 1 };

  static const int8_T c[8] = { 1, 3, 2, 3, 1, 3, 2, 0 };

  static const real_T d[4] = { -0.45, -0.05, 0.238, 0.4493 };

  static const real_T e[4] = { 0.261, 0.401, 0.55, 0.667019027484144 };

  static const real_T f[4] = { 0.55, 0.738900634249471, 1.05, 1.45 };

  real_T outputMFCache[303];
  static const int8_T b_0[4] = { 3, 2, 1, 2 };

  static const real_T c_0[4] = { -0.426, -0.0257, 0.197674418604651, 0.5
};

  static const real_T d_0[4] = { 0.548, 0.931289640591966, 1.05, 1.45 };

  static const real_T e_0[4] = { 0.212473572938689, 0.373, 0.601, 0.76 };

  real_T area;
  real_T inputMFCache_0[5];
  static const int8_T b_1[8] = { 1, 2, 1, 0, 1, 3, 2, 2 };

  static const int8_T b_2[4] = { 1, 3, 1, 2 };

  static const real_T c_1[3] = { 0.583333333333333, 1.0, 1.41666666666667
};

  static const real_T d_1[3] = { 0.0833333333333333, 0.5, 0.916666666666667
};
```

```
real_T rtb_defuzzifiedOutputs;
real_T rtb_antecedentOutputs[4];
real_T rtb_aggregatedOutputs[101];
int32_T i;
real_T tmp[3];
real_T tmp_0[101];
real_T tmp_1[101];
real_T tmp_2[101];
real_T rtb_TmpSignalConversionAtSFun_0;

/* Outputs for Atomic SubSystem: '<S8>/Fuzzy Logic Controller' */
/* MATLAB Function: '<S17>/Evaluate Rule Antecedents' */
area = 0.0;
inputMFCache[0] = trapmf_k(0.848275862069, d);
inputMFCache[1] = trapmf_k(0.848275862069, e);
inputMFCache[2] = trapmf_k(0.848275862069, f);
inputMFCache[3] = 2.7101000202178775E-7;
inputMFCache[4] = 0.0982002191835399;
inputMFCache[5] = 0.91468969691693336;
for (i = 0; i < 4; i++) {
  rtb_defuzzifiedOutputs = (b[i] == 1);
  if (b[i] == 1) {
    if (rtb_defuzzifiedOutputs > inputMFCache[c[i] - 1]) {
      rtb_defuzzifiedOutputs = inputMFCache[c[i] - 1];
    }
  } else {
    if (rtb_defuzzifiedOutputs < inputMFCache[c[i] - 1]) {
      rtb_defuzzifiedOutputs = inputMFCache[c[i] - 1];
    }
  }

  if (c[i + 4] != 0) {
    if (b[i] == 1) {
      if (rtb_defuzzifiedOutputs > inputMFCache[c[i + 4] + 2]) {
        rtb_defuzzifiedOutputs = inputMFCache[c[i + 4] + 2];
      }
    } else {
      if (rtb_defuzzifiedOutputs < inputMFCache[c[i + 4] + 2]) {
```

```
          rtb_defuzzifiedOutputs = inputMFCache[c[i + 4] + 2];
        }
      }
    }

    area += rtb_defuzzifiedOutputs;
    rtb_antecedentOutputs[i] = rtb_defuzzifiedOutputs;
  }

  /* MATLAB Function: '<S17>/Evaluate Rule Consequents' incorporates:
   *  Constant: '<S17>/Output Sample Points'
   */
  trapmf_n(rtConstP.pooled2, e_0, tmp_2);
  trapmf_n(rtConstP.pooled2, d_0, tmp_1);
  trapmf_n(rtConstP.pooled2, c_0, tmp_0);
  for (i = 0; i < 101; i++) {
    rtb_aggregatedOutputs[i] = 0.0;
    outputMFCache[3 * i] = tmp_2[i];
    outputMFCache[1 + 3 * i] = tmp_1[i];
    outputMFCache[2 + 3 * i] = tmp_0[i];
  }

  for (i = 0; i < 4; i++) {
    for (inputID = 0; inputID < 101; inputID++) {
      rtb_defuzzifiedOutputs = rtb_aggregatedOutputs[inputID];
      if (outputMFCache[(3 * inputID + b_0[i]) - 1] >
rtb_antecedentOutputs[i])
      {
        rtb_TmpSignalConversionAtSFun_0 = rtb_antecedentOutputs[i];
      } else if (rtIsNaN(outputMFCache[(3 * inputID + b_0[i]) - 1])) {
        rtb_TmpSignalConversionAtSFun_0 = rtb_antecedentOutputs[i];
      } else {
        rtb_TmpSignalConversionAtSFun_0 = outputMFCache[(3 * inputID +
b_0[i]) -
          1];
      }

      if (rtb_aggregatedOutputs[inputID] <
rtb_TmpSignalConversionAtSFun_0) {
```

```
        rtb_defuzzifiedOutputs = rtb_TmpSignalConversionAtSFun_0;
      }

      rtb_aggregatedOutputs[inputID] = rtb_defuzzifiedOutputs;
    }
  }

  /* End of MATLAB Function: '<S17>/Evaluate Rule Consequents' */

  /* MATLAB Function: '<S17>/Defuzzify Outputs' incorporates:
   *  Constant: '<S17>/Output Sample Points'
   *  MATLAB Function: '<S17>/Evaluate Rule Antecedents'
   */
  if (area == 0.0) {
    /* SignalConversion: '<S1>/TmpSignal ConversionAtAnimation1Inport1' */
    rtDW.TmpSignalConversionAtAnimatio_m[0] = 0.5;
  } else {
    rtb_defuzzifiedOutputs = 0.0;
    area = rtb_aggregatedOutputs[0];
    for (i = 0; i < 100; i++) {
      area += rtb_aggregatedOutputs[i + 1];
    }

    if (area == 0.0) {
      /* SignalConversion: '<S1>/TmpSignal ConversionAtAnimation1Inport1'
*/
      rtDW.TmpSignalConversionAtAnimatio_m[0] = 0.5;
    } else {
      for (i = 0; i < 101; i++) {
        rtb_defuzzifiedOutputs       +=       rtConstP.pooled2[i]       *
rtb_aggregatedOutputs[i];
      }

      /* SignalConversion: '<S1>/TmpSignal ConversionAtAnimation1Inport1'
incorporates:
       *  Constant: '<S17>/Output Sample Points'
       */
      rtDW.TmpSignalConversionAtAnimatio_m[0] = 1.0 / area *
        rtb_defuzzifiedOutputs;
```

```
  }
}

/* End of MATLAB Function: '<S17>/Defuzzify Outputs' */
/* End of Outputs for SubSystem: '<S8>/Fuzzy Logic Controller' */

/* Outputs for Atomic SubSystem: '<S7>/Fuzzy Logic Controller' */
/* MATLAB Function: '<S13>/Evaluate Rule Antecedents' */
area = 0.0;
inputMFCache_0[0] = 0.93720317541791687;
inputMFCache_0[1] = 0.00015430101324937512;
inputMFCache_0[2] = 0.67097436269442956;
inputMFCache_0[3] = 0.34353650586224221;
inputMFCache_0[4] = 0.00068597615844099256;
for (i = 0; i < 4; i++) {
  rtb_defuzzifiedOutputs = 1.0;
  if (b_1[i] != 0) {
    if (1.0 > inputMFCache_0[b_1[i] - 1]) {
      rtb_defuzzifiedOutputs = inputMFCache_0[b_1[i] - 1];
    } else {
      rtb_defuzzifiedOutputs = 1.0;
    }
  }

  if (rtb_defuzzifiedOutputs > inputMFCache_0[b_1[i + 4] + 1]) {
    rtb_defuzzifiedOutputs = inputMFCache_0[b_1[i + 4] + 1];
  }

  area += rtb_defuzzifiedOutputs;
  rtb_antecedentOutputs[i] = rtb_defuzzifiedOutputs;
}

/* MATLAB Function: '<S13>/Evaluate Rule Consequents' incorporates:
 *  Constant: '<S13>/Output Sample Points'
 */
tmp[0] = -0.416666666666667;
tmp[1] = 0.0;
tmp[2] = 0.416666666666667;
```

```
  trimf_g(rtConstP.pooled2, tmp, tmp_2);
  trimf_g(rtConstP.pooled2, d_1, tmp_1);
  trimf_g(rtConstP.pooled2, c_1, tmp_0);
  for (i = 0; i < 101; i++) {
    rtb_aggregatedOutputs[i] = 0.0;
    outputMFCache[3 * i] = tmp_2[i];
    outputMFCache[1 + 3 * i] = tmp_1[i];
    outputMFCache[2 + 3 * i] = tmp_0[i];
  }

  for (i = 0; i < 4; i++) {
    for (inputID = 0; inputID < 101; inputID++) {
      rtb_defuzzifiedOutputs = rtb_aggregatedOutputs[inputID];
      if (outputMFCache[(3 * inputID + b_2[i]) - 1] >
rtb_antecedentOutputs[i])
      {
        rtb_TmpSignalConversionAtSFun_0 = rtb_antecedentOutputs[i];
      } else if (rtIsNaN(outputMFCache[(3 * inputID + b_2[i]) - 1])) {
        rtb_TmpSignalConversionAtSFun_0 = rtb_antecedentOutputs[i];
      } else {
        rtb_TmpSignalConversionAtSFun_0 = outputMFCache[(3 * inputID +
b_2[i]) -
          1];
      }

      if (rtb_aggregatedOutputs[inputID] <
rtb_TmpSignalConversionAtSFun_0) {
        rtb_defuzzifiedOutputs = rtb_TmpSignalConversionAtSFun_0;
      }

      rtb_aggregatedOutputs[inputID] = rtb_defuzzifiedOutputs;
    }
  }

  /* End of MATLAB Function: '<S13>/Evaluate Rule Consequents' */

  /* MATLAB Function: '<S13>/Defuzzify Outputs' incorporates:
   *  Constant: '<S13>/Output Sample Points'
   *  MATLAB Function: '<S13>/Evaluate Rule Antecedents'
```

```
   */
  if (area == 0.0) {
    /* SignalConversion: '<S1>/TmpSignal ConversionAtAnimation1Inport1' */
    rtDW.TmpSignalConversionAtAnimatio_m[1] = 0.5;
  } else {
    rtb_defuzzifiedOutputs = 0.0;
    area = rtb_aggregatedOutputs[0];
    for (i = 0; i < 100; i++) {
      area += rtb_aggregatedOutputs[i + 1];
    }

    if (area == 0.0) {
      /* SignalConversion: '<S1>/TmpSignal ConversionAtAnimation1Inport1'
*/
      rtDW.TmpSignalConversionAtAnimatio_m[1] = 0.5;
    } else {
      for (i = 0; i < 101; i++) {
        rtb_defuzzifiedOutputs       +=       rtConstP.pooled2[i]       *
rtb_aggregatedOutputs[i];
      }

      /* SignalConversion: '<S1>/TmpSignal ConversionAtAnimation1Inport1'
incorporates:
       *  Constant: '<S13>/Output Sample Points'
       */
      rtDW.TmpSignalConversionAtAnimatio_m[1] = 1.0 / area *
        rtb_defuzzifiedOutputs;
    }
  }

  /* End of MATLAB Function: '<S13>/Defuzzify Outputs' */
  /* End of Outputs for SubSystem: '<S7>/Fuzzy Logic Controller' */
}

/* Model step function for TID2 */
void ZJ_step2(void)                    /* Sample time: [2.0s, 0.0s] */
{
  /*   SignalConversion:   '<S7>/TmpSignal   ConversionAtAnimation1Inport1'
incorporates:
```

```
   *  Constant: '<S4>/Vyhodnotit Gap Fit analyzu Y'
   *  Constant: '<S6>/Odhad casoveho vytizeni Y'
   */
  rtDW.TmpSignalConversionAtAnimation1[0] = 0.1620689655172;
  rtDW.TmpSignalConversionAtAnimation1[1] = 0.1896551724138;

  /*  SignalConversion:  '<S8>/TmpSignal  ConversionAtAnimation1Inport1'
incorporates:
   *  Constant: '<S3>/Odhad nákladů Y'
   *  Constant: '<S5>/Odhad casove narocnosti vyvoje Y'
   */
  rtDW.TmpSignalConversionAtAnimatio_a[0] = 0.848275862069;
  rtDW.TmpSignalConversionAtAnimatio_a[1] = 0.9103448275862;
}

/* Model initialize function */
void ZJ_initialize(void)
{
  /* Registration code */

  /* initialize non-finites */
  rt_InitInfAndNaN(sizeof(real_T));
}

/*
 * File trailer for generated code.
 *
 * [EOF]
 */
```

11.2 Příloha 2. Zdrojová Gap-Fit analýza

Gap-Fit analýza ÚJV Řež, a. s.

Podkladem pro Gap-Fit analýzu je dokument UJV_Analyza_procesu_a_pozadavku-2-0.docx

Celkem hodin dle filtru 470

Požadavek	Popis	Oblast	Původ	Gap/Fit	Doba vývoje (hod)	Priorita	Poznámka NAVISYS	Poznámka
POZ01.	Pro UJV zachovat funkčnost při vytvoření nového projektu doplnění etap a úloh s účty, které k nim přísluší	PROJEKTY	Analýza	Gap	0	1	Funkce na vytvoření úloh a řádků plánování. Zahrnuto v požadavku POZ02	
POZ02.	Zachovat a upravit funkci, která vytváří etapy a v nich jednotlivé úlohy – možnost zvolit, kolik etap se má vytvořit a které úlohy se v nich mají vytvořit. Dodatečně bude možné vytvořit jakoukoli etapu nebo i dílčí úlohu, úlohy, které již budou existovat, budou přeskočeny (funkce neskončí chybou)	PROJEKTY	Analýza	Gap	4	1		
POZ03.	Vytvořit report, který vyhodnotí jen projekt sám nebo projekt včetně jeho podprojektů	PROJEKTY	Analýza	Gap	x	x	storno	
POZ04.	Vytvořit report, který zpětně změní/doplní dimenzi do položek projektu a věcných položek pro klasifikaci projektu, když se tento údaj změní na kartě (možnost vybrat projekt, kód dimenze, hodnotu dimenze, období, čísla účtů).	PROJEKTY	Analýza	Gap New	8	4	Doplnění pouze do položek projektu a věcných položek + související tabulky dimenzí. Nutno stanovit striktní pravidla a omezení při použití.	
POZ05.	Zachovat funkčnosti plánování včetně možností používaného reportingu.	PROJEKTY	Analýza	Gap	16	1	Požadavek nezahrnuje vytvoření reportů. Zahrnuto je vytvoření detailu k řádku cash flow, použití tohoto nastavení v průvodci vytvořením prodejní faktury, vytvoření samostatných řádků na faktuře, možnost potlačení jejich tisku, přidání polí na kartě	V rámci požadavku i funkčnost, kdy se při návrhu faktury bere číslo výnosového účtu ze řádku plánování příslušné úlohy. Funkč

							projektu a v úloze projektu.	nost stavů etap, jejich uzavírání ve vazbě na stav celého projektu.
POZ06.	Navrhnout možné řešení fakturace divize EGP s rozdělením na střediska pomocí standardních funkcí NAV.	PROJEKTY	Analýza	Gap	x	x	storno	
POZ07.	Vytvořit reporty Termínové hlášení, Přehled fakturací oddělení, Rozpracovanost útvaru a Externí a interní subdodávky	PROJEKTY	Analýza	Gap	24	2	24 h = převod reportů NAV do nové verze	
POZ08.	Přenos rozpočtu projektů do jednoho fin. rozpočtu dle nastavení – vždy vytvořit nové položky, neimportovat rozdíly. Na položce rozpočtu bude vyplněno středisko i pole Číslo projektu. Při vytváření rozpočtu doplňovat s číslem projektu také stav zajištěnosti a úlohu projektu. Upravit kopírování rozpočtů a přenášet s číslem projektu také stav zajištěnosti a úlohu projektu.	ROZPOČTY	Analýza	Gap New	12	1	Úprava stávající funkce. Navíc doplnění funkčnosti - přenos řádku typu Zdroj. Nová tabulka rozdělení dle tabulky 50022? pro definici zdroj/skupina zdrojů/vše versus účet.	
POZ09.	Import dat do fin.rozpočtu jednotlivými rozpočtáři se zajištěním ochrany před možným vzájemným přepsáním či vymazáním dat	ROZPOČTY	Analýza	Fit		1	Možnost pomocí nastavení předdefinovat uživateli filtry dimenzí ve fin. rozpočtu (dimenze Středisko, Rozpočtář), neumožnit filtry dimenzí změnit.	Standard s nahrám odpovědnou osobou, v tom případě FIT.
POZ09.01	Jet Report na porovnání rozpočtů	ROZPOČTY	Analýza	Gap New	12	3	Možno řešit reportem v NAV	
POZ10.	Vytvoření šablon s nezbytnými údaji a zabezpečením k úspěšnému importu finančního rozpočtu z Excelu (strukturu finančních dat potřebných pro plánování zajistí oddělení controlling)	ROZPOČTY	Analýza	Gap New	2	2	Součinnost NAVISYS při vytvoření šablon.	

POZ11.	Účtování práce na projekty v ÚJV s použitím současné metodiky dle sazeb. Ostatní společnosti dle skutečnosti.	VYKAZOVÁNÍ PRÁCE	Analýza	Gap	10	1	Sazby dle kategorie zaměstnance. Kategorie zaměstnance nastavena na skupině zdrojů. Vkládání při validaci čísla zdroje v deníku projektů. Převod funkce na výpočet skutečné sazby a doplnění do deníku. Předpokladem je předchozí načtení mzdových složek ze mzdového SW (není předmětem tohoto požadavku).	
POZ12.	Možnost zaznamenávat nemoci v deníku zdrojů – neprůměrovat mzdu jako celek, ale navázat na mzdové složky.	VYKAZOVÁNÍ PRÁCE	Analýza	Gap New	32	3	Před účtováním závěrky, nutná neustálá aktualizace mzdových složek na rozúčtování mezd, některé složky mají definovaný účetní předpis. Komunikace NAV a Personální systém	
POZ13.	Možnost kopírovat řádky deníku zdrojů z předešlého měsíce.	VYKAZOVÁNÍ PRÁCE	Analýza	Fit		2	Kopie do Excelu z deníku projektů před zaúčtováním, poté kopie z Excelu do NAV	
POZ14.	Nad deníkem zdrojů se tiskne sestava pro kontrolu - Pracovní výkaz. Tyto sestavy je nutné zachovat. Např. možnost zkontrolovat, jaká částka se zaúčtuje na projekt, jaký objem hodin se účtuje za zdroj, středisko, projekt.	VYKAZOVÁNÍ PRÁCE	Analýza	Gap	8	1	Sestava v NAV nad deníkem zdrojů Pracovní výkaz. Přesunout nad deník projektů.	
POZ15.	V případě rozhodnutí o používání časových výkazů možnost vykazovat souhrnně za měsíc a vidět ve výkazu měsíční pracovní fond. Možnost změnit pro každého zaměstnance dle skutečnosti.	VYKAZOVÁNÍ PRÁCE	Analýza	Fit		3	Vytvoření měsíčního fondu na zvolený den měsíce (např. poslední den)	
POZ16.	Zachovat stávající pořizování pracovních výkazů importem ze speciální tabulky EGPI	VYKAZOVÁNÍ PRÁCE	Analýza	Gap	6	2	Import do deníku projektů místo deníku zdrojů	

POZ17.	Nahradit v současnosti používaný specifický report pro rozpočítání režií CVŘ.	REŽIE	Analýza	Gap	8	1	Možnost nastavit v systému parametry pro rozpočítání režií. Rozpočítání po divizích (filtr na část střediska), podle mzdových nákladů rozpočítané režie, s vyloučením urč. projektů. Správní režie za celou firmu, opět s vyloučením vybraných projektů, na které správní režie nelze účtovat.	Na workshopu prezentace řešení pomocí nákladového účetnictví
POZ18.	Zaintegrovat rozvrhové základny do NAV pro zaúčtování nákladů z prvotních dokladů pro CVŘ	REŽIE	Analýza	Fit		1	Vytvoření řádků rozdělení včetně šablon nad nákupními řádky. Pokud využití periodických fin. deníků a dodatečného rozpočítání nákladů po zaúčtování prvotního dokladu, potom fit. Řešení musí respektovat workflow nák. požadavků etapu 3	Po workshopu rozhodnuto, že budou periodické deníky rozdělení (nebo nákladové účetnictví), tedy Fit
POZ19.	Upravit funkčnosti NAV 2013 tak, aby se popsaná účetní metodika výpočtu a účtování nedokončené výroby používala jako dosud.	MĚSÍČNÍ ZÁVĚRKA	Analýza	Gap New	16	1	Bude zachován původní report a převeden, se zápisem na karty projektů až PO ZAÚČTOVÁNÍ	

POZ220.	Zachovat funkce pro zpracování měsíční závěrky (mimo funkčnosti Cenotvorby-Skladové režie). Řešení v jiném systému by vyžadovalo nového dodavatele a podstatné navýšení rozpočtu.	MĚSÍČNÍ ZÁVĚRKA	Analýza	Gap	80	1	Kooperace při vývoji 40h. Jedná se o funkce: - Rozúčtování mezd (R 92205) - Výpočet přímých a režijních nákladů (R 92206) - Přeúčtování režijních nákladů (R 90830) - Výpočet provozní režie (R 90815) - Výpočet správní režie (R 90826) - Přeúčtování mzdové složky (R 50013) - report Aktualizace zaměstnanců/prodejců/zdroj Hromadná rozúčtování sníží chybovost a čas v produkci.	
POZ221.	Zjistit rozhodnutí, zda zachovat/začít používat/zrušit VP fakturaci interních projektů s ohledem na rozdělení projektů na etapy (nelze pak použít).	MĚSÍČNÍ ZÁVĚRKA	Analýza	Gap	x	x	storno	
POZ222.	Přepočet směnných kurzů se účtuje s obecným střediskem ne se střediskem na původním dokladu.	MĚSÍČNÍ ZÁVĚRKA	Analýza	Gap	3	1		
POZ223.	Navrhnout detailně metody alokace odpisů na projekty.	DLOUHODOBÝ MAJETEK	Analýza	Gap New	24	4	Návrh + zpracování požadavku na různé metody alokace (1 projekt (2h), pevné rozdělení na více projektů (16h), proměnné rozdělení dle vykázaných výkonů souvisejícího zdroje (předchozích 16h+6h)).	
POZ224.	Vytvořit report pro sloučení operativní evidence a dlouhodobého majetku. Společnost UJV dodá poklady, které karty operativní evidence se budou slučovat do DM	DLOUHODOBÝ MAJETEK	Analýza	Gap New	16	1		
POZ225.	Navrhnout řešení pro modul investic.	INVESTICE	Analýza	Gap New	16	3		

POZ26.	Kontrolovat, aby nemohl existovat účet v jednotlivých společnostech bez vazby na společnou účetní osnovu.	ÚČETNÍ OSNOVA	Analýza	Gap New	2	2		
POZ27.	Ve společné účetní osnově vyplnit MD a Dal účet konsolidace účtem účetní osnovy ČEZ.	ÚČETNÍ OSNOVA	Analýza	Fit		2	Vyplní ÚJV	
POZ28.	Nad společnou účetní osnovou vytvořit účetní schéma pro manažerskou výsledovku a pomocí reportu v JetReports ji tisknout z účetních dat v jednotlivých společnostech.	ÚČETNÍ OSNOVA	Analýza	Gap New	4	3	Součinnost při tvorbě JetReportu	
POZ29.	Umožnit certifikovat karty kontaktů	KONTAKTY	Analýza	Gap New	x	x	storno	Souvisí s POZ31.
POZ30.	Vytvořit nové pole na kartě kontaktu s vazbou na kontakt v centrální databázi kontaktů.	KONTAKTY	Analýza	Gap New	x	x	storno	
POZ31.	Vytvořit funkci na přenášení kontaktů do dceřiných společností	KONTAKTY	Analýza	Gap New	x	x	storno	Nutné je interně realizovat POZ30. - bez tohoto bodu nemá tento požadavek smysl, protože by se vždy vytvářely nové kontakty
POZ32.	Funkce na přečíslování lidských zdrojů a souvisejících tabulek pro potřeby HRIS	HRIS	Analýza	Gap New	1	1	Na kartu lidského zdroje doplnit pole pro číslo zaměstnance v HRIS. Bude naplněno v rámci parametrizace.	
POZ33.	Dodatečná analýza pro komunikace HRIS a NAV 2013. Odsouhlasit, co se nahraje z NAV do HRIS. Z HRIS do NAV přenášet kromě jiných dat také fyzický počet pracovníků, přepočtený počet pracovníků a průměrný počet pracovníků.	HRIS	Analýza	Gap New	16	1	Pouze přenos funkčností do verze NAV2013 - zbytek v požadavku pro verzi NAV2009 (UJV14142)	
POZ34.	Upravit přepočet mezd tak, aby se i sociální a zdravotní pojištění rozpočítalo na projekty. Spočítat procento sociálního a zdravotní pojištění za zaměstanavatele.	HRIS	Analýza	Gap New	x	x	storno	
POZ35.	V NAV2013 vytvořit číselník mzdových složek z KS Programu a nastavit, jak se budou mzdové složky v NAV2013 účtovat	HRIS	Analýza	Gap	0	1	Řeší se v rámci požadavku POZ33	

POZ36.	Vytvářet příkazy pro zahraniční platby (nastavení).	OSTATNÍ	Analýza	Fit		1	Nastavení v rámci parametrizace systému	
POZ37.	Převést nebo navrhnout alternativu pro import řádků fin. deníku z MS Excel.	OSTATNÍ	Analýza	Fit		1	Ctrl + C, Ctrl + V z Excelu	
POZ38.	Prověřit funkčnost globálních filtrů ve formulářích a sestavách a ocenit úpravu doplnění globálních filtrů do sestav	OSTATNÍ	Analýza	Fit		1	Bude řešeno nastavením globálních filtrů na rolích v Nastavení uživatelů	
POZ39.	Úpravy reportů a dataportů, které nejsou výslovně specifikovány v jiném požadavku, dle specifikace ÚJV a dcer - standardní reporty, specifické reporty	OSTATNÍ	Analýza	Gap	64	1		Specifikovat každý objekt
POZ40.	Navrhnout evidenci obalových materiálů ve standardu NAV2013	OSTATNÍ	Workshop	Fit		2	Parametrizace, zaškolení uživatelů	
POZ41.	Zprovoznit systém upomínek k pohledávkám	OSTATNÍ	Workshop	Fit		2	Parametrizace, zaškolení uživatelů	
POZ42.	V sestavě vydaného pokladního dokladu volitelně zobrazovat čísla účtů	OSTATNÍ	Workshop	Gap	3	1		
POZ43.	Pod kartou smlouvy zobrazit seznam projektů, na které je smlouva navázaná	OSTATNÍ	Workshop	Gap New	2	3		
POZ44	Nákupní požadavky a vazba na SP	OSTATNÍ	Workshop	Gap	24	1	WF1 + WF2 (řešení nákupních požadavků a objednávek)	
POZ45.	Scanování příloh (došlých faktur, smluv)	OSTATNÍ	Workshop	Gap	4	1	Převod stávající funkčnosti	
POZ46.	Inventarizace majetku pomocí čteček	DLOUHODOBÝ MAJETEK	Workshop	Gap	12	3	Převod stávající funkčnosti + úprava - již nebude OE	
POZ47.	Pole na Prodejcích/Nákupčích – tituly, nadřízená osoba, adresa. Úpravy pro EGPI v pokladně, tisk adresy na pokladním dokladu	OSTATNÍ	Workshop	Gap	4	1		
POZ48.	Pole na kartě zaměstnance - to co je na kartě prodejce a kartě zdroje, musí být na kartě zaměstnance v personalistice + pole pro systemizační číslo	OSTATNÍ	Workshop	Gap	2	1		
POZ49.	Pole na kartě majetku a vytváření knih odpisů dle nastavení tříd DM	DLOUHODOBÝ MAJETEK	Workshop	Gap	4	1		
POZ50.	Pole přenášené při archivaci prodejních a nákupních dokladů	OSTATNÍ	Workshop	Gap	2	1		
POZ51.	Výpis svázaný s příkazem – kontrola typu a čísla vyrovnání dokladu	OSTATNÍ	Workshop	Gap	4	1		
POZ52.	Zachovat funkčnost tisku prodejní faktury s řádky faktury nebo souhrnně (fakturace se účtuje na různá střediska, tisk pro zákazníka je pouze ve výsledné částce).	OSTATNÍ	Workshop	Gap	0	1	Řešeno v rámci požadavku POZ05	
POZ53.	Přidaná pole v hodnotách dimenzí převést do nového NAV	OSTATNÍ	Workshop	Gap	1	1	Kromě pole VP úrok	

POZ54.	Umožnit nastavení oprávnění na listy deníku projektů pro výkazy práce	VYKAZOVÁNÍ PRÁCE	Workshop	Fit		1	Nastaví ÚJV po zaškolení na používání globálních filtrů v Nastavení uživatelů, viz také POZ38.	
POZ55.	Přenést kontroly z deníku zdrojů do deníku projektů při vkládání dat i při účtování	VYKAZOVÁNÍ PRÁCE	Návrh	Gap New	8	1	Včetně schválení řádků ekonomem	
POZ56.	Účtovat do projektů DPH nastavenou na nákladový účet	OSTATNÍ	Workshop	Gap	1	1		
POZ57.	Do řádku příkazu doplnit popis z položky dodavatele. Popis z položky dodavatele tisknout v reportu Bankovní příkaz	OSTATNÍ	Workshop	Gap New	2	4	Požadavek CVŘ, lze jej realizovat změnou procesu, potom FIT	
POZ58.	Do položek zákazníka zobrazit sloupec Úroveň posl. vydané upomínky.	OSTATNÍ	Workshop	Fit		1		
POZ59.	Na kartu DM doplnit pole: způsob pořízení, vyřazení, budova	DLOUHODOBÝ MAJETEK	Workshop	Gap	1	1		
POZ60.	Nový formulář pro ICT nad kartou DM.	DLOUHODOBÝ MAJETEK	Workshop	Gap	2	2		
POZ61.	Označení karty DM, která bude k vyřazení.	DLOUHODOBÝ MAJETEK	Workshop	Fit		1	Využití standardního pole "V údržbě"	
POZ62.	Kontrola účtování přímých a režijních nákladů v závislosti na zajištěnosti a typu projektu, kontrola účtování na projekt pouze v rámci divize - středisko účetního případu musí náležet stejné divizi jako středisko projektu.	OSTATNÍ	Workshop	Gap	1	1	Na projekty se zajištěností jinou než 1 a všechny projekty N se musí účtovat pouze režijní náklady. Na projekty se zajištěností 1 a různé od N se musí účtovat přímé náklady.	
POZ63.	Převést report Cash Flow řešený v JetReports, aby jej bylo možno používat s NAV 2013.	OSTATNÍ	Workshop	Gap	8	2		
POZ64.	Na kartu projektu doplnit pole na nový číselník Druh projektu.	PROJEKTY	Workshop	Gap New	1	2		
					470			
	V projektu	360						
	Zůstatek	110						
Legenda k vybraným sloupcům:								
Původ	*Analýza - požadavek vznikl v průběhu analýzy*							
	Workshop - požadavek byl zadán během workshopu analýzy nebo návrhu							

Gap/Fit	*Gap - požadavek není v systému řešen, bude nutno provést programové úpravy, u těchto požadavků bude uvedena časová náročnost zpracování v hodinách ve sloupci "Doba vývoje (hod)"*							
	Fit - požadavek je součástí standardní funkčnosti							
	NSS - požadavek je řešen v základním řešení Navisys dodávaném jako nadstavba systému							
	BIZ4BuildIn - požadavek je řešen v modulu BIZ4BuildIn							
Priorita	*1 - požadavek bude zpracován prioritně, při ostrém spuštění NAV bude funkčnost popisovaná v požadavku součástí systému*							
	2 - řešení požadavku bude součástí systému do 1 měsíce od ostrého spuštění							
	3 - řešení požadavku bude součástí systému do 3 měsíců od ostrého spuštění, případně bude požadavek na základě dodatečného rozhodnutí přesunut do priority 4							
	4 - požadavky této priority nebudou součástí implementace. K jejich řešení je možné se vrátit v budoucnu, doporučujeme po určité době práce s novým systémem.							

11.3 Příloha 3. Schéma pro zachcycení názoru experta - prázdné schéma, funkce příslušnosti

Scénář:

FZJ	FZJ4	FZJ3	FZJ2	FZJ1	FZJ3	FZJ4	FZJ3
Požadavek	**Popis**	**Gap/Fit**	**Doba vývoje (hod)**	**Náklady na customizaci (CZK)**	**Priorita**	**Časové vytížení uživatelů, Nároky na klíčové uživatele při vývoji a při provozu**	**Standard vs Firemní logika popis**
POZ07.	Vytvořit reporty Terminové hlášení, Přehled fakturací oddělení, Rozpracovanost útvaru a Externí a interní subdodávk,	Gap	24	10 000,00	2	Převod reportů NAV do nové verze, nutná kooperace při zadání a analýze 38h. V produkci nutná aktualizace nastavení, update verzí.	Nová verze NAV prioritizuje řešení reportů v externích systémech.
POZ12.	Možnost zaznamenávat nemoci v deníku zdrojů – neprůměrovat mzdu jako celek, ale navázat na mzdové složky.	Gap	32	15 000,00	3	Před účtováním závěrky, nutná neustálá aktualizace mzdových složek na rozúčtování mezd, některé složky mají definovný účetní předpis. Komunikace NAV a Personální systém.	Pomůže při vykazování práce na dotační projekty zejména ekonomům.
POZ20.	Zachovat funkce pro zpracování měsíční závěrky (mimo funkčnosti Cenotvorby-Skladové režie). Řešení v jiném systému by vyžadovalo nového dodavatele a podstatné navýšení rozpočtu.	Gap	80	30 000,00	1	Kooperace při vývoji 120h. Jedná se o funkce:- Rozúčtování mezd (R 92205) - Výpočet přímých a režijních nákladů (R 92206) - Přeúčtování režijních nákladů (R 90830) - Výpočet provozní režie (R 90815) - Výpočet správní režie (R 90826) - Přeúčtování mzdové složky (R 90813) - report Aktualizace zaměstnanců/prodejců/zdroj Hromadná rozúčtování sníží chybovost a čas v produkci úspora za 100h měsíčně.	Z hlediska auditu a vykazování je závěrka klíčová. Know How rozúčtování by mělo být konsolidováno do jednoho systému.

Schéma pro zadání dílčích a konečného výsledku: