ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

katedra řídicí techniky

Grafické prostředí pro návrh fuzzy systémů

Diplomová práce

Vedoucí práce: Doc. Ing. Petr Horáček, CSc.

Diplomant: Martin Langer leden 2003

Anotace

Tato diplomová práce je příspěvkem k návrhu grafického rozhraní mezi návrhářem fuzzy systému a programovým prostředkem, který návrh, validaci a implementaci fuzzy systému podporuje. Shrnuje základní poznatky o fuzzy množinách a fuzzy logice a naznačuje obecný postup používaný při tvorbě fuzzy systémů. Po tomto teoretickém rozboru následuje přehled komerčních i akademických programů pro návrh fuzzy systémů. Jsou definována srovnávací kritéria, na základě kterých jsou vybrané programy porovnány. Důraz je kladen především na uživatelské hledisko (snadnost ovládání, vzhled, přehlednost atd.). Výsledky porovnávací studie vedou k návrhu a implementaci nového grafického rozhraní programu pro návrh fuzzy systémů splňujícího definovaná kritéria.

Annotation

The thesis deals with software applications supporting design, validation and implementation of fuzzy systems. It summarizes fuzzy sets and fuzzy logic basics and describes common procedure for manual design of a fuzzy system. After this theoretical introduction there is an overview of commercial and academical products used in the specified domain. The evaluation criteria are defined and tested applications are compared. The user point of view (ease of control, design, lucidity and so on) is emphasized. The results are used as a basis for proposing a genuine user interface of a software environment supporting design of fuzzy systems. Its characteristics satisfy the specified criteria and eliminate drawbacks of the existing applications.

Prohlášení

Prohlašuji, že jsem svou diplomovou (bakalářskou) práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW, atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb. , o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne ……. …….

Podpis

Obsah

1. ÚVOD 1

2. FUZZY SYSTÉM, CHARAKTERISTIKA A POSTUP NÁVRHU 2

2.1. Princip teorie fuzzy množin a fuzzy logiky 2
2.1.1. Definice fuzzy množin 2
2.1.2. Operace s fuzzy množinami 4

2.2. Odraz reality ve fuzzy systému 6
2.2.1. Fuzzifikace 7
2.2.2. Defuzzifikace 7

2.3. Obecný postup při návrhu fuzzy systému 8

3. KRITÉRIA PRO HODNOCENÍ A POROVNÁVÁNÍ PROGRAMŮ 10

3.1. Kritéria obecného vývojového prostředí 10
3.1.1. Různé druhy ovládání 11
3.1.2. Používání standardů a zavedených metod ovládání 11
3.1.3. Přehlednost ovládání 11
3.1.4. Jednotný systém ovládání 11
3.1.5. Nápověda a pomoc v nouzi 12
3.1.6. Operace Zpět (Undo) a Opakovat (Redo) 12
3.1.7. Nastavení prostředí na míru 13
3.1.8. Umělá „inteligence“ programu a nástroje usnadnění 13
3.1.9. Bezchybnost algoritmu programu 13
3.1.10. Úplnost programu 13
3.1.11. Možnosti ladění a kontroly výsledného produktu 14
3.1.12. Čitelnost a přehlednost výstupních dat 14
3.1.13. Použitelný formát vstupních a výstupních dat 14
3.1.14. Komunikace aplikace s jinými programy 15
3.1.15. Komunikace programu se speciálními periferiemi 15
3.1.16. Zdravotní kritéria 15
3.1.17. Estetika programu 15
3.1.18. Ostatní kritéria 16

3.2. Kritéria vývojového prostředí pro fuzzy systémy 16
3.2.1. Struktura fuzzy systému 16
3.2.2. Vstupy a výstupy 17
3.2.3. Jazykové proměnné 17
3.2.4. Pravidla 18
3.2.5. Výpočet 18
3.2.6. Ladění fuzzy systému 19
3.2.7. Rozšířené funkce programu 19

3.3. Kritéria – shrnutí 19

4. SROVNÁNÍ EXISTUJÍCÍCH PROGRAMŮ 20

4.1. Fuzzy Logic Toolbox 21
4.1.1. Obecný pohled 22
4.1.2. Tvorba fuzzy systému 23
4.1.3. Ladění a výstupní data fuzzy systému 25
4.1.4. Celkové hodnocení 26

4.2. Fuzzy knihovna v ECANSE 26
4.2.1. Obecný pohled 27
4.2.2. Tvorba a ladění fuzzy systému 27
4.2.3. Celkové hodnocení 28

4.3. Fuzzy Control Manager 29
4.3.1. Obecný pohled 30
4.3.2. Tvorba fuzzy systému 30
4.3.3. Ladění a výstupní data fuzzy systému 32
4.3.4. Celkové hodnocení 33

4.4. FuzzyControl ++ 34
4.4.1. Obecný pohled 35
4.4.2. Tvorba fuzzy systému 35
4.4.3. Ladění a výstupní data fuzzy systému 36
4.4.4. Celkové hodnocení 37

4.5. FuzzyTECH 37
4.5.1. Obecný pohled 38
4.5.2. Tvorba fuzzy systému 39
4.5.3. Ladění a výstupní data fuzzy systému 42
4.5.4. Celkové hodnocení 43

4.6. LFLC 2000 44
4.6.1. Obecný pohled 45
4.6.2. Tvorba a ladění fuzzy systému 46
4.6.3. Celkové hodnocení 47

4.7. Srovnávací studie vybraných produktů 47

5. OPTIMÁLNÍ ROZHRANÍ VÝVOJOVÉHO PROSTŘEDÍ PRO NÁVRH FUZZY SYSTÉMŮ 48

5.1. Struktura programu 48

5.2. Ovládání 48

5.3. Tvorba fuzzy systému 49

5.4. Ladění a výstupní data fuzzy systému 50

6. IMPLEMENTACE UŽIVATELSKÉHO ROZHRANÍ VÝVOJOVÉHO PROSTŘEDÍ 52

6.1. Aplikace z pohledu uživatele 52
6.1.1. Grafický model – popis ovládání 52
6.1.2. Grafický model – XML formát 55
6.1.3. 2D a 3D graf 60
6.1.4. Časový graf 62

6.2. Uživatelské rozhraní z pohledu programátora 65
6.2.1. Grafický model 65
6.2.2. 2D a 3D graf 68
6.2.3. Časový graf 70

7. ZÁVĚR 72

1. Úvod

Cílem této diplomové práce je porovnat existující a vytvořit vlastní grafické rozhraní softwarového nástroje pro manuální návrh fuzzy systémů. Návrh grafického rozhraní vyžaduje znalost funkce programu samotného a proto bude nejprve nutné seznámit se s teoretickými základy fuzzy množin a fuzzy logiky. Následně je třeba utvořit si představu, jaký je obecný postup při návrhu běžného fuzzy systému. Po těchto dvou krocích budeme sice mít dostatečné znalosti k testování existujících programů po stránce funkční, nikoliv však uživatelské, což je hlavním úkolem této práce. K tomu abychom mohli porovnávat ovládání, vzhled a všechny ostatní aspekty práce návrháře s aplikací je nutné určit srovnávací kritéria. Této problematice je věnována celá jedna kapitola. Po ní následuje vlastní srovnání testovaných produktů podle definovaných hledisek. Výsledky jsou doplněny obrázky, které vytvoří lepší představu o analyzovaném programu. Posledním krokem je návrh a realizace vlastního grafického rozhraní vývojového prostředí pro tvorbu fuzzy systémů. Použijí se zkušenosti získané nejen při úvodním teoretickém rozboru problematiky, ale také poznatky získané při testování jednotlivých, již existujících aplikací.

2. Fuzzy systém, charakteristika a postup návrhu

Za fuzzy systém můžeme považovat každý systém, který s využitím teorie fuzzy množin generuje ze vstupních dat data výstupní. Pro pochopení principu návrhu takového fuzzy systému je nutné porozumět alespoň částečně fuzzy množinám a fuzzy logice.

2.1. Princip teorie fuzzy množin a fuzzy logiky

Teorie fuzzy množin je poměrně mladá a bohužel ne vždy kladně přijímaná část matematiky. Na jejím vývoji a propagaci se podíleli například Łukasiewicz, Zadeh a jiní. Pro její vysvětlení je třeba začít u klasických množin. Základním kamenem klasické teorie množin je fakt, že každý přípustný prvek do dané množiny buď patří, nebo nepatří. Žádná jiná varianta nemůže nastat a obě varianty současně také nemohou nastat. Příslušnost prvku do množiny se tedy dá vyjádřit veličinou, která nabývá hodnot 1 nebo 0. Tato dvouhodnotová logika je sice zřejmá a snadno se s ní počítá, v mnoha případech ale není vhodná pro popis reálného světa. Můžeme se o tom přesvědčit na mnoha paradoxech, které jsou často známé již od starověku.

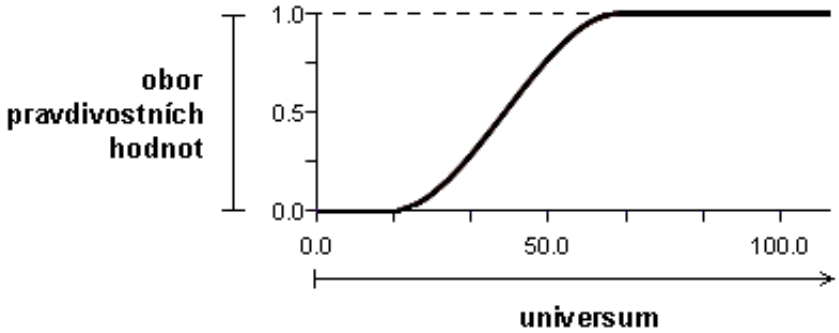
Jedním z nich je paradox hromady [1]. Je obecně přijímáno, že velké množství zrnek písku je nazýváno hromadou. Podle klasické teorie množin si tedy zavedeme prvek **Skupina zrnek písku** a množinu **Hromady,** do níž patří všechny Skupiny zrnek písku, o nichž tvrdíme, že jsou hromadou. Co se ale stane v případě, že z hromady odebereme jedno zrnko písku? Zřejmě ještě stále bude hromadou. Co když budeme tuto operaci ale neustále opakovat? Pak se zákonitě musíme dostat do situace, kdy nám zbude jedno jediné zrnko písku. To už ale samozřejmě není nazýváno hromadou. Během odebírání jednotlivých zrnek jsme se tedy někdy museli dostat do situace, kdy minulá skupina zrnek byla hromadou a nová skupina zrnek již hromadou není. My však nejsme schopni určit, v kterém kroku se tak stalo. Tato situace je způsobena tím, že se snažíme realitu popsat pouze pomocí dvou hodnot. Skupina zrnek písku **je** nebo **není** hromadou. Jiné možnosti nepřipouštíme. Podobných příkladů najdeme v běžném životě mnoho. Jedná se vlastně o všechny situace, kdy se snažíme objektu přisoudit nějakou vlastnost, u které není přesně dáno, jestli ji má nebo nemá. Rozhodování je navíc relativní. Někdo považuje 100 000 Kč za velkou částku, jiný ne. Navíc dost záleží na kontextu, v němž se rozhodujeme. Výše jmenovaná částka za nové auto nebude určitě považována za velkou, ale pokud ji někdo zaplatí za televizor, tak už to většině lidí bude připadat přemrštěné.

2.1.1. Definice fuzzy množin

Jedním z řešení problému věrného popisu reality je zavedení vícehodnotové logiky. O daném prvku nebudeme pouze tvrdit, že do množiny „patří“ či „nepatří“, ale povolíme i používání výrazů jako „spíše patří“, „spíše nepatří“ a podobných. Charakteristická funkce [1], která stanovuje příslušnost prvku do množiny, tedy bude například nabývat hodnot $\{0, 0.25, 0.5, 0.75, 1\}$. Pokud půjdeme ještě dále, můžeme jako obor pravdivostních hodnot [1] stanovit třeba celý interval $\langle 0,1 \rangle$. Tak se nám naskýtá možnost matematicky popsat vágní pojmy z běžného jazyka, jako jsou „skoro“, „téměř“, atd. Právě tímto se fuzzy množiny odlišují od klasických množin. U fuzzy množiny A může charakteristická funkce μ_A (nazývaná též funkce příslušnosti) nabývat libovolné hodnoty uvnitř oboru pravdivostních hodnot. Typický příklad takové funkce je :

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

Definiční obor hodnot, na němž je daná funkce příslušnosti definovaná, se nazývá universum [1]. Možná grafická reprezentace je ukázána na následujícím obrázku.

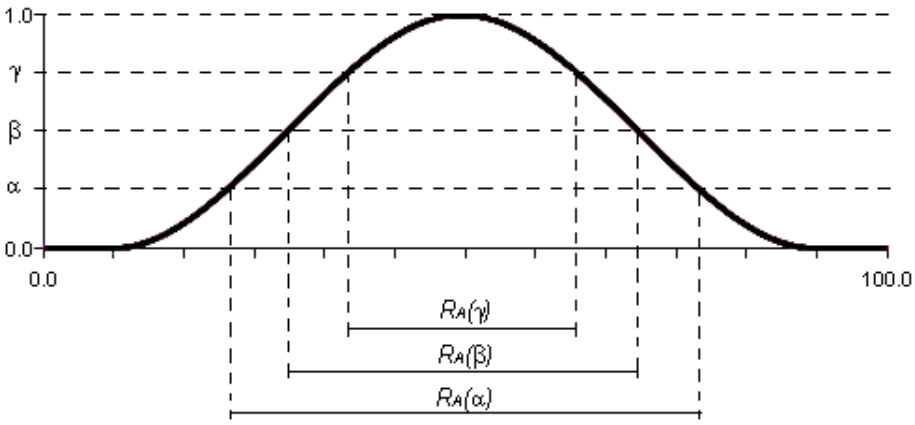


Obr. 2.1: Příklad funkce příslušnosti u fuzzy množiny

Dalším způsobem, jak se dá fuzzy množina popsat, jsou řezy a hladiny [1]. Zjednodušeně řečeno, α-řez fuzzy množiny A je interval či skupina intervalů, na níž je hodnota funkce příslušnosti větší nebo rovna hodnotě α. Matematicky zapsáno :

$$R_A(\alpha) = \{x \in X : \mu_A(x) \geq \alpha\}$$

Celá fuzzy množina se pak dá zapsat systémem řezů, kde α je z celého oboru pravdivostních hodnot. Grafická reprezentace tohoto popisu je ukázána na následujícím obrázku.



Obr. 2.2: Řezy fuzzy množiny na hladinách α, β, γ

Pro pochopení fuzzy množin je potřeba si uvědomit rozdíl mezi vágností informace používané ve fuzzy popisu světa a klasickou pravděpodobností. Rozdíl je především v původu neurčitosti. Při používání fuzzy přístupu je již samotný popis jevu neurčitý a zůstává neurčitý po celou dobu zkoumání. Kdežto pravděpodobnost hovoří o nejistotě před provedením pokusu (bližší zkoumání reality), ale po něm již žádná neurčitost není. Výsledkem je jednoznačná odpověď.

2.1.2. Operace s fuzzy množinami

Operace s fuzzy množinami jsou operace výrokového počtu s pravděpodobnostním ohodnocením z intervalu $\langle 0,1 \rangle$. Fuzzy logika je funkcionální [1], což znamená, že stupeň příslušnosti bodu $x \in X$ k danému výsledku operace závisí jen na jeho stupni příslušnosti k jednotlivým operandům. V následujících odstavcích jsou popsány některé základní operace potřebné k realizaci běžných fuzzy systémů.

Fuzzy negace

Fuzzy negace je definována jako unární operace předpisem

$$\neg : \langle 0,1 \rangle \to \langle 0,1 \rangle \quad \text{tak, že } \alpha \le \beta \Rightarrow \neg\beta \le \neg\alpha, \neg\neg\alpha = \alpha$$

Tento zápis je dosti „volný“. Umožňuje definovat mnoho funkcí, které mu vyhovují. To je značný rozdíl oproti klasické logice. Tam byla negace pouze jedna. Z výše uvedeného je také vidět, že dokonce nemusí platit $\neg\alpha = 1-\alpha$. Pro popis fuzzy negace se obecně používají generátory, což jsou parametrické funkce splňující daný předpis. Příkladem může být Yagerova třída negací popsaná vztahem

$$\neg\alpha = (1-\alpha^{\omega})^{\frac{1}{\omega}}, \quad \omega > 0,$$

nebo Gödelova zobecněná negace definovaná

$$\neg\alpha = \begin{cases} 1 & pro\ \alpha = 0 \\ 0 & jinak \end{cases}.$$

Fuzzy konjunkce

Fuzzy konjunkce je definována jako binární operace $\wedge : \langle 0,1 \rangle^2 \to \langle 0,1 \rangle$ splňující pro všechna $\alpha, \beta, \gamma \in \langle 0,1 \rangle$ axiomy:

$$\begin{aligned} &\alpha \wedge \beta = \beta \wedge \alpha \\ &\alpha \wedge (\beta \wedge \gamma) = (\alpha \wedge \beta) \wedge \gamma \\ &\beta \le \gamma \Rightarrow \alpha \wedge \beta \le \alpha \wedge \gamma \\ &\alpha \wedge 1 = \alpha \end{aligned}$$

Stejně jako u negace, i v tomto případě taková definice poskytuje dostatečnou volnost pro definování mnoha rozličných fuzzy konjunkcí. Mezi základní patří

Standardní konjunkce

$$\alpha \wedge \beta = \min(\alpha, \beta)$$

Łukasiewiczova konjunkce

$$\alpha \wedge \beta = \begin{cases} \alpha + \beta - 1 & pro\ \alpha + \beta - 1 > 0 \\ 0 & jinak \end{cases}$$

Součinová

$$\alpha \wedge \beta = \alpha \cdot \beta$$

Drastická disjunkce

$$\alpha \wedge \beta = \begin{cases} \alpha & pro\ \beta = 1 \\ \beta & pro\ \alpha = 1 \\ 0 & jinak \end{cases}$$

Yagerova třída konjunkcí

$$\alpha \wedge \beta = \max(1 - ((\alpha - 1)^{\omega} + (\beta - 1)^{\omega})^{\frac{1}{\omega}}, 0), \quad \omega > 0$$

Která z nich bude v aplikaci využita, je nutné rozhodnout na základě významu jednotlivých fuzzy množin a potažmo celých fuzzy tvrzení. Pokud například budeme zkoumat konjunkci „pokrm je chutný a zdravý“, určitě budeme chtít rozlišit situaci ½ ∧ ½ od situace 1 ∧ ½ . Pokud použijeme standardní konjunkci, bude u obou výsledek ½, při použití Łukasiewiczovy konjunkce bude v prvním případě výsledek nulový. To je zase příliš přísné. Je proto nutné dobře zvážit, v jaké situaci chceme danou konjunkci použít.

Fuzzy disjunkce

Fuzzy disjunkce je definována jako binární operace $\vee : \langle 0,1 \rangle^2 \to \langle 0,1 \rangle$ splňující pro všechna $\alpha, \beta, \gamma \in \langle 0,1 \rangle$ axiomy:

$$\begin{aligned} &\alpha \vee \beta = \beta \vee \alpha \\ &\alpha \vee (\beta \vee \gamma) = (\alpha \vee \beta) \vee \gamma \\ &\beta \leq \gamma \Rightarrow \alpha \vee \beta \leq \alpha \vee \gamma \\ &\alpha \vee 0 = \alpha \end{aligned}$$

Opět je známo mnoho funkcí, které odpovídají této definici. I v tomto případě záleží na konkrétní aplikaci, kterou disjunkci použijeme. Mezi základní z nich patří

Standardní disjunkce

$$\alpha \vee \beta = \max(\alpha, \beta)$$

Łukasiewiczova disjunkce

$$\alpha \vee \beta = \begin{cases} \alpha + \beta & pro\ \alpha + \beta < 1 \\ 1 & jinak \end{cases}$$

Součtová disjunkce

$$\alpha \vee \beta = \alpha + \beta - \alpha \cdot \beta$$

Drastická disjunkce

$$\alpha \vee \beta = \begin{cases} \alpha & pro\ \beta = 0 \\ \beta & pro\ \alpha = 0 \\ 1 & jinak \end{cases}$$

Fuzzy implikace

Pro tuto operaci není ustálená axiomatická definice, proto například v [1] považují za fuzzy implikaci jakoukoliv operaci $\rightarrow : \langle 0,1 \rangle^2 \rightarrow \langle 0,1 \rangle$, která se na $\{0,1\}^2$ shoduje s klasickou implikací. Z nejčastěji používaných fuzzy implikací jmenujme například:

$$\alpha \rightarrow \beta = \sup\{\gamma : \alpha \wedge \gamma \leq \beta\}$$
$$\alpha \rightarrow \beta = \neg\alpha \vee \beta$$
$$\alpha \rightarrow \beta = \neg\alpha \vee (\alpha \wedge \beta)$$

Tímto ukončím popis základů fuzzy množin a fuzzy logiky. Zájemce o podrobnější informace odkazuji na skripta [1] a [2], popřípadě knihy [3] a [4]. Pro tuto práci je podstatné zjištění, že u jednotlivých fuzzy operací existuje mnoho různých způsobů, kterými mohou být realizovány. Uživatel si při tvorbě fuzzy systému musí být této situace vědom a musí pro svůj příklad zvolit tu nejvhodnější.

2.2. Odraz reality ve fuzzy systému

Informace o struktuře světa a jeho zákonitostech je reprezentována výrazy typu „zrychlení je malé a kladné“. Veličina „zrychlení“, které se výrok týká, se nazývá jazyková proměnná. Ta může nabývat určitých hodnot „malé a kladné“. Tyto jazykové hodnoty odpovídají příslušným fuzzy množinám a jejich funkcím příslušnosti. Každému výrazu přísluší určitý význam. Příklad: rychlost je malá, pokud je menší než 40 km/h, střední, pokud je mezi 40 km/h a 100 km/h a velká, pokud je větší než 100 km/h. Zobrazení, které přiřazuje každému výrazu jeho význam, se nazývá sémantika jazyka (*M*) [2].

Jazykovou proměnnou pak můžeme definovat jako proměnnou, jejíž hodnoty jsou výrazy nějakého jazyka. Zapisuje se jako šestice *<L, T, X, G, M>*. Význam jednotlivých symbolů je následující: *L* je název jazykové proměnné, *T* je množina termů (jazykových hodnot - odpovídá fuzzy množinám), *X* je universum, *G* je syntaktické pravidlo (generativní gramatika - viz dále) a *M* je sémantické pravidlo (míra souhlasu termu s jeho významem). Termy lze spojovat do složených termů: „rychlost je velká a vzdálenost malá“ nebo také upravovat jejich význam pomocí jazykových operátorů (hedges): „rychlost je přibližně velká“. Generativní gramatika generuje termy (tvar a pozice funkcí příslušnosti). Pokud takto lze vygenerovat všechny termy obsažené v jazykové proměnné, nazýváme ji strukturovanou.

Další důležitou otázkou je význam složených výrazů. Ten zjevně záleží na významu jednotlivých elementárních termů z nichž se skládá. Samotné skládání je realizováno pomocí logických spojek *a*, *nebo* a atd. Proto lze použít operace s fuzzy množinami, které jsou definovány v předchozí kapitole. Celý postup realizace obrazu reality pomocí fuzzy modelu vede na strukturu systému, která je popsána pomocí následujícího obrázku.

Obr. 2.3: Základní struktura fuzzy systému

IN : Vstupní bod systému. Slouží k získání informaci o okolním světě (nejčastěji v podobě naměřených veličin)

Fuzzifikace : Převod naměřených dat na fuzzy hodnoty (míry příslušnosti do vstupních fuzzy množin)

TP : Tabulka pravidel. Je to množina výroků většinou ve tvaru „JESTLIŽE … PAK …“, která postihuje vliv vstupů na výstupy. Jinými slovy tabulka pravidel reprezentuje sadu fuzzy operací, při nichž se ze vstupních fuzzy hodnot generují výstupní fuzzy hodonty.

Defuzzifikace : Převod výstupních fuzzy hodnot (míra příslušnosti do výstupních fuzzy množin) na reálná data.

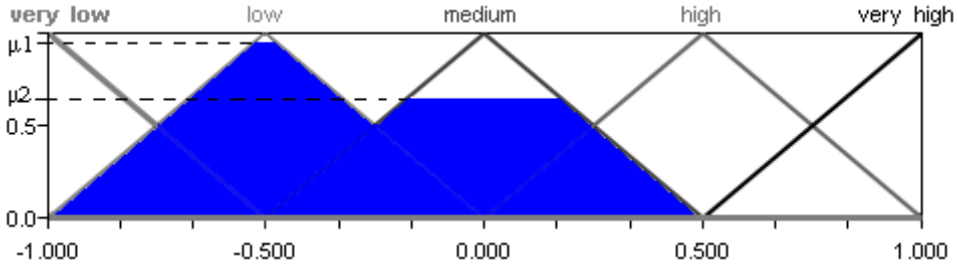
OUT : Výstupní bod systému.

2.2.1. Fuzzifikace

Fuzzifikace je operace, kdy je ostré hodnotě z univerza přiřazen stupeň příslušnosti do dané fuzzy množiny. Pro každou hodnotu z univerza tedy musí být možné určit stupeň příslušnosti do libovolné fuzzy množiny a zároveň by mělo platit, že žádný prvek by neměl mít nulový stupeň příslušnosti pro všechny fuzzy množiny [2]. Prakticky to znamená určit počet a tvar funkcí příslušnosti reprezentujících navržené fuzzy množiny. To se provádí většinou na základě zkušeností experta v oblasti zájmu fuzzy systému. Druhý způsob je získávání z dat. Nejpoužívanější tvary funkcí příslušnosti pro vstupní jazykové proměnné jsou trapezoid a S-funkce (Obr. 2.5 – a,b).

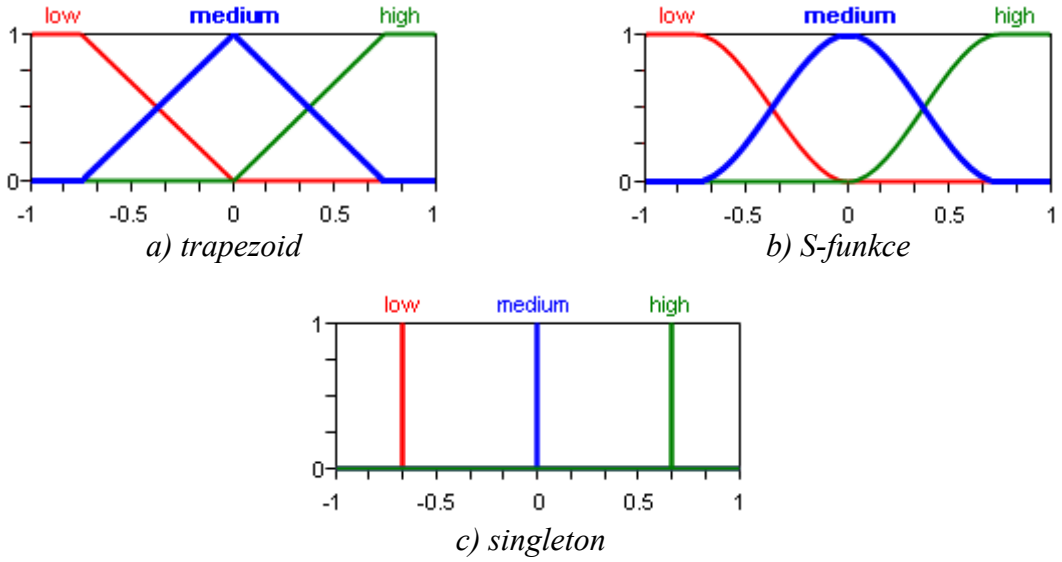
2.2.2. Defuzzifikace

Defuzzifikace je operace, při niž je fuzzy množině přiřazena ostrá hodnota, která ji reprezentuje. Výpočet tohoto čísla je možné realizovat mnoha způsoby. Všechny vycházejí z reprezentace míry splnění závěrů jednotlivých pravidel, která je ukázána na Obr. 2.4. Zde jsou vidět stupně příslušnosti (μ_1, μ_2) do jednotlivých výstupních fuzzy množin získaných pomocí Mamdaniho inference [2].



Obr. 2.4: Výstupní fuzzy množina při použití Mamdaniho inference.

Mezi nejpoužívanější defuzzifikační metody patří například Centrování (bere v úvahu i minoritní funkce příslušnosti), Střední hodnota součtů, První maximum, Střed maxim atd. Jejich podrobnější popis je nad rámec této práce a lze ho nalézt v [2]. Tvary funkcí příslušnosti u výstupních jazykových proměnných jsou různé. Některé často používané jsou ukázány na následujícím obrázku.



Obr. 2.5: Příklady tvarů funkcí příslušnosti.

2.3. Obecný postup při návrhu fuzzy systému

Manuální návrh fuzzy systému má několik fází. Předně je nutné definovat vstupy, výstupy a přibližnou vnitřní strukturu modelu (počet tabulek pravidel, jejich propojení …) a případně provést v aplikaci nebo někde mimo ni předzpracování použitých signálů (filtrace, integrace, absolutní hodnota, …).

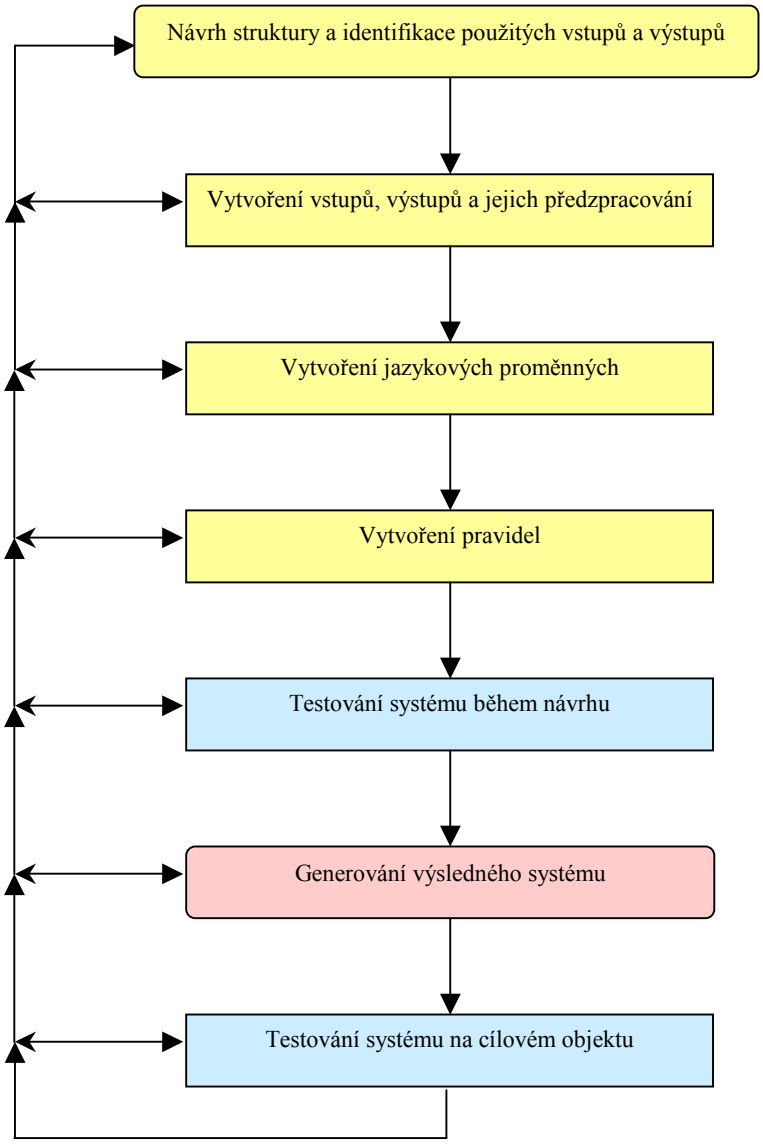
U každého ostrého vstupu je třeba převést ho z ostrých hodnot na fuzzy hodnoty (fuzzifikace). To znamená, každé fyzické veličině přiřadit jazykovou proměnnou, u níž je třeba určit jméno (např. „rychlost“), universum (např. 0 až 200) a pro větší přehlednost i jednotku (např. km/h). Dalším krokem je volba třídy tvarů funkcí příslušnosti používaných fuzzy množin. Jejich vlastnosti závisí na požadované ale také realizovatelné možnosti měření veličiny a na struktuře budoucích pravidel. Fuzzy množiny se nejlépe definují pomocí svých funkcí příslušnosti, proto je vhodné začít u nich. Je třeba určit jejich počet, tvar, pojmenování (př. „malá“, „střední“, „velká“) a rozložení. Tato fáze návrhu je dosti intuitivní a do značné míry subjektivní. Neexistuje obecný postup, jak ji realizovat. Každý inženýr musí sám rozhodnout, co bude považovat například za „malou rychlost“ a co již za „střední rychlost“. Záleží na zkušenostech, míře neurčitosti, konkrétním případě, atd. U vstupních proměnných bývá zvykem používat funkce příslušnosti ve tvaru trapezoidu, S-funkce, popřípadě nějakém podobném. U výstupních proměnných je postup opačný. Z fuzzy hodnot je třeba vytvořit ostré hodnoty (defuzzifikace). Postup je obdobný jako u vstupních proměnných. Také se musí nadefinovat fuzzy množiny a jejich funkce příslušnosti. Někdy je vhodný kromě výše jmenovaných druhů také singleton, který ve většině případů urychluje výpočet defuzzifikace. Již v tuto chvíli by návrhář měl mít alespoň přibližnou představu o v budoucnu použité metodě defuzzifikace

V následujícím kroku se navrhnou pravidla. Nyní je již třeba přesněji stanovit budoucí strukturu systému a určit, které výstupy závisí na kterých vstupech. Jinak řečeno kolik a v jakém řazení bude použito pravidlových bloků (souborů pravidel). Pro generování vlastních pravidel opět není žádný obecný postup. Zde přichází ke slovu znalosti experta z oblasti, které se fuzzy systém týká. Ten musí na základě svých vědomostí a zkušeností určit, které vstupy ovlivňují jaké výstupy a v jaké míře se projeví na výsledném signálu. Tato práce často bývá zdlouhavá a pro dostatečně přesné určení pravidel bývá třeba mnoho iteračních kroků.

Po těchto fázích, během nichž vznikne základní fuzzy systém, nastává fáze ladění a testování systému. Při ní je nutné otestovat celý model, zda pracuje podle našich představ a

zda neobsahuje nějaké chyby. Během těchto kroků se často zjistí, že je nutné některou, již vytvořenou, část fuzzy systému pozměnit nebo úplně přepracovat. Proto je nutné mít neustále možnost snadno upravovat vlastní strukturu modelu, jazykové proměnné i samotná pravidla. Celý návrh se tak mění na posloupnost iteračních kroků, při nichž se neustále opakují operace jako definice (úprava) fuzzy množiny, vytvoření (pozměnění) pravidla a opětné testování. Během tohoto postupu je nutné již přesně určit veškeré parametry potřebné pro správnou činnost systému. Kromě jiného se jedná o použitý způsob inference, defuzzifikace, atd.

V posledním kroku je již systém hotový, je ale třeba převést ho do stavu potřebného pro jeho aplikaci v průmyslu. Je proto nutné vygenerovat například zdrojový kód v jazyce C či jiném, který se chová přesně stejně jako navržený systém. Další možnosti jsou programy pro PLC, mikroprocesory a podobně. Tím ale návrh nekončí. Vytvořenou průmyslovou aplikaci je nutné opět testovat, kontrolovat její správnou činnost a v případě nutnosti opět upravit navržený systém. Základní kroky návrhu jsou uvedeny v následujícím schématu. Zpětné kroky jsou možné do jakékoliv fáze vývoje.



Obr. 2.6: Schéma postupu práce při návrhu fuzzy systému

3. Kritéria pro hodnocení a porovnávání programů

Před vlastním návrhem optimálního vývojového prostředí pro vytváření fuzzy systémů si musíme položit otázku, co považujeme za **optimální** prostředí z ergonomického hlediska. Podle definice z [13] je ergonomie věda o zákonitostech lidské práce. Vychází ze zkoumání vztahu člověk-stroj-prostředí. Zjišťuje optimální antropometrické, fyziologické a psychologické parametry pro maximální pracovní výkon i pro relaxaci. Ve spolupráci s pracovním lékařstvím a jinými obory projektuje pracovní i životní prostředí, navrhuje pracovní nástroje a další. Z toho tedy vyplývá, že ergonomické vývojové prostředí je takové, které umožní každému uživateli, nebo alespoň velké většině uživatelů, s minimálním úsilím a minimálními negativními následky vyprodukovat co nejkvalitnější výsledný projekt (v našem případě fuzzy systém). Tato definice sice zní přesvědčivě, při hlubším zamyšlení ale dává mnohem více otázek, než na které odpovídá. Předem je zjevné, že aby bylo prostředí považováno za ergonomické, musí co nejlépe odpovídat mnoha kriteriím. To ale znamená, že se mohou vyskytovat prostředí, která jednomu kritériu dokonale vyhovují, ale druhému již méně. U dalšího prostředí to může být třeba přesně naopak. Není tedy možné nějaký program nazývat optimálním nebo ergonomickým, bez toho abychom dodali, jaká kritéria jsme použili a na které parametry jsme kladli důraz.

Ilustrujícím příkladem může být hypotetická aplikace, která pro začínající uživatele obsahuje sadu funkcí a podpůrných metod, které ho za běhu informují o ovládání jednotlivých částí programu, předdefinovávají za něj potřebné proměnné a různými jinými způsoby mu ulehčují první kroky s aplikací. To vše je určitě užitečná věc, která nováčkovi umožní velmi rychle proniknout do ovládání aplikace a tudíž ve svém důsledku zvýší efektivitu jeho práce. Po nějaké době se ale ze začínajícího uživatele stane zkušený, který již nepotřebuje automatické nápovědy. Nepotřebuje ani automaticky vyplňovat všechny proměnné. Pokud v tuto chvíli nelze vypnout podpůrné funkce pro začátečníka, začnou být na obtíž a práci brzdí. Z toho je vidět, že pokud se na tento program budeme dívat jako nový uživatel, bude nám připadat velmi dobrý a budeme se spíše přiklánět k tomu, abychom ho zařadili do skupiny optimálních vývojových prostředků. Pokud se ale na něj budeme dívat z pohledu zkušeného uživatele, budeme tudíž mít jiná kritéria kvality než začátečník, zařadíme ho spíše mezi ty špatné produkty.

Předcházející příklad ukázal, jak důležitá je volba vhodných kritérií a jejich zveřejnění při celkovém hodnocení daného produktu. Následující kapitoly budou tedy věnovány především stanovení a vysvětlení jednotlivých kritérií, která budou později použita k hodnocení jednotlivých vývojových prostředí pro návrh fuzzy systémů. Některá se možná budou zdát samozřejmá, a tudíž téměř nadbytečná, v praxi se ale můžeme setkat s mnoha programy, které mají značné nedostatky právě v těchto oblastech.

Jednotlivá kritéria se dají rozdělit do dvou velkých skupin. Jednak to jsou obecná kritéria, která se používají pro porovnávání všech softwarových produktů. A dále to jsou kritéria týkající se přímo programů, které se zabývají vývojem fuzzy systémů. Pro obě tyto skupiny musí platit, že při jejich maximálním splnění je aplikace v jistém smyslu optimální. Pro některé skupiny z dále jmenovaných požadovaných vlastností programů existují i oficiální normy. Já jsem ale bohužel žádnou takovou neměl k dispozici.

3.1. Kritéria obecného vývojového prostředí

Kritéria používaná pro hodnocení a porovnávání všech programových vývojových prostředí jsou velmi různá. Obecně pro všechny z nich platí, že při jejich dodržení by měl program být snadno a přehledně ovladatelný. Právě kvůli různorodosti je obtížné je nějakým způsobem seřadit. Jak bude vysvětleno později, ani jejich důležitost a priorita při testování

není přesně stanovena. Z tohoto důvodu jsou řazena přibližně tak, jak se jejich dodržování, či spíše nedodržování, projeví při práci s aplikací.

3.1.1. Různé druhy ovládání

U většiny programů je požadováno více druhů ovládání. Jedná se především o ovládání pomocí hlavního menu, toolbarů, kontextových menu, klávesových zkratek a ovládacích oken (okna, kde jsou nastavovány parametry …). Tato potřeba je vyvolána odlišností jednotlivých uživatelů, kteří preferují různé způsoby obsluhy programu. Dalším důvodem této rozmanitosti je vývoj samotného uživatele. Určitě není moudré začátečníka nutit používat klávesové zkratky, ale pokročilý uživatel se bez nich často již téměř neobejde.

3.1.2. Používání standardů a zavedených metod ovládání

Další důležitou oblastí, především pro nového uživatele, je otázka, zda ovládání odpovídá standardům a běžně zavedeným způsobům. U hlavního menu je to například běžné pojmenování a řazení jednotlivých podmenu a samotných položek. Zvykem je začínat položkou „Soubor“…„Úpravy“…„Nápověda“. V prvním podmenu jsou shora položky „Nový“…„Otevřít“…„Uložit“…„Tisk“, atd. U toolbaru se jedná o řazení a vzhled základních a často používaných ikon. Standardní je začínat ikonami v pořadí obdobném, jaké jsem vyjmenoval u prvního podmenu. U kontextového menu jde například o jeho samotné otevíraní. Standardní je vyvolávat ho pomocí pravého tlačítka myši. Snad nejhorší stav je u klávesových zkratek. Zde je dodržování obecných pravidel méně rozšířené. Navíc se objevují i programy, které mají různé klávesové zkratky v různých jazykových verzích. Ale nikoho snad nepřekvapí, pokud se pomocí „Ctrl+O“ otevře nový projekt, „Ctrl+S“ uloží stávající a pomocí „F1“ se vyvolá nápověda. U ovládacích a výběrových oken programu je také zvykem nastavovat některé prvky podle standardů. Je tomu tak i přesto, že jednotlivá okna jsou často velmi odlišná. Jedná se například o tlačítka „OK“ a „Cancel“ spolu s jejich grafickými symboly, atd.

Dodržování výše jmenovaných standardů je velmi důležité pro přehlednost programu. Nový uživatel se s aplikací naučí rychleji zacházet (ovládání je intuitivní), a tudíž i dříve bude vytvářet kvalitnější produkty. Velký význam má standardní ovládání i pro zkušené uživatele. Často se stává, že se používá více softwarových prostředků současně a během práce se mezi nimi mnohokrát přechází. Pokud by každý z nich měl úplně odlišné ovládání, činilo by to problémy i zkušeným uživatelům. Například pokud použiji obecně známou ikonu se zcela jiným významem, zavedenou klávesovou zkratkou vyvolám úplně jinou operaci, než operátor ze svých zkušeností s ostatními programy očekává a tak podobně.

3.1.3. Přehlednost ovládání

Pokud program splňuje předcházející kritérium, nemusí ještě nutně obsahovat přehledné ovládání. Velmi totiž záleží na logickém uspořádání ovládání a potažmo také na chování celé aplikace. Ovládací prvky, které se týkají stejné vlastnosti programu, by měly být u sebe a měly by být viditelně odděleny od ostatních. Pokud je to možné, musí být ovládací prvky co nejblíže k objektům, kterých se týkají. Jestliže toto nebude splněno, může se snadno stát, že ne zcela zasvěceného uživatele zaskočí, když stiskem ikony vyvolá akci v úplně jiném okně, než kde se samotná ikona nachází.

3.1.4. Jednotný systém ovládání

Předchozí odstavce nastínily problém přehlednosti ovládání, jeho uspořádání a intuitivnosti. Tyto bezpodmínečně nutné vlastnosti ale nezískáme, pokud budeme pouze splňovat předcházející kritéria, i když to nám k tomu velmi pomůže. Důležité je také udržovat v ovládání a posléze i v chování celého programu nějaký řád, vytvořit si jakási vlastní interní

pravidla. Pokud tedy program vyhovuje kritériu o standardních způsobech ovládání, zbývá ještě dát řád funkcím, které jsou typické pouze pro naši aplikaci. Pokud se třeba na více místech téhož programu objeví stejná nebo velmi podobná ikona, měla by znamenat to samé. Stejné je to s názvy položek hlavního a kontextových menu. Častou chybou větších aplikací bývá odlišné chování některých jejich autonomních částí. Je to zřejmě způsobeno tím, že každou část vytvářel někdo jiný, to ale programátory nikterak neomlouvá.

3.1.5. Nápověda a pomoc v nouzi

Velmi důležitou částí programu, možná někdy i tou nejdůležitější, je nápověda. Ta musí sloužit k zodpovězení všech běžných dotazů, které mohou uživatele zajímat. Jako taková by měla být rozdělena do několika částí.

Jednak je to vlastní často obsáhlý text, který by měl popisovat k čemu samotný program slouží, jaké jsou podmínky jeho použití a jaké výstupy můžeme očekávat. Dále pak samozřejmě výrobce a místo, kam se můžeme obracet v případě problémů s aplikací. Neméně důležitou informací je i verze samotného programu a kompatibilita mezi starými a aktuální. Dále by se zde měl nacházet popis vlastního ovládání programu. Tato část by měla být koncipována jako učebnice, která se dá číst sekvenčně a nového uživatele naučí krok za krokem s aplikací pracovat. Pokud je uživatel již zkušenější, zcela jistě využije nápovědu, která vyřeší konkrétní problémy. Pro tyto účely by měl každý program obsahovat seznam tématických okruhů, ze kterých si uživatel může vybrat a najít tak odpověď na svou otázku, aniž by postupně procházel celou nápovědu. K obdobnému účelu by měl program obsahovat i vyhledávací aparát, který podle zadaného dotazu (často řetězec znaků) najde stránky, které by uživatele mohly zajímat.

Velmi důležitou částí je odpovídat na otázky týkající se nejen ovládání a postupu při práci, ale také způsobu, jak program pracuje uvnitř. Uživateli musí být umožněno, aby si ověřil, zda program pracuje tak, jak předpokládá, a tedy zda správně interpretuje dosažené výsledky. V praxi je to například upozornění, jaké výpočetní metody a algoritmy se v programu používají a jaké z toho plynou pro uživatele důsledky.

Další vhodnou funkcí je volání nápovědy přímo z místa vzniku dotazu. Mám tím na mysli například možnost volání odpovídající nápovědy přímo z ovládacího (nastavovacího) okna. Nebo otevření nápovědy pomocí poklepání myši na dotazovaný objekt. Další možnost usnadnění práce je režim pro začínající uživatele. Tato funkce vede uživatele během prvních používání daného programu a umožní mu tak velmi rychle pochopit a naučit se ovládání. Samozřejmostí nápověd jsou dynamické odkazy na související témata a nápovědy na tlačítcích (tzv. tooltipy).

Obecně lze říci, že program by měl být koncipován tak, aby byla nápověda co nejméně potřeba a pokud již vznikne nějaká nejasnost či dotaz, měl by se uživatel dozvědět požadovanou informaci co nejrychleji.

3.1.6. Operace Zpět (Undo) a Opakovat (Redo)

Pokud má být práce s jakýmkoliv programem pohodlná, musí bezpodmínečně obsahovat funkce „Zpět“ a „Opakovat“. Tyto dvě funkce mají umožňovat volně se pohybovat dozadu (Zpět) a dopředu (Opakovat) v posloupnosti operací, které byly uživatelem provedeny. Nestačí pouhá přítomnost těchto dvou metod. Musí být také zajištěno, že bude možné vrátit každý provedený krok. Pokud tomu tak ve výjimečném případě není, měl by být uživatel informován o tom, že jeho operaci nebude možné opravit. Toto je nutné dodržovat, protože při vynechávání či nevracení některých kroků nastává zmatek nejen v tom, co lze vrátit, ale hlavně v tom, co již bylo vráceno. Další podmínkou bezpečné práce je tedy, aby bylo vždy dobře vidět, který krok se vrátil, případně kterého objektu se týkal. V ideálním případě jsou jednotlivé kroky přesně pojmenovány a uživatel se mezi nimi může libovolně

pohybovat pouze pomocí výběru jedné položky ze seznamu. Poslední podmínkou je, aby takový seznam byl dostatečně dlouhý. Musí být možné se vrátit nejen o jeden či dva kroky, ale i o mnohem víc.

3.1.7. Nastavení prostředí na míru

Velmi vhodným doplňkem klasických ovládacích prvků je takzvané nastavení ovládání na míru. Jde především o možnost skrýt, přemístit či přerovnat jednotlivé ikony v toolbarech, nebo obdobně manipulovat s celými toolbary. Další častou možností je nastavit si vlastní klávesové zkratky. Stejně jako ovládání, tak i chování vlastního programu může být vhodné nastavovat podle potřeb uživatele. V úvodu o optimálním vývojovém prostředí jsem zmínil příklad s pomocnými funkcemi, sloužícími k vedení nového uživatele krok za krokem při prvních krocích s aplikaci. Tuto funkci je ale nutno umožnit zapnout a vypnout podle potřeb daného operátora. Podobné to může být i s mnoha jinými metodami. Samostatnou kapitolou je pak nastavování parametrů, kterými jsou například barva, fonty, atd. Důležitou funkcí je však v takových případech možnost defaultního nastavení. Tato funkce může být využita nejen pokud se uživatel dostane při nastavování parametrů do takové situace, ze které se neumí vrátit, ale může být využita také náhodným (zastupujícím) pracovníkem, který se v prostředí vytvořeném předchozím uživatelem nemůže zorientovat.

3.1.8. Umělá „inteligence“ programu a nástroje usnadnění

Další kritérium, pomocí kterého se dají hodnotit jednotlivé programy, jsou nástroje usnadnění. Jedná se o takové funkce, které uživateli „usnadní“ práci. Je to například automatické nastavování parametrů. Provádění automatických kontrol a následných oprav, automatické doplňování rozepsaného textu, atd. Často platí, že pro uživatele je mnohem příjemnější, když daný parametr může nastavit tím, že vybere jednu možnost z nabízených, než aby si musel pamatovat jednotlivá jména a pracně je ručně vypisovat. Také možnost automatizace práce v podobě nastavitelných maker je často vítanou funkcí programu. Zajímavým příkladem pokusu o usnadnění je přizpůsobování se samotného programu uživateli. Typickým příkladem je Microsoft Word verze 9. Tento program nejenže hledá a opravuje v uživatelem napsaném textu chyby, ale tvůrci tohoto prostředí šli ještě dále. Do vlastního programu vkomponovali jistý druh umělé „inteligence“, pomocí něhož se uživatelské prostředí snaží měnit podle chování operátora tak, aby vyhovovalo jeho potřebám. Jedná se vlastně o jakési automatické nastavování parametrů programu. To se projevuje například v hlavním menu, kdy jsou přímo viditelné pouze často používané položky, atd. Toto je jistě zajímavý způsob usnadnění, musí být ale kvalitně proveden, protože jinak uživateli spíše práci přidělá než ulehčí. U všech výše jmenovaných způsobů ulehčení práce platí, že je třeba umožnit uživateli, aby je v případě, kdy mu nevyhovují, mohl deaktivovat.

3.1.9. Bezchybnost algoritmu programu

Toto kritérium by zřejmě mělo být jako první, neboť pokud pracuje chybně samotná aplikace, pak je veškerá práce s ní zbytečná. Správnost algoritmů a celého programu se ale jaksi předpokládá, přestože pro tento závěr není často logické opodstatnění. Uživatel by z toho důvodu měl být informován, jak daná aplikace uvnitř pracuje, jaké využívá metody výpočtu a podobně.

3.1.10. Úplnost programu

Program by měl být úplný. To znamená, že by měl pokud možno plně pokrývat celou problematiku v oblasti, jíž se zabývá. Je velmi nepříjemné, když uživatel uprostřed práce zjistí, že některou operaci, kterou chtěl v budoucnu provést, tato aplikace nepodporuje. V takových případech je nucen snažit se využít i jiných nástrojů, které mu program poskytuje,

ale které jsou původně určeny k úplně jinému účelu. Případně musí pro tento jediný případ použít jiný program. Tak či onak je taková práce dosti zdlouhavá a často i fyzicky a psychicky náročná. V praxi lze nalézt mnoho odstrašujících případů, kdy třeba grafický editor umožňuje pracovat pouze s velmi omezeným seznamem formátů, který zcela určitě nezahrnuje všechny běžně používané. Nebo textový editor, který neumí napsaný text vytisknout, atd. Pokud se autorům aplikace nepodaří vyhnout takovým situacím, je nutné, aby umožnili nepodporovanou operaci provést v jiném programu. Což samozřejmě znamená, že musejí umožnit export jejich projektu do jiné aplikace. Tomuto tématu se věnuje jedno z následujících kritérií.

3.1.11. Možnosti ladění a kontroly výsledného produktu

Při práci na projektu je vhodné mít před jeho dokončením a prohlášením za hotový možnost ověřit, zda náš výtvor neobsahuje nějaké chyby nebo jestli dokonce není celý špatně. K takovému účelu by měly sloužit nástroje pro ladění a kontrolu výsledného produktu. Pokud je to technicky možné, měly by uživateli umožnit testovat jeho výtvor například pomocí zkušebních dat a přehledně zobrazovat nejen výstupní data, ale také operace a mezivýsledky, které se v projektu vyskytují. To znamená, že by mělo být možno testovat nejen celý produkt jako celek, ale i jednotlivé jeho části. Další velmi vhodnou funkcí jsou automatické opravy, které uživateli usnadní prácí s návrhem testů a sami vyzkouší a zkontrolují části aplikace, kde se například často vyskytují chyby nebo které jsou svým chováním podezřelé. Je vhodné, aby tyto automatické testy probíhaly nejen na konci práce na projektu, ale aby bylo možné je aktivovat i během samotného vývoje projektu. To pak může ušetřit spoustu času s předěláváním projektu, ve kterém došlo k chybě již na počátku.

3.1.12. Čitelnost a přehlednost výstupních dat

Toto kritérium je zcela samozřejmé. Nezáleží totiž pouze na správnosti získaného výsledku, ale také na jeho čitelnosti. Pokud nejsou výstupní data dostatečně přehledná, může se snadno stát, že se uživatel při jejich interpretaci dopustí chyby. Proto pokud je například výstupem programu nějaká posloupnost čísel, měl by uživatel mít možnost zobrazit si je graficky, atd.

3.1.13. Použitelný formát vstupních a výstupních dat

Dalším velmi podstatným hlediskem, podle něhož se pozná kvalitní program, je formát jeho vstupních a výstupních dat. Není totiž jen důležité, jak se samotný produkt vytváří, ale také jaká data může načítat a jak se dá nakonec upotřebit. V některých programech se dokonce setkáváme se situací, kdy nám práce jde díky vhodnému ovládání rychle od ruky, náš projekt si můžeme v aplikaci úspěšně odsimulovat, pokud se ale pokusíme výsledný produkt v praxi použít, zjistíme, že to kvůli nevhodnému formátu nelze, nebo že jsou s tím značné problémy. V takovém případě je zřejmé, že veškerý komfort při návrhu je nám k ničemu, protože musíme začít znovu a s jiným programem. Z tohoto příkladu je vidět, že výstup musí být vhodně zvolen podle předpokládaného využití programu.

V posledních několika letech sílí ve výpočetní technice trend ukládat a exportovat data v takovém formátu, aby byla použitelná i v jiných aplikacích. Je to pochopitelné. Při návrhu či vytváření složitějších projektů se velmi často používá mnohem více než pouze jeden program. Často ani není práce na dané úloze v jedné aplikaci ukončena a již je třeba použít její mezivýsledky v jiné. To tedy vede k požadavku na možnost ukládání, či exportu dat nejen na konci práce na projektu, ale i během ní. V praxi by to mělo vypadat tak, že program bude podporovat všechny „standardní“ formáty týkající se jeho oblasti činnosti, které jsou v současné době k dispozici. A naopak by měl poskytovat možnost jiným programům, aby mohly svá data ukládat tak, aby je mohl přesít.

Pro tento účel byl vytvořen například jazyk XML [7]. Ten se v dnešní době používá především jako formát, v němž jsou ukládána veškerá data, která mají být čitelná i pro ostatní programy. Jeho výhodou je, že je dostatečně obecný a pružný na to, aby vyhovoval všem aplikacím. Pokud je to tedy možné, pak by každý program měl podporovat XML nebo nějaký podobný formát.

3.1.14. Komunikace aplikace s jinými programy

Bezprostředně na předcházející kapitolu navazuje požadavek, aby byl program schopen během své činnosti komunikovat s okolními programy. A to ne pouze pomocí dat, která jsou ukládána do souboru a která si později nějaký jiný program přečte, ale také přímo za běhu poskytovat vlastní a číst cizí data. Takové uspořádání umožňuje například uživateli používat více programů současně a to někdy dokonce tak, že o tom ani on sám nemusí vědět. Pokud by tuto možnost neměl, musel by ručně přenášet data z jedné aplikace do druhé, což je samozřejmě značně neefektivní. V praxi se vlastně jedná o dva trochu odlišné problémy.

Jednak jde o vlastní přenos dat, kdy jeden program vypočte nějaká data, pošle je jinému programu, který je použije. Tato komunikace se odehrává například pomocí OPC [9] či DDE [9] protokolu. Druhý způsob komunikace je možnost volání samotné aplikace nebo jedné její funkce z jiné aplikace. To můžeme vidět například u prohlížeče Internet Explorer, který pro čtení dokumentů ve formátu *.doc volá program MS Word a pro čtení dokumentů ve formátu *.pdf program Acrobat Reader. Obě tyto aplikace se na chvíli stanou součástí internetového prohlížeče. Tento postup je možný díky využití technologie COM [8].

3.1.15. Komunikace programu se speciálními periferiemi

Pokud je to účelné, je vhodné, aby program podporoval komunikaci se speciálními periferiemi. Tím mám na mysli i jiná zařízení než jsou standardní klávesnice, monitor, myš, atd. Jedná se například o sériovou či paralelní linku nebo o speciální vstupně-výstupní zásuvné karty do PC. Tato možnost je mnohdy velmi vítaným rozšířením aplikace.

3.1.16. Zdravotní kritéria

Program by neměl mít negativní vliv na lidské zdraví. V tomto případě jde především o různé rušivé blikání a kmitání obrázků, tabulek a všech zobrazovaných částí aplikace. To má vliv nejen na zrak, ale také na psychiku a soustředění. Podrobnější analýza těchto jevů je nad rámec této práce, proto se spokojíme s konstatováním, že v programu by nic nemělo zbytečně blikat, ani jinak škodit lidskému organismu.

3.1.17. Estetika programu

Estetika je další stránka programů, která se hodnotí velmi subjektivně. Je jednoznačné, že pokud se nám nějaká aplikace líbí, rádi jí používáme a naše činnost je tak často efektivnější. Je ale také jasné, že každý považuje za pěkné něco jiného. Proto je velmi těžké navrhnout takový design programu, který by se líbil všem, nebo by alespoň nikomu nevadil. Další oblastí v této problematice je motivace pracovníka. Vzhled by neměl být až příliš strohý. Je důležité zvýraznit podstatné ovládací prvky a funkce. Měly by vlastně „vyzývat“ uživatele k jejich použití.

Obecné zásady naštěstí přeci jen existují. Vzhled programu by měl odpovídat jeho funkci a předpokládanému zaměření uživatele. Měl by být přehledný. To znamená například, že pokud se v programu vyskytuje několik oddělených částí, je účelné, když každá má svůj osobitý vzhled. Spolu ale musejí utvářet vzhledově jednolitý celek. To znamená, že musejí obsahovat dostatek společných prvků. Vše by samozřejmě mělo být vkusné a mělo by vyhovovat výše jmenovaným kritériím, která se týkají dodržování standardů. Celkově se dá říci, že otázka estetiky programu je spíše z oblasti psychologie a vlivu prostředí na lidské

vnímání a psychiku, než z technických oborů, kterými se zaobíráme my. Je ale nutné ji nepodceňovat a věnovat její řešení dostatek času.

3.1.18. Ostatní kritéria

Mezi ostatní kritéria patří například nároky programu na použité PC. Naštěstí doby, kdy se počítal každý bajt na disku a každý bit v paměti, už jsou dávno pryč. To ovšem neznamená, že nyní program může mít libovolné parametry. Měl by mít takové nároky, jaké odpovídají jeho funkcím. Aby například pro vykreslení obyčejného statického 3D grafu nepožadoval grafickou kartu s 3D akcelerátorem a podobně.

3.2. Kritéria vývojového prostředí pro fuzzy systémy

Každý program, a tedy i ten pro návrh fuzzy systémů, má svá specifika. Oblasti, ve kterých jsou na něj kladeny zvláštní nároky a požadavky. V několika následujících kapitolách se je pokusíme zmapovat a určit tak vhodná kritéria pro hodnocení a porovnávání této skupiny aplikací. V mnoha případech nepůjde o zcela nové podmínky kladené na činnost programu, ale pouze o konkretizaci kritérií uvedených v minulé kapitole věnované obecnému pohledu na aplikace. Jednotlivá kritéria jsou řazena podle toho, jak se během práce na vývoji fuzzy systémů projevují.

3.2.1. Struktura fuzzy systému

Strukturou fuzzy systému je myšleno pořadí zapojení jejich jednotlivých funkčních členů. To znamená pořadí spojení vstup - tabulka pravidel, tabulka pravidel - tabulka pravidel, tabulka pravidel - výstup a případné jiné alternativy.

Možné struktury

Optimální program pro návrh fuzzy systému musí umožňovat snadnou práci s jednoduchými i složitými strukturami. Musí podporovat dostatečný počet vstupů, výstupů a pravidel. Z tohoto důvodu je vhodné, aby bylo možné vytvořit více nezávislých tabulek pravidel, které mohou být mezi sebou propojovány. Vznikne tak hierarchický fuzzy systém. Důležitou vlastností tohoto aplikace je druh veličiny, která jednotlivé pravidlové bloky spojuje. Může jí být ostrá hodnota nebo fuzzy hodnota. Je vhodné pokud program umožňuje oba způsoby. Uživateli dále musí být ponechána při návrhu struktury dostatečná volnost, což znamená, že smí vytvářet téměř libovolné kombinace zapojení. Návrh struktury by tedy měl být omezen pouze smysluplností spojení a parametry PC.

Úprava struktury

Na počátku vytvořenou strukturu musí jít jednoduše editovat. Rozpojovat spojení, přidávat nové prvky a obecně měnit všechny parametry, které byly během vývoje projektu nastaveny.

Kontrola správnosti struktury

Správnost (smysluplnost) struktury by měla být programem kontrolována již během návrhu systému. Je velmi nemilé, když se uživatel o své chybě či omylu dozví až na konci práce a musí pak předělávat celý projekt.

Přehlednost struktury

Dalším významným parametrem je přehlednost struktury. Pro rychlou práci je velmi podstatné, aby bylo již na první pohled zřejmé, jaké prvky se v systému nacházejí a který prvek (vstup, pravidlo,…) co ovlivňuje. Je naprosto nepřístupné, aby byly některé vztahy mezi veličinami zobrazeny, například spojovací čárou, a jiné vztahy na téže úrovni nikoliv.

3.2.2. Vstupy a výstupy

První členy ze struktury fuzzy systému, se kterými se uživatel setká, jsou vstupy a výstupy. Aplikace by měla podporovat vstupy a výstupy zpracovávající jak ostré hodnoty, tak fuzzy množiny (fuzzy čísla). Program by měl umožňovat jejich přidávání, odebírání a editaci během celého procesu návrhu systému. Pro větší rozsah použití aplikace mohou být vstupní porty doplněny o předzpracovatelské funkce, které signál například filtrují, popřípadě provádějí jeho derivaci, či integraci. Takové řešení může dosti snížit počet potřebných vstupů a výstupů a tudíž zjednodušit a zpřehlednit celý systém.

3.2.3. Jazykové proměnné

Jak bylo řečeno v kapitole o základních vlastnostech fuzzy systémů, v pravidlech se pracuje ne přímo s hodnotami vstupů a výstupů, ale s jejich obrazem ve fuzzy množinách (funkcích příslušnosti). Uživatel proto musí mít možnost jednak vlastní funkce příslušnosti vidět a jednak je také editovat. A to i v případě, že jsou automaticky generovány. Systém totiž může velmi těžko správně fungovat, pokud operátor nemá přesnou představu o tvaru a umístění fuzzy množin, popřípadě pokud si je nemůže upravit podle svých potřeb.

Podporované typy funkcí příslušnosti

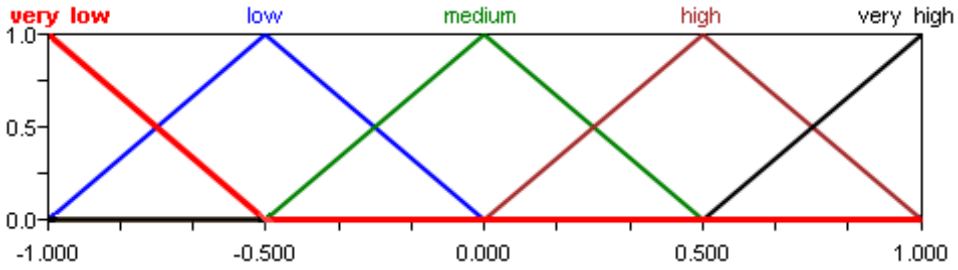
V reálném životě se setkáme s mnoha velmi odlišnými realizacemi fuzzy systémů a proto i požadavky na tvar a umístění jednotlivých funkcí příslušnosti jsou velmi odlišné. Program by měl uživateli dávat na výběr z několika druhů. Jsou to zejména funkce ve tvaru trapezoidu, S-funkce a singleton (Obr. 2.5) . Ideální by bylo, kdyby si operátor mohl vytvořit libovolnou funkci podle svých představ. Stejně jako tvar, tak také počet a universum, na kterém jsou jednotlivé fuzzy množiny definovány, by měl být libovolný, omezený pouze možnostmi PC.

Zobrazení funkcí příslušnosti

K tomu, aby byl tvar a umístění všech funkcí příslušnosti patřících k jednomu vstupu/výstupu dobře patrný, je nutné zobrazovat je graficky a všechny současně. To sice poskytne uživateli dostatečnou informaci o jejich vzhledu, ale nikoliv jejich přesnou číselnou hodnotu. Proto každý kvalitní editor funkcí příslušnosti musí zobrazovat informace také číselně, a to dostatečně přesně.

Vytvoření nové funkce příslušnosti

K vytvoření nových funkcí příslušnosti, ať na počátku práce či během práce, by měl program poskytovat funkce jak pro automatické generování, tak pro čistě ruční návrh. Automatickým generováním je myšleno například vytvoření standardního tvaru a pojmenování funkcí příslušnosti, pokud uživatel zadá pouze jejich počet. Příklad takto vygenerovaných funkcí je na následujícím obrázku.



Obr. 3.1: Příklad pěti automaticky generovaných funkcí příslušnosti

Editace existující funkce příslušnosti

Editace existující funkce příslušnosti musí být co nejsnadnější. To předpokládá, že úpravy bude možné provést jak pomocí myši, tak pomocí klávesnice. Obzvláště první jmenovaný způsob je náročné správně realizovat. Uživatel by měl při něm jednoduše táhnutím měnit tvar a pozici jednotlivých funkcí příslušnosti. Velmi vhodné je doplnění o funkce jako je inverze, atd.

3.2.4. Pravidla

Pravidla jsou zcela určitě nejdůležitější částí celého fuzzy systému. Proto musí být jejich vytváření a editace jednoduchá a hlavně přehledná. Kritérií na hodnocení práce s nimi je opět několik.

Používané logické operátory

V lidském jazyce je používáno mnoho různých spojek s dosti odlišným významem. Pokud tedy chceme popsat reálný svět pomocí pravidel ve fuzzy systému, je nutné i zde používat různé fuzzy operátory. Z tohoto důvodu by měli aplikace umožňovat spojovat vstupní i výstupní části pravidel pomocí rozličných spojek. Zapomínat by se nemělo ani na negaci. Ta nemusí být realizována pouze pomocí unárního operátoru, ale třeba automatickým generováním inverzní fuzzy množiny.

Vytváření nové tabulky pravidel

Při vytváření nové tabulky pravidel by měl mít uživatel možnost vytvořit jak úplně čistou, tak nějakým způsobem předem vyplněnou tabulku pravidel. Takový postup do značné míry šetří čas a usnadňuje v této části návrhu často mechanickou práci. Důležité je také, ve které fázi návrhu je možné tabulku pravidel vytvořit. Správně by měla jít vytvořit kdykoliv a ne například až po vytvoření vstupů a výstupů. Toto pravidlo nutí uživatele postupovat podle předem určené posloupnosti kroků, což je omezující.

Zobrazení pravidel

Reprezentace pravidel musí být co nejpřehlednější. Musí být zřejmé, které výstupy závisí na kterých vstupech. To ovšem v mnoha případech znamená, že k dispozici musí být několik možností zobrazení. Zcela určitě to je tabulka, ve které jsou napsána jednotlivá pravidla v textové formě a to buď přímo větami (IF … THEN …) nebo matematickým zápisem (… => …). Druhý vhodný způsob, který by měl program umožňovat, je matice pokrytí, což je maticově vyjádřená závislost jednoho výstupu na jednom nebo kombinaci dvou vstupů. Na obrázku Obr. 4.9 je ukázán příklad tohoto zobrazení. Obecně mohou být pravidla velmi složitá, všechna se ale dají rozepsat do na několika jednodušších, která lze zobrazit pomocí tabulky nebo matice pravidel. Proto pokud aplikace obsahuje pouze tyto dva způsoby zobrazení není pro návrháře nikterak omezující.

Editace pravidel

Pro editaci pravidel platí stejná kritéria jako pro editaci funkcí příslušnosti. Editovat by mělo jít jak v tabulce, tak v matici pokrytí. Pokud jsou pravidla psána ručně, neměl by být uživatel nucen pamatovat si přesné znění názvů funkcí příslušnosti, ale měl by si je vybírat z nabízené množiny.

3.2.5. Výpočet

V úvodní kapitole byly zmíněny matematické základy práce s fuzzy množinami. Je z nich vidět, že mnoho operací lze provádět různými způsoby. Výsledek výpočtu výstupů fuzzy systému může být tedy v závislosti na použitých metodách dosti různý. Je proto

nezbytné, aby každá aplikace obsahovala přinejmenším popis použitých forem konjunkce, disjunkce, implikace a defuzzifikace. Vhodným rozšířením použitelnosti programu je možnost volby metod z několika nabízených nebo dokonce jejich přizpůsobení potřebám uživatele.

3.2.6. Ladění fuzzy systému

Dalším významným kritériem pro porovnávání programů je možnost ladění vytvořeného fuzzy systému. K jeho splnění by aplikace měla obsahovat několik nástrojů, používaných k testování celého nebo jen některé časti systému. Základním prvkem je možnost grafického zobrazení závislosti libovolného výstupu na množině vstupů. Dále je nutné poskytnout nástroje k simulování systému. To znamená umožnit přivádět na vstup uměle (ručně, ze souboru) generované signály a sledovat, jak na ně systém reaguje. V tomto momentě by měly být vidět stupně příslušnosti vstupních/výstupních signálů do jednotlivých fuzzy množin a míry splnění jednotlivých pravidel. Hodnoty by měly být zobrazeny číselně i graficky a měla by být dána možnost jejich uložení na disk.

3.2.7. Rozšířené funkce programu

Posledním kritériem v této oblasti jsou rozšířené funkce programu. Jsou tím myšleny funkce, které aplikace nemusí ke svému správnému chodu vůbec obsahovat, jejich přítomnost ale rozšíří použitelnost programu. Jedná se například o učení ve fuzzy systémech [5], shlukovou analýzu [6] a jiné funkce sloužící například k automatickému generování některých částí systému.

3.3. Kritéria – shrnutí

Mnoho výše jmenovaných kritérií je hodnoceno relativně. Jednoznačná není ani jejich důležitost či pořadí, v jakém by měla být zmiňována. Velmi také záleží na tom, k jakému účelu je testovaný program určen. Aplikace pro rychlý návrh jednoduchých fuzzy systémů má určitě jiné nároky na metody práce se strukturou modelu než aplikace určená k návrhu rozsáhlých hierarchických fuzzy systémů. Je ale pravda, že většina kritérií by měla být vždy alespoň částečně splněna, aby mohl být program prohlášen za kvalitní nebo alespoň za použitelný. Obecně můžeme podmínky rozdělit do tří skupin (toto rozdělení je ale také subjektivní).

První jsou ty kritéria, která musejí být bezpodmínečně splněna, aby byl program vůbec použitelný. Sem patří především: bezchybnost algoritmu programu, přehlednost ovládání, možnost základní manipulace se strukturou systému, jazykovými proměnnými, funkcemi příslušnosti a pravidly. Často ale také úplnost programu, možnosti ladění a kontroly výsledného projektu a použitelný formát vstupních a výstupních dat.

Druhou skupinu tvoří pravidla, která musejí být alespoň částečně splněna, aby byl program použitelný. Sem patří například: používání standardů, jednotný systém ovládání, nápověda a pomoc v nouzi, funkce „Zpět“ a „Opakovat“, některé metody usnadnění, komunikace aplikace s jinými programy a periferiemi a zdravotní a estetická kritéria.

Do třetí skupiny patří všechny ostatní podmínky. Na jejich splnění funkčnost a použitelnost aplikace příliš nezáleží, je ale plusem, pokud je program podporuje. Do této skupiny patří například různé druhy ovládání, přizpůsobení prostředí na míru, rozšířené funkce a ostatní.

4. Srovnání existujících programů

Tato kapitola se věnuje přehledu a srovnání jednotlivých programů týkajících se návrhu a simulace fuzzy systémů. Tyto produkty se dají rozdělit do tří základních skupin. Jednak to jsou čistě akademické aplikace, které vznikají na mnoha vysokých školách a výzkumných pracovištích po celém světě. Ty se vyznačují tím, že často obsahují nové, zprvu pouze v tomto programu používané, algoritmy. Jde například o učení ve fuzzy systémech, shlukovou analýzu a jiné. Protože je kladen důraz právě na vnitřní strukturu systémů, bývá u nich problém s nedokonalostí ovládání nebo horšími vlastnostmi při praktickém použití. Opačným extrémem jsou demonstrační programy, které mají studenty naučit základům fuzzy množin a fuzzy logiky. Druhou skupinou jsou aplikace ryze komerčního původu a užití. Ty většinou používají klasické, dostatečně vyzkoušené algoritmy. To ovšem neznamená, že se v nich nemůže vyskytnout nic nového, ale jejich předností je spíše jejich všestrannost při návrhu a naopak často specielní komunikační funkce, umožňující návrháři rychlou a pohodlnou práci s jistou třídou technických prostředků, např. PLC, mikroprocesor, na kterých navržený fuzzy systém a jeho implementace nakonec poběží. Poslední skupinu tvoří aplikace, které jsou volně šiřitelné a nezasahují ani do jedné z výše jmenovaných skupin.

Programy používané pro návrh fuzzy systémů tedy mohou být velmi odlišné a to zejména s ohledem na charakter jeho cílového užití. Některá kritéria pro jejich porovnání jsou však společná. Vždy budeme chtít, aby se aplikace dobře ovládala, vše bylo přehledné, abychom měli dostatečnou volnost při návrhu fuzzy množin a pravidel a aby námi vytvořený systém byl plně funkční. Kromě těchto obecnějších požadavků je ale také nutné uvědomit si, co od programu očekáváme a k čemu ho chceme používat. Pokud například budeme pracovat pouze s malými fuzzy systémy, spokojíme se většinou s jednodušším programem, který nám neumožní vytvářet hierarchické struktury. Pokud ale budeme pracovat s velkým množstvím vstupů a výstupů, určitě budeme tuto možnost vyžadovat.

Proto při hodnocení jednotlivých produktů nebudeme dbát pouze na to, jestli splňují všechna kritéria stanovená v minulé kapitole, ale také na to, k jakému účelu lze daný produkt použít, případně zda má všechny výrobcem slibované parametry. Protože pořadí ani důležitost jednotlivých kritérií není přesně určeno, budeme zkoumat nejprve obecnější vlastnosti programu a až poté jednotlivé funkce sloužící k sekvenčnímu návrhu a testování fuzzy systému. Zaměříme se přitom spíše na komerční produkty, protože ty bývají v praxi častěji využívány. Na počátku každého rozboru programu se nachází tabulka se základními informacemi o daném produktu. Tyto informace mají sloužit k hrubé představě o aplikaci. Zdůrazněme, že většina z testovaných verzí byla k dispozici pouze jako demoverze, proto některé parametry mohou být u „ostrých“ verzí odlišné. Jedná se především o maximální počty vstupů, výstupů a pravidel ve vytvořených fuzzy systémech. Veškeré programy byly testovány na stejném počítači s následujícími parametry.

Procesor	-	Celeron 900MHz
Paměť	-	500MB (133MHz)
Pevný disk	-	20GB
Operační sys.	-	Windows 2000 (Service Pack 2)

Aplikací zabývajících se tvorbou fuzzy systému je nepřeberné množství, proto bylo velmi těžké zvolit jen několik málo, které se mají otestovat. Bohužel nebylo v našich silách analyzovat všechny (zdaleka ne každá je volně k dostání byť jen v testovací verzi). Vybrali jsem proto ty, které jsou, podle našeho názoru, zajímavé svými vlastnostmi. Pro přesnější představu jsme v příloze A vypsali několik z mnoha používaných produktů. Tento seznam

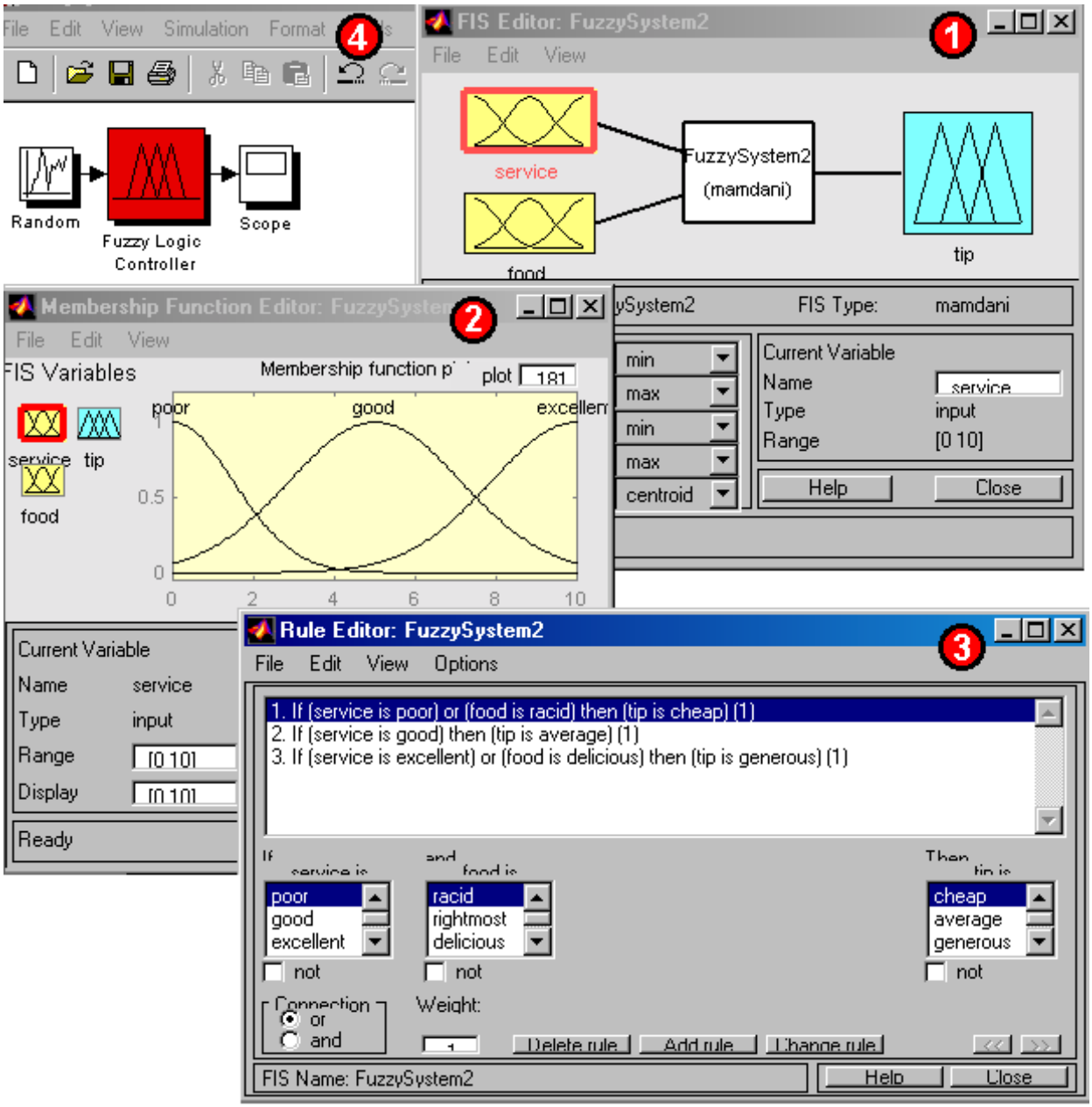
obsahuje převážně významné aplikace oficiálně zaregistrované na území USA. Jako zdroj informací jsme použili internetovou adresu [23], kde lze nalézt úplný seznam všech těchto produktů a internetovou adresu [24], která je věnována přímo aplikacím používaným pro vývoj fuzzy systémů.

4.1. Fuzzy Logic Toolbox

Základní informace o testovaném programu	
Název	Fuzzy Logic Toolbox
Verze	2.0.1 (R11)
Operační systém	MS Windows, Unix
Datum vytvoření	16.9. 1998
Výrobce	Math Works Inc.
Velikost	4,5MB
Druh	Komerční produkt
Stručný popis	Program je toolbox matematického programu Matlab. Slouží k vytváření a simulaci jednoduchých nehierarchických fuzzy systémů.
Maximální počty jednotlivých částí vytvořeného fuzzy systému	
Vstupy	Přesný počet není znám. Experimentálně ověřeno do počtu 100.
Výstupy	Přesný počet není znám. Experimentálně ověřeno do počtu 100.
Fce. přís. u vstupu	9
Fce. přís. u výstupu	9
Pravidla	Přesný počet není znám. Experimentálně ověřeno do počtu 100.

Fuzzy Logic Toolbox je nesamostatnou částí Matlabu, to znamená, že nemůže sám fungovat bez použití Matlabu. Licence na Fuzzy Logic Toolbox umožňuje jeho případnou instalaci.. Ovládání tohoto toolboxu je rozděleno do dvou oblastí, které se svými funkcemi v podstatě překrývají. Jde o ovládání pomocí příkazové řádky nebo pomocí speciálních GUI nástrojů (Graphical User Interface). Podrobněji se oběma způsobům budeme věnovat v následující podkapitole.

Jako každá součást Matlabu je i tato koncipována jako SDI (Single-Document Interface), což v praxi znamená, že každé ovládací okno je zcela samostatné, může se libovolně pohybovat po celé obrazovce, zavírat, otevírat a minimalizovat bez toho, aby tím ovlivnilo okolní okna. Uživateli to přináší volnost při manipulaci s jednotlivými dialogy a hlavně větší pracovní plochu. Na druhé straně v případě, že je otevřeno více aplikací, způsobuje tato koncepce mírný zmatek na monitoru. Na obrázku Obr. 4.1 je ukázána typická pracovní plocha pro případ, že vytváříme fuzzy projekt pomocí GUI nástrojů. Je zde zobrazen celkový náhled na strukturu fuzzy systému (č.1), editor funkcí příslušnosti (č.2), editor pravidel (č.3) a Simulinkový blok zastupující vytvořený fuzzy systém (č.4), který se stane komponentou blokového simulačního schématu. Podrobnější informace o aplikaci lze nalézt v [14].



Obr. 4.1: Typický vzhled pracovní plochy při práci s Fuzzy Logic Toolboxem v Matlabu

4.1.1. Obecný pohled

Při prvním setkáním s Fuzzy Logic Toolboxem bude uživatel zcela jistě potřebovat pročíst si alespoň zběžně manuál. Ten je velmi dobře připraven. Nacházejí se zde kapitoly popisující základy fuzzy množin a fuzzy logiky, mnoho příkladů a podrobný popis, jak od základů vytvořit nový fuzzy systém za použití příkazové řádky nebo GUI. K dispozici je také nápověda. Ta se dá z příkazové řádky i z prostředí GUI vyvolat přímo k jednotlivým funkcím. Druhý způsob je listovat v ní jako v abecedním seznamu obsahujícím informace o veškerých nástrojích toolboxu.

Jak již bylo několikrát řečeno, ovládání je možné buď z klávesnice nebo pomocí GUI nástrojů, přičemž oba tyto přístupy se mohou libovolně kombinovat. Jelikož je Matlab jako takový příkazově orientovaný program, nevyhneme se ani při zvolení druhého způsobu psaní některých příkazů. To samozřejmě snižuje komfort při návrhu.

Pokud se uživatel rozhodne používat výhradně příkazovou řádku, bude manuál potřebovat dvojnásob. K dispozici má dostatečné množství funkcí pro vytvoření, modifikaci a

zobrazení fuzzy systému. Nevýhodou tohoto postupu je nutnost pamatovat si alespoň přibližné znění jednotlivých příkazů a špatná přehlednost aktuálně vytvořeného fuzzy systému. Naopak velkou výhodou tohoto postupu je, že fuzzy systém může být vytvořen a měněn automaticky (bez aktuálního zásahu operátora) z m-souboru (druh textového dávkového souboru používaného v prostředí Matlabu).

Ovládání pomocí GUI je zcela určitě jednodušší, rychlejší a hlavně přehlednější. Na druhou stranu pomocí něj nelze vytvořit vše. Některé specializované funkce existují pouze v příkazové podobě. Jedná se například o vytvoření vlastního vzoru funkce příslušnosti (viz dále). Přestože při tomto způsobu manipulace se systémem máme k dispozici jednotlivá okna s grafickou reprezentací příkazů, neodpovídá ovládání všem kritériím stanoveným v předcházející kapitole. Omezující je především fakt, že k dispozici je vždy jeden jediný způsob, jak danou operaci v GUI provést. Úplně chybí kontextová menu, toolbar a klávesové zkratky. Jedinými ovládacími prvky jsou tady hlavní menu a tlačítka. Oba tyto prvky splňují požadavky na standardní tvar a funkci. Jsou přehledně uspořádány a vykazují ve všech oknech GUI stejné vlastnosti. V úvodu bylo řečeno, že při současném otevření více aplikací je na pracovní ploše mírný zmatek, mezi jednotlivými okny se ale dá přepínat velmi snadno. Slouží k tomu tlačítka, která po stisknutí otevřou příslušný dialog s editorem funkcí příslušnosti, tabulky pravidel či jiné okno vývojového prostředí.

Asi největší nevýhodou tohoto GUI je nefunkčnost operací „Zpět“ a „Opakovat“. Přesněji řečeno funkce „Opakovat“ úplně chybí a funkce „Zpět“ obsahuje pouze dva minulé kroky. Navíc se do historie ukládají pouze některé provedené operace, což způsobuje nejasnosti v tom, která operace bude po případném použití funkce „Zpět“ vrácena. Položka „Undo“ („Zpět“) se vyskytuje ve všech ovládacích oknech, ale akce provedená po jeho stisku může být odlišná. To opět vnáší nepřehlednost do používání této funkce.

Obecně se prostředí GUI vyznačuje možná až přílišnou strohostí. Není možné ho upravit podle aktuálních potřeb uživatele. Grafika je jednoduchá, ale ve většině případů postačující. Nic příliš nebliká. Jistým nedostatkem je nedokonalost vykreslování oken při jejich zmenšení (viz Obr. 4.1 – č.4). Další drobnou vadou grafiky jsou problémy se zobrazováním základního schématu a tabulky pravidel, pokud se počet jazykových proměnných zvýší nad určitou mez (řádově desítky). Toto množství je ale tak vysoké, že se v praxi stejně nikdy s jednou tabulkou pravidel nevyužije. Hardwarové nároky jsou přiměřené.

4.1.2. Tvorba fuzzy systému

Již v úvodní tabulce se základními informacemi o programu bylo řečeno, že tento vývojový prostředek umožňuje tvořit pouze nehierarchické fuzzy systémy. To znamená, že k dispozici je jedna tabulka pravidel a sada vstupů a výstupů s odpovídajícími fuzzy množinami a funkcemi příslušnosti. Celkové schéma je dobře patrné v hlavním okně GUI nástrojů nazvaném *FIS Editor* (Obr. 4.1 – č.1). Pokud bude uživatel chtít zkonstruovat hierarchický fuzzy systém, musí si vytvořit několik samostatných podsystémů a ty pak v Simulinku spojit do jedné struktury (jedná se o spojení pomocí proměnných s ostrými hodnotami). Každý subsystém je pak reprezentován jedním blokem, který je ukázán na Obr. 4.1 – č.4. Tento způsob má nevýhodu v menší přehlednosti celkového modelu. Některé informace o jeho struktuře jsou obsaženy pouze v zapojení Simulinkového schématu a některé pouze ve *FIS Editorech*.

Při vytváření nového systému je předdefinováno schéma s jednou vstupní a jednou výstupní jazykovou proměnnou. Přidávání nebo odebírání dalších je velmi snadné a provádí se pomocí hlavního menu. Každá proměnná může být v libovolnou chvíli přejmenována, a to i na jméno, které již používá jiná. To je poměrně citelný nedostatek, protože výsledný systém může být dosti nepřehledný. Dalším nedostatkem je nemožnost přemístění jazykové

proměnné do jiné pozice na základním schématu. U výstupů je možné použít buď klasické nebo Takagi Sugeno výstupní jazykovou proměnnou. Ne však obě současně. Všechny výstupy musejí být buď jednoho nebo druhého druhu.

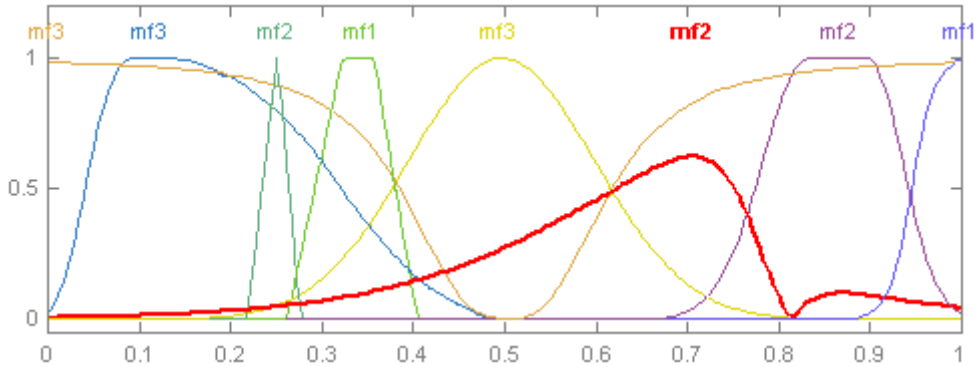
Počet vstupů a výstupů je zcela jistě dostatečný. Nepodařilo se mi zjistit, jaké jsou přesné limity pro jejich počet, ale z praktických testů vím, že jich může být více než sto. Takové číslo u jedné tabulky pravidel již pozbývá reálný význam. Při tak velkém počtu se také objevují problémy s grafikou a ovládáním celého systému (viz výše).

K editování fuzzy množin slouží například GUI nástroj nazvaný *Membership Function Editor* (Obr. 4.1 – č.2). Fuzzy Logic Toolbox podporuje nezvykle mnoho druhů funkcí příslušnosti. Některé z nich jsou ukázány na obrázku (Obr. 4.2). Každý druh má svou předlohu, kterou si uživatel podle jména vybere při vytváření nové funkce příslušnosti. Velmi zajímavou vlastností programu je možnost vytváření vlastních vzorových funkcí příslušnosti. Některé dokonce mohou být subnormální (nemají maximum rovno jedné) a nekonvexní (mají více maxim).

Funkce příslušnosti se přidávají pomocí hlavního menu. Nové funkce příslušnosti se vždy automaticky rozmístí do úvodního rozestavení podle svého typu a počtu. Takové rozmístění je u prvních funkcí často postačující a nemusí se měnit. Automaticky jsou také generována jména. Není ovšem možné zvolit si jejich sadu tak, aby výsledné pojmenování vypadalo například podobně jako na obrázku Obr. 3.1. Další přidané funkce je samozřejmě nutné většinou upravovat. Problematické je přidávání uživatelem definovaných funkcí příslušnosti. Operátor si musí pamatovat jméno předlohy a počet a význam parametrů. Vše pak musí ručně zapsat do úvodního dialogu.

Zobrazení funkcí příslušnosti je grafické a u právě vybrané křivky také číselné. Nevýhodou je absence barevného odlišení. Při vybrání nějaké funkce příslušnosti je odpovídající křivka dostatečně zviditelněna včetně všech svých význačných bodů.

Editace křivek je možná dvěma způsoby. Prvním je přesné stanovení parametrů pomocí číselných hodnot význačných bodů. Nevýhoda tohoto postupu je, že počet a význam parametrů se s typem funkce příslušnosti mění. Protože nikde v bezprostřední blízkosti není vysvětlen jejich aktuální význam, musí si uživatel vše pamatovat nebo vyhledávat v nápovědě. Druhý způsob editace je pomocí myši. Je sice dosti pružný, uživatel si ale na něj musí zvyknout. Obtížné je například uchopit kotvící bot, protože plocha, na kterou se kurzor myši musí trefit, je poměrně malá. Snadno se dají měnit i jména jednotlivých funkcí příslušnosti. Opět je zde problém s možnou duplicitou pojmenování (viz výše). Nastavit lze také číselný rozsah celé fuzzy množiny. Všechny provedené změny se automaticky promítají do dalších částí systému. Kontrolovány jsou i situace, kdy je vymazána nějaká funkce příslušnosti používaná v tabulce pravidel. V takovém případě je uživatel na nastalou situaci dostatečně důrazně upozorněn.



Obr. 4.2: Příklady tvarů funkcí příslušnosti ve Fuzzy Logic Toolboxu (uměle obarveno)

Pravidla jsou v GUI uspořádána do řádků přehledné tabulky, která se dá zobrazit a editovat nástrojem *Rule Editor*. Po vytvoření nové tabulky pravidel je na počátku úplně prázdná. Program neposkytuje žádné možnosti k jejímu automatickému naplnění. Uživatel musí vše dělat ručně. Nová pravidla se vytvářejí velmi snadno pouhým výběrem potřebných jazykových proměnných a jejich funkcí příslušnosti z automaticky generovaného seznamu. Odebírání také nečiní žádný problém. Editace existujících pravidel se provádí stejným způsobem jako přidání nového. Poměrně velkým nedostatkem je nemožnost přemísťování jednotlivých řádků tabulky. Další možností modifikace je nastavování váhy jednotlivých pravidel.

Způsob jejich zobrazení je možné volit v několika formátech. Předně je to klasický slovní zápis pomocí vět IF…THEN… Zajímavým doplňkem je volba jazyka, ve kterém bude daný „text“ zapsán. K dispozici je angličtina, němčina a francouzština. Tento způsob je nejbližší lidskému vyjadřování, je ale dosti rozsáhlý a u složitějších pravidel tudíž i nepřehledný. Druhou možností zobrazení pravidel je symbolický zápis ve tvaru (…==…) => (…). Takový formát má výhodu především ve své jazykové nezávislosti. Poslední způsob je reprezentace pomocí indexů. Pravidlo s významem „IF InVar1 is MF1 and InVar2 is MF3 THEN OutVar1 is MF1“ se vypíše jako „1, 3 (1)“. Je vidět, že jeho hlavní předností je extrémně krátký zápis, který může být výhodný při zobrazování velkých tabulek pravidel. Mezi jednotlivými formáty je samozřejmě možné kdykoliv libovolně přepínat. Zobrazovat pravidla pomocí matice pokrytí bohužel není ve Fuzzy Logic Toolboxu možné. Nejsou ani k dispozici žádné vložené komentáře, které by mohly zpřehledňovat složitější tabulky.

4.1.3. Ladění a výstupní data fuzzy systému

K ladění vytvořeného fuzzy systému slouží dva základní GUI nástroje. První je *Surface Viewer*. Ten zobrazuje graf závislosti vybraného výstupu na jednom nebo dvou vstupech. Grafika je velmi pěkná, manipulace v podobě rotace je jednoduchá, nastavování parametrů jako jsou vstupní a výstupní proměnné a jejich rozsah je velmi snadné, vše je poměrně přehledné. Příjemnou zajímavostí je velké množství volitelných pohledů na danou závislost. Od 3D grafu, přes zobrazení vrstevnice až k vektorovému poli. Drobnou nevýhodou je fakt, že nelze nastavit hodnoty ostatních (nezobrazovaných) vstupů, které samozřejmě také mají vliv na tvar dané závislosti. Druhý nástroj pro ladění je *Rule Viewer*. Zde se přehledně graficky zobrazují míry splnění předpokladů a závěrů jednotlivých pravidel. Vstupní data mohu snadno simulovat ručním nastavováním nebo je mohu „on-line“ načítat z Matlabu.

Výsledný fuzzy systém lze použít přímo v Matlabu k dalším výpočtům. Velmi snadno ho mohu zařadit do Simulinkového schématu a zobrazovat tak například jeho časové průběhy výstupů. Už ale nemám možnost zobrazovat vnitřní proměnné. Právě díky úzkému spojení s Matlabem je možné výsledný systém převádět do různých formátů, jako je jazyk C, simulovat jeho chování s umělými i reálnými daty (on-line), nechat ho komunikovat s jinými programy a různými jinými způsoby ho testovat a používat v reálných systémech.

Díky příkazům umožňujícím vznik a ovládání celého fuzzy systému je možné vytvářet ho i v jiných programech pouze tím, že do textového souboru uloží posloupnost potřebných příkazů. Další možností je generování přímo souboru *.fis, který obsahuje ve formě podobné XML běžně čitelný zdrojový kód pro vytvoření systému. V obou případech stačí výsledný soubor otevřít v Matlabu a ten z něj vytvoří plně funkční fuzzy systém.

Z ostatních funkcí usnadňujících práci na fuzzy systému či rozšiřujících možnosti aplikace lze použít učení ve fuzzy systémech a shlukovou analýzu [6]. K dispozici není žádný komplexní nástroj, který by automaticky generoval dokumentaci k vytvořenému systému. V Matlabu však existuje mnoho dílčích funkcí, které umožňují vytisknout nebo přehledně zobrazit jednotlivé části fuzzy systému.

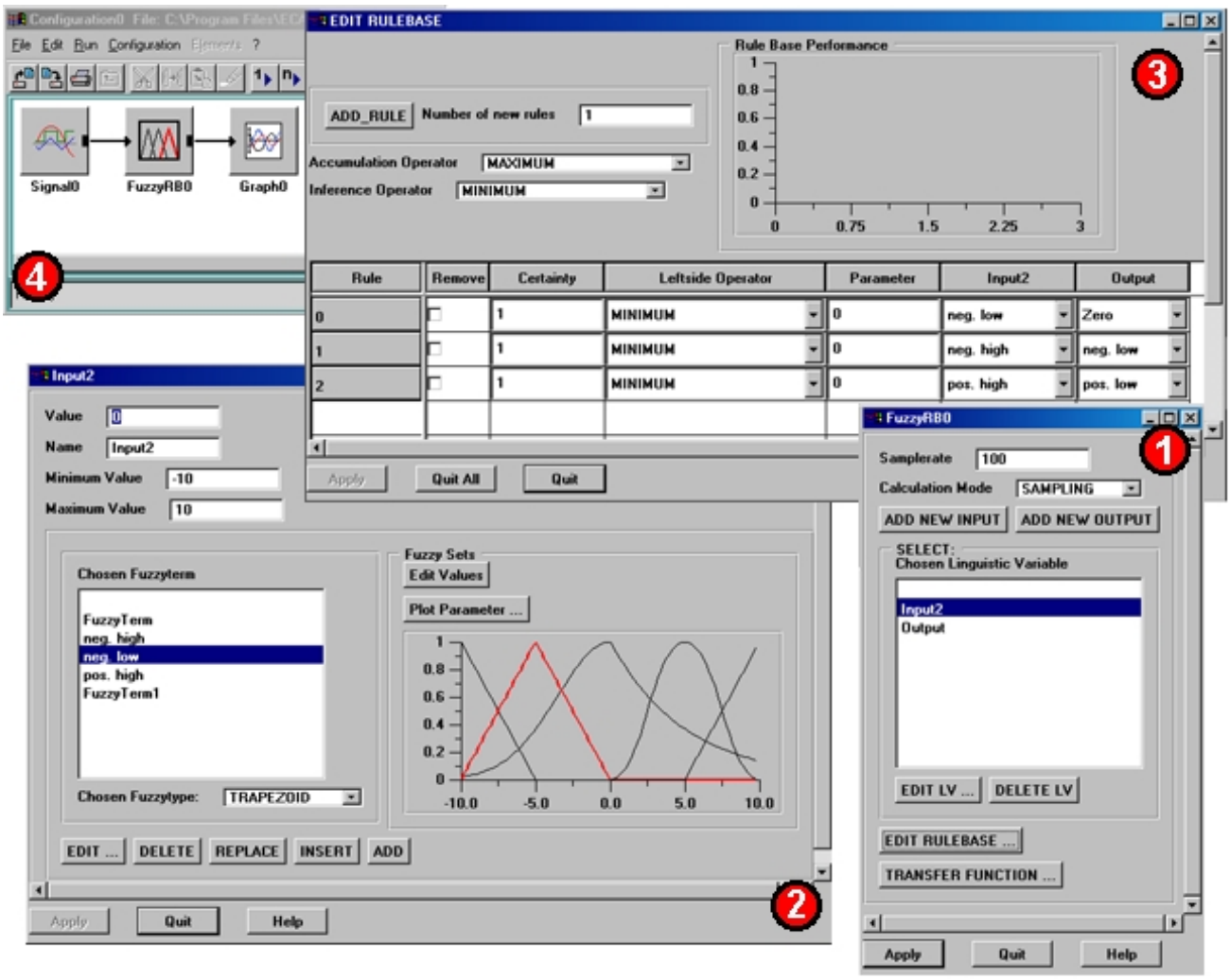
4.1.4. Celkové hodnocení

Celkově je vývoj a editace fuzzy systému za pomoci Fuzzy Logic Toolboxu poměrně přímočarý a přehledný. Před prvním použitím tohoto programu je ale nutné prostudovat alespoň zběžně kvalitně připravený manuál nebo nápovědu. Vlastní ovládání aplikace neposkytuje takový komfort jako některé jiné produkty. To je ovšem způsobeno omezeními, které na program klade Matlab. Zajímavé je velké množství možných tvarů funkcí příslušností. Vytvořený systém je přehledný a dá se poměrně dobře ladit. Vše je ale nepatrně složitější a často se nevyhneme použití příkazové řádky. Kvůli nehierarchické struktuře je toolbox vhodný spíše pro menší fuzzy systémy.

4.2. Fuzzy knihovna v ECANSE

Základní informace o testovaném programu	
Název	ECANSE 2000 (Environment for Computer Aided Neural Software Engineering)
Verze	Demo 2.03
Operační systém	MS Windows
Datum vytvoření	2000
Výrobce	Siemens AG
Druh	Komerční produkt
Stručný popis	Program je knihovnou simulačního programu ECANSE. Slouží k vytváření a simulaci jednoduchých nehierarchických fuzzy systémů.
Maximální počty jednotlivých částí vytvořeného fuzzy systému	
Vstupy	Přesný počet není znám. Experimentálně ověřeno do počtu 100.
Výstupy	Přesný počet není znám. Experimentálně ověřeno do počtu 100.
Fce. přís. u vstupu	Přesný počet není znám. Experimentálně ověřeno do počtu 100.
Fce. přís. u výstupu	Přesný počet není znám. Experimentálně ověřeno do počtu 100.
Pravidla	Přesný počet není znám. Experimentálně ověřeno do počtu 100.

Zkoumaná knihovna je nesamostatnou součástí programu ECANSE (rozsáhlý simulační program podobný matlabskému Simulinku). Uživatel nemá možnost ovlivnit, zda se s ECANSE nainstaluje nebo nikoliv. Aktivuje se jednoduše vložením příslušného bloku do simulačního schématu. Vlastní ovládací okna jsou koncipována jako u SDI aplikace a jsou většinou modální. To samozřejmě zhoršuje přehlednost celého systému. Jejich chování je navíc poměrně nestandardní. Aktivní „modální“ okno sice neumožní provádět změny v jiném dialogu, umožní ale přesunout do popředí jiná okna. Snadno se tak může stát, že uživatel po ukončení práce v modálním okně ho zapomene zavřít, klikne na jiné okno, to se bez problémů zobrazí, ale po pokusu provést v něm nějakou změnu se nic nestane. Taková situace je zcela určitě dosti matoucí. Operátor se navíc musí vrátit k „zapomenutému“ modálnímu dialogu, což nemusí být při práci s mnoha okny nebo aplikacemi až tak samozřejmé. Celá situace tak může vyústit ve zbytečné hledání blokujícího okna. Na obrázku Obr. 4.3 je ukázáno několik pracovních oken používaných při návrhu fuzzy systému. Je zde zobrazen celkový náhled na strukturu fuzzy systému (č.1), editor funkcí příslušnosti (č.2), editor pravidel (č.3) a simulační blok zastupující vytvořený fuzzy systém (č.4). Podrobnější informace o aplikaci lze nalézt v [15].



Obr. 4.3: Typický vzhled pracovní plochy při práci s fuzzy knihovnou v ECANSE

4.2.1. Obecný pohled

Fuzzy knihovna v ECANSE má jednoduchou strukturu, přesto práce s ní přináší nejeden problém. Již samotné ovládání je dosti neintuitivní a často pouze pomocí klávesnice. Úplně chybí hlavní a kontextové menu, toolbar a klávesové zkratky. K dispozici není ani nejzákladnější nápověda. Veškeré informace musí uživatel získávat z manuálu. Ten je naštěstí proveden velmi kvalitně. Ostatní pomocné funkce usnadňující práci, jako je nastavení prostředí na míru, automatické doplňování a podobné funkce usnadnění, také chybí.

Jednotlivá okna jsou často nepřehledná. Ovládací tlačítka jsou někdy nevhodně umístěna. Citelnou vadou je pevná velikost většiny dialogů. Ty se nedají zmenšovat a často tak zabírají na obrazovce zbytečně mnoho místa. Grafika není příliš kvalitní, ale nikde nic nebliká, nikde se nic zbytečně nepřekresluje. Celkový vzhled není příliš lákavý. Hardwarové nároky programu jsou úměrné jeho funkcím, ale při opravdu velkém počtu vstupů (>100) se stává program nestabilním.

4.2.2. Tvorba a ladění fuzzy systému

Fuzzy knihovna v ECANSE neumožňuje vytvářet hierarchické fuzzy systémy. K dispozici je tedy pouze jedna tabulka pravidel a sada vstupů a výstupů. Jejich přidávání a odebírání je velmi snadné. Neobsahují možnosti žádných předzpracování nebo úpravy signálů. Nové vstupy i výstupy se automaticky pojmenovávají a přidávají se do společného seznamu v hlavním ovládacím okně (Obr. 4.3 – č.1). Jelikož nikde u jejich jména není značka,

zda jde o vstup či o výstup, nelze po přejmenování jednoduše říci, jakého typu je daný prvek seznamu. Není ani možné upravovat pořadí jednotlivých prvků. Protože výše jmenovaný seznam je jediným zobrazením struktury modelu, může být systém někdy dosti nepřehledný.

Jazykové proměnné (vstup, výstup) se editují v modálním dialogu ukázaném na obrázku Obr. 4.3. – č.2. Mohou obsahovat velké množství funkcí příslušnosti. K dispozici je dostatečný počet předdefinovaných typů, které se dají upravovat i do dosti neobvyklých tvarů. U nové jazykové proměnné jsou některé funkce příslušnosti vygenerovány automaticky. Uživatel ovšem nemá možnost zvolit si jejich počet ani pojmenování. Úvodní umístění navíc naprosto nevyhovuje standardním potřebám. Operátor tedy téměř vždy musí do vytvořené struktury zasahovat. Přidávání a odebírání dalších funkcí příslušnosti je jednoduché a nevyžaduje od uživatele pamatovat si žádná jména vzorových křivek. Zobrazení funkcí příslušností je grafické, ale nepříliš přehledné. Křivky jsou černobílé bez jakéhokoliv popisu. Čárová grafika je dosti kostrbatá. Jedině právě vybraná křivka je dostatečně odlišena od ostatních. Způsob editace tvaru a polohy existujících funkcí příslušnosti je naprosto nevyhovující. Jediná možnost, jak změnit parametry fuzzy množiny, je úprava číselných parametrů. Ty mají u každého typu křivky jiný význam, který není nikde poblíž vysvětlen. Měnit se dá i rozsah definičního oboru vstupní/výstupní proměnné. Veškeré úpravy se bez problémů promítají do celého projektu.

Editor pravidel je ukázán na obrázku Obr. 4.3 – č.3. Neumožňuje žádné automatické generování pravidel. Uživatel musí vkládat všechna pravidla ručně. Při vytváření nového jsou všechny nastavitelné parametry vybírány z nabízených možností, takže uživatel nemusí vůbec použít klávesnici. Také editace a mazání pravidel jsou snadné. Nastavitelná je také váha jednotlivých pravidel. Jedinou nevýhodou je nemožnost přemisťování řádků tabulky a absence komentářů. To činí větší systémy nepřehledné. Naopak pozitivní je fakt, že je k dispozici nepřeberné množství druhů logických spojek používaných při práci s pravidly.

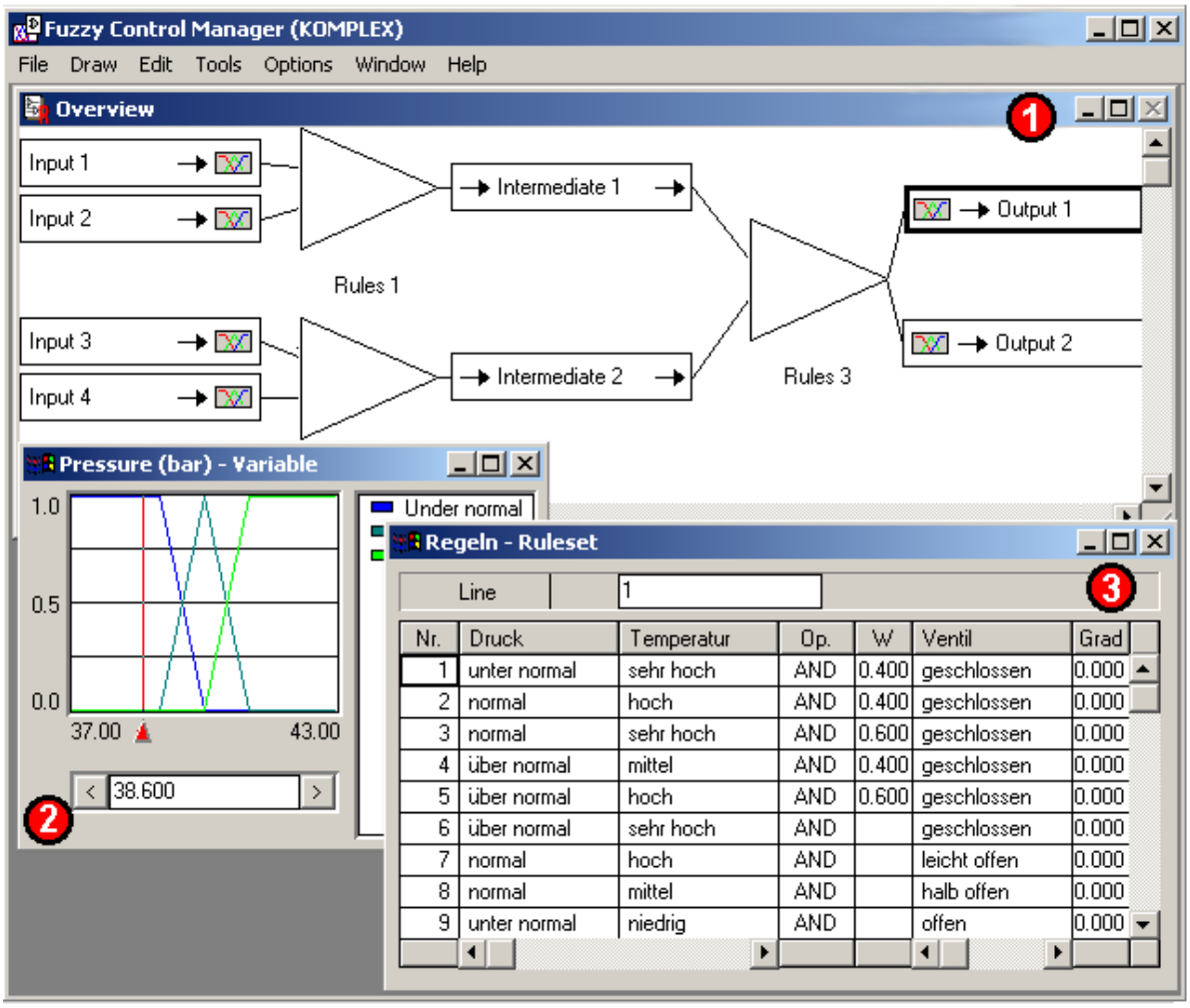
Pro ladění výsledného systému lze použít statický graf závislosti jednoho výstupu na jednom nebo dvou vstupech, přičemž hodnota ostatních se nedá nastavovat. Zobrazení je poměrně přehledné, umožňuje libovolnou rotaci a volbu počtu stupňů barev. Veškeré ovládání ale opět probíhá výhradně pomocí klávesnice. Druhým nástrojem pro monitorování je sloupcový graf ukazující míru splnění jednotlivých pravidel. U jazykových proměnných naopak chybí jakékoliv zobrazení stupně příslušnosti k jednotlivým fuzzy množinám. Simulaci nelze provádět ručně, ale jelikož je výsledný fuzzy systém součástí ECANSE, je možné přivést na vstup počítačem generovaný signál a časové průběhy výstupů pak zobrazovat do grafu. Tímto způsobem ale není samozřejmě možné sledovat vývoj vnitřních proměnných. Spojení s ostatními programy obstarává výhradně prostředí ECANSE, takže na něm a na jeho verzi závisí, s kterými periferiemi a aplikacemi bude možno komunikovat. Knihovna neobsahuje žádné nadstandardní funkce pro práci s fuzzy systémy ani podporu pro tvorbu dokumentace. Výsledný fuzzy systém nelze uložit do samostatného souboru. Vždycky musí být součástí nějakého simulačního schématu. Výsledný soubor je v textové formě a dá se tedy z vnějšku editovat.

4.2.3. Celkové hodnocení

Fuzzy knihovna v ECANSE sice obsahuje veškeré potřebné prostředky pro tvorbu funkčních menších fuzzy systému, její ovládání je ale zcela nevyhovující dnešním požadavkům uživatelů. Obzvláště nevhodně je vytvořen editor funkcí příslušnosti. Nástroje pro ladění výsledného systému také nesplňují všechna stanovená kritéria. Zejména jedná-li se o monitorování časového vývoje systému. Jedinou zajímavou částí programu je možnost volby velkého množství logických operátorů pro práci s pravidly.

4.3. Fuzzy Control Manager

Základní informace o testovaném programu	
Název	Fuzzy Control Manager
Verze	1.5.1e (demo)
Operační systém	MS Windows
Datum vytvoření	1996
Výrobce	TransferTech GmbH
Velikost	3.5MB
Druh	Komerční produkt
Stručný popis	Program je samostatný produkt, který slouží k vytváření a simulaci složitých i zpětnovazebních hierarchických fuzzy systémů.
Maximální počty jednotlivých částí vytvořeného fuzzy systému	
Vstupy	Přesný počet není znám. Experimentálně ověřeno do počtu 100.
Výstupy	Přesný počet není znám. Experimentálně ověřeno do počtu 100.
Fce. přís. u vstupu	Přesný počet není znám. Experimentálně ověřeno do počtu 100.
Fce. přís. u výstupu	Přesný počet není znám. Experimentálně ověřeno do počtu 100.
Pravidla	Přesný počet není znám. Experimentálně ověřeno do počtu 100.



Obr. 4.4: Typický vzhled pracovní plochy při práci s programem Fuzzy Control Manager

Fuzzy Control Manager sice patří, pokud jde o datum vytvoření, mezi starší programy, obsahuje ale několik zajímavých funkcí, které se v jiných aplikacích nevyskytují. Proto jsem se ho rozhodl zařadit mezi testované produkty. Je koncipován jako MDI (Multiple-Document Interface), přičemž dokumentem není myšlen samotný projekt, ale jakýkoliv dialog. To znamená, že existuje jediné hlavní okno, které ohraničuje pracovní plochu využívanou tímto programem a všechna ostatní (kromě modálních) se zobrazují uvnitř něj. Tato koncepce má výhodu v přehlednosti jednotlivých otevřených dialogů. Na první pohled je vidět, které okno patří ke které aplikaci. Nevýhodou je částečně zmenšený pracovní prostor a vzájemná závislost dceřinných oken. Ta se negativně projevuje například ve chvíli, kdy změníme jedno okno z maximalizovaného na menší velikost. Současně se nám totiž stejným způsobem změní i ostatní okna., což je ve většině případů nežádoucí. Tento nedostatek není způsoben programem jako takovým, nýbrž samotným operačním systémem Windows, který takto přistupuje k řešení obdobných problémů. Přestože tato aplikace je koncipována jako MDI, neumožňuje pracovat současně s více než jedním projektem. Pokud to uživateli nevyhovuje, je nucen spustit další kopii Fuzzy Control Manageru. Na obrázku Obr. 4.4 je zobrazeno hlavní okno programu obsahující základní nástroje pro práci s fuzzy systémem. Celkový pohled na strukturu systému je v okně s číslem 1. Funkce příslušnosti se editují pomocí dialogu 2 a pravidla například pomocí dialogu 3. Podrobnější informace o aplikaci lze nalézt v [16].

4.3.1. Obecný pohled

Ovládání Fuzzy Control Manageru není příliš dobře řešeno. Uživatel má k dispozici často jen jeden způsob. Většinou je to hlavní nebo kontextové menu. Obě jsou řazena poměrně přehledně a blíží se běžně používaným menu. Klávesové zkratky jsou zastoupeny pouze sporadicky. Jejich význam navíc často neodpovídá dnešním standardům. Úplně chybí toolbary. Celkově se dá říci, že ovládání často působí dosti těžkopádným dojmem. Zcela nevhodné je například reagování na první stisk levého tlačítka myši tím, že se do projektu vloží nějaký prvek. Konkrétně tato vlastnost působí začínajícím uživatelům velké problémy. Dalším zásadním nedostatkem je absence funkcí „Zpět“ a „Opakovat“. V programu není jediná možnost, jak si práci s aplikací zjednodušit, například přizpůsobením některých ovládacích prvků nebo deaktivováním „obtěžujících“ funkcí. Jedním ze světlejších míst této aplikace je používání automatického doplňování jmen jazykových proměnných a fuzzy množin při práci s tabulkou pravidel.

Pro případ potřeby je připraven kvalitní manuál a nápověda. Lze v něm snadno listovat, vyhledávat podle tématu či předdefinovaných hesel. Bohužel chybí přímé propojení aktuálně prováděné akce s nápovědou. Pokud tedy například edituji některou část fuzzy systému a chci se o ní něco dozvědět, musím otevřít hlavní nápovědu a v ní se teprve „proklikat“ k odpovídajícímu tématu. Dalším drobným nedostatkem je absence fulltextového vyhledávání. Pro začínající uživatele je připraveno několik ukázkových projektů demonstrujících nejen možnosti při vytváření fuzzy systému, ale také způsoby jeho propojení s okolím (například pomocí DDE).

Z estetického hlediska je na aplikaci znát, že vznikla již před delší dobou. Kromě hlavního schématu systému není grafika příliš kvalitní, a tudíž ani přehledná. Při zvětšování oken dochází často ke značnému blikání. Velmi málo jsou využívány barvy. Program pak působí černobílým dojmem. Většina dialogů má konstantní velikost. Nelze tudíž plně využít celou pracovní plochu. Hardwarové nároky jsou vlivem použitých jednoduchých technologiích minimální.

4.3.2. Tvorba fuzzy systému

Program umožňuje vytvářet jednoduché i velmi složité hierarchické fuzzy systémy. Na pracovní ploše lze zkonstruovat i struktury se zpětnými vazbami, program ale s takovými

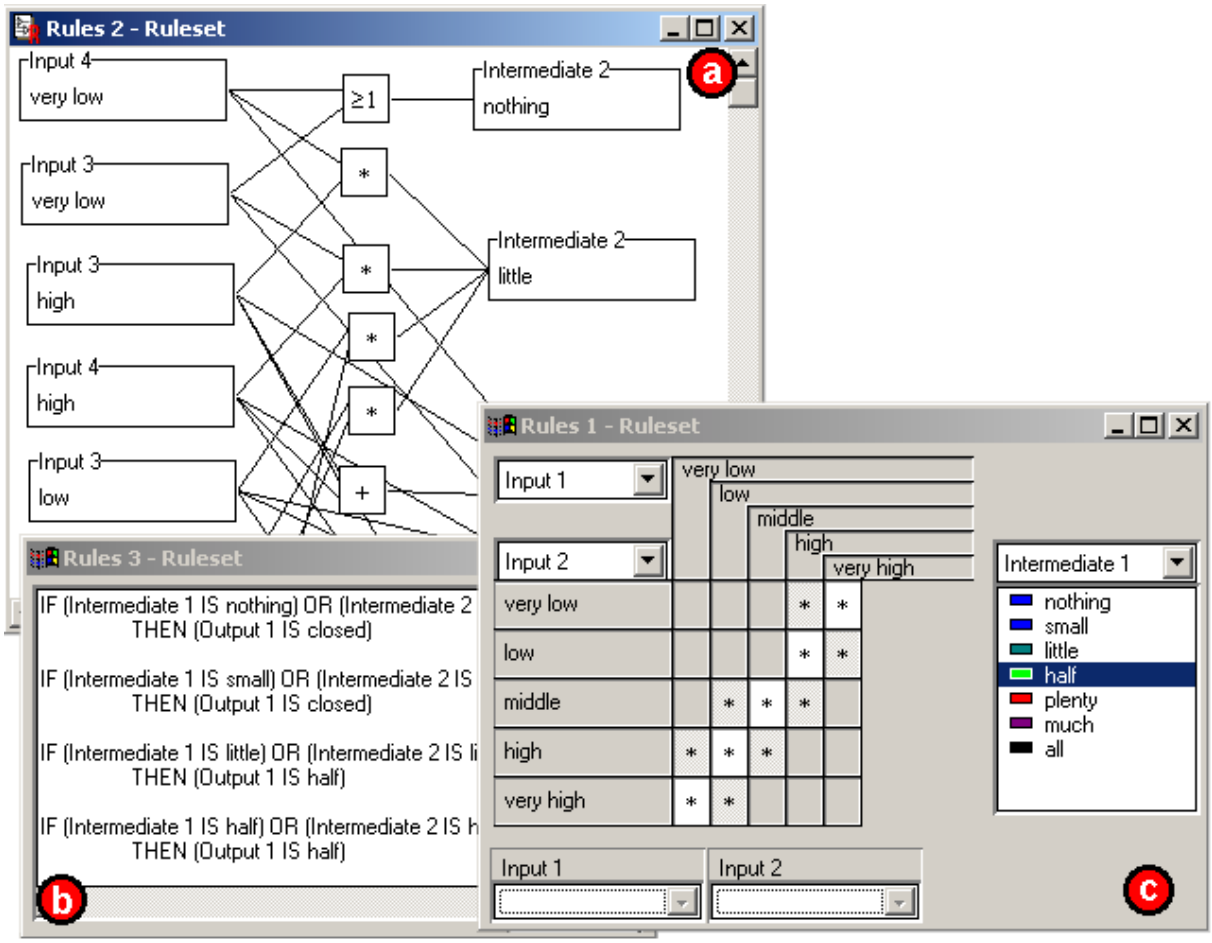
modely neumí dále pracovat. Uživatel přitom na tuto skutečnost není nikdy upozorněn. Operátor si může nadefinovat velké množství jazykových proměnných, které se samy podle zapojení mění na vstupní, výstupní či mezilehlé. Přidávání a odebírání prvků je velmi snadné a provádí se buď pomocí hlavního menu nebo pomocí myši. Propojování jednotlivých částí systému se také provádí pomocí myši. Přípustnost takto vzniklého zapojení je programem okamžitě kontrolována a v případě neshody je uživatel varován nebo přímo opraven. Výsledná struktura je zobrazena velmi přehledně pomocí jasně odlišených bloků (viz Obr. 4.4 – č.1). Ty jsou pospojovány čarami podle toho, jak na sobě závisí. V systému není žádný prvek, který by neměl viditelnou grafickou reprezentaci. Drobnou nevýhodou je absence zarovnání na mřížku a lomených nebo zaoblených čar. Bloky jsou spojeny prostě úsečkami. To samozřejmě u větších projektů zapříčiňuje horší přehlednost. Citelným nedostatkem je nemožnost pracovat současně s více bloky. Pokud chce například uživatel přesunout skupinu jazykových proměnných na jiné místo, musí tak učinit u každého prvku zvlášť. Vstupní a výstupní body systému jsou jednoznačně určeny vstupními a výstupními jazykovými proměnnými. U čtených nebo výsledných signálů není možné provádět žádné předzpracování či jiné úpravy.

Při přidávání nové jazykové proměnné musí uživatel vždy vyplnit její jméno (kontrola jedinečnosti jmen je prováděna automaticky). Program nenabízí žádné defaultní pojmenování, což dosti zdržuje. Dalšími nastavitelnými parametry, které jsou již předvyplněny, jsou jednotka a definiční rozsah (universum). Po této proceduře se objeví prázdné editační okno (Obr. 4.4 – č.2), do kterého je nutné vložit žádaný počet funkcí příslušnosti. K dispozici jsou tři typy. Trapezoid, singleton a obecná lomená čára (fuzzy množina může být subnormální). Omezující je, že všechny křivky v jedné jazykové proměnné musí být stejného druhu. Aplikace neumožňuje hromadné přidávání fuzzy množin a nově přidané nemají ani vhodný tvar. Jedinými usnadňujícími nástroji jsou automatické rovnoměrné nebo proporcionální rozmístění existujících termů. Funkce příslušnosti jsou zobrazeny pouze graficky. Navíc ještě nepříliš kvalitně. Popisy hodnot na obou osách jsou naprosto nepostačující. Jednotlivé křivky jsou sice barevně odlišeny, přesto i při malém počtu funkcí začíná být graf nezřetelný. Právě vybraný term je zvýrazněn jen částečně. Uživatel má pouze omezené možnosti, jak zjistit přesné číselné parametry křivek. Z těchto důvodů není editace tvaru existujících funkcí příliš přehledná, a to i přesto, že se provádí jednoduše táhnutím myši. Pozitivní je, že veškeré změny provedené v tomto dialogu se promítají i do ostatních částí vývojového prostředí.

Velmi netradiční je velký počet možných logických spojek a operací s fuzzy množinami. K dispozici jsou například AND, OR , NOT, AVG, PRODUCT, SUM, GAMA a jiné. U každého pravidla může být použita jiná kombinace těchto několika operátorů. Bohužel nikde není popsáno, co který přesně provádí, natož aby se daly zvolit způsoby jejich výpočtu (viz kapitola Operace s fuzzy množinami).

Další zajímavostí tohoto produktu je způsob vytváření, editace a hlavně zobrazení pravidel. K dispozici jsou celkem čtyři typy reprezentace. Bohužel po počátečním zvolení jednoho z nich nelze přejít na žádný jiný. To dosti snižuje použitelnost daného řešení. Další nevýhodou je, že ani v jednom z nich není možné automaticky vygenerovat jakoukoliv část stavového prostoru. První způsob zobrazení je klasická tabulka pravidel (Obr. 4.4 – č.3). Její vyplňování a editace je snadná. Uživatel si nemusí pamatovat žádná jména fuzzy množin ani jazykových proměnných. Stačí pouze vybírat potřebné prvky z nabízených množin. Také mazání je bezproblémové. Drobným nedostatkem je nemožnost libovolného prohazování jednotlivých řádků tabulky. Aplikace umožňuje pouze automatické setřídění podle vstupů či výstupů. Druhým způsobem zobrazení pravidel je prostý text (Obr. 4.5 b). Zde si již uživatel musí veškerá jména pamatovat a dbát na správnou syntaxi. Samotný textový editor je velmi kvalitní a jako jediná část programu poskytuje funkci „Zpět“. Po dokončení práce s pravidly je nutné nechat aplikaci celý kód zkontrolovat. Třetí způsob zobrazení pravidel je pomocí

matice (Obr. 4.5 c). Ta je poněkud odlišná od běžně používaných. Na jednotlivých stranách jsou sice vstupní proměnné a jejich fuzzy množiny, prvky ale nejsou funkce příslušnosti výstupní matice, ale rozložení jednotlivých logických spojek pro vybraný závěr pravidla (term). Takové zobrazení je poněkud nezvyklé, ale má svou vlastní vypovídací hodnotu. Vytváření a editace matice je snadné. Stačí pouze pomocí kontextového menu vyplnit jednotlivá políčka. Asi nejneobvyklejší způsob reprezentace pravidel je grafická síť obsahující na jedné straně termy vstupních jazykových proměnných a na straně druhé termy výstupních jazykových proměnných. Mezi nimi je spleť spojnic a operátorů. Vše je ukázáno na obrázku Obr. 4.5 a. Z něj je vidět, že tento zajímavý způsob může být použit pouze pro jednodušší pravidlové bloky. Grafické ztvárnění a ovládání má stejné nedostatky jako schéma celého systému. Navíc ne zcela správně v tomto bloku funguje propagace změn provedených v jiných částech aplikace. Ne zřídka se tak stane, že bloky „záhadně putují“ po pracovní ploše. Celkově je to ale velmi zajímavý nápad jiného zobrazení pravidel.



Obr. 4.5: Některé způsoby zobrazení pravidel v programu Fuzzy Control Manager

4.3.3. Ladění a výstupní data fuzzy systému

Fuzzy Control Manager obsahuje velkou škálu nástrojů určených k ladění a testování vytvořených fuzzy systémů. Vstupní data používaná k testování se dají generovat několika způsoby. Jednak to mohou být reálná (on-line) data získávaná z jiných programů nebo vnějších zařízení. Spojení je možné přes DDE, DLL a vložené ovladače k externím zařízením. V demo verzi nebyl bohužel žádný z vyjmenovaných způsobů povolen. Druhou skupinu vstupních signálů tvoří off-line data. Posloupnosti hodnot se načítají buď ze souboru nebo se generují pomocí interního data-generátoru. Ten ale umožňuje vytvářet pouze lineární

posloupnosti. Dalším drobným nedostatkem je absence ručního generátoru, kde by se daly vytvářet libovolné posloupnosti vstupních signálů.

Zobrazovat lze všechny průběhy výstupních signálů. K dispozici jsou dva nástroje: *Data Writer* a *Time Data Writer*. V prvním z nich se posloupnosti zobrazují bez měřítka na obou osách. Je ale možné je v omezené míře zvětšovat, zmenšovat, roztahovat na celou obrazovku a ukládat do souboru. Veškeré ovládání je realizováno pouze pomocí tlačítek a je dosti nepružné. Pokud je provedeno více měření, jsou získaná data řazena za sebe. Ve druhém zobrazovacím okně je již popsána alespoň vodorovná osa – je na ní napevno absolutní čas měření (relativní čas nelze nastavit). Naměřené průběhy nelze ani zvětšovat ani ukládat. Pokud je provedeno více pokusů, pak je každé měření zobrazováno zvlášť. Takové uspořádání je velmi vhodné pro porovnávání jednotlivých měření. Pokusy sice nelze zobrazit současně do jednoho grafu, dá se ale mezi nimi poměrně snadno přepínat. V editoru vstupních jazykových proměnných je během simulace vidět aktuální hodnota vstupu, nikoliv však stupeň příslušnosti k daným fuzzy množinám. V editoru výstupních jazykových proměnných je zobrazeno již obojí – jak aktuální hodnota výstupu, tak stupeň příslušnosti k fuzzy množinám. Bohužel není možné ani u vstupních ani u výstupních jazykových proměnných prohlížet nebo ukládat naměřená data jako časové posloupnosti. Grafické i číselné hodnoty míry splnění jednotlivých pravidel jsou k dispozici pouze u klasické tabulky pravidel. Při zvolení ostatních způsobů zobrazení pravidel není tyto informace možné získat. Vliv dvou vybraných vstupů na jeden výstup lze zjistit pomocí nástroje 3D graf. Po grafické stránce není příliš kvalitní. Umožňuje vytvářet pouze „drátěnou“ kostru spočtené nelineární plochy. Ovládání (rotace a nastavování zobrazovaných proměnných) je opět dosti těžkopádné, přesto je výsledné zobrazení poměrně přehledné. Celkově řečeno, k dispozici jsou rozmanité monitorovací a ladící prvky. Pro optimální ovládání a plné využití jejich funkcí by ale měly být lépe zpracovány.

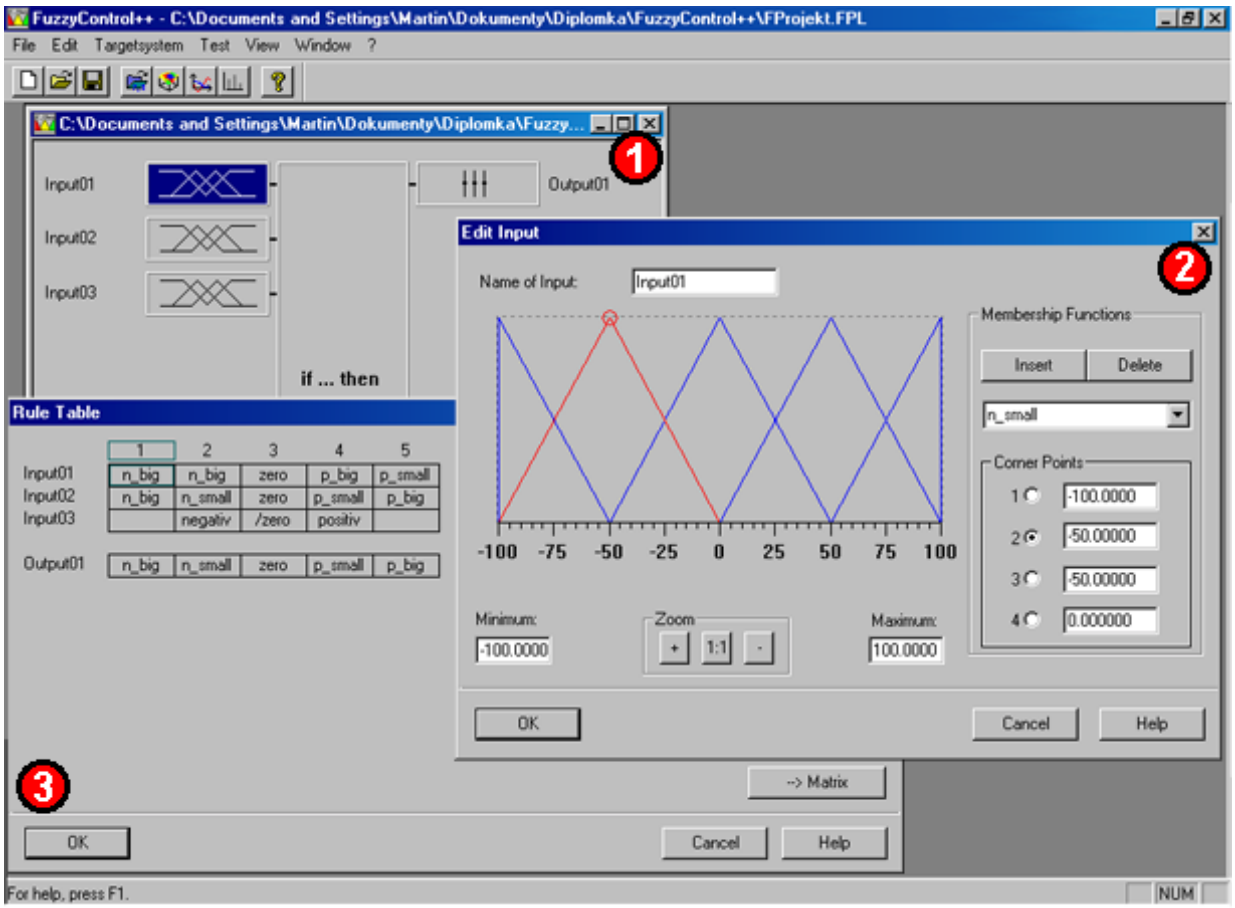
Výstupní data je možné generovat ve formátu zdrojového kódu jazyka C nebo zdrojového kódu pro některé druhy mikroprocesorů. K dispozici je také přímé spojení s programem MS Excel. Vytvořený projekt se defaultně ukládá do binárního souboru *.fp. Není ho tedy možné upravovat ani generovat z jiného programu než z Fuzzy Control Manageru. V této aplikaci sice není žádná speciální funkce určená přímo ke generování dokumentace, dá se ale poměrně snadno vytisknout jak tabulka pravidel, tak všechny funkce příslušnosti.

4.3.4. Celkové hodnocení

Fuzzy Control Manager je jedním z mála produktů, kde lze bez větších problémů vytvářet opravdu velké hierarchické fuzzy systémy. Má zajímavé možnosti zobrazení a editace pravidlového bloku. Celkový dojem dosti kazí nepříjemné ovládání, nekvalitní grafické zpracování (především monitorovacích prvků) a do konce nedotažená koncepce různých reprezentací pravidel. Nástroje pro ladění a generovaná výstupní data jsou postačující. Poměrně zajímavá možnost je ukládání a porovnávání několika měření. Stejně jako u pravidlových bloků by ale tyto nástroje potřebovaly zlepšit ovládání a celkově doladit.

4.4. FuzzyControl ++

Základní informace o testovaném programu	
Název	FuzzyControl++
Verze	Demo, Version 4.0
Operační systém	MS Windows 95 a výše
Datum vytvoření	1998
Výrobce	Siemens AG
Velikost	16MB (včetně NeuroSystems)
Druh	Komerční produkt
Stručný popis	Program je samostatný produkt, který slouží k vytváření a simulaci jednoduchých nehierarchických fuzzy systémů.
Maximální počty jednotlivých částí vytvořeného fuzzy systému	
Vstupy	8
Výstupy	4
Fce. přís. u vstupu	7
Fce. přís. u výstupu	9
Pravidla	200



Obr. 4.6: Typický vzhled pracovní plochy při práci s FuzzyControl++

FuzzyControl++ je komplexní, zcela autonomní program sloužící k návrhu a testování fuzzy systémů se zaměřením na systémy implementované pomocí PLC. Celá aplikace je koncipována jako MDI, myšleno ve stejném smyslu jako u předcházející aplikace. Negativní

vlastností této aplikace je časté používání modálních oken při zobrazování jednotlivých částí fuzzy systému. Výsledkem je například nemožnost současného otevření tabulky pravidel a editorů funkcí příslušnosti. To samozřejmě zhoršuje přehlednost celého systému. Na obrázku Obr. 4.6 je ukázáno hlavní okno aplikace, které obsahuje tři základní formuláře pro návrh fuzzy systému. Pod číslem 1 je zobrazen celkový náhled na strukturu fuzzy systému, editor funkcí příslušnosti je označen číslem 2 a editor pravidel číslem 3. Podrobnější informace o aplikaci lze nalézt v [17].

4.4.1. Obecný pohled

FuzzyControl++ má přehledné a dosti intuitivní ovládání, které respektuje veškeré obecně přijímané standardy používané v této oblasti. Přesto obsahuje některé nedostatky. Uživatel má například ve většině případů na výběr pouze ze dvou způsobů, jak danou operaci provést. Jedním je hlavní menu a druhým tlačítko toolbaru. Klávesové zkratky jsou zastoupeny pouze symbolicky a kontextová menu chybí úplně. Drobnou odchylkou od běžně používaného uspořádání je umístění ikon v některých oknech na pravý horní okraj místo na levý. Na tuto maličkost se ovšem dá snadno zvyknout. Zásadním nedostatkem je úplná absence funkcí „Zpět“ a „Opakovat“. Prostředí sice téměř neumožňuje upravovat si jednotlivé nástroje takzvaně na míru, poskytuje ale dostatečný komfort při nastavování a vyplňování parametrů. Není třeba si pamatovat téměř žádná jména, hodnoty editovaných proměnných se totiž často vybírají z nabízených možností. Uživatel až na malé výjimky nemusí vůbec použít klávesnici. Vše může ovládat pouze pomocí myši.

V případě nouze je k dispozici kvalitní manuál a help umožňující vyhledávání podle abecedy, podle témat a podle klíčových slov. Velmi pěkně jsou provedeny tématické nápovědy zobrazované přímo k aktuálně editovaným částem fuzzy systému. Program obsahuje také mnoho již vytvořených ukázkových projektů. Do detailů jsou provedeny i takové drobnosti jako tooltipy u jednotlivých ikon v toolbaru.

Z estetického hlediska je program proveden velmi pěkně. Ikony jsou čitelné a výstižné. Čárová grafika je čistá. Obzvláště povedený je 3D graf (viz níže). Až na výjimku při změně velikosti nikde žádné okno zbytečně nebliká. Hardwarové požadavky jsou vzhledem k náročnosti některých funkcí překvapivě malé.

4.4.2. Tvorba fuzzy systému

Aplikace umožňuje vytvářet jednoduché fuzzy systémy s jednou tabulkou pravidel a s poměrně omezeným počtem vstupů a výstupů. Jejich množství je ale kvůli nepřehlednosti velkých tabulek většinou postačující. Při vytváření nového systému uživatel zadává počáteční počet vstupů a výstupů. Ty jsou pak automaticky pojmenovány a připojeny k nové tabulce pravidel. Přidávání a odebírání existujících prvků je velmi jednoduché a realizuje se pomocí hlavního menu. Celkový pohled na strukturu fuzzy systému je ukázán na obrázku Obr. 4.6 - č.1. Zobrazení je velmi přehledné a snadno se pomocí něj odkrývají další „podčásti“ modelu. U vstupů i výstupů lze editovat jejich jméno (je kontrolována duplicita), ale není možné měnit jejich pozici na výše jmenovaném schématu. Vstupní ani výstupní signály nejsou žádným způsobem filtrovány ani jiným způsobem upravovány.

Jazykové proměnné se editují v modálním okně, které je na obrázku Obr. 4.6 pod číslem 2. Poměrně nemilým překvapením je pouze jeden možný typ funkcí příslušnosti. U vstupní jazykové proměnné to je funkce ve tvaru trapezoidu a u výstupní jazykové proměnné to je singleton. Žádné jiné typy nejsou k dispozici a uživatel si je ani nemůže dodefinovat. Po vytvoření nového fuzzy systému je nutné do prázdných jazykových proměnných přidat požadovaný počet funkcí příslušnosti. Dá se tak učinit najednou, stačí pouze zadat dané číslo. Automaticky se vygeneruje standardní rozložení, které se v mnoha případech již nemusí měnit. Také úvodní pojmenování je často postačující. Drobnou nevýhodou je pouze jedna

sada těchto jmen. Další přidávání a odebírání funkcí příslušnosti je velmi snadné a realizuje se pomocí dvou tlačítek. Jednotlivé křivky jsou reprezentovány především graficky. Zobrazení je sice pěkné a čisté, neobsahuje však barevné odlišení ani pojmenování jednotlivých funkcí příslušnosti. Jediná dobře rozpoznatelná křivka je právě ta vybraná. Další nevýhodou je poměrně nepohodlné zvětšování pouze pomocí tlačítek a nikoliv kurzoru myši. Druhou možností, jak současně zjistit přesný tvar funkcí příslušnosti, jsou číselné hodnoty parametrů. Ty jsou zobrazované vždy k aktuálně vybrané křivce a jejich význam je zřejmý (hlavně kvůli tomu, že je k dispozici pouze jeden typ f.p.). Editace tvaru a pozice funkcí příslušnosti je velmi snadná a realizuje se buď pomocí táhnutí myší nebo nastavením číselných parametrů. Oba způsoby jsou přehledné a poměrně komfortní. Nastavovat se dá také universum. Veškeré změny se automaticky promítají do dalších částí fuzzy systému.

Pravidla se editují pomocí modálního okna, které je na obrázku Obr. 4.6 pod číslem 3. Program neposkytuje žádnou možnost na počátku prázdnou tabulku nějakým způsobem automaticky vyplnit. K dispozici je reprezentace pravidel jak pomocí tabulky, kde trochu netradičně odpovídají jednotlivým pravidlům sloupečky, tak pomocí matice pokrytí, v níž je vidět závislost jedné výstupní jazykové proměnné na jedné nebo dvou vstupních, přičemž hodnoty ostatních vstupů se dají libovolně nastavovat. Obě zobrazení jsou velmi přehledná a umožňují snadnou editaci. V tabulce pravidel je jednoduchým způsobem možné vkládat, mazat a dokonce také přemisťovat jednotlivá pravidla. Při jejich editaci není nutné používat klávesnici, vyplňovaná jména funkcí příslušnosti jsou vždy nabízena, a to včetně svých negací. K dispozici bohužel není úprava váhy ani dočasná deaktivace pravidel. Není možné nastavovat ani žádné druhy fuzzy operací používaných při výpočtu výsledného výstupu. Zato je k dispozici parametr určující hodnotu každého výstupu, pokud není v danou chvíli pokryt žádným pravidlem.

4.4.3. Ladění a výstupní data fuzzy systému

Vytvořený systém se dá poměrně snadno testovat. Slouží k tomu několik zdařilých vzájemně propojených nástrojů umožňujících sledovat vývoj vnitřních proměnných fuzzy systému. Prvním z nich je graf závislosti jednoho výstupu na jednom až třech vstupech. Je zde použita velmi čistá grafika, pěkná paleta barev a netradiční černé pozadí. Jediným nedostatkem je malé zdůraznění orientace jednotlivých os. Přesto je graf dostatečně přehledný. Snadno se dají měnit jednotlivé proměnné i parametry zobrazení. Naprosto výjimečná je možnost nastavování hodnot ostatních nezobrazovaných vstupů, které samozřejmě mohou také mít vliv na výsledný graf. Autorům se dokonce podařilo uplatnit vliv třetího vstupu, který může v čase měnit svou hodnotu od minima k maximu a 3D graf sleduje jeho změny. Vzniká tak jakýsi 4D graf. Další radikální změnou v používání 3D grafu je dynamické přepočítávání a překreslování zkoumané závislosti v závislosti na aktuální hodnotě nezobrazovaných vstupů během časové simulace systému. Jakoby samozřejmostí je možnost nastavení automatické rotace grafu kolem jedné z os.

Druhým neméně zajímavým ladicím nástrojem je sloupcový graf zobrazující poměrně netradiční cestou jednak míru splnění jednotlivých pravidel a jednak jejich vliv na jednotlivé funkce příslušnosti sledovaného výstupu. Estetická stránka grafu je opět poměrně zdařilá a značnou měrou přispívá k celkové přehlednosti tohoto zobrazení. Po ukončení testů je možné nechat si vygenerovat statistiku, která nás informuje, jaká pravidla měla během simulace největší vliv na zkoumanou funkci příslušnosti daného výstupu.

Dalším velmi užitečným nástrojem pro testování fuzzy systému je časový graf. Spolu se simulačními nástroji tvoří základní nástroj pro zkoumání „dynamických“ vlastností systému. Na vstupní porty je možné přivádět buď vnější data z připojených PLC popřípadě WINCC (on-line), nebo data čtená ze souboru, popřípadě vnitřní signály generované pomocí předdefinovaných funkcí (off-line). Bohužel není možné nastavovat hodnotu vstupů ručně.

Časový graf je graficky asi nejhůře zpracovaným ladícím nástrojem. Přesto je dostatečně přehledný. Umožňuje snadné přidávání a odebírání křivek, měnění jejich pořadí a podobné základní operace. Bohužel není možné zobrazovat více jak čtyři proměnné současně. Navíc to mohou být pouze vstupy nebo výstupy. Není tedy možné zobrazit časové závislosti vnitřních veličin. Naměřená data je možné uložit do souboru. Jako ve většině případech, nevhodný se také jeví absolutní čas na horizontální ose. Jediným výrazným nedostatkem při ladění fuzzy systému je absence nástroje pro sledování stupně příslušnosti vstupních jazykových proměnných k jednotlivým fuzzy množinám.

Výsledný systém je možné exportovat do vybraných PLC Simatic S7-300 a S7-400. Naopak úplně chybí generování kódu v některém běžně používaném programovacím jazyce (C, C++, JAVA, …). Program neumí generovat ani zdrojové soubory používané v jakémkoliv z matematických simulačních programů. Chybí také převod do XML formátu. K dispozici nejsou žádné rozšířené funkce ani žádný komplexní nástroj pro tvorbu dokumentace. Celý projekt se ukládá do souboru *.fpl, který je textový. Dá se tedy snadno editovat, či přímo generovat v jiné aplikaci. Zajímavou možností je spuštění programu FuzzyControl++ přímo z prostředí Microsoft Word a jiných podobných produktů.

4.4.4. Celkové hodnocení

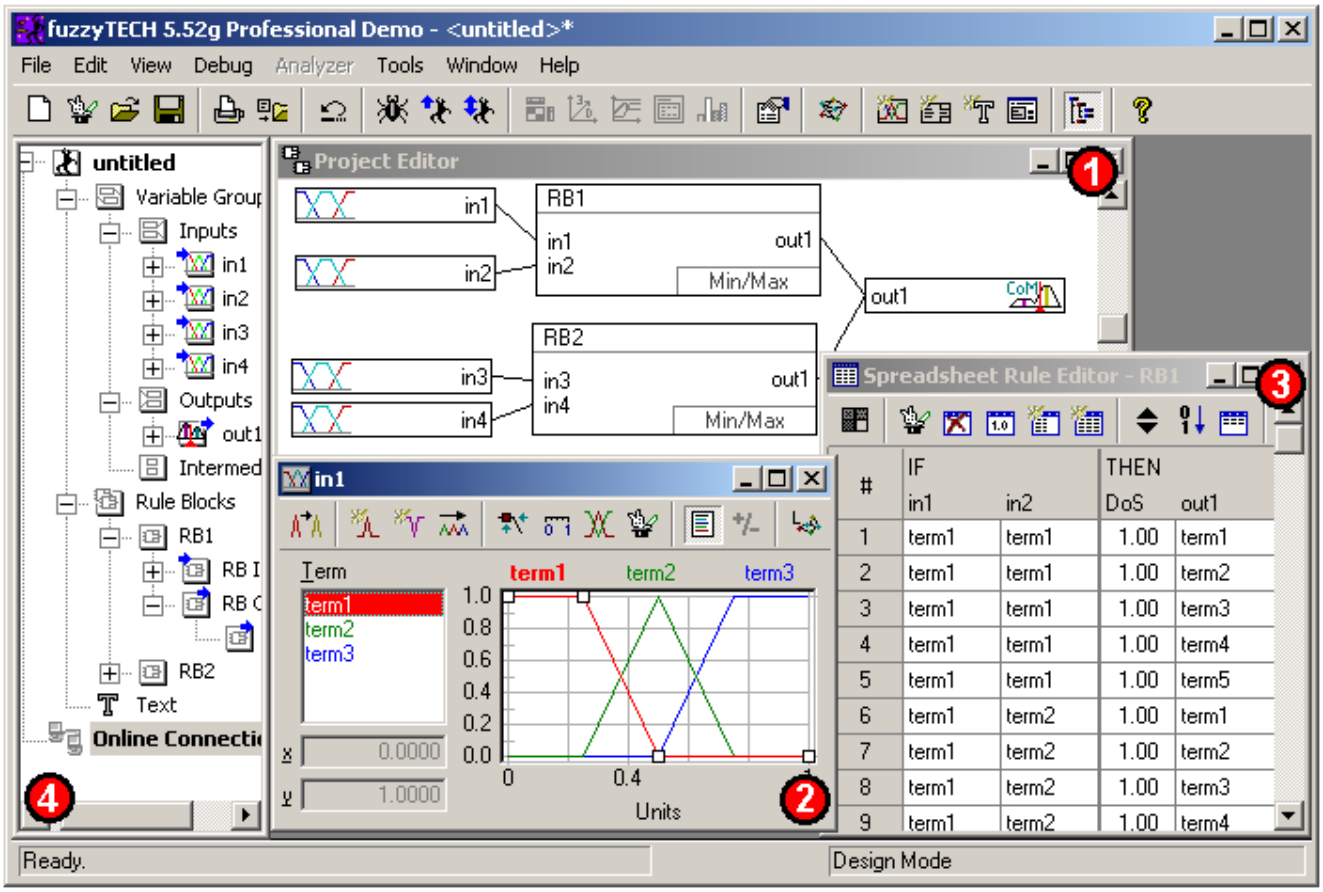
FuzzyControl++ je poměrně zdařilá aplikace umožňující tvorbu jednodušších fuzzy systému s omezeným počtem vstupů a výstupů. Celý program se snadno ovládá, je přehledný a poskytuje mnoho zajímavých ladicích nástrojů. Obzvláště zajímavý je dynamický 3D graf závislosti jednoho výstupu na jednom až třech vstupech. Škoda jen, že je tento produkt úzce specializován na programovatelné automaty PLC. K dispozici není žádný generátor kódu v nějakém programovacím jazyce ani spojení s jakýmkoli simulačním nebo jiným programem.

4.5. FuzzyTECH

Základní informace o testovaném programu	
Název	FuzzyTECH
Verze	5.52g Professional Demo
Operační systém	MS Windows 95 a vyšší
Datum vytvoření	2001
Výrobce	Inform
Velikost	26,7MB
Druh	Komerční produkt
Stručný popis	Komplexní program je samostatný produkt, který slouží k vytváření a simulaci hierarchických fuzzy systémů.
Maximální počty jednotlivých částí vytvořeného fuzzy systému	
Vstupy	255 (u jednoho pravidlového bloku maximálně 11)
Výstupy	255 (u jednoho pravidlového bloku maximálně 9)
Fce. přís. u vstupu	32
Fce. přís. u výstupu	32
Pravidla	Přesný počet není znám. Experimentálně ověřeno do počtu 100.

FuzzyTECH je v současné době asi nejpoužívanějším komerčním produktem pro návrh složitých i jednoduchých hierarchických fuzzy systémů. Slouží nejen k práci s klasickými fuzzy systémy, ale podporuje také učení ve fuzzy systémech a shlukovou analýzu. Celá aplikace je koncipována jako MDI ve smyslu jednoho hlavního okna, které ohraničuje pracovní plochu používanou všemi ostatními dceřinnými okny. Ve verzi, která byla

k dispozici, nebylo možné současně otevřít více projektů. Pokud tomu je i u ostré verze, pak je to poměrně zbytečné omezení. Jelikož nelze spustit více jak jednu kopii FuzzyTECHu, uživatel by tak vůbec nemohl na jednom PC pracovat s více projekty současně. Typický vzhled pracovní plochy je ukázán na obrázku Obr. 4.7. V okně s číslem 1 je zobrazena celá struktura fuzzy systému, editor funkcí příslušnosti je pod číslem 2 a editor pravidel má číslo 3. Užitečnou pomůckou při návrhu je i zobrazení celého projektu v podobě stromové struktury (okno číslo 4 – *Tree View*). Podrobnější informace o aplikaci lze nalézt v [18].



Obr. 4.7: Typický vzhled pracovní plochy při práci s FuzzyTECHem.

4.5.1. Obecný pohled

FuzzyTECH má dosti bohaté a přesto velmi přehledné ovládání. K dispozici jsou všechny běžné způsoby. Hlavní menu obsahuje standardně uspořádané všechny funkce, které jsou v aplikaci k dispozici. Přestože jich je velké množství, podařilo se autorům udržet jeho hloubku na přijatelné míře. Význam většiny položek je zřejmý na první pohled. Přesto se najdou i takové, u kterých je nutné použít nápovědu, aby se uživatel dozvěděl, čeho se týkají (např.: View/Lines….). Všechny základní nebo často používané příkazy mají svou klávesovou zkratku. Ty mají běžně používaný tvar, takže není problém si je zapamatovat. Dalším kvalitně provedeným ovládacím prvkem jsou četná kontextová menu. Jejich rozsah bohatě pokrývá všechny nejdůležitější operace při návrhu fuzzy systému. Vše výborně doplňují nápadité toolbary. Neomezují se pouze na hlavní okno aplikace, ale nacházejí se ve všech podstatných ovládacích oknech. Použité ikony jsou u standardních operací klasické a u specifických operací programu poměrně nápadité. Za zmínku určitě stojí pevný řád, který byl při jejich návrhu striktně dodržován. Pokud se například ikona týká učení ve fuzzy systému, obsahuje „L“, pokud způsobí přidání nějakého prvku do fuzzy systému obsahuje symbol „*“, atd. K přehlednosti celého projektu dosti přispívá možnost vkládání informačních textů na

pracovní plochu a komentáře psané přímo k jednotlivým blokům. Velkým kladem je bezchybná práce funkce „Zpět“. Funkce „Opakovat“ bohužel opět zcela chybí. Ovládání jednotlivých bloků schématu a práci s nimi se budeme věnovat v části o tvorbě fuzzy systému.

Přestože je ovládání intuitivní, je kvůli velkému počtu možných funkcí občas potřeba použít nápovědu. Ta je vytvořena velmi precizně. Obsahuje všechna důležitá témata seřazená jak tématicky, tak podle abecedy. K dispozici je vyhledávání podle názvů kapitol i fulltextově. Pro úplné začátečníky jsou připraveny stránky se základy fuzzy množin a fuzzy systémů. Vše je doplněno několika výstižnými příklady a podrobným postupem při návrhu fuzzy systému za použití FuzzyTECHu. Vhodným doplňkem je seznam všech klávesových zkratek a přímý dynamický odkaz na webové stránky FuzzyTECH Home Page. Velmi praktické se jeví generování nápovědy týkající se přímo objektu, s nímž uživatel právě pracuje. Pokud například upravuje tabulku pravidel, bude se vyvolaná nápověda týkat přímo tabulky pravidel. Propracovaný je i systém tooltipů. U ovládacích prvků dostatečně popisuje jejich funkci a u jednotlivých částí fuzzy systému odpovídá jejich popisu zadaném v parametrech bloku. Pro získání podrobnějších informací o některých částech fuzzy systému proto není někdy ani nutné daný blok otevírat.

Pracovní prostředí se dá snadno modifikovat tak, aby vyhovovalo každému uživateli. Lze upravovat viditelnost všech toolbarů, status baru, tloušťky čar, barvy některých prvků a dokonce i měřítko při zobrazování celého schémat systému. Samozřejmostí jsou funkce pro práci s otevřenými okny, jako jsou jejich různá řazení a podobně. Vhodným doplňkem je možnost automatického otevření a uspořádání všech editorů funkcí příslušnosti nebo všech tabulek pravidel. K usnadnění práce slouží mnoho „wizardů“ používaných pro poloautomatické vytváření některých nebo přímo všech částí fuzzy systému. Pokud je nutné někde nastavovat nějaké parametry, provádí se tak většinou pomocí prostého výběru jednoho prvku z nabízeného seznamu. Uživatel je tak ušetřen častému psaní.

Z estetického hlediska je program poměrně povedený. Ikony jsou zajímavé, tvary jednotlivých bloků systému jsou dostatečně odlišené, hojně jsou používány barvy. Grafika je čistá. Drobnou vadou je špatně zvolená transparentní barva u všech ikon. Pokud je ve Windows nastavena jiná než šedivá barva jako barva okrajů oken, vypadají tlačítka toolbaru nepěkně. Problémy jsou i s častým blikáním. To se projevuje například při překreslování 3D grafu nebo při změně velikosti všech oken. Hardwarové nároky jsou vzhledem k poskytovaným funkcím přiměřené.

4.5.2. Tvorba fuzzy systému

FuzzyTECH umožňuje vytvářet složité hierarchické struktury obsahující velké množství vstupů a výstupů. K dispozici jsou čtyři základní „stavební“ prvky: vstupní, mezilehlé a výstupní jazykové proměnné a pravidlové bloky. Všech těchto součástí může být v jednom systému velké množství. Při vytváření nového projektu může uživatel využít několik pomůcek pro usnadnění práce. Základní je takzvaný „Wizard“. Ten slouží k rychlému návrhu jednodušších systému. Operátor si zvolí počet použitých jednotlivých bloků, program za něj nastaví všechny základní parametry, přičemž mu umožní kdykoliv do návrhu vstoupit a upravit libovolné nastavení. Takto vygenerovaný základní model je ve většině případů velmi kvalitní a nevyžaduje u běžných systémů výraznější změny. Druhý způsob vytvoření nového systému je pouhé zadání počtu jednotlivých částí budoucího projektu. Od předcházejícího postupu se liší tím, že program neumožní ruční úpravu parametrů během generování nového modelu. Opět ale platí, že výsledný systém je velmi kvalitní a nemusí se příliš měnit. Poslední způsob vytváření nového modelu je postupné vkládání všech žádaných prvků. V tomto případě je zde ale malé omezení. Z neznámých důvodů nelze vytvořit nezapojený pravidlový blok, přestože při automatickém generování takové vznikají.

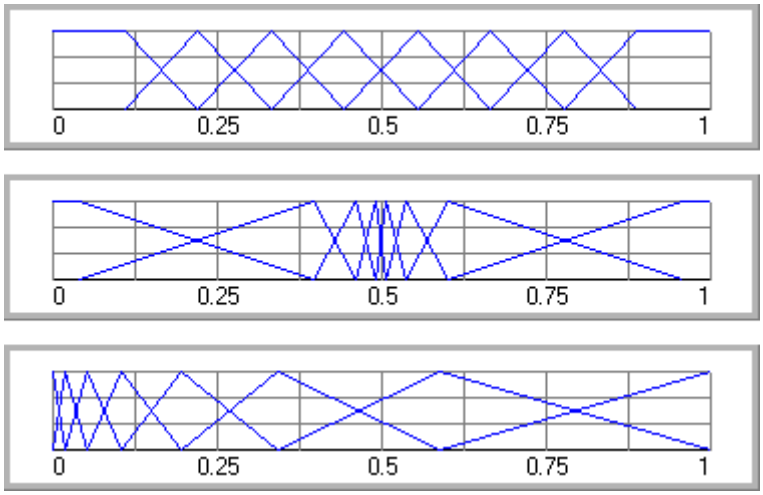
Celý fuzzy systém je zobrazen ve dvou podobách. Jednak je to blokové schéma (Obr. 4.7 - č.1), které dává přehlednou informaci o struktuře, a dále je to stromové schéma (Obr. 4.7 - č.4), kde jsou přehledně vidět všechny použité prvky. Obě podoby se výborně doplňují a jsou hojně využívány zejména při práci s většími projekty. Blokové schéma je sice názornější, má ale několik nedostatků. Jedná se zejména o způsob zobrazení a ovládání. Uživatel nemá možnost pracovat s více bloky naráz. Pokud chce tedy například přesunout celý projekt na jiné místo pracovní plochy, je nucen provést to prvek po prvku. Další nevýhodou jsou čáry reprezentující závislost jednotlivých bloků. Ty jsou pouhou přímkovou spojnicí vstupních a výstupních bodů bloků. Při větším počtu prvků se pak stává systém nepřehledný. Zcela určitě by pomohlo, kdyby čáry byly lomené (viz např. Simulink) nebo zaoblené (viz např. Stateflow), každopádně aby se dal upravovat jejich tvar. Mnohým uživatelům také může činit problémy samotná pracovní plocha, na níž jsou bloky umístěny. Ta má na počátku nepřiměřeně velké rozměry a navrhovaný systém se vyskytuje uprostřed. Pokud operátor popojede pomocí rolovátek mimo aktuálně používanou oblast, není úplně snadné vrátit se do původní polohy. Zásadním nedostatkem se jeví nezobrazování mezilehlých jazykových proměnných. Naopak lze kladně hodnotit dvě různé „aktivní“ oblasti na pravidlových blocích. Při dvojitém kliknutí na první z nich se objeví tabulka pravidel a při té samé operaci na druhou oblast se objeví obecné parametry pravidlového bloku. Oba dialogy jsou velmi často používané, proto je výhodné, že není třeba pro vyvolání ani jedné z nich požívat kontextové menu nebo jiný složitější ovládací prvek. Práce se stromovou reprezentací fuzzy systému je podobná práci s blokovým schématem. Na dvojitý klik se u odpovídajících prvků objeví stejný dialog, klávesové zkratky jsou obdobné, atd. Výhoda tohoto zobrazení je především v přehlednosti. Jednotlivé elementy systému jsou řazeny podle svého typu a jména. Zde jsou již zobrazeny opravdu všechny prvky vyskytující se v modelu (včetně mezilehlých jazykových proměnných). Drobný nesoulad obou reprezentací je ve zvýrazňování vybraných prvků. V každé z nich může být označen jiný prvek.

Editace vytvořené struktury je velmi snadná. Bloky se přesouvají pomocí myši nebo klávesnice. Přidávání nových vstupů, výstupů a pravidlových bloků se realizuje pomocí hlavního nebo kontextového menu, případně tlačítkem v toolbaru. Spojování a odpojování elementů se provádí jednoduchým výběrem předchůdců a následníků u pravidlového bloku. Dosti užitečnou funkcí je možnost kopírování již existujících bloků. Po každé operaci program automaticky kontroluje, zda je v souladu s pravidly návrhu a v případě chyby ihned upozorní nebo opraví uživatele.

Jazykové proměnné ve FuzzyTECHu často splývají se vstupními a výstupními proměnnými. Jejich vkládání do systému bylo popsáno v předchozím odstavci. Při vytváření nové jazykové proměnné si uživatel může vybrat z mnoha druhů vstupních, výstupních a mezilehlých. Jejich podrobný popis je nad rámec této práce a dá se zjistit v manuálu k testované aplikaci. Hlavní rozdíl mezi nimi je v použité implementaci fuzzifikace (on-line, off-line, …) a metodě defuzzifikace. Lze vytvořit dokonce jeden druh používaný pouze k monitorování vnějších signálů (bez dalšího zpracování ve fuzzy systému) a jeden druh určený ke generování výstupní hodnoty bez souvislosti s navrženým fuzzy systémem. Zvolený druh jazykové proměnné již není možné v budoucnu měnit. Po výběru typu funkce příslušnosti je opět k dispozici „Wizard“ s obdobnými vlastnostmi jako u generování celého systému. Nastavovat lze úvodní počet a pojmenování funkcí příslušnosti, universum, použité jednotky a barvu nové proměnné. Takto vytvořené jazykové proměnné se editují v okně ukázaném na Obr. 4.7 pod číslem 2. Funkce příslušnosti jsou dvou typů: L-Shape (lomená čára) a S-Shape V obou případech může mít křivka maximum menší než je maximum oboru pravdivostních hodnot. Mezi oběma typy lze kdykoliv v průběhu návrhu přecházet. Menším nedostatkem u některých projektů může být absence typu „Singleton“. Termy jsou přehledně

graficky zobrazeny pomocí barevně odlišených křivek. Právě vybraná funkce příslušnosti je dobře označena a u důležitých bodů je zobrazena také jejich přesná číselná hodnota. Modifikace všech parametrů křivek (barva, jméno, typ, zarovnání na mřížku, …) je velmi snadná. Tvar se mění pomocí táhnutí myší. Veškeré změny se ihned promítnou ve všech částech fuzzy systému.

Za povšimnutí také stojí množství poloautomatických operací, které lze s funkcemi příslušnosti provádět. Jsou to například inverze, převod do defaultního nebo uživatelem nadefinované rozmístění. Při tomto rozmístění si operátor volí například rozsah krajních funkcí příslušnosti a nelinearitu celého rozestavení. Některé příklady automaticky generovaného postavení křivek jsou na následujícím obrázku.



Obr. 4.8: Příklady automaticky generovaných rozmístění funkcí příslušnosti.

Pravidla určující závislost výstupních proměnných na vstupních se definují v pravidlových blocích. Jejich přidávání je popsáno v úvodu této kapitoly. Při vytvoření nového je opět k dispozici výkonný „Wizard“. Během několika málo kroků je automaticky (v případě nutnosti i ručně) nastavena většina potřebných parametrů. Pravidla mohou být zobrazena dvěma způsoby. První je ve formě tabulky (Obr. 4.7 - č. 3) a druhý je ve formě matice. Mezi oběma tvary lze kdykoliv přecházet. Tabulku lze automaticky vyplnit všemi kombinacemi vstupů a výstupů (generují se celá pravidla) nebo všemi kombinacemi pouze vstupů (generují se pravidla bez závěrů) nebo pomocí speciálního nástroje, kde si určíme vliv jednotlivých vstupů na dané výstupy (generují se celá pravidla). Ruční přidávání, mazání a editování pravidel je velmi snadné. Provádí se většinou jen za použití myši. Řádky bohužel není možné libovolně prohazovat. Program pouze umožňuje automaticky seřadit pravidla podle jednotlivých proměnných. Při návrhu je okamžitě kontrolována duplicita. U pravidel lze nastavovat jejich váhu. K dispozici ale nejsou ani vložené komentáře do tabulky ani možnost deaktivace pravidla (bez jeho vymazání). Pro přehlednější orientaci ve velkých tabulkách může uživatel použít zjednodušený tvar pravidel. V tomto módu ovšem není možné pravidla upravovat.

V matici pravidel odpovídají řádky a sloupce fuzzy množinám vstupních jazykových proměnných a jednotlivé prvky matice fuzzy množinám jedné výstupní jazykové proměnné. Zobrazení lze modifikovat tak, aby byly výstupní fuzzy množiny reprezentovány buď svým jménem nebo zvolenou barvou. Obzvláště efektní je nastavení, kdy jsou zvoleny různé odstíny téže barvy. Matice pak vypadá například jako na obrázku Obr. 4.9. Zkoumané jazykové proměnné lze velmi snadno měnit. Ostatní (nezobrazované) vstupy lze nastavit na libovolnou hodnotu. Lze tak pokrýt celý stavový prostor.

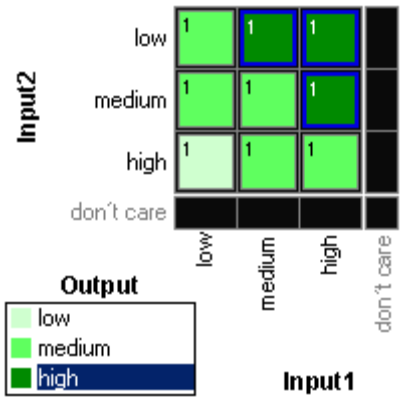
Další užitečnou pomocí při návrhu pravidel je velká volnost ve výběru použitých fuzzy operátorů (viz kapitola: Operace s fuzzy množinami). Pro spojování výstupů pravidel lze zvolit maximum nebo omezený součet. Pro spojování vstupů pravidel lze zvolit fuzzy operace jako maximum, minimum, průměr, gama a jejich kombinace. Matematické definice jednotlivých operací jsou uvedeny v následujících vztazích. Význam symbolů: μ_i je stupeň příslušnosti do i-té vstupní jazykové proměnné, λ a γ jsou parametry a n je počet vstupních jazykových proměnných. Podrobnější informace lze nalézt v nápovědě k testované aplikaci.

$$\mu_{Min-Max} = (1-\lambda)\min_{i=1..n}(\mu_i) + \lambda \max_{i=1..n}(\mu_i)$$

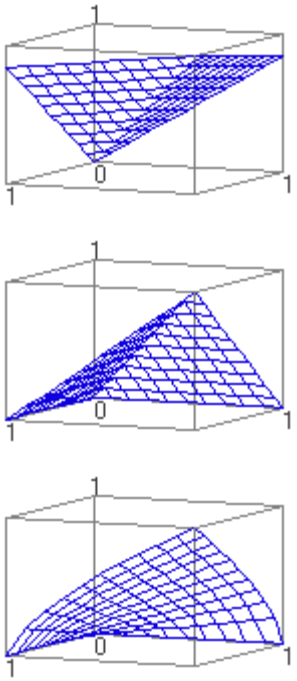
$$\mu_{\min-Avg} = (1-\lambda)\min_{i=1..n}(\mu_i) + \lambda \sum_{i=1}^{n} \mu_i / n$$

$$\mu_{gama} = \left(\prod_{i=1}^{n} \mu_i\right)^{1-\gamma} \left(1 - \prod_{i=1}^{n}(1-\mu_i)\right)^{\gamma}$$

Tvar „přenosové funkce“ (závislost výstupu na vstupech) si lze v aplikaci jednoduše prohlédnout a pro některé zvolené parametry je ukázán na obrázku Obr. 4.10. Shora to jsou $\mu_{Min-Max}$ ($\lambda = 0.8$), $\mu_{Min-Avg}$ ($\lambda = 0$) a μ_{gama} ($\gamma = 0.4$).



Obr. 4.9: Maticová reprezentace pravidel



Obr. 4.10: Některé možné operace se vstupy

4.5.3. Ladění a výstupní data fuzzy systému

Pro ladění a monitorování poskytuje FuzzyTECH velké množství kvalitních nástrojů. Vstupní data mohou být získávána on-line i off-line. Data posílaná na vstupy fuzzy systému mohou být generována například ručně, čtena ze souboru nebo sériové linky, popřípadě získávána přímým spojením s okolními programy nebo periferiemi počítače (přes DDE, sdílené soubory, sériovou linku, Motorola SDI, TCP/IP a jiné). Trochu možná chybí nějaký vestavěný generátor vstupních signálů. Tento problém se se současnými nástroji musí řešit

vygenerováním a uložením posloupnosti vstupních signálů mimo FuzzyTECH a následným načtením takto vytvořeného souboru.

Naměřená data se mohou buď ukládat do souboru, nebo zobrazovat přímo v aplikaci. Mezi základní nástroje patří časový graf. Ten umožňuje sledovat průběhy téměř všech měřitelných veličin vyskytujících se ve fuzzy systému. Jsou to především vstupy, výstupy, ale také míry splnění pravidel a stupně příslušnosti signálů do zvolených fuzzy množin. Samotný graf je proveden dostatečně přehledně. Snadno se mění počet a druh zobrazovaných proměnných a barva jim příslušejících křivek. Změna měřítka se nastavuje již o něco hůře. Provádí se pouze pomocí volby v toolbaru, takže není možné zvětšit libovolné místo grafu, ale pouze poslední naměřené hodnoty. Drobným nedostatkem je také absence popisu vodorovné osy při některých způsobech simulace (např. ruční simulace).

Druhým často používaným nástrojem je 3D graf. Ten zobrazuje závislost vybraného výstupu na jednom nebo dvou vstupech.. Toto zobrazení nelineární plochy není pouze statické, ale mění se v závislosti na nezobrazených vstupech. Při změně zobrazených vstupů se vykresluje do 3D grafu trajektorie, po níž se pohybují. Vlastní grafické provedení grafu je pěkné a přehledné, ale pouze do chvíle, než se začne graf měnit (vyvíjet, rotovat atd.). V takovém případě celé zobrazení dosti bliká. Ovládání grafu není vysloveně špatné, je ale nutné si na něj zvyknout. Možná je změna vstupních i výstupních proměnných, kterých se graf týká, nastavení hustoty testovaných bodů, změna použité barevné palety, ruční nebo automatická rotace a skrývání či mazání trajektorie (viz výše). K dispozici není zvětšování a zmenšování grafu. Obdobné informace jako 3D graf poskytuje také tzv. *Transfer Plot*. Ten zobrazuje 3D graf promítnutý do roviny. Jedná se například o různé řezy.

Další nástroj pro monitorování stavu systému je analyzátor pravidel. Ten zobrazuje vliv pravidel na míru příslušnosti výstupu k jednotlivým fuzzy množinám zkoumané výstupní jazykové proměnné. Podobná informace je také přímo v tabulce pravidel. Zde je graficky i číselně znázorněna míra splnění jednotlivých pravidel. Zajímavou funkcí je statistika. Ta obsahuje například informace o tom, kolikrát bylo během celého měření dané pravidlo aktivováno, popřípadě jaká byla jeho nejvyšší míra aktivace. V editorech jazykových proměnných je vidět aktuální stav proměnné a číselné hodnoty míry příslušnosti do všech fuzzy množin. Ale pouze u výstupních jazykových proměnných je tato hodnota vidět také graficky v podobě sloupcových grafů.

Výsledný projekt se ukládá do textového souboru *.ftl. Je ho tady možné načíst, vytvořit nebo modifikovat v jiné aplikaci. Aplikace umožňuje generovat fuzzy systém ve formě zdrojového kódu formátu C, Java, m-soubor do Matlabu a FTR. Export je také možný do mnoha druhů PLC. Mezi mnoho dalších zajímavých funkcí patří například učení ve fuzzy systémech, shluková analýza a jiné.

K dokonalosti je dovedena možnost automatického generování dokumentace k projektu. Vytvořený dokument ve formátu MS Word obsahuje podrobný grafický i textový popis všech částí fuzzy systému. Další netradiční, ale při velkých projektech užitečný, nástroj je *Revision Control*. Slouží k udržování přehledu o důležitých verzích projektu, postupu práce a pracovnících, kteří na něm pracovali během jeho vývoje.

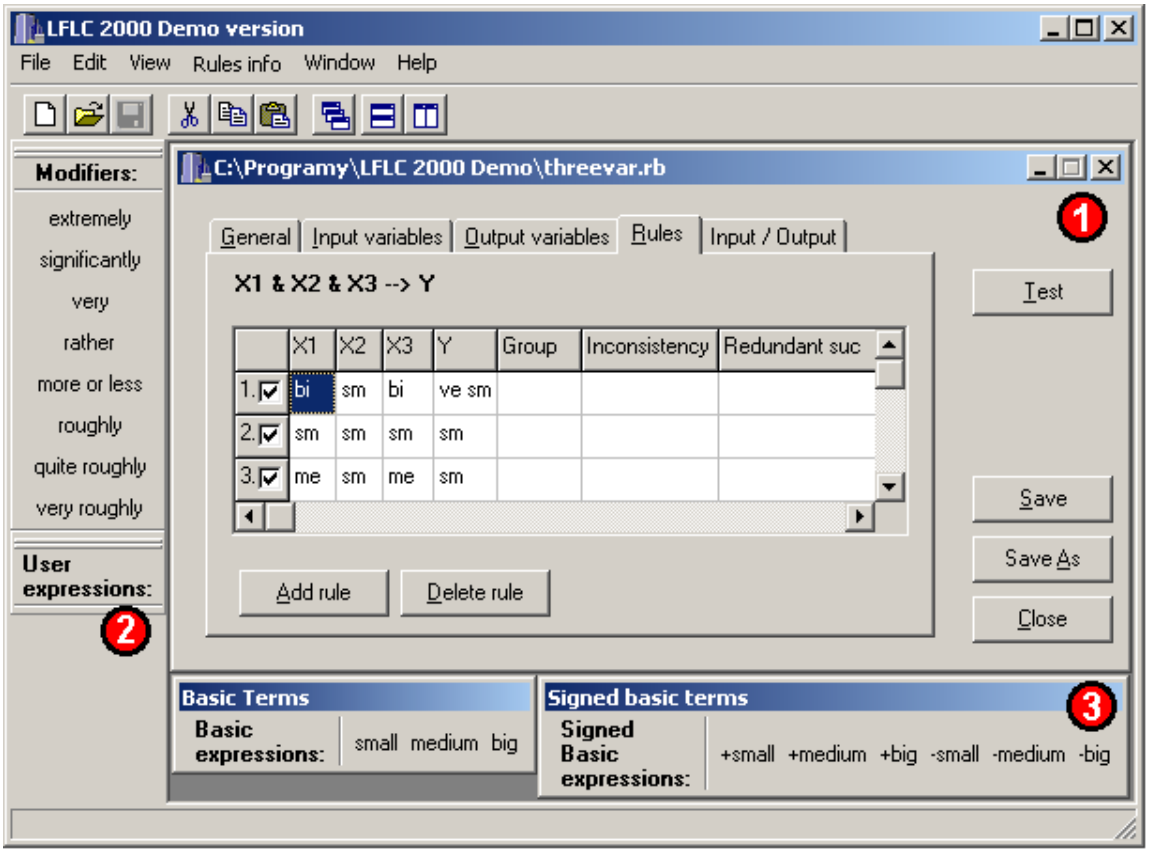
4.5.4. Celkové hodnocení

FuzzyTECH je zcela určitě nejkvalitnějším testovaným programem pro návrh fuzzy systémů. Jeho síla je především v jednoduchém ovládání a nepřeberném množství funkcí a možností, které poskytuje. Velmi kvalitně je provedeno automatické vyplňování položek (generování funkcí příslušnosti a pravidel), monitorování a ladění výsledného systému. Za zmínku také stojí možnost generování rozsáhlé dokumentace k projektu a podpora pro víceuživatelskou práci na jednom úkolu. Pokud má tato aplikace nějaké nedostatky, pak je to

ve zpracování a ovládání blokového schématu a v dosti velkém blikání obrazovky při každé změně pracovního okna. Ale žádná z těchto chyb není natolik závažná, aby výraznějším způsobem omezovala použitelnost aplikace. O kvalitách programu svědčí i fakt, že podle něj byla vytvořena norma pro aplikace zabývající se softwarovým návrhem fuzzy systémů. Její jméno je *IEC 1131-7 Fuzzy Logic Standard* a lze ji nalézt na oficiálních stránkách organizace *International Electrotechnical Commission (IEC)*, které jsou *http://www.iec.ch*.

4.6. LFLC 2000

Základní informace o testovaném programu	
Název	Linguistic Fuzzy Logic Controller 2000
Verze	Demo 0.1.7.10
Operační systém	MS Windows
Datum vytvoření	2002
Výrobce	Ostravská univerzita, Ústav pro výzkum a aplikace fuzzy modelování
Velikost	1,3MB
Druh	Akademický produkt
Stručný popis	Program je samostatný produkt, který slouží k vytváření a simulaci jednoduchých nehierarchických fuzzy systémů.
Maximální počty jednotlivých částí vytvořeného fuzzy systému jsou v beta verzi velmi odlišné od ostré verze, proto je zde neuvádím.	



Obr. 4.11: Typický vzhled pracovní plochy při práci s LFLC 2000

LFLC 2000 je akademický produkt sloužící k návrhu jednoduchých fuzzy systémů. Je koncipován jako MDI aplikace a umožňuje současně pracovat s několika projekty. Jeho struktura (Obr. 4.11) je poněkud odlišná od předchozích aplikací. Projekt je reprezentován

vždy pouze jedním oknem (číslo 1), který obsahuje několik záložek. Ty se týkají jednotlivých částí fuzzy systému (obecná charakteristika, vstupní a výstupní proměnné, pravidla a učení). Takovéto uspořádání sice šetří místem a udržuje přehled, která část patří ke kterému projektu, je ale nevhodné kvůli nemožnosti současného otevření více částí modelu.

Za zmínku také stojí samotná filozofie programu. Ten využívá velké množství automaticky vygenerovaných fuzzy množin a jim příslušných funkcí příslušnosti k tomu, aby uživatele oprostil od nutnosti jejich definování. Aplikace se pak chová, jako by rozuměla výrokům ve formě „rychlost je velmi velká a vzdálenost od překážky je více méně malá“. Nevýhoda tohoto postupu je, že ne vždy mohou defaultní fuzzy množiny vyhovovat. Jelikož operátor nic nedefinoval, nemusí mít správnou představu o významu jednotlivých jazykových proměnných a jejich modifikátorů. Podrobnější rozbor tohoto problému není náplní této práce. Zájemci mohou nalézt další informace v manuálu testovaného programu [19].

4.6.1. Obecný pohled

Aplikace LFLC se ovládá především pomocí funkčních tlačítek, která jsou k dispozici na každé záložce hlavního okna projektu. Jejich pojmenování a poloha dobře odpovídá jejich významu. Druhý způsob ovládání je pomocí hlavního menu. To je vzhledem k malému počtu funkcí programu nepříliš rozsáhlé. Uspořádáno je poměrně přehledně a dodržuje veškeré platné standardy. Drobnou nevýhodou je jeho neúplnost. Ani zdaleka neobsahuje všechny operace, které je možno se systémem provádět. Mnoho jich lze realizovat pouze díky výše jmenovaným funkčním tlačítkům. Mírně matoucí je také submenu Edit, které se celé věnuje pouze tabulce pravidel a to bez ohledu na to, zda je tabulka právě viditelná nebo ne. Při použití některé z položek pak nemusí být vůbec patrný výsledek operace. Třetím ovládacím prvkem jsou klávesové zkratky. Těch je v celém programu poměrně hodně. Jejich tvar odpovídá zažitým zvyklostem, takže není těžké si je zapamatovat. Ovládací prvek toolbar je zastoupen spíše jen v symbolické podobě. Obsahuje opravdu jen ty nejzákladnější funkce (Otevřít, Uložit, …). Vzhled a pořadí tlačítek je standardní. Nejméně využívána jsou kontextová menu. Až na jednu výjimku se v celé aplikaci vůbec nevyskytují. Výrazným nedostatkem je absence funkcí „Zpět“ a „Opakovat“. Celkově však lze říci, že manipulace s programem je sice nezvyklá, ale poměrně přehledná a po nějakém čase i lehce zvládnutelná. Pokud však do aplikace přibudou další funkce, nebude již možné pokračovat v nastaveném formátu ovládání pomocí tlačítek, protože by vše začalo být dosti nepřehledné.

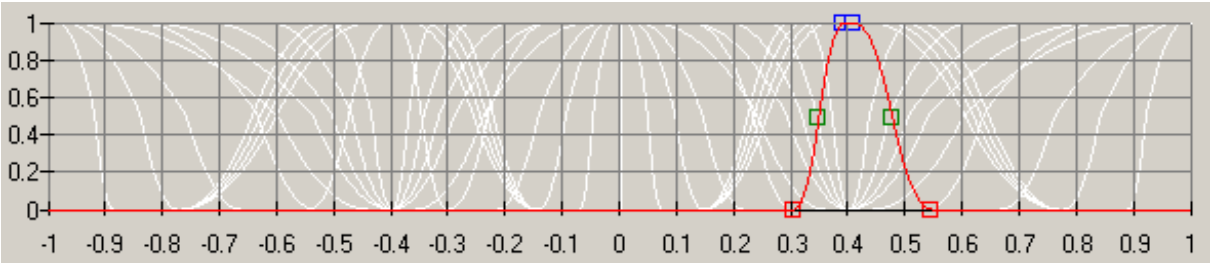
V aplikaci je pouze skromná nápověda. Má formu spíše manuálu, ale obsahuje všechny potřebné informace pro zvládnutí programu. Není v ní možné žádným způsobem vyhledávat nebo ji otevírat pouze k danému tématu. Vždy je nutné začínat v hlavním výběru a z něj se postupně propracovávat až k žádané informaci. Úplně chybí podrobný návod, jak od začátku vytvořit nový projekt. Tooltipy jsou převážně pouze v toolbaru. U ostatních tlačítek je operátor většinou odkázán pouze na jeho jméno nebo manuál. Spolu s testovací verzí je dodáváno i několik vzorových projektů, na nichž si uživatel může udělat představu o možnostech aplikace.

Při vyplňování parametrů je uživatel někdy nucen ručně vyplnit zcela zbytečně i ty položky, které by mohly mít nastavenou defaultní hodnotu. Jedná se například o jméno nové proměnné. Na druhou stranu při práci s tabulkou pravidel není nutné nic psát, stačí pouze vybírat nabízené položky. Škoda jen, že podobná zjednodušení nejsou všude v programu. Vývojové prostředí se dá jen minimálně upravovat. K dispozici je jen několik málo funkcí pro automatické uspořádání polohy otevřených oken nebo skrytí nepoužívaných výběrů. Z estetického hlediska je program proveden až příliš střídmě. Grafika je nevýrazná, nepoužívají se téměř žádné barvy. V mnoha případech navíc dochází k nadměrnému nepříjemnému blikání. Hardwarové nároky jsou zanedbatelné.

4.6.2. Tvorba a ladění fuzzy systému

Aplikace umožňuje pracovat pouze s jednou tabulkou pravidel. Celková struktura modelu je tedy velmi jednoduchá. Při vytváření nového projektu není nutné nic vyplňovat, ale uživatel nemá ani možnost ovlivnit, kolik bude mít vstupů, výstupů a podobně. Projekt v každé fázi svého návrhu musí mít alespoň jeden vstup, jeden výstup a jedno úplné pravidlo. I u nového systému je tato podmínka splněna. To je dosti omezující, protože v systému jsou tak části, které nejenže většinou musíme upravit, ale dokonce je ani nelze smazat. Vkládání nových vstupů a výstupů by byla velmi jednoduchá operace, kdyby uživatel při každém novém prvku nemusel ručně vyplňovat jeho jméno (duplicita je automaticky kontrolována). Mazání a nastavování parametrů jako je jméno, universum a diskretizace (hrubost proměnné) lze provádět dostatečně snadno. Přehlednost celé struktury je trochu omezena pouze jedním pracovním oknem, které může být k jednomu projektu otevřeno. Naštěstí je složitost systému velmi malá a tak pouhý seznam obecných parametrů, vstupů, výstupů a pravidel stačí.

Práce s jazykovými proměnnými je asi největší slabinou celého programu. Jsou vyjmenovány ve druhé (vstupy) a třetí (výstupy) záložce hlavního okna projektu. Jejich pořadí nelze měnit. Hlavním nedostatkem je editování funkcí příslušnosti. Těch obsahuje proměnná ihned po vytvoření velké množství (23). Jedná se o takzvané standardní fuzzy množiny a nelze měnit jejich počet, tvar ani pojmenování. Jejich podoba je ukázána na obrázku Obr. 4.12. Uživatel si kromě nich může nadefinovat ještě vlastní. K dispozici jsou tři typy: trapezoid, S-funkce a trojúhelník. První problém je se zobrazením funkcí příslušnosti. Standardní a uživatelem nadefinované fuzzy množiny nelze vykreslit do jednoho obrázku. Samotné zobrazení je velmi nekvalitní a nepřehledné. Křivky nejsou nijak odlišeny. V grafu mají všechny stejnou barvu a nejsou u nich jména. Jejich výběr je možný pouze z vedle zobrazeného seznamu. Pokud ale nevíme, jak se daná křivka jmenuje, musíme procházet celý seznam. Jedině právě vybraná funkce příslušnosti je dostatečně zvýrazněná. Editace tvaru křivek je možná buď v grafu táhnutím myší nebo nastavením číselných parametrů. První způsob je poměrně snadný, ale druhý činí značné problémy. Změněné hodnoty parametrů nejsou ihned zobrazeny v grafu, u standardních fuzzy množin dokonce lze jejich hodnotu měnit, ale po zavření a opětovném otevření editačního okna jsou veškeré změny ztraceny.



Obr. 4.12: Zobrazení fuzzy množin v programu LFLC 2000

Pravidla jsou uspořádána do přehledné tabulky (Obr. 4.11 – č.1). Ta na počátku obsahuje pouze jedno pravidlo. Nejsou k dispozici žádné funkce pro automatické vyplnění tabulky. Veškerá pravidla musí být přidávána ručně. Provádí se tak buď přímo vpisováním jmen jednotlivých fuzzy množin nebo pomocí výběru nabízených položek ze seznamů (Obr. 4.11 – č.2 a 3). Editace a mazání pravidel jsou také snadné operace. U pravidel lze nastavit jejich pořadí a aktivitu, nikoliv však váhu. Velmi zajímavými nástroji jsou hlídací funkce tabulky. Ty uživatele upozorní na výskyt opakujících se pravidel, nekonzistenci (ze stejných předpokladů plynou různé závěry) a přebytečná pravidla (pravidlo se stejným závěrem, ale užším předpokladem). Při výpočtu lze použít několik způsobů inference a defuzzifikace.

Pro ladění vytvořeného systému není k dispozici mnoho nástrojů. Jako vstupní data lze použít pouze ručně generované posloupnosti. K dispozici není žádný generátor, žádné načítání ze souboru, ani on-line propojení s okolím. U vstupů lze monitorovat jejich aktuální hodnotu a fuzzy množinu s největším stupněm příslušnosti. U výstupů lze sledovat výsledek infernce a jejich hodnotu a fuzzy množinu s největším stupněm příslušnosti. U pravidel je zobrazeno pouze, zda je aktivováno nebo nikoliv. Už ale není k dispozici informace o míře aktivace. Výše jmenované grafy nebo hodnoty jsou umístěny poměrně přehledně na jednom místě. Horší už to je s grafickým zpracováním, které není příliš kvalitní. Druhým nástrojem pro testování systému je 2D nebo 3D graf závislosti jednoho výstupu na jednom nebo dvou vstupech. Zde jsou nedostatky grafiky asi nejvíce patrné. Zcela chybí možnost zaznamenávání a zobrazování časových průběhů jednotlivých veličin. Naměřené hodnoty nelze uložit do souboru ani jiným způsobem archivovat. Mezi rozšířené funkce programu patří učení z dat.

Výsledný systém se ukládá do textového soboru se jménem *.rb. Lze ho tedy vytvářet či editovat v jiné aplikaci. Celý projekt lze pomocí COM technologie vložit do jiné aplikace. Možné je také použití speciálního bloku do Simulinku a S-funkce do Matlabu. Oběma se jako parametr dává výše jmenovaný soubor. Žádné jiné propojení s okolím není možné. Celý proces vývoje ani finální produkt nelze nijak dokumentovat a dokonce ani vytisknout.

4.6.3. Celkové hodnocení

LFLC 2000 je sice zajímavý program pro realizaci fuzzy systémů, má ale velké množství nedostatků. Mezi hlavní patří práce s fuzzy množinami, nedostatečné testovací a monitorovací prostředky a omezené možnosti propojení programu s okolím. Na druhou stranu obsahuje velmi zajímavé a užitečné kontrolní funkce v tabulce pravidel. Specifický je také přístup k samotné fuzzy logice a návrhu fuzzy systémů. Podrobnosti o těchto nových metodách lze nalézt v [10], [11] a [12].

4.7. Srovnávací studie vybraných produktů

Testované produkty spadají do dvou částečně odlišných skupin. První tvoří nesamostatné nástroje, které jsou pouhou součástí většího celku. Druhou tvoří autonomní aplikace, které ke své činnosti již nepotřebují žádné jiné programy. Obě skupiny mají svá specifika a je vhodné k nim přihlédnout. Ne všechny programy mají stejné použití, proto nelze jednoznačně stanovit jejich pořadí.

Oba nesamostatné nástroje mají velkou výhodu ve svém těsném spojení s matematicky mnohem výkonnějším programem. Pro jejich použití není potřeba generovat žádné dávkové soubory ani nastavovat parametry spojení. Mají ale poměrně velké nedostatky v ovládání a v grafickém zpracování. Problémy jsou také s testováním vnitřních proměnných. Oba produkty jsou určeny pro vytváření jednodušších nehierarchických fuzzy systémů. Porovnáváme-li pouze zástupce programů uvnitř této kategorie, vyjde jednoznačně lépe Fuzzy Logic Toolbox do Matlabu než do ECANSE (ten je celkově nejhorší z testovaných produktů).

Mezi všemi programy, tedy i samostatnými, výrazně převyšuje FuzzyTECH. Je nejlepší téměř ve všech ohledech: ovládání, možnosti testování, komunikace s okolím, použitelnost výsledných dat, vzhled, atd. FuzzyControl++ je také bez výraznějších problémů použitelný. Ostatní produkty nejsou sice vyloženě špatné, ale mají již většinou nějaký závažnější nedostatek. Přesto se i v nich občas najde nějaká funkce, která je něčím výjimečná, kterou ostatní programy nemají. Podrobný přehled hodnocení testovaných aplikací podle jednotlivých kritérií je uveden v příloze B.

Rozsah diplomové práce neumožňuje provést analýzu všech existujících programových nástrojů podporujících návrh fuzzy systémů. Výběr programů byl volen tak, aby pokrývaly co největší možné spektrum. Přesto nelze říci, že vybraný vzorek je reprezentativní pro všechny aplikace.

5. Optimální rozhraní vývojového prostředí pro návrh fuzzy systémů

Z výše uvedených příkladů programů pro návrh a simulaci fuzzy systémů je zřejmé, že navrhnout a hlavně implementovat kvalitní aplikaci, která by vyhovovala všem kritériím, je velmi obtížné. Předem je nutné stanovit, k čemu má daný produkt sloužit a jaké má mít základní vlastnosti. Optimální vývojový prostředek by měl umožňovat jednoduchým způsobem vytvářet malé, nestrukturované, ale i složité hierarchické systémy, měl by splňovat všechna výše jmenovaná kritéria. Proto některá z nich již nebudou dále explicitně zmíněna ale implicitně se bude předpokládat, že jim navrhované prostředí a jeho uživatelské rozhraní vyhoví. Program by se měl poučit z chyb vyjmenovaných testovaných produktů a inspirovat dobrými nápady.

5.1. Struktura programu

To, zda má být aplikace ve formě MDI nebo SDI ve smyslu jednoho hlavního okna a vnořených dceřinných oken (nikoliv projektů), nelze jednoznačně rozhodnout. Oba způsoby mají své výhody i nevýhody. Samozřejmostí musí být možnost současné práce na více projektech a vícenásobné spuštění aplikace. Při práci s okny je třeba dbát na to, aby nebyla zbytečně modální a neblokovala tak přístup k ostatním částem programu. Celé prostředí musí jít přizpůsobit různému rozlišení monitoru. U oken by se v rámci možností měla dát libovolně nastavovat jejich velikost a pozice.

5.2. Ovládání

Ovládání programu musí být jednoduché a přehledné. Mělo by obsahovat všechny možné ovládací prvky, aby si každý uživatel mohl zvolit ten svůj nejoblíbenější. Hlavní menu by nemělo být příliš hluboké, ale mělo by obsahovat veškeré podstatné operace, které lze v aplikaci realizovat. Diskutabilní je „mapování“ některých podmenu, která se mění podle aktuálně upravované části systému. Hojně by měly být používány vkusné a přehledné toolbary a kontextová menu. Samozřejmostí jsou standardní klávesové zkratky pro často používané funkce. Naprostou nezbytností jsou dobře pracující nástroje „Zpět“ a „Opakovat“. Hloubka historie by měla být dostatečně dlouhá. Při zpětném i dopředném kroku je nutné dát uživateli najevo, která část modelu se právě změnila. Vhodným doplňkem je textový nebo jiný seznam všech provedených kroků. Uživatel se pak v nich může mnohem snadněji orientovat.

Neodmyslitelnou částí aplikace je nápověda. Ta by měla umožňovat vyhledávání podle témat, klíčových slov, v rejstříku a fulltextově. Několik kapitol by mělo být věnováno obecnému popisu fuzzy systému, použitým metodám a samozřejmě ovládání celého programu. Velmi vítanou částí je podrobný popis postupu návrhu nového systému (takzvaně krok za krokem). Zajímavou možností by byl mód pro začátečníky, v němž by tento postup nebyl pouze staticky popsán, ale přímo by se realizoval v aplikaci. K dispozici by měly být příklady již existujících projektů pokrývající co nejlépe celou oblast možností aplikace. Nápovědu by mělo jít vyvolat přímo k libovolnému tématu. Všechna tlačítka a toolbary by měly obsahovat výstižné tooltipy. Jako doplněk k nápovědě by měl být manuál, který je určen spíše k sekvenčnímu čtení, než k vyhledávání. Samozřejmostí je identifikace autora produktu a možnost spojení na něj. Ideální je aktivní přímý odkaz na internetové stránky zabývající se aplikací.

Uživatel by při práci neměl být příliš zatěžován nutností vyplňovat parametry. O vše by se měl starat program. Nastavovat všechny defaultní hodnoty voleb tak, aby co nejlépe

vyhovovaly jak podmínkám aplikace, tak požadavkům uživatele. Při nastavování parametrů by se pokud možno mělo dát vyhnout použití klávesnice, protože to ve většině případů zdržuje. Pokud ale přeci jen je nutné něco psát ručně, měl by mít uživatel k dispozici nápovědu, jaké jsou platné hodnoty právě vyplňované proměnné. Vhodným doplňkem je automatické dokončování rozepsaného textu. Tato funkce, stejně jako ostatní funkce usnadnění, by měla jít podle potřeb deaktivovat. Prostředí by mělo jít upravovat podle požadavků uživatele. Jedná se například o volbu tlačítek v toolbarech, nastavení vlastních klávesových zkratek, aktivace či deaktivace používaných nástrojů, atd. Veškerá nastavení by měla zůstat stejná i při znovuotevření aplikace. Vzhled aplikace by měl být vkusný a dostatečně výrazný. Barvy by se měly používat přiměřeně. Grafy by měly být přehledné a čitelné, bez zbytečného blikání.

5.3. Tvorba fuzzy systému

Jelikož se předpokládá tvorba rozsáhlých fuzzy systémů, musí být jejich struktura zobrazena dostatečně přehledně. Vhodné je celý model rozdělit na víceméně nezávislé části, které reprezentují nějaký logický oddíl systému. Jedná se například o vstupy, výstupy, pravidlové bloky, atd. Každá taková komponenta se pak dá jednoduše reprezentovat blokem s danými vlastnostmi. Nabízejí se hned dva způsoby zobrazení struktury těchto bloků. První je blokové schéma a druhý je stromová struktura. Nejvhodnější je, pokud aplikace obsahuje obojí. Blokové schéma musí umožňovat libovolnou manipulaci jak se samotnými prvky, tak s jejich grafickou reprezentací. Musí v něm jít komponenty snadno přidávat, otevírat editační okna, kopírovat i mazat. Vzhled bloků různého typu musí být dostatečně odlišný (tvarem, barvou, …). Vhodné je, pokud lze nastavovat jejich velikost, barvu, použitý font, atd. Propojování a změna polohy by se mělo provádět jednoduše táhnutím myší. Samozřejmostí je možnost práce s více prvky současně. Pokud existují omezení v navrhované struktuře, měla by být v rámci možností patrná již na první pohled. Pokud například nelze nějaké prvky vůbec propojit, mělo by to být graficky znázorněno. Samotná pracovní plocha, na níž je schéma umístěno, by měla mít jen takový rozměr, aby poskytovala dostatečný prostor pro vytvořený systém, ne větší. Jako vzor pro ideální ovládání blokových schémat může posloužit například Simulink. Stromová struktura nedává informace o propojení jednotlivých prvků, ale přehledně shrnuje, které komponenty jakého druhu jsou použity, jak se jmenují, atd. Možnosti ovládání z tohoto prostředku by měly být stejné jako u blokového schématu. Otevíraná editační okna, klávesové zkratky i všechno ostatní by mělo být obdobné. Pěkný příklad kvalitní stromové struktury je ve FuzzyTECHu.

Při tvorbě vlastního systému by mělo být během vytváření každé nové části (i celého projektu) možno zvolit její základní parametry. Toto nastavování by uživatele nemělo zdržovat a mělo by se dát přeskočit = nastavení defaultních parametrů. U nové aplikace to konkrétně je počet vstupů, výstupů, pravidlových bloku, popřípadě jejich propojení, ale také používané metody výpočtu, atd. Přidávání nových bloků, jejich úprava a mazání by se mělo realizovat co nejjednodušeji. Vhodné je kontextové menu, otevření editačního okna po dvojitém kliku na danou komponentu a jiné. Možné je využití klávesových modifikátorů. Při stisku například CTRL by se na dvojitý klik mohlo otevřít jiné okno než bez použití této klávesy. Nutností je okamžitá kontrola správnosti vytvořeného schématu. Není přípustné, aby uživatel delší dobu pracoval na systému, který není realizovatelný.

Pro práci s jazykovými proměnnými musí aplikace obsahovat speciální nástroje. Nastavitelné musí být jméno, universum, popřípadě jednotky. Při vytváření nové proměnné je opět vhodné, pokud si uživatel může dané parametry zvolit. Navíc by měl jít nastavit úvodní počet, rozmístění, tvar a pojmenování fuzzy množin. Ty by měly být alespoň tří druhů: trapezoid, S-funkce a singleton. Je ale vhodné, pokud má uživatel k dispozici i jiné druhy,

nebo pokud si dokonce může nadefinovat své vlastní vzorové křivky (viz Matlab). Zobrazení funkcí příslušnosti musí být přehledné. To předpokládá jak grafickou reprezentaci pro názornost, tak číselné hodnoty pro přesnost. V grafu musí být viditelné všechny současně, měly by být barevně odlišeny a pojmenovány. Samozřejmostí je popsání obou os včetně jednotek, zvětšování libovolné části grafu a jiné základní operace. Vhodné je, pokud je ještě někde vedle seznam použitých fuzzy množin se stejnými jmény a barvami. Zvláště zvýrazněná by měla být vybraná křivka. Samotný výběr by měl být možný jak v grafu, tak v přilehlém seznamu. Editovat by měly jít všechny parametry fuzzy množin. Podstatné je zejména upravování tvaru a polohy funkcí příslušnosti. Realizovat by se mělo jednak pomocí myši a jednak pomocí nastavení přesných číselných parametrů křivek. Vhodným doplňkem je zarovnávání na nastavitelnou mřížku a různé nástroje pro automatické generování sady fuzzy množin s uživatelem definovaným rozložením (viz FuzzyTECH). Veškeré změny se musejí automaticky okamžitě propagovat do celého systému.

Pravidla by měla být zobrazena přinejmenším v podobě přehledné tabulky a matice. Je ale vhodné, pokud jsou k dispozici i jiné způsoby. Jako například blokové schéma (Fuzzy Control Manager) nebo seznam vět typu IF…THEN…, matematický zápis popřípadě symbolický (Matlab). Mezi jednotlivými reprezentacemi musí jít kdykoliv libovolně přecházet a v každé z nich upravovat daná pravidla. Při vytváření nové tabulky je opět užitečné, pokud je dána možnost automatického nebo ručního nastavení základních parametrů. Vhodné jsou generátory pravidel, které naplní tabulku pravidly podle potřeb uživatele (všechny kombinace vstupů, vliv vstupu na výstup atd.). Ruční přidávání a editování by mělo probíhat bez nutnosti psát názvy fuzzy množin. Ideální je výběr položek z nabízených možností. U pravidel by měla jít nastavit jejich váha a aktivita a pořadí v tabulce. Užitečnými nástroji by byly hlídací funkce pro kontrolu duplicity a nekonzistence pravidel. Matice pravidel by měla splňovat stejné podmínky pro ovládání jako tabulka. Zobrazené fuzzy množiny výstupní proměnné je vhodné reprezentovat jak jejich jménem, tak barevnou škálou. Matice pak vypadá jako pohled shora na 3D graf závislosti výstupu na vstupech.

5.4. Ladění a výstupní data fuzzy systému

Nástroje pro ladění a monitorování fuzzy systému poskytují největší volnost pro kreativitu softwarových návrhářů. Posloupnosti vstupních dat by se měly dát generovat ručně, speciálními generátory, načtením ze souboru nebo přímo on-line z jiného programu nebo periferie počítače. Zajímavou myšlenkou je také přivádění testovacích signálu doprostřed systému. Monitorovat by mělo jít časové průběhy všech myslitelných veličin: vstupy, výstupy, jazykové proměnné, stupně příslušnosti do jednotlivých fuzzy množin, míry splnění pravidel, atd. Časový graf by měl umožňovat libovolné zvětšování, nastavovaní barev a parametrů zobrazení. Umožňovat přizpůsobení velmi rozličných signálů pro zobrazení do jednoho grafu. Samozřejmostí je ukládání naměřených dat do souboru. Účelnou pomůckou může být statistické zpracování měření nebo možnost uchovávat a porovnávat v programu více výsledků simulace. U některých veličin je navíc vhodné, pokud jsou zobrazeny i jako sloupcový graf, popřípadě přímo číslem (míry splnění pravidel a stupně příslušnosti do fuzzy množin). Již tradičním nástrojem je 2D a 3D graf závislosti výstupu na jednom nebo dvou vstupech. Zobrazení by opět mělo poskytovat veškerý komfort při ovládání a upravování zobrazené plochy. Nutností je možnost nastavení ostatních (nezobrazovaných) vstupů do žádaného stavu. Výhodou je, pokud graf není jen statický, ale pokud se vyvíjí souběžně s měřenými daty. Jako nadstandardní, ale zajímavé, funkce mohou být například 4. rozměr v podobě v čase se periodicky měnící třetí vstupní proměnné nebo zobrazování aktuální polohy a trajektorie měřených vstupů.

Výsledný projekt by se měl ukládat do textového souboru s dokumentovanou strukturou. Nejvhodnější je přímo formát XML. Téměř nutností je generování zdrojového kódu v jazyce C a dávkového souboru pro Matlab nebo jiný matematický program. Vítané jsou generátory programů pro různá PLC a mikroprocesory. Velmi užitečné je, poskytuje-li program možnost přímé komunikace se svým okolím (DDE, DLL, COM, OPC, …). Samostatnou kapitolou je generování dokumentace. Ta by měla obsahovat veškeré informace o projektu: strukturu, fuzzy množiny a jejich funkce příslušnosti, tabulky pravidel a obecné nastavení aplikace.

6. Implementace uživatelského rozhraní vývojového prostředí

Uživatelské rozhraní, sledující doporučení kapitoly 5, bylo naprogramováno v prostředí Microsoft Visual Studio .NET a testováno na beta verzi programového nástroje podporujícího návrh, ladění a realizaci fuzzy systémů firmy ProTyS a.s. Z programovacích jazyků bylo zvoleno C#, což je poměrně nový, čistě objektový jazyk s podobnou strukturou, jako má například Java. Taková volba má řadu výhod, mezi něž například patří: vysoká kvalita a profesionálnost vývojového prostředí, přehledné a rozsáhlé knihovny funkcí pro vytváření aplikaci pro Windows 98 a výše, bezpečnost a „jednoduchost“ jazyka, atd. Výsledná aplikace je sice o něco náročnější na výkon počítače než podobný program vytvořený třeba v C++, ale je zase skromnější než například aplikace v Javě. Minimální požadavky pro spuštění programu jsou:

- Operační systém – Microsoft Windows 98 nebo vyšší.
- Microsoft Internet Explorer 5.01 (je volně ke stažení) nebo vyšší.
- Microsoft .NET Framework (je volně ke stažení)
- 64 MB RAM
- 5 MB místa na pevném disku.

Vytvořený program je modulární. Hlavní aplikace volá více či méně nezávislé funkce z přiložených dynamických knihoven *.dll.

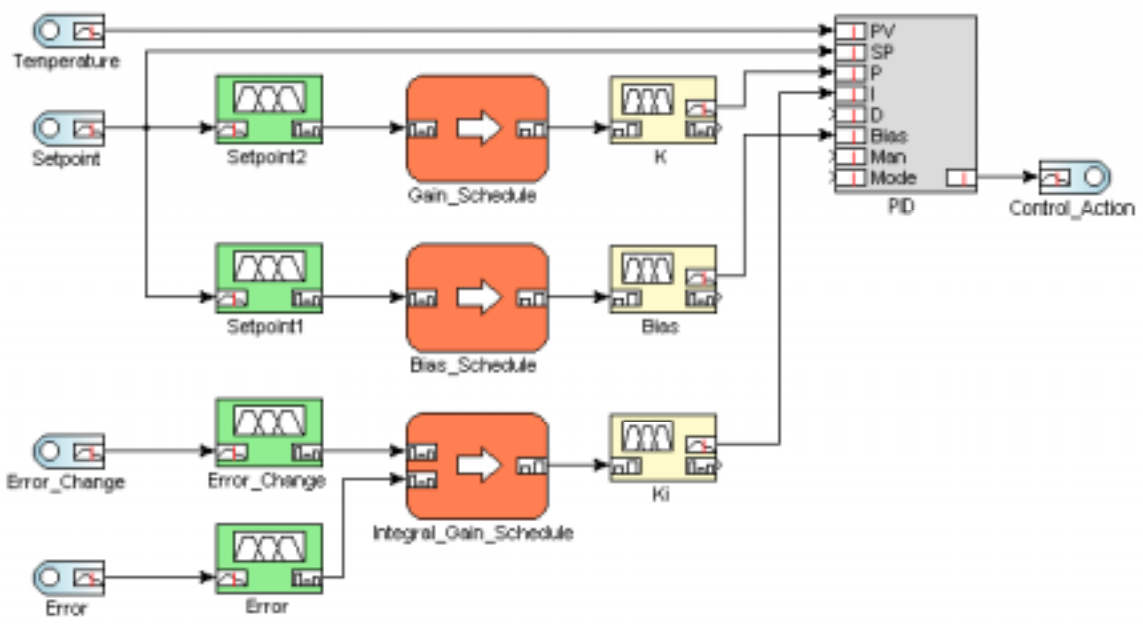
6.1. Aplikace z pohledu uživatele

Program je určen pro návrh hierarchických fuzzy systémů. Lze v něm vytvářet jak jednoduché struktury obsahující jeden pravidlový blok, tak složité modely s více vstupy, výstupy a rozličně pospojovanými pravidlovými bloky. Celý systém je udržován v takzvaném **fuzzy jádře**, což je srdce aplikace. Provádí veškeré výpočty, simulace, realizuje spojení s okolními aplikacemi a podobně. Dalo by se říci, že tato část je ta nejdůležitější v celém programu. Ne však z pohledu této práce. Fuzzy systém obsažený v jádře musí být totiž nějakým způsobem vytvořen. A k tomu právě slouží nástroje GUI. Základní jsou stromová struktura systému a grafický model. Ten je grafickou reprezentací fuzzy systému obsaženého v jádře. Je v něm možné jednotlivé části systému přidávat, editovat i mazat. Již vytvořený model je dále potřeba ladit a otestovat. Za tímto účelem aplikace obsahuje několik nástrojů pro simulaci a monitorování. Jmenovitě to jsou například časový graf, 2D a 3D graf, sloupcové grafy u jazykových proměnných a jednotlivých pravidel, atd. Část uživatelského rozhraní, podporujícího návrh, ladění a monitorování fuzzy systému, byla vytvořena v rámci diplomové práce. Jedná se o **grafický model**, **2D graf**, **3D graf** a **časový graf**. Jejich charakteristika je uvedena v následujícím textu.

6.1.1. Grafický model – popis ovládání

Grafický model je grafická reprezentace systému definovaného ve fuzzy jádře. Každému prvku definovanému v jádře odpovídá právě jeden prvek definovaný v grafickém modelu. Naopak to ale neplatí. V grafickém modelu se vyskytují tři základní druhy zobrazovaných prvků: **grafické členy**, **čáry** a **text**. Grafické členy jsou grafickou podobou **komponent** ve fuzzy jádře. Lze s nimi libovolně manipulovat, přesouvat, měnit jejich velikost, barvu atd. Čáry jsou grafickou podobou spojení mezi jednotlivými komponentami. Jedná se o rozvětvené lomené čáry, které lze libovolně tvarovat. Texty jsou prvky, které nejsou v jádře obsaženy a slouží pouze pro lepší orientaci ve vytvořeném fuzzy systému.

Výsledné blokové schéma pak vypadá například jako na obrázku Obr. 6.1. To se vykresluje do samostatného okna, ve kterém je takzvaná pracovní plocha (prostor pro umisťování objektů). Ta má vždy jen takovou velikost, která je právě potřeba k zachycení celého schématu. Dynamicky tedy mění svou velikost podle velikosti schématu.

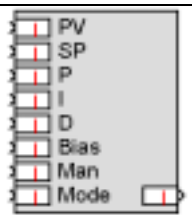


Obr. 6.1: Grafický model hierarchického fuzzy systému

Grafické členy

V grafickém modelu je definováno osm druhů grafických členů. Jejich vzhled, jméno a význam je uveden v následující tabulce. Konkrétní parametry, jako jsou způsoby zadávání a zobrazování funkcí příslušnosti a pravidel, se netýkají grafického modelu a nejsou tedy předmětem této práce.

	Input Port	Předzpracovává a uchovává vstupní data fuzzy systému.
	Output Port	Obsahuje výstupní data fuzzy systému.
	Input Linguistic Variable	Obsahuje jazykové termy a je užíván pro fuzzifikaci hodnotového signálu.
	Rule Block	Obsahuje pravidla, provádí inferenci založenou na fuzzy pravidlech a počítá stupeň příslušnosti výstupních jazykových proměnných v závislosti na vstupních jazykových proměnných.
	Intermediate Linguistic Variable	Tvoří most mezi dvěma spojenými pravidlovými bloky.
	Output Linguistic Variable	Obsahuje jazykové termy a je užíván pro výpočet výstupního hodnotového signálu na základě stupně příslušnosti do jednotlivých termů.

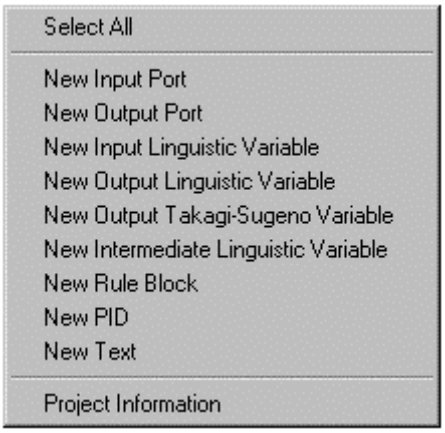
	Output Takagi-Sugeno Variable	Obsahuje parametry funkčních termů a je užíván pro výpočet výstupního hodnotového signálu na základě stupně příslušnosti do jednotlivých termů.
	**PID controller**	Odpovídá průmyslové verzi PID regulátoru.

Grafické členy lze do projektu přidávat mnoha způsoby: z hlavního menu, z toolbaru a také z kontextového menu. Toto kontextové menu se vyvolá pomocí pravého tlačítka myši, které je stisknuto nad prázdným místem pracovní plochy. Jeho struktura je ukázaná na obrázku Obr. 6.2. Mazání objektů se provádí buď pomocí klávesy DELETE nebo přes kontextové menu grafického objektu (viz dále).

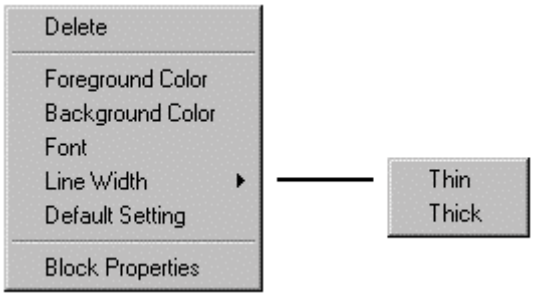
Vzhled jednotlivých komponent byl volen tak, aby se od sebe dostatečně lišily a již na první pohled bylo zřejmé, o jakou komponentu jde. Jelikož se v systému mohou vyskytovat čtyři druhy pinů (vstupní a výstupní místa grafických členů), bylo nutné udělat také čtyři odlišné grafické reprezentace. Takové rozlišení je důležité nejen pro rozpoznání o jaký druh pinu jde, ale také pro poznání, které dva lze spojit. Není totiž možné spojit dva piny různého druhu (s různým „obrázkem“). Vlastní spojování se provádí pomocí speciálních dialogů. Pro lepší orientaci má každý grafický člen pod sebou napsané své jméno.

Pracovat lze současně s jedním nebo s několika objekty (grafické členy nebo texty). První možnost výběru prvku je prosté kliknutí na jeho grafickou reprezentaci. Druhý způsob je výběrový obdélník vytvořený táhnutím myší. Třetí způsob je výběr všech objektů pomocí položky výše jmenovaného kontextového menu. Pokud před započetím jedné nebo druhé operace byly nějaké bloky vybrány, jsou tyto odznačeny a do výběru se počítají pouze nové objekty. Pokud chce uživatel k již vybraným prvkům přidat nějaký nový, musí být u obou operací stisknuté CTRL. Vybraný objekt (objekty) jsou dostatečně viditelně označeny.

U grafických členů lze nastavovat tyto grafické parametry: polohu, velikost, barvu pozadí, barvu popředí, tloušťku čáry a font používaný pro psaní jména. Poloha se mění buď pomocí táhnutí myší nebo klávesnicí (kombinace šipek a CTRL). Pozice bloků není úplně libovolná, ale je zarovnávána na mřížku. Výsledné schéma je pak estetičtější a přehlednější. Velikost bloků se mění pouze táhnutím myší za libovolný roh. Ostatní parametry se nastavují pomocí kontextového menu grafického členu (Obr. 6.3), které se vyvolá stiskem pravého tlačítka myši nad vybraným prvkem (prvky).



Obr. 6.2: Kontextové menu grafického modelu



Obr. 6.3: kontextové menu grafického členu

Čáry

Spojení mezi komponentami je v grafickém modelu reprezentováno rozvětvenými lomenými čarami. Jejich tvar je po vytvoření spoje určen programem. Je ho ale možné pomocí táhnutí myši libovolně měnit. Pohybovat lze vždy pouze buď s celou lomenou čárou, nebo s jedním přímým úsekem anebo s jedním bodem. Čáry nemají žádné omezení na počet zlomů, větví nebo podobně. Pouze musejí být vždy pravoúhlé. Jejich poloha je, stejně jako u grafických objektů, zarovnávána na mřížku.

Texty

Texty jsou používány pro komentování jednotlivých částí fuzzy systému a pro celkové zvýšení přehlednosti schématu. Operace s nimi je snadná. Vytváření, mazání a přesun je stejný jako u grafických členů. Nastavovat u nich lze pouze barvu a font. Provádí se tak pomocí kontextového menu, které se vyvolá stiskem pravého tlačítka myši nad editovaným textem.

Autoplacer (Autorouter)

Zvláštní pozornost by si zasloužila funkce Autoplacer spolu s Autoroutrem. Obě společně provádějí uspořádání grafického modelu v případě, že se ze souboru načte fuzzy systém, u něhož není nebo je poškozená informace o grafické reprezentaci. V takovém případě je vytvořeno defaultní nastavení, které vypadá například stejně jako na Obr. 6.1.

6.1.2. Grafický model – XML formát

Fuzzy systém, a tudíž i grafický model, lze uložit a načíst do textového souboru ve formátu XML. Úsek definující oblast XML, která patří ke grafickému modelu, je element GRAPHICS_MODEL. Začíná tagem `<GRAPHICS_MODEL VERSION="1.0">`, pokračuje definicí elementů prvků grafického modelu a končí tagem `</GRAPHICS_MODEL>`. Obsahuje jeden nepovinný atribut: VERSION, který určuje verzi XML, v níž je daný kód napsán. Pokud není uveden, program předpokládá, že verze je stejná jako ta, kterou používá. Pokud element chybí, nebo je prázdný, je vytvořeno defaultní nastavení grafického modelu. Přibližná struktura základního elementu:

```
<GRAPHICS_MODEL VERSION="1.0">
     ⋮
     <INPUT_PORT NAME="velocity">
           ⋮
     </INPUT_PORT>
     ⋮
     <LINE ID="1">
           ⋮
     </LINE>
     ⋮
     <TEXT>
           ⋮
     </TEXT>
   ⋮
</GRAPHICS_MODEL>
```

Element GRAPHICS_MODEL může obsahovat některé z následujících vnořených elementů: INPUT_PORT, OUTPUT_PORT, INPUT_LINGUISTIC_VARIABLE, OUTPUT_LINGUISTIC_VARIABLE, OUTPUT_TAKAGI_SUGENO_VARIABLE, PID, RULE_BLOCK, INTERMEDIATE_LINGUISTIC_VARIABLE, LINE a TEXT. Jejich význam je popsán v následujícím textu.

Grafické členy

Grafické členy musejí přesně odpovídat struktuře příslušných komponent v jádře. Elementy reprezentující grafické členy v XML začínají tagem pojmenovaným podle typu členu a obsahují povinný atribut NAME, který definuje jméno členu. Například Input Port začíná tagem `<INPUT_PORT NAME="velocity">`, Input Linguistic Variable začíná tagem `<INPUT_LIGUISTIC_VARIABLE NAME="velocity">`, atd. Ukončovací tagy mají stejný systém. Komponenta fuzzy jádra, které grafický prvek odpovídá, je jednoznačně určena právě typem a jménem. Pravidla pro mapování grafických členů a komponent jádra jsou následující. Pokud nějaká komponenta nemá definovaný odpovídající platný grafický člen, je vytvořena jeho defaultní podoba. Pokud neexistuje ke XML specifikaci grafického členu odpovídající komponenta, je tento člen ignorován. Pokud je nějaký člen definován v XML opakovaně, je platná pouze první definice a ostatní jsou ignorovány. Pořadí, v jakém jsou grafické členy v XML definovány, je stejné jako pořadí prvků v grafickém modelu a ovlivňuje posloupnost, v níž se jednotlivé členy vykreslují. Příklad počátečního tagu grafického členu:

```
<INPUT_PORT NAME="velocity" />
```

Každý grafický člen může obsahovat následující vnořené elementy: INPUT, OUTPUT, LOCATION, SIZE, BACK_COLOR, FORE_COLOR, LINE_WIDTH a FONT. Žádný z nich není sice povinný, ale je doporučeno, aby byly definovány alespoň elementy INPUT, OUTPUT a LOCATION. Pokud některý z elementů chybí, je nastavena jeho defaultní hodnota.

Element INPUT definuje vstupní piny grafického členu. Má dva atributy : DESCRIPTION a LINK. Atributy DESCRIPTION a LINK jsou používány k mapování vstupních pinů grafického členu na vstupní piny komponenty v jádře. Atribut DESCRIPTION je povinný pouze pro hodnotové vstupy u PID a Output Takagi-Sugeno Variable. U všech ostatních grafických členů je ignorován. U PID má význam jmen jednotlivých pinů (P_GAIN, I_GAIN, D_GAIN, BIAS, MANUAL_CONTROL, MODE_SWITCH, SET_POINT a PROCESS_VARIABLE). U Output Takagi-Sugeno Variable má význam pořadového čísla vstupního hodnotového pinu v odpovídající XML definici komponenty (první má číslo 1 a počítány jsou pouze hodnotové vstupy). Atribut LINK je povinný pro všechny typy grafických členů. Definuje jméno připojeného grafického členu. Pravidla pro mapování vstupních pinů grafických členů a vstupních pinů komponent jsou následující. Pokud nemá nějaký vstupní pin komponenty definován odpovídající vstupní pin grafického členu, pak je vytvořena jeho defaultní podoba. Pokud neexistuje ke XML specifikaci vstupního pinu grafického členu odpovídající vstupní pin komponenty, je tento pin ignorován. Pokud je nějaký vstupní pin definován v XML opakovaně, je platná pouze první definice a ostatní jsou ignorovány. Element INPUT může obsahovat dva nepovinné vnořené elementy: LINE_ID a LINE_PIN_ID. Oba slouží k definování připojené čáry. LINE_ID odpovídá atributu ID uvedené v úvodním tagu LINE a LINE_PIN_ID odpovídá atributu ID u elementu čáry OUTPUT. Příklad použití elementu INPUT:

```
<INPUT DESCRIPTION="P_GAIN" LINK="velocity">
     <LINE_ID>1</LINE_ID>
     <LINE_PIN_ID>1</LINE_PIN_ID>
</INPUT>
```

Element OUTPUT definuje výstupní pin grafického členu. Má povinný atribut LINK a může obsahovat nepovinné vnořené elementy LINE_ID a LINE_PIN_ID. Význam všech je stejný jako u vstupního pinu grafického členů. Příklad použití elementu OUTPUT:

```
<OUTPUT LINK="angle">
     <LINE_ID>1</LINE_ID>
     <LINE_PIN_ID>1</LINE_PIN_ID>
</OUTPUT>
```

Element LOCATION definuje pozici levého horního rohu grafického členu. Obsahuje dva povinné atributy: X a Y. Platné hodnoty pro oba dva jsou nezáporná celá čísla. Bod [0,0] odpovídá levému hornímu rohu pracovní plochy. Příklad použití elementu LOCATION:

```
<LOCATION X="20" Y="100"/>
```

Element SIZE definuje velikost grafického členu. Obsahuje dva povinné atributy: WIDTH a HEIGHT. Platné hodnoty pro oba dva jsou nezáporná celá čísla. Příklad použití elementu SIZE:

```
<SIZE WIDTH="50" HEIGHT="30"/>
```

Elementy BACK_COLOR a FORE_COLOR definují barvy používané k vykreslení pozadí a popředí grafického členu. Oba mají tři povinné atributy: R (červená), G (zelená) a B (modrá). Platné hodnoty u všech těchto atributů jsou celá čísla v rozsahu 0 až 255. Příklad použití elementů BACK_COLOR a FORE_COLOR:

```
<BACK_COLOR R="100" G="100" B="100"/>
<FORE_COLOR R="0" G="0" B="255"/>
```

Element LINE_WIDTH definuje šířku okrajové čáry. Může nabývat dvou hodnot: 1 (tenká) nebo 2 (tlustá). Příklad použití elementu LINE_WIDTH:

```
<LINE_WIDTH>1</LINE_WIDTH>
```

Element FONT definuje font používaný pro psaní jmen grafických členů. Má čtyři nepovinné atributy: NAME, STYLE, SIZE a SCRIPT. Pokud není některý z nich uveden, je místo něj nastavena defaultní hodnota. NAME obsahuje jméno fontu. STYLE specifikuje styl písma. Jeho platné hodnoty jsou: “Regular”, “Bold”, “Italic”, “Underline” a “Strikeout”. SIZE určuje velikost fontu. SCRIPT specifikuje GDI (Graphics Device Interface) znakovou sadu.

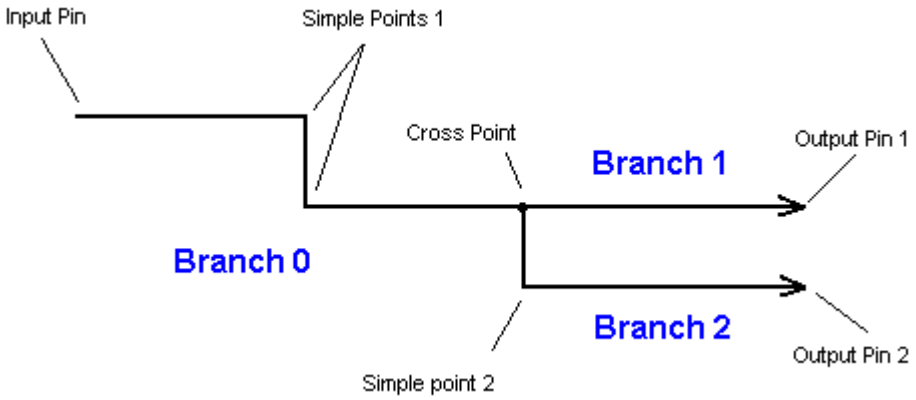
```
<FONT NAME="Arial" STYLE="Regular" SIZE="8" SCRIPT="1" />
```

Příklad XML reprezentace grafického členu PID:

```
<PID NAME="PID_Controller">
     <INPUT DESCRIPTION="P_GAIN" LINK="input">
           <LINE_ID>1</LINE_ID>
           <LINE_PIN_ID>1</LINE_PIN_ID>
     </INPUT>
     <INPUT DESCRIPTION="I_GAIN" LINK="input">
           <LINE_ID>1</LINE_ID>
           <LINE_PIN_ID>2</LINE_PIN_ID>
     </INPUT>
     ⋮
     <OUTPUT LINK="output">
           <LINE_ID>2</LINE_ID>
           <LINE_PIN_ID>1</LINE_PIN_ID>
     </INPUT>
     <LOACTION X="20" Y="100"/>
     <SIZE WIDTH="50" HEIGHT="30"/>
     <BACK_COLOR R="100" G="100" B="100"/>
     <FORE_COLOR R="0" G="0" B="0"/>
     <LINE_WIDTH>1</LINE_WIDTH>
     <FONT NAME="Arial" STYLE="Regular" SIZE="8" SCRIPT"1" />
</INPUT_PORT>
```

Čáry

Element LINE je XML reprezentací lomené čáry spojující dva nebo více grafických členů. Pokud některé spojení nemá definováno své parametry, pak je vytvořena defaultní čára. Každá čára má jeden povinný atribut ID. To je jednoznačné identifikační číslo, které se nesmí v celém grafickém modelu u žádné čáry opakovat. Povolené hodnoty jsou od 1 do 2,147,483,647. Element LINE obsahuje povinný vnořený element BRANCH. Element BRANCH obecně definuje jednu větev z úvodního bodu (vstupní pin spoje nebo rozvětvující bod) do koncového bodu (výstupní pin spoje nebo rozvětvující bod). Element BRANCH může obsahovat čtyři typy vnořených elementů: INPUT, OUTPUT, POINTS a BRANCH. Jejich význam je vidět z následujícího příkladu.



Obr. 6.4: Schéma lomené čáry

Na obrázku Obr. 6.4 je vidět schéma lomené čáry skládající se ze tří větví. Větev ***Branch 0*** je základní větev a obsahuje jeden vstupní pin (*Input Pin*), tři vnitřní body (*Simple Points 1* a *Cross Point*) a žádný výstupní pin. Základní větev se dále dělí na dvě pokračující větve: ***Branch 1*** a ***Branch 2***. Větev ***Branch 1*** neobsahuje žádný vstupní pin, žádný vnitřní bod, ale má jeden výstupní pin (*Output Pin 1*). Větev ***Branch 2*** neobsahuje žádný vstupní pin, ale má jeden vnitřní bod (*Simple Point 2*) a jeden výstupní pin (*Output Pin 2*). Tato struktura je v XML popsána následujícím elementem:

```
<LINE ID="1">
     <BRANCH>
           <INPUT ID="1"/>
           <POINTS>[100,10] [100,20] [150,20]</POINTS>
           <BRANCH>
                 <OUTPUT ID="1"/>
           </BRANCH>
           <BRANCH>
                 <OUTPUT ID="2"/>
                 <POINTS>[150,30]</POINTS>
           </BRANCH>
     </BRANCH>
</LINE>
```

Element INPUT reprezentuje vstupní pin (bod připojený k nějakému grafickému členu). Tento element je pro základní větev povinný a v ostatních je ignorován. Obsahuje jeden povinný atribut ID. To je jednoznačné identifikační číslo, které se nesmí v rámci jedné čáry u žádného vstupního pinu opakovat. Povolené hodnoty jsou od 1 do 2,147,483,647. Pozici není nutné udávat, protože je jednoznačně určena pozicí připojeného pinu u grafického členu.

Element OUTPUT reprezentuje výstupní pin. V každé věty se musí vyskytovat buď jeden výstupní pin nebo nejméně jedna pokračující větev. Element OUTPUT obsahuje jeden povinný atribut ID. To je jednoznačné identifikační číslo, které se nesmí v rámci jedné čáry u žádného vstupního pinu opakovat. Povolené hodnoty jsou od 1 do 2,147,483,647. Pozici není nutné udávat, protože je jednoznačně určena pozicí připojeného pinu u grafického členu.

Element POINTS definuje vnitřní body větve. Body jsou definovány svou x-ovou a y-ovou souřadnicí, které jsou uzavřené do hranaté závorky a odděleny čárkou. Dva body jsou odděleny mezerou.

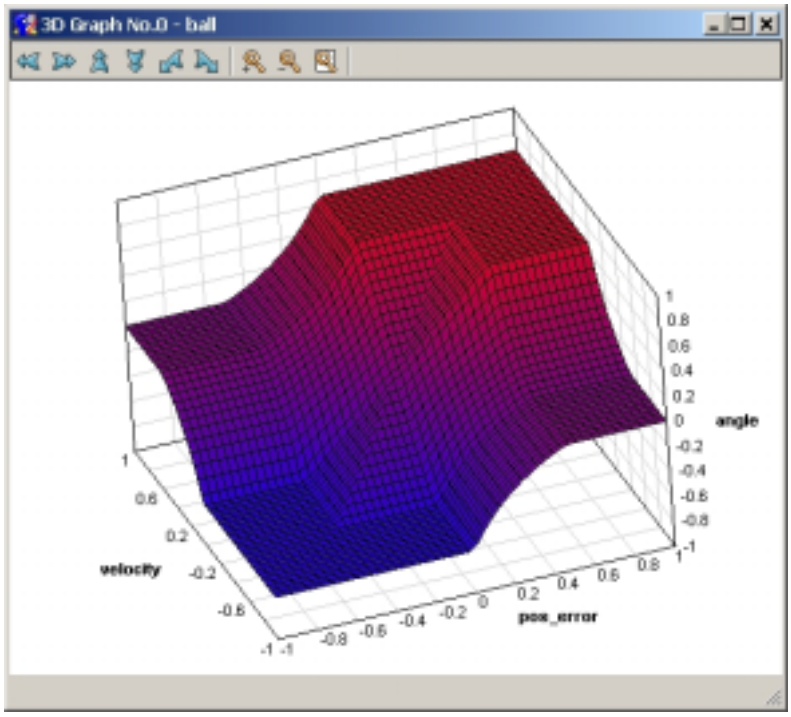
Texty

Text je v XML reprezentován elementem TEXT. Ten může obsahovat následující vnořené elementy : STRING, LOCATION, FORE_COLOR a FONT. Element STRING definuje zobrazený text. To je jediný povinný element, který navíc nesmí být prázdný. Význam a pravidla použití ostatních elementů jsme popsali v předchozím textu. Příklad:

```
<TEXT>
     <STRING>Displayed text</STRING>
     <LOCATION X="20" Y="50">
     <FORE_COLOR R="0" G="0" B="0">
     <FONT NAME="Arial" STYLE="Regular" SIZE="8" SCRIPT"1"/>
</TEXT>
```

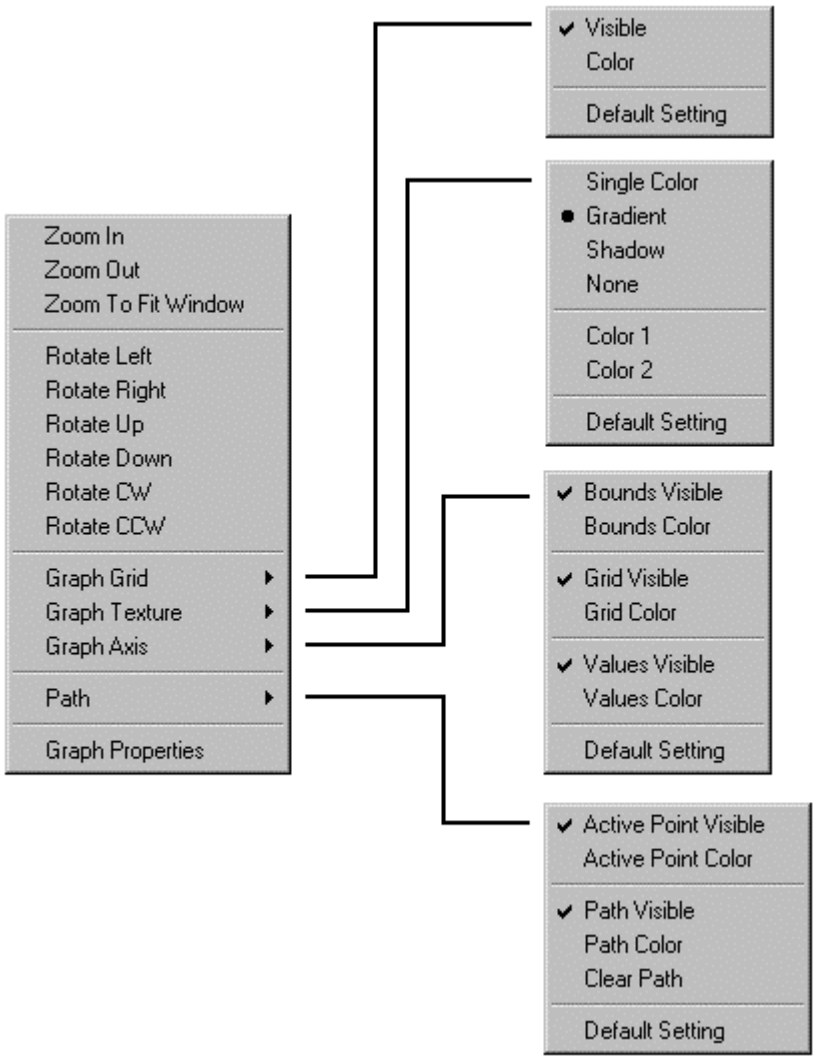
6.1.3. 2D a 3D graf

2D a 3D graf slouží k zobrazení závislosti jednoho výstupu na jednom (2D) nebo dvou (3D) vstupech. Při vytváření se zvolí sledované veličiny, jejich rozsah a počet vzorků (10 až 50) na rozsah. Jelikož hodnota zkoumaného výstupu málokdy závisí pouze na testovaných vstupech, je nutné sledovat změny i ostatních (nezobrazených) vstupů. Při výpočtu hledané závislosti se jejich hodnota považuje za konstantní a odpovídá aktuálnímu stavu. Pokud je graf již vytvořen a změní se nějaký nezobrazovaný vstup, stane se aktuální závislost neplatnou a musí být přepočítána. Pokud se změní jeden ze zobrazovaných vstupů, je možné do grafu vykreslit jeho trajektorii. Tato funkce se nazývá *„Path“* a její ovládání je popsáno v následujícím textu. Vypočtená závislost je vykreslena v okně, které je na obrázku Obr. 6.5.



Obr. 6.5: Hlavní okno 3D grafu

Graf je možné upravovat podle potřeb uživatele. Ovládání je trojího druhu: pomocí toolbaru, kontextového menu a myši. Měnit je možné pozici, velikost, barvu, způsob vykreslení grafu, ale také základní parametry jako jsou zobrazované vstupní a výstupní proměnné, rozsah a počet vzorků. Pro změnu orientace grafu o větší úhel lze použít toolbar nebo kontextové menu. Pokud ovšem potřebujeme jemnější manipulaci, je možné rotovat graf pomocí táhnutí myši. Pokud je stisknuté levé tlačítko a myš se pohybuje vodorovně, otáčí se graf doleva a doprava. Při stisku stejného tlačítka a svislém pohybu myši se otáčí graf nahoru a dolů. A pokud je stisknut SHIFT, levé tlačítko a myš se pohybuje vodorovně, otáčí se graf ve směru hodinových ručiček, popřípadě proti nim. Hlavním ovládacím prvkem je ale kontextové menu. To obsahuje veškeré operace, které lze s grafem provádět. Na obrázku Obr. 6.1 je přehledně zobrazena jeho struktura. Význam jednotlivých položek je vysvětlen v následujícím textu.



Obr. 6.1: Struktura kontextového menu 3D grafu

- **Zoom In** – Zvětší graf.
- **Zoom Out** – Zmenší graf.
- **Zoom To Fit Window** – Změní velikost grafu tak, aby co nejlépe vyplnil okno.
- **Rotate Left** – Rotuje graf doleva.
- **Rotate Right** – Rotuje graf doprava.

- **Rotate Up** – Rotuje graf nahoru.
- **Rotate Down** – Rotuje graf dolů.
- **Rotate CW** – Rotuje graf ve směru hodinových ručiček.
- **Rotate CCW** – Rotuje graf proti směru hodinových ručiček.
- **Graph Grid** – Nastaví viditelnost a barvu mřížky na ploše grafu.
- **Graph Texture** – Nastaví parametry použité textury. Ta může být jednobarevná, gradientní, stínovaná nebo může být graf bez textury. Barva se volí změnou položek Color1 a Color2.
- **Graph Axis** – Nastaví parametry os. Lze zvolit viditelnost a barvu u hranic, mřížky a popisu os.
- **Path** – Nastaví parametry vykreslené trajektorie zobrazovaných vstupních veličin. Lze zvolit viditelnost a barvu právě aktivního bodu nebo celé trajektorie. Pomocí položky **Clear Path** lze celou trajektorii vymazat.
- **Graph Properties** – Otevře okno s parametry grafu. V něm lze nastavit zkoumané vstupní a výstupní proměnné, rozsah a počet vzorků.

Některé výše uvedené operace lze také provést pomocí toolbaru. Jeho podoba je ukázána na obrázku Obr. 6.7. Jednotlivá tlačítka jsou zleva: Rotate Left, Rotate Right, Rotate Up, Rotate Down, Rotate CCW, Rotate CW, Zoom In, Zoom Out a Zoom To Fit Window. Jejich význam je stejný jako u stejnojmenných položek v kontextovém menu.

Obr. 6.7: Toolber 3D grafu

6.1.4. Časový graf

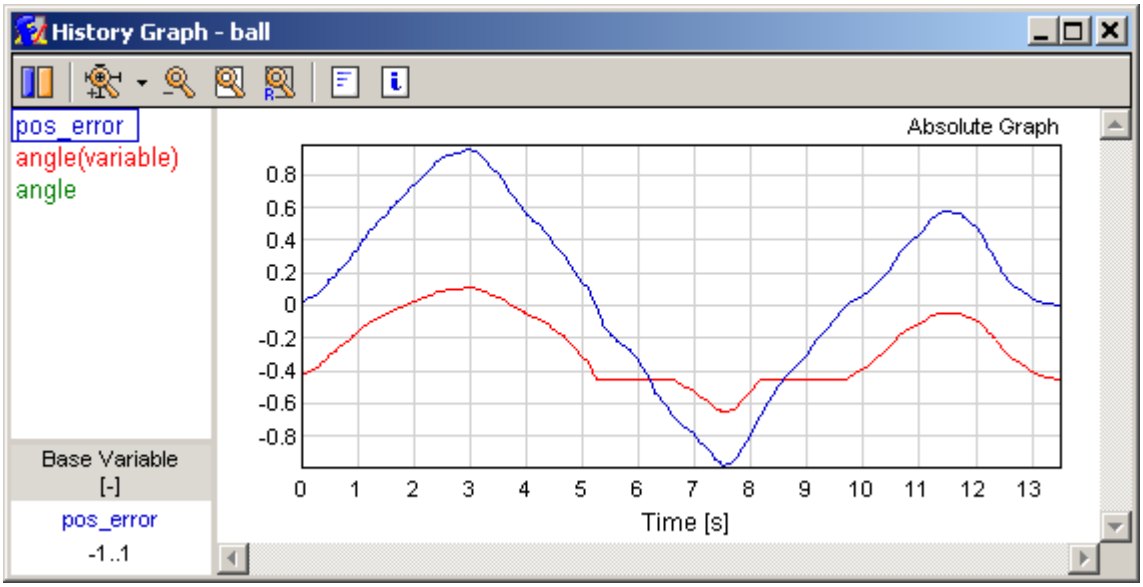
Časový graf je součást monitorovacích nástrojů. Je používán k zobrazování naměřených nebo nasimulovaných dat. Při jeho vytváření se nastavují zobrazované veličiny a maximální počet vzorků (hloubka historie). Nový graf je vykreslen v okně ukázaném na obrázku Obr. 6.8. To se skládá ze tří částí (podoken): okno s vlastním grafem, legenda a okno s parametry bázové křivky (viz dále). Jejich význam je následující:

Okno s vlastním grafem – Je umístěno v levé části hlavního okna časového grafu a nedá se skrýt. Zde jsou vykresleny naměřené křivky. Jednotlivé vzorky nejsou interpolovány, ale jednoduše spojovány úsečkami. Maximální počet zobrazených křivek je 1000. Mřížka vertikální i horizontální osy je nastavována automaticky. Rozsah zobrazovaných hodnot může být změněn pomocí zvětšování nebo zmenšování grafu. Název a jednotky vodorovné osy jsou popsány pod touto osou. Jméno a jednotky svislé osy nejsou zobrazeny, protože v grafu jsou zobrazovány většinou veličiny s různými parametry. Některé vlastnosti zobrazení lze měnit pomocí kontextového menu časového grafu (Obr. 6.9). To se aktivuje po stisku pravého tlačítka myši nad tímto oknem.

Legenda – Je umístěna v levé horní části hlavního okna časového grafu. Může být skryta nebo opět zviditelněna pomocí jednoho z tlačítek toolbaru. V legendě je zobrazen seznam vykreslených proměnných. Barva a pořadí odpovídá křivkám ve vlastním grafu. Pokud uživatel chce například, aby proměnná se jménem „Input1“ byla vykreslena přes všechny ostatní, musí být toto jméno jako první ve výše jmenovaném seznamu. Pořadí položek se mění jednoduše táhnutím myši (drag and drop operace). První položka v seznamu je nazývána bázová křivka a je používána jako referenční při nastavení relativního formátu grafu (viz dále). Pomocí pravého tlačítka myši může být vyvoláno kontextové menu legendy. Umožňuje nastavit barvu a vertikální rozsah proměnné, nad níž byla daná operace provedena.

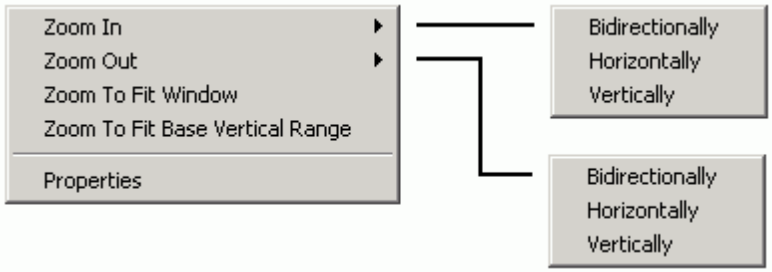
Okno s parametry bázové křivky – Je umístěno v levém dolním rohu hlavního okna časového grafu. Může být skryta nebo opět zviditelněna pomocí jednoho z tlačítek toolbaru. Jsou v něm zobrazeny základní informace o bázové křivce. Jsou to zejména jméno, barva, jednotka a vertikální rozsah.

Vytvořený graf může být buď absolutní nebo relativní. Přepínání se provádí pomocí prvního tlačítka toolbaru. Text informující o tom, v jakém stavu z těchto dvou se nachází, je umístěn v pravém horním rohu okna. Pokud je graf absolutní, jsou zobrazované hodnoty na svislé ose platné pro všechny křivky. Pokud je ale relativní, platí tyto hodnoty pouze pro takzvanou bázovou křivku. To je křivka, která je jako první jmenovaná v legendě ke grafu. V relativním módu jsou dále všechny křivky mapovány tak, aby měly stejný vertikální rozsah jako bázová křivka. Rozsah je platný pouze pro tento graf a s rozsahem jazykové proměnné v celém projektu má společnou pouze svou počáteční hodnotu.



Obr. 6.8: Základní okno časového grafu

Ovládání grafu je realizováno třemi základními způsoby: pomocí dvou kontextových menu, toolbaru a myši. Některé operace je možné vyvolat z několika z nich, neplatí to však obecně. Všechny základní úkony lze vykonat pomocí hlavního kontextového menu časového grafu. To se vyvolá stiskem pravého tlačítka nad oblastí vlastního grafu. Jeho struktura je ukázána na obrázku Obr. 6.9. Význam jednotlivých položek je popsán v následujícím textu.



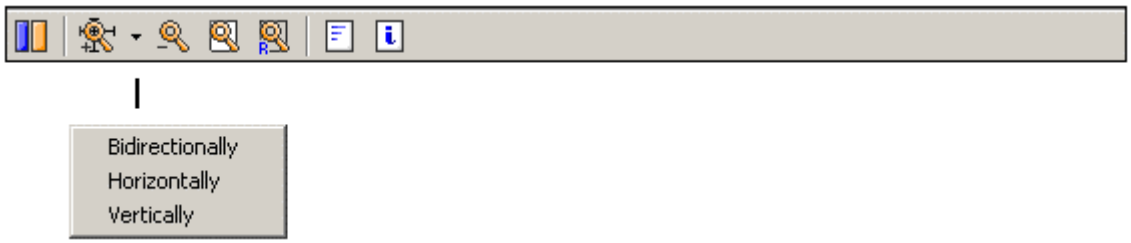
Obr. 6.9: Kontextové menu časového grafu

Obr. 6.10: Kontextové menu legendy časového grafu

- **Zoom In** – Zvětší měřítko grafu. Maximální velikost závisí na aktuálním vertikálním rozsahu bázové křivky a maximální naměřené hodnotě.
- **Zoom out** – Zmenší měřítko grafu. Minimální hodnota závisí na maximální zobrazené hodnotě.
- **Zoom To Fit Window** – Zvětší graf tak, aby byly vidět všechny naměřené hodnoty a přitom bylo vyplněno celé okno.
- **Zoom To Fit Base Vertical Range** – Změní měřítko grafu tak, aby vertikální rozsah bázové křivky vyplňoval celé okno.
- **Properties** – Otevře okno s parametry grafu (zobrazené křivky).

Druhé kontextové menu se vyvolá stiskem pravého tlačítka nad legendou. Týká se vybrané proměnné a umožňuje nastavit její barvu a vertikální rozsah. Pro změnu barvy se používá standardní dialog z Windows a pro změnu rozsahu je určen speciální dialog. Struktura kontextového menu legendy je ukázána na obrázku Obr. 6.10.

Dalším ovládacím prvkem je toolbar. Obsahuje tři skupiny tlačítek: tlačítko pro změnu relativního/absolutního zobrazení, skupinu tlačítek pro změnu měřítka a tlačítka pro ovládání jednotlivých částí okna časového grafu. Jeho vzhled je ukázán na obrázku Obr. 6.11. a význam jednotlivých tlačítek je popsán v následujícím textu.



Obr. 6.11: Toolbar časového grafu

- **Relative/Absolute Graph** – Změní graf na relativní/absolutní.
- **Zoom In** – Zvětší měřítko grafu. Druh zvětšení závisí na vybrané položce menu náležícího k toolbaru. Jsou možné tři způsoby této operace: změna velikosti v obou směrech, změna velikosti ve vodorovném a ve svislém směru. Obrázek na tlačítku odpovídá právě zvolenému způsobu. Vybrané nastavení neplatí jen pro zvětšování a zmenšování pomocí toolbaru, ale také pomocí myši (viz dále).
- **Zoom Out** – Zmenší měřítko grafu. Pro přesnější popis viz Zoom In.
- **Zoom To Fit Window** – Zvětší graf tak, aby byly vidět všechny naměřené hodnoty a přitom bylo vyplněno celé okno.
- **Zoom To Fit Base Vertical Range** – Změní měřítko grafu tak, aby vertikální rozsah bázové křivky vyplňoval celé okno.
- **Show/Hide Legend** – Skryje/zviditelní legendu.
- **Show/Hide Base Curve Parameters** – Skryje/zviditelní okno s parametry bázové křivky.

Poslední možností, jak ovládat časový graf, je použití myši. Jejím táhnutím (operace drag and drop) lze zvětšit libovolnou část grafu. Tento nástroj podléhá stejnému nastavení, jaké je uvedeno v položce Zoom In u toolbaru.

6.2. Uživatelské rozhraní z pohledu programátora

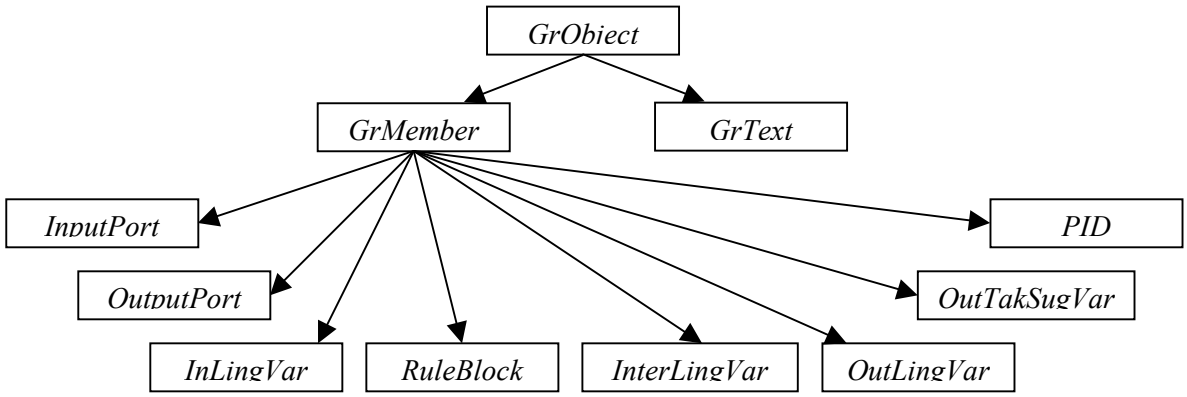
Kompletní aplikace, program pro manuální návrh fuzzy systémů, se skládá z centrální části a okolních modulů v podobě .NET dynamických knihoven *.dll. Centrální část má za úkol zprostředkovávat a synchronizovat komunikaci mezi periferními funkcemi. Soubory obsahující moduly vytvořené v rámci diplomové práce se jmenují GrahicsModel.dll (grafická model), MeshPlot.dll (2D a 3D graf) a Graphs.dll (časový graf). Jejich vnitřní struktura je rozsáhlá (zdrojový kód čítá 16267 + 6393 + 5579 = 28239 většinou ručně psaných řádků), proto ji v následujícím textu pouze naznačíme. S výhodou byly často použity přednosti objektově orientovaného programování. Veškeré prvky programu jsou reprezentovány pomocí tříd, která jsou uspořádány do přehledné hierarchie. Tento způsob práce umožňuje snadnější vývoj celé aplikace. Nepíše se tolik kódu a je tedy menší pravděpodobnost chyby. Vzniklý úsek programu je pak do značné míry modulární. Jednotlivé vlastnosti tříd nebo samotné třídy lze snadno měnit a upravovat tak celkový vzhled a funkci aplikace. U takto velkého projektu je také důležité, že reprezentace pomocí tříd zajišťuje umístění dat a funkcí s nimi pracujícími na jedno místo (zapouzdření). Výsledný zdrojový kód je pak podstatně přehlednější, než kdyby se objekty nepoužily.

6.2.1. Grafický model

Jak již bylo řečeno, grafický model obsahuje tři druhy objektů: grafické členy (funkční bloky), text a čáry. Jejich vnitřní reprezentace v programu je nastíněna v následujících odstavcích.

Grafické členy a text

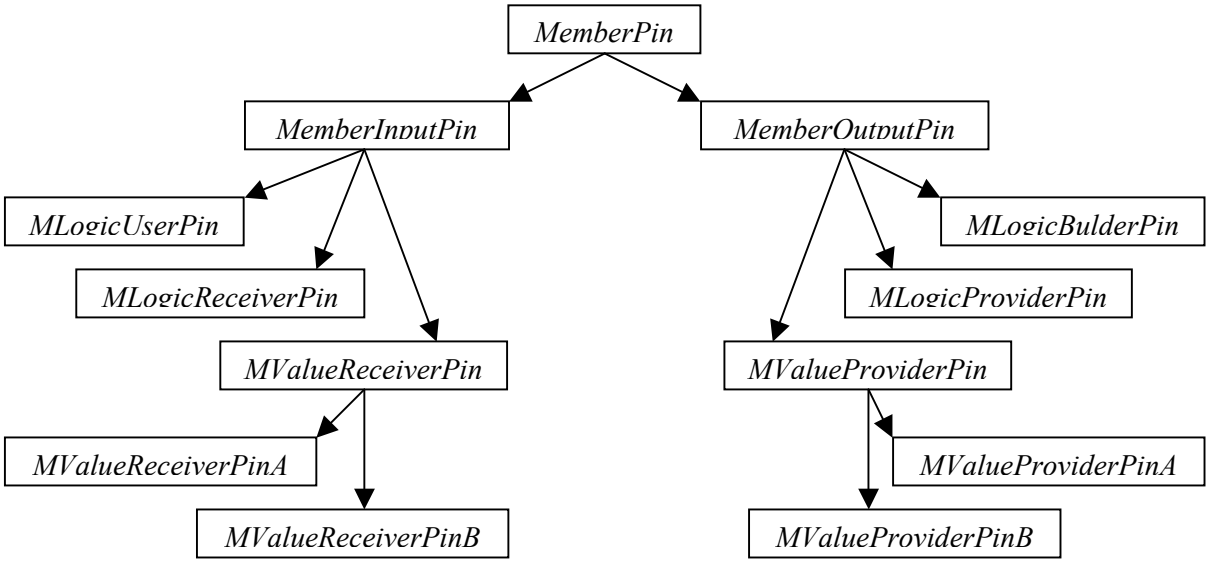
Oba tyto objekty jsou realizovány třídami odvozenými od základní třídy nazvané *GrObject*. Hierarchie tříd je uvedena na obrázku Obr. 6.12. *GrObject* obsahuje základní informace o objektu (poloha, stav vybrání, …) a předpisy několika společných metod. Od ní jsou odvozeny dvě třídy. První je *GrText,* což je koncová třída reprezentující text umístěný na pracovní plochu a druhá je *GrMember,* která slouží jako základ pro třídy odpovídající jednotlivým typům grafických členů (*InputPort*, *OutputPort*, *InLingVar*, *RuleBlock*, *InterLingVar*, *OutLingVar*, *OutTakSugVar* a *PID*). Tyto koncové třídy grafických členů se liší především ve metodách obstarávající vykreslování objektu na plochu. Většina ostatních funkcí je ale již realizováno ve třídě *GrMember*.



Obr. 6.12: Hierarchie tříd grafických objektů

Při podrobnějším popisu vnitřní struktury grafických členů se musíme zmínit o jejich pinech. Ty jsou odvozeny od společného základu s názvem *MemberPin*. Hierarchie tříd pinů

je uvedena na obrázku Obr. 6.13. Je z ní patrné, že zvlášť jsou reprezentovány vstupní piny a zvlášť výstupní. Hlavní funkcí základní třídy je udržování informace o rodičovském grafickém členu a připojených čarách. Kdežto funkce koncových tříd je čistě grafická. Obstarávají vykreslování daného pinu.



Obr. 6.13: Hierarchie tříd pinů grafických členů

Struktura celého systému je v programu udržována dvojím způsobem. K dispozici je pole obsahující všechny grafické členy. Budeme ho nazývat základní seznam grafických členů (nebo zjednodušeně prvků). Ten se používá pokud je třeba rychle nějaký prvek vyhledat nebo naopak pracovat se všemi současně (operace vykreslování objektů, autoplace, …). Druhý způsob reprezentace struktury jsou odkazy na předchůdce a následníky uvnitř všech grafických členů. Tyto odkazy nejsou přímé, ale jsou realizovány přes piny a spojovací čáry. Připomeňme jen, že výsledná struktura nemusí odpovídat souvislému grafu, ale může se skládat z několika nezávislých subgrafů.

Čáry

Spojovací čáry jsou realizovány pomocí zvláštní třídy *GrConnection*, která obsahuje jednotlivé body lomené čáry. Ty jsou pěti druhů: *InputPin*, *FixedInputPoint*, *NormalPoint*, *FixedOutputPoint*, a *OutputPin*. Jejich význam je následující. *InputPin* a *OutputPin* slouží ke spojování čáry s příslušným pinem grafického členu. Jsou reprezentovány objekty třídy *ConnectionPin*. a nelze ručně měnit jejich polohu (je určena protilehlým pinem) ani větvit v nich spojovací čáru. *NormalPoint* je běžný bod umožňující snadnou manipulaci a větvení čáry. Je reprezentován objekty třídy *ConnectionPoint*. Body *FixedInputPoint* a *FixedOutputPoint* jsou zvláštním případem běžných bodů. Jejich poloha je ale pevně určena následujícím nebo předcházejícím krajním bodem čáry. Není ji tedy možné ručně měnit, je ale možné v nich provádět větvení. Struktura a tvar celého spoje je v *GrConnection* udržována dvojím způsobem. Jednak je k dispozici pole obsahující všechny krajní body a jednak každý bod obsahuje přímý odkaz na své předchůdce a následníky.

Autoplacer

Statická třída nazvaná *Autoplacer* má za úkol rozmístit grafické členy na pracovní ploše tak, aby zabírali minimální plochu, a zároveň aby se co nejméně křížili spojnice mezi nimi.

Tato úloha nemusí mít pouze jedno řešení. Jeho nalezení je dosti obtížné a předpokládá, že se mnohokrát projde celý seznam grafických členů. To by ovšem trvalo velmi dlouho, proto jsme se rozhodli slevit z nároků na nalezené rozmístění a hledat pouze suboptimální řešení. Heuristika použitá k jeho nalezení částečně využívá procházení stavového prostoru a částečně postupuje podle hesla „nejlepší nejdřív“. Cesta (způsob umístění následujícího prvku), která se zdá být v danou chvíli nejlepší, nemusí být nejlepší z globálního hlediska. Právě proto je nalezené řešení pouze suboptimální. Stručný popis této heuristiky je následující:

1) Určení relativní hodnoty souřadnice x (x_{rel}) u všech grafických členů.
   a) Prochází se celý základní seznam grafických členů a vybírají se ty, které nemají žádného předchůdce. U nich se nastaví x_{rel} na hodnotu 1 a pokračuje se následujícím bodem.
   b) Expanduje se do všech následníků a u každého se nastaví x_{rel} na hodnotu o jedno větší než u předchůdce. Tímto způsobem se stanoví relativní souřadnice x u každého grafického členu. Pokud se dostaneme do situace, kdy se snažíme nastavit x_{rel} u prvku, který již byl ohodnocen (přišli jsem do něj z jiného předchůdce), vybereme z aktuální a nové hodnoty relativní polohy tu, která je větší.
2) Určení relativní hodnoty souřadnice y u všech grafických členů.
   a) Vytvoří se virtuální mřížka, která má tolik sloupečků, kolik je maximální hodnota x_{rel} mezi všemi prvky.
   b) Prochází se celý základní seznam grafických členů a vybírají se ty, které nemají žádného předchůdce a nejsou umístěny ve virtuální tabulce. U nich se pokračuje podle bodu 2 c). Po projití celého základního seznamu grafických členů se pokračuje bodem 2 e).
   c) Ve vybraném (pracovním) subgrafu se pokusíme nalézt prvek, k němuž vede od prvního členu cesta, která vychází a zase vstupuje vždy pouze do prvních (horních) pinů grafických členů. Tento prvek v dané větvi pracovního subgrafu nemusí vůbec existovat. V takovém případě se pokračuje v prohledávání základního seznamu grafických členů podle bodu 2 b). Druhá možnost je, že hledaných prvků může být hned několik. Vybere se ale vždy ten, který se nalezne jako první. To je jeden z důvodů proč je výsledné řešení pouze suboptimální. Tento zvolený grafický člen se vloží do virtuální tabulky do sloupečku s číslem x_{rel} na první volnou pozici. Dále se pokračuje podle následujícího bodu heuristiky.
   d) Expanduje se do všech předchůdců a následníků podle pořadí jak jsou připojeny k právě umístěnému prvku a každý se stejným způsobem jako předchozí prvek umístí do virtuální tabulky. Takto se projde celý pracovní subgraf. Po té se pokračuje v prohledávání základního seznamu grafických členů podle bodu 2 b).
   e) Po prohledání celého základního seznamu v bodě 2 b) nemusí ještě být všechny prvky umístěny do virtuální tabulky. Tento stav je způsoben tím, že ne v každém subgrafu musí nutně existovat grafický člen odpovídající podmínce 2 c). V takovém případě následuje další procházení celého základního seznamu grafických členu. Tentokrát se hledají prvky, které nemají žádného následníka a nejsou umístěny ve virtuální tabulce. S nimi se pak zachází stejně jako s prvky nalezenými v bodě 2 c). Po tomto druhém projití celého základního seznamu grafických členů jsou již všechny prvky umístěny ve virtuální tabulce. Může se tedy pokračovat bodem 3).
3) Výpočet absolutní pozice všech grafických členů.
   a) Projde se celá virtuální tabulka. Postup je uspořádaný od levého horního rohu (prvek [1,1]) k pravému spodnímu rohu tabulky. Podle umístění prvků na daných pozicích se určí jejich absolutní poloha na pracovní ploše. Horizontální vzdálenost levých okrajů sousedních prvků je pevně stanovena a nezáleží na jejich velikosti. Kdežto vertikální

vzdálenost sousedních prvků je limitována pouze minimální vzdáleností spodního okraje horního grafického členu a horního okraje spodního grafického členu.

Výhodou této heuristiky je, že není příliš časově náročná a výsledek, přestože je pouze suboptimální, v drtivé většině případů naprosto vyhovuje našim potřebám. Příklad automatického rozmístění je vidět na obrázku Obr. 6.1.

6.2.2. 2D a 3D graf

K pochopení vnitřní funkce 2D a 3D grafu je nutné znát některé části matematické algebry. Ty lze získat například v [21] nebo [22]. Jelikož práce s dvourozměrným obrazem je jen podmnožinou operací prováděných s trojrozměrnými objekty, zaměříme se především na popis kroků při zobrazení 3D grafu. Po vytvoření nového objektu 3D grafu (třída *Graph3D*) máme k dispozici dvourozměrné pole dat. Čísla řádků a sloupců určují souřadnice x, y a prvky matice jsou souřadnice z. Tato data je nutné převést na posloupnost polygonů (v našem případě čtyřúhelníků), jejichž krajní body jsou reprezentovány uspořádanou trojicí u = [x,y,z].

Prvním krokem při převodu 3D scény do 2D obrazu je určení souřadné soustavy v rovině obrazu. Tu jsme zvolili tak, aby orientace os x (horní okraj monitoru) a y (levý okraj monitoru) byla stejná jako orientace příslušných os v oblasti předlohy. Osa z pak směřuje kolmo na obrazovku. Druhým krokem je mapování velikosti a polohy grafu aby se celý vešel do vymezeného prostoru v zobrazovacím okně. K tomu je nutné nalézt minimální a maximální souřadnice bodů. Z nich se vypočítá posun a změna velikosti grafu v daném směru. Způsob provedení těchto dvou operací bude popsán dále. Takto přetransformovaný obraz se převede do dvou rozměrů pouhým vynecháním souřadnice z při vykreslování na projekční plochu (monitor). Celkový obraz pak vypadá, stejně jako když se na scénu budeme dívat kamerou umístěnou uprostřed zobrazovací plochy namířenou na střed předlohy. V obrázku nebude uplatněna perspektiva. To je samozřejmě dobře, protože nezobrazujeme reálný svět, ale graf popisující matematickou závislost výstupu na dvou vstupech.

Po převodu souřadnic z reálných na zobrazované je nutné upravit viditelnost jednotlivých polygonů. Ta musí odpovídat vzdálenosti od kamery (obrazovka). Z toho důvodu vypočteme z polohy krajních bodů střed každého polygonu a celé pole těchto prvků seřadíme od těch, které mají největší hodnotu souřadnice z k těm s nejmenší hodnotou. Při vykreslování pak postupujeme v poli stejným směrem, takže se nejvzdálenější objekty vykreslí nejdřív a pak jsou překryty bližšími.

Posledním krokem před samotným vykreslením na monitor je určení barvy jednotlivých polygonů. Pokud je nastavena jednobarevná textura, mají všechny objekty stejnou barvu. Pokud je nastavena gradientní textura, opět se vypočtou středy polygonů. Tentokrát ale ne ze souřadnic zobrazovaných bodů (ty jsou již transformované), ale z původních souřadnic. Barva jednotlivých objektů pak odpovídá velikosti souřadnice z středního bodu polygonu. Pokud je nastavena stínovaná textura, je nutné spočítat úhel, který svírá normála v reálných souřadnicích každého polygonu se zobrazovací rovinnou. Podle jeho velikosti se pak určí barva. K výpočtu normály se použije vektorový součin dvou úseček svázaných s povrchem polygonu. Prakticky jsou použity tři body z okraje polygonu.

S vykresleným 3D grafem je možno libovolně manipulovat. Rotovat ho kolem kterékoliv osy, zvětšovat a zmenšovat. Všechny tyto operace, stejně jako úvodní transformace 3D předlohy do 2D obrazu, je možné realizovat pomocí násobení matic. Jejich obecný popis je následující:

$$u_1 = A \cdot u_0$$

Kde u_0 je původní bod se souřadnicemi $[x_0, y_0, z_0]$, A je transformační matice a u_1 je výsledný bod se souřadnicemi $[x_1, y_1, z_1]$. Pro dvourozměrné objekty jsou u_0 a u_1 vektory o délce 3 a A je matice 3x3. Pro operace s trojrozměrnými objekty (náš případ) jsou u_0 a u_1 vektory o délce 4 a A je matice 4x4. Konkrétní operace jsou:

Posun

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}$$

Význam symbolů: *dx* je posun ve směru osy x, *dy* je posun ve směru osy y a *dz* je posun ve směru osy z.

Změna velikosti

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}$$

Význam symbolů: *sx* je změna velikosti ve směru osy x, *sy* je změna velikosti ve směru osy y a *sz* je změna velikosti ve směru osy z.

Rotace kolem osy x, y, z

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(u) & \sin(u) & 0 \\ 0 & -\sin(u) & \cos(u) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(u) & 0 & \sin(u) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(u) & 0 & \cos(u) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(u) & \sin(u) & 0 & 0 \\ -\sin(u) & \cos(u) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}$$

Význam symbolů: *u* je úhel o který daný objekt rotuji. Pokud je rotace opačným směrem je nutné vyměnit znaménka u funkcí sinus.

Pokud je současně provedeno s grafem několik operací, pro program to znamená pouze několikrát za sebou vynásobit původní bod různými transformačními maticemi (A_1 až A_n). Pozor záleží na pořadí násobení. Výsledný vztah pak má tvar:

$$u_1 = A_n \cdot A_{n-1} \cdot \ldots \cdot A_2 \cdot A_1 \cdot u_0$$

Knihovny Visual Studia .NET bohužel obsahují pouze funkce pracující s dvourozměrnými objekty. Proto jsme byli nuceni veškeré maticové operace naimplementovat sami. Při realizaci algoritmu převodu 3D obrazu na 2D jsme se snažili aby byl výpočet co nejrychlejší a práce s 3D grafem tak byla plynulá. Zde se ale projevila slabost jazyka C#. Nejenže výpočty trvají déle než pokud bychom použili například klasické C++, ale výrazně pomalejší je i vykreslování na obrazovku.

6.2.3. Časový graf

Operace s časovým grafem jsou obdobné jako s 2D grafem. Jediný rozdíl je ve složitější implementaci zvětšování a zmenšování měřítka grafu.V časovém grafu je při změně měřítka zachována původní velikost oblasti, na kterou je graf vykreslen (u 2D a 3D grafu ne) a mění se pouze jeho velikost uvnitř této kreslící plochy. Z toho důvodu je nutné nejen transformovat hodnoty z reálných souřadnic do zobrazovaných, ale navíc je třeba omezovat i zobrazovanou část grafu.

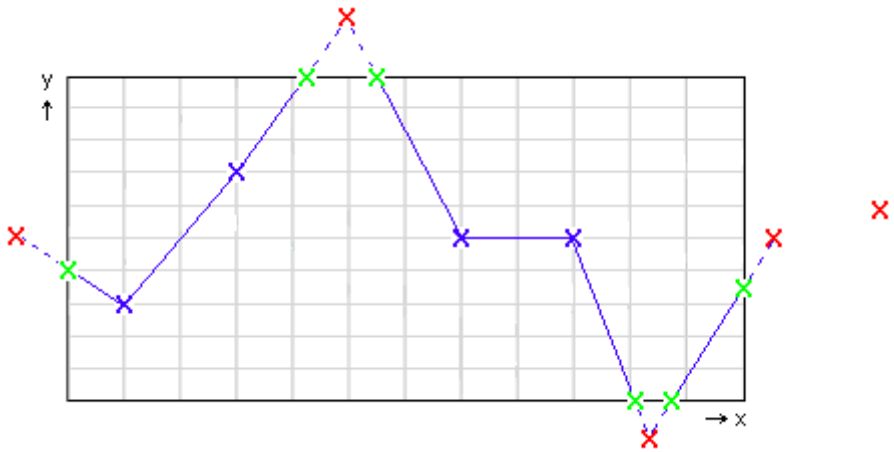
První způsob, který by mohl vést k zobrazení pouze potřebné části grafu, je vykreslení celého grafu (i mimo okno) a následné překrytí nežádoucích bodů bílou plochou. Takové řešení lze použít pouze pokud jsou hodnoty zobrazovaných bodů relativně malé. C# totiž umožňuje vykreslovat body jen se souřadnicemi do cca $[x, y] = [\pm 1 \times 10^6, \pm 1 \times 10^6]$. Pokud by například byl graf mnohonásobně zvětšen, mohl by bod jedna mít například zobrazované souřadnice [0,0] a následující bod dva zobrazované souřadnice $[1, 1 \times 10^7]$. Při pokusu o vykreslení úsečky mezi body jedna a dva by pak nastala chyba.

Lepší řešení, které jsme použili i my, je ořezávání grafu přesně na velikost kreslící plochy. Tento postup je patrný z obrázku Obr. 6.14. Na něm je vidět, že každý bod ležící mimo kreslící plochu může být zastoupen žádným jedním nebo dvěma dalšími body. K výpočtu těchto bodů je použita jednoduchá analytická geometrie. Postup výpočtu zástupných bodů probíhá zleva doprava (od nejmenší hodnoty souřadnice x k největší) a je popsán následujícím postupem. Aktuálně zkoumaný bod je označován jako p_k, předchozí bod je p_{k-1} a následující p_{k+1}.

1) Pokud je souřadnice x zkoumaného bodu p_k menší než poloha levého okraje kreslící plochy a následující bod p_{k+1} nemá souřadnici x větší než tento levý okraj, není bod p_k vykreslen ani zastoupen žádným jiným bodem.
2) Pokud je souřadnice x zkoumaného bodu p_k menší než poloha levého okraje kreslící plochy a následující bod p_{k+1} má souřadnici x větší než tento levý okraj, pak následuje jedna z těchto variant.
   a) Jestliže úsečka mezi body p_k a p_{k+1} protíná obdélník ohraničující kreslící plochu, pak je bod p_k zastoupen jedním bodem ležícím na levé hraně tohoto obdélníku.
   b) Jestliže není splněna varianta 2 a), není bod p_k vykreslen ani zastoupen žádným jiným bodem
3) Pokud je souřadnice x zkoumaného bodu p_k mezi levým a pravým okrajem kreslící plochy (včetně), nastává jedna z následujících variant.

a) Jestliže je souřadnice y bodu p_k mezi horní a spodní hranou kreslící plochy (včetně), je bod vykreslen přímo a není potřeba žádný zástupný bod.
b) Jestliže není splněn varianta 3 a), je zkoumána úsečka mezi body p_{k-1} a p_k a úsečka mezi body p_k a p_{k+1}.U obou se zjišťuje, zda protínají kreslící plochu. Pokud ano, pak je bod p_k zastoupen jedním nebo dvěma body ležícími na horní respektive spodní hraně obdélníku ohraničujícího kreslící plochu.

4) Pokud je souřadnice x zkoumaného bodu p_k větší než pravý okraj kreslící plochy a souřadnice x předcházejícího bodu p_{k-1} je menší než tato hranice, pak následuje jedna z těchto variant.
   a) Jestliže úsečka mezi body p_{k-1} a p_k protíná obdélník ohraničující kreslící plochu, pak je bod p_k zastoupen jedním bodem ležícím na pravé hraně tohoto obdélníku.
   b) Jestliže není splněna varianta 4 a), není bod p_k vykreslen ani zastoupen žádným jiným bodem
5) Pokud je souřadnice x zkoumaného bodu p_k větší než pravý okraj kreslící plochy a souřadnice x předcházejícího bodu p_{k-1} není menší než tato hranice, pak není bod p_k vykreslen ani zastoupen žádným jiným bodem.



Obr. 6.14: Struktura bodů jedné křivky v časovém grafu.

Na obrázku Obr. 6.14 je zobrazena struktura bodů jedné křivky v časovém grafu. Kreslící plocha je ohraničena černým obdélníkem, přímo zobrazované body jsou modře, nezobrazované body jsou červeně a zástupné body jsou zeleně.

Při výpočtu zobrazovaných souřadnic z reálných hodnot je nutné dávat pozor na omezení přesnosti práce jazyka C# s čísly s pohyblivou desetinou čárkou. Tyto vlastnosti jsou naštěstí dobře známi a prostředí .NET zaručuje, že vytvořený program se bude chovat stejně na každém počítači, kde se spustí. Protože vykreslovací metody v C# umožňují pracovat pouze s datovým type *float* ($\pm 1.5 \times 10^{-45}$ až $\pm 3.4 \times 10^{38}$ s přesností na 7 číslic), jsou veškeré vnitřní operace uměle převáděny do číselného typu *double* (5.0×10^{-324} až 1.7×10^{308} s přesností na 15-16 číslic). Důvodem není větší rozsah, ale větší přesnost a potažmo také stabilita výpočtu.

7. Závěr

Mezi oblasti, které by bylo dobré v navržené části uživatelského rozhraní v budoucnu zlepšit patří v grafickém modelu - doplnění možnosti kopírování existujících grafických členů, spojování prvků pouhým táhnutím myší, zjednodušení práce s čarami, dokonalejší autoplacer a autorouter, atd. Ve všech grafech – urychlení výpočtu a vykreslení grafu, možnost rolování, atd. Úprav a vylepšení, která se nabízejí, je velké množství.

V rámci diplomové práce byly porovnány vybrané komerční a akademické programy pro návrh fuzzy systémů. Byly analyzovány jejich vlastnosti se zaměřením na uživatelské rozhraní a dokumentovány přednosti i nedostatky. Přestože počet vybraných programů zdaleka neodpovídal celkovému počtu existujících produktů, lze považovat výběr za rozumný, zejména s ohledem na možné zobecnění závěrů analytické studie. Na základě teoretických poznatků a zkušeností získaných z porovnávací studie byla navržena část uživatelského rozhraní programového prostředí pro návrh, ladění a implementaci fuzzy systémů zahrnující **grafický model** fuzzy systému, **2D graf**, **3D graf** a **časový graf.** Navržené a implementované moduly vyhovují v práci definovaným kritériím hodnocení programů pro návrh fuzzy systémů.

Poděkování

Rád bych na tomto místě poděkoval vedoucímu diplomové práce panu docentu Horáčkovi za věnovaný čas a konstruktivní připomínky. Dále pak bych chtěl poděkovat ing. Renatě Pytelkové, ing. Janu Kolínskému a ing. Jaroslavu Křížovi za pomoc při realizaci programu pro návrh fuzzy systémů.

Použitá literatura

[1] Navara, M. - Olšák, P.: *Základy fuzzy množin*. Vydavatelství ČVUT, Praha, 2002.

[2] Vysoký, P.: *Fuzzy řízení*. Vydavatelství ČVUT, Praha, 1996.

[3] Klir, G.J- Yuan, B.: *Fuzzy Sets and Fuzzy Logic. Theory and Applications.* Prentice Hall PTR, New Jersey, 1995.

[4] Driankov,D. - Hellendoorn, H. - Reinfrank, M.: *An Introduction to Fuzzy Control.* Springer-Verlag, Berlin, 1993.

[5] Jang, J.S.R. - Sun, C. Mitzutani, T.: *Neuro-Fuzzy and Soft Computing: a Computation Approach to Learning and Machine Intelligence.* Prentice Hall, Inc., 1997.

[6] Hoppner, F. - Klawonn, F. - Kruse, R. - Runkler, T.: *Fuzzy cluster analysis.* John Wiley & Sons. Ltd., 1999.

[7] Marchal, B.: *XML v příkladech*. Computer Press, Praha, 2000.

[8] Box, D.: *Essential COM.* Eddison-Wesley, New York, 1998.

[9] Iwanitz, F. - Lange, J.: *OLE for Process Control.* Hüthing GmbH & Co. KG Heidelberg, Germany, 2001.

[10] Novák, V.: *Fuzzy množiny a jejich aplikace.* SNTL, Praha, 1990.

[11] Novák, V., Perfilieva, I., Močkoř, J.: *Mathematical Principles of Fuzzy Logic.* Kluwer, Boston, 1999.

[12] Novák, V.: *Základy fuzzy modelování.* BEN, Praha, 2000.

[13] *Velký slovník naučný.* Diderot, Praha, 1999.

[14] Uživatelská příručka - *Fuzzy Logic Toolbox User's Guide (elektronická podoba),* The MathWorks Inc., 1999.

[15] Uživatelská příručka – *ECANSE (elektronická podoba).* Siemens AG Austria, Program and System Engineering, PSE NLT, Gudrunstrasse 11, A-1100 Vienna, Austria, 1998.

[16] Uživatelská příručka - *Fuzzy Control Manger (elektronická podoba)* . TransferTech GmbH, Cyriaksring 9A, D-38118 Braunschweig, Germany, 1996.

[17] Uživatelská příručka - *FuzzyControl++ (elektronická podoba).* Siemens AG, A&D SH 23, Hugo-Junkers-Straße 7, 90411 Nürnberg, Germany ,1998.

[18] Uživatelská příručka - *FuzzyTECH Help (elektronická podoba)*. INFORM GmbH, Pascalstr. 23, D-52076 Aachen, Germany, 2001.

[19] Uživatelská příručka - *LFLC 2000 (elektronická podoba).* Ústav pro Výzkum a Aplikace Fuzzy Modelování, Ostravská Univerzita v Ostravě, 30. Dubna 22, Ostrava, 2002.

[20] Gunnerson, E.: *Začínáme programovat v C#* . Computer Press, Praha, 2001.

[21] Smutný, V.: *Přednášky z předmětu Robotika.* ČVUT, Praha, 2002.

[22] Krajník, E.: *Maticový počet.* ČVUT, Praha, 2000.

[23] Internetová adresa *http://tarr.uspto.gov.* Seznam zaregistrovaných značek fuzzy aplikací na území USA.

[24] Internetová adresa *http://www.it.uom.gr/pdp/DigitalLib/Fuzzy/fuzzy_soft.htm*. Přehled některých aplikací pro vývoj fuzzy systémů.

Příloha A

Přehled některých významných aplikací oficiálně zaregistrovaných na území USA a několika dalších zajímavých produktů zabývajících se návrhem fuzzy systémů.

(zdrojem informací byly internetové adresy [23] a [24])

Seznam několika vybraných produktů používaných pro vývoj fuzzy systémů		
Název produktu	**Výrobce**	**Kontakt na výrobce**
FIDE	Aptronix	http://www.aptronix.com/fide/fide.htm
Fuzzy Logic	Wolfram Research	http://www.wolfram.com/products/applications/fuzzylogic/
Fuzzy Logic Toolbox	Mathworks	http://www.mathworks.com/products/fuzzylogic/
fuzzyTECH	Inform	http://www.fuzzytech.com
Fuzzy Control Design Toolbox	Addlink	http://www.addlink.es
Xfuzzy 2.1	Centro Nacional de Microelectronica, Sevilla	http://www.imse.cnm.es/Xfuzzy/
fuzzyBUILDER	Intel	http://www.intel.com/design/auto/
FPDynamics	Predictive Dynamix, Inc.	http://www.predictivedynamix.com
Fuzzy Control Design Toolbox (FCD)	ExpertControl inc.	http://www.expertcontrol.com
FUZZY KNOWLEDGE BUILDER	FUZZY SYSTEMS ENGINEERING	http://www.fuzzysys.com/fkbprod.html
FUZZYSTUDIO	SGS-THOMSON	http://eu.st.com/stonline/
FUZZYLOG	Buhler Ltd.	(Tento projekt již není aktivní)
XFUZZY	Expersoft Corporation California	http://www.imse.cnm.es/Xfuzzy/index.htm
FuzzyDesigner	University of Kaiserslautern	http://www.e-technik.uni-kl.de/litz/software/fuzzy.htm
PROFUZZY	Siemens AG	http://www.sea.siemens.com/ssi/include/trademark.html
FuzzyControl	Siemens AG	http://www.ad.siemens.de/fuzzycontrol/
FuzzyControl++	Siemens AG	http://www.ad.siemens.de/fuzzycontrol/
NeuroSystems	Siemens AG	http://www.ad.siemens.de/fuzzycontrol/
Fuzzylab 1.1	Anjan Rakshit	http://www.simtel.net/pub/pd/5781.html
fuzzyLAB	Microchip Technology inc. USA	http://www.mocrochip.com
FS-FUZZYSOFT	FUZZY SOFT LTD.	(Tento projekt již není aktivní)
FS-FUZZYEXPERT	FUZZY SOFT LTD.	(Tento projekt již není aktivní)
Fuzzy Query	Sonalysts Inc.	http://fuzzy.sonalysts.com/
FlexTool	CynapSys	http://cynapsys.com
FID Fuzzy Development Tool	FDG-Austria	http://www.fdg.donau-uni.ac.at/FID.HTM
ECANSE 2000	Siemens AG	http://www.siemens.at/pse/ecanse/
Fuzzy Control Manager	TransferTech GmbH	http://www.transfertech.de/www/soft_e.htm
LFLC 2000	Ostravská univerzita	http://ifm.osu.cz/
A-B Flex	Allen-Bradley	(Tento projekt již není aktivní)

Příloha B

Přehled výsledků testování aplikací podle jednotlivých kritérií.

Výsledky testů aplikací pro návrh fuzzy systémů						
Kritérium	**FLT**	**ECANSE**	**FCM**	**FC++**	**FTECH**	**LFLC**
Různé druhy ovládání	✗	✗	○	○	✓	✗
Používání standardů	○	✗	○	○	✓	✗
Přehlednost ovládání	✓	✗	✗	✓	✓	✓
Jednotný systém ovládání	✓	✓	○	✓	✓	✓
Manuál a Nápověda	✓	○	○	○	✓	✓
Operace „Zpět“ a „Opakovat“	○	✗	✗	✗	○	✗
Prostředí na míru	✗	✗	✗	✗	✓	✗
Nástroje usnadnění	○	✗	○	✓	✓	○
Bezchybnost algoritmu	✓	✓	○	✓	✓	✓
Úplnost programu	○	✗	○	✓	✓	✗
Možnosti ladění	✓	○	○	✓	✓	✗
Přehlednost výstupních dat	○	✗	○	✓	✓	✗
Použitelnost výstupních dat	✓	✓	○	✓	✓	✗
Komunikace s jinými programy	✓	✓	✓	✓	✓	○
Komunikace s periferiemi	✓	✓	✓	✓	✓	✗
Zdravotní kritérium	○	○	✓	✓	○	○
Estetika programu	✗	✗	○	✓	✓	○
Nároky programu na HW	✓	○	✓	✓	✓	✓
Možné struktury systému	○	○	✓	○	✓	✗
Editování struktury modelu	○	✗	✓	✓	✓	○
Kontrola správnosti struktury	○	○	✗	✓	✓	✓
Přehlednost struktury	✓	✗	✓	✓	○	✓
Druhy I/O	✗	✗	✗	✗	○	✗
Operace s I/O	○	○	✗	✗	○	✗
Typy jazykových proměnných	✓	✓	✗	✗	✓	○
Zobrazení funkcí příslušnosti	○	✗	○	○	✓	✗
Vytváření nových fcí. přísluš.	✓	✗	✗	✓	✓	○
Editace funkcí příslušnosti	○	✗	✗	✓	✓	○
Zobrazení pravidel	✓	✓	✓	✓	✓	✓
Vytváření nových pravidel	✓	○	✓	✓	✓	✓
Editace pravidel	✓	○	✓	✓	✓	✓
Vyhodnocování fuzzy systému	○	✓	○	○	✓	✓
Simulace systému	✓	○	○	✓	✓	○
Zobrazení výsledků	✓	✗	○	✓	✓	✗
Rozšířené funkce	✓	✗	○	○	✓	○

Zkratky jednotlivých názvů testovaných programů jsou:

FLT	Fuzzy Logic Toolbox pro Matlab
ECANSE	Fuzzy toolbox programu ECANSE
FCM	Fuzzy Control Manger
FC++	FuzzyControl++
FTECH	FuzzyTECH
LFLC	LFLC 2000

Kritéria jsou pojmenována a řazena stejně jako v textu. Jejich splnění je hodnoceno třemi způsoby:

✓	Vyhovuje
●	Vyhovuje pouze částečně
✗	Nevyhovuje