ACTA UNIVERSITATIS AGRICULTURAE ET SILVICULTURAE MENDELIANAE BRUNENSIS

Ročník LVIII 25 Číslo 3, 2010

TEORETICKÝ MODEL SYSTÉMU PRO OPTIMALIZOVANÝ PROCES VÝBĚRU PROGRAMOVÉHO VYBAVENÍ

J. Rybička, P. Talandová, J. Přichystal

Došlo: 18. prosince 2009

Abstract

RYBIČKA, J., TALANDOVÁ, P., PŘICHYSTAL, J.: *Theoretical model of system for optimized software selection process.* Acta univ. agric. et silvic. Mendel. Brun., 2010, LVIII, No. 3, pp. 233–242

Selection of a suitable software is an everyday problem for many users. This process if often ineffective, as the users usually work only with a restricted set of programs and are unable to have appropriate knowledge about all programs available. The paper therefore deals with the model of a system for optimized software selection process. Applications of this kind are already available online, but they are usually aimed at narrow-band area, recommending often commercial programs only. Users also cannot influence the process of selection. The system described in this paper removes this insufficiency. A mathematical model is designed, which works with input sets of users' requirements and programs' properties, recommending an optimal solution. The system is designed as open, extensible and accessible and is oriented on users and their needs.

document, optimization, program properties, software selection, user requirements

Pro zpracování dat na počítači existuje široká škála programového vybavení. Uživatel přicházející s konkrétními požadavky stojí před úkolem vybrat z dostupné množiny programů takový systém, který pokud možno co nejlépe vyhoví při splnění požadovaných vlastností a umožní realizovat zamýšlený proces.

Programových systémů je velké množství a často disponují i velkým množstvím služeb. Uživatel se těžko může orientovat ve všech systémech a jejich nabídkách, zvláště když se situace dynamicky mění, vznikají nové verze programů a jejich služeb. Uživatelé tedy často pracují s omezenou množinou programů, aniž by uvažovali o efektivitě zpracování, lepších variantách a vhodnosti volby pro daný úkol.

Uveďme příklad zpracování vektorového obrázku. Aniž bychom zdaleka vyčerpali všechny možnosti, můžeme pro tento účel použít:

- vektorový editor InkScape, Adobe Illustrator, Corel Draw a další,
- nabídku Kreslení v programu MS Word nebo OpenOffice.org Writer,
- naskenování předlohy a převod trasovacím programem (například Corel Trace),
- tabulkový kalkulátor s nabídkou vykreslení grafů s doplňkem libovolných dalších obrazových objektů,
- prostředí `picture` v systému LaTeX,
- balík `pdftricks` v systému TeX,
- knihovnu Graph v jazyce Turbo Pascal,
- přímou editaci souboru formátu SVG v libovolném programovém editoru,
- některý ze systémů typu CAD,
- specializované programy pro kreslení schémat a diagramů, například MS Visio a další.

Z tohoto i neúplného přehledu vyplývá, že každá varianta skýtá někdy zcela zásadně odlišné možnosti, které běžný uživatel není schopen souhrnně vyhodnotit. S tím souvisí i skutečnost, že mnohdy jsou požadavky uživatele předem determinovány zamýšleným programem, což vede k apriorním omezením.

Nepochybně je nad síly každého jedince ovládnout množství informací o současném programovém vybavení v potřebné šíři i hloubce. Podobně jako se ukazuje, že souhrn znalostí není možné řešit formou tištěných encyklopedií a slovníků, jako

tomu bylo do nedávné minulosti, ale spíše kolaborativní a postupnou formou jako u projektu Wikipedia, tak i znalosti o programech a jejich službách je účelné soustřeďovat v elektronické podobě, z níž pak lze automatizovaně získat hledanou informaci.

Cílem článku je prezentace myšlenky automatizovaného systému informací o programových produktech a o jejich službách, který na základě požadavků uživatele doporučí vhodné použití vybraného programu.

PŘEHLED LITERATURY A SOUČASNÉHO STAVU

Nachází-li se uživatel v situaci, kdy potřebuje řešit určitý problém a potřebuje vybrat odpovídající program, může sáhnout po tištěných zdrojích (přehledy a srovnání vybraných programů převážně v odborných časopisech), po elektronických zdrojích (srovnávání různých produktů) nebo po specializovaných aplikacích zaměřených na konkrétní okruh požadavků.

V oblasti srovnávání různých výrobků je dobře znám například časopis Test (Test, 2009), srovnávající vlastnosti vybraných reprezentantů okruhu výrobků daného typu. Je však zaměřen na produkty (mezi ně obvykle nepatří software) a požadavky uživatele (v tomto kontextu spíše zákazníka) řeší jen okrajově.

V oblasti volby softwaru lze nalézt doporučení pro výběr, která jsou buď zcela obecná, nebo jsou zaměřena na určitou aplikační oblast (např. Komoski a Plotnick, 1995; Ekonomické softwary, 2009).

Podstatně častější jsou přehledové články nebo souhrnné tabulky vybraných programů s parametry zvolenými podle očekávaných požadavků uživatele. Představitelem takového zdroje může být přehled editorů pro pořizování zdrojových textů v systému TEX (Editory, 2009). Tyto tabulky jsou však statické a uživatel nemůže ovlivnit typ ani počet zobrazených programů ani jejich vlastností.

Částečně užitečný může být zdroj porovnávající produkty podobného typu na základě zvolených kritérií respektujících předpokládané požadavky uživatele. Příkladem může být srovnání kancelářských balíků (Skovajsová, 2001) nebo operačních systémů (Drdla, 2006).

Z hlediska uživatele je nejvhodnější takový přístup, kdy je proces výběru softwaru přizpůsoben a uživatel si může volit, které parametry jsou pro něj podstatné. Proces výběru je v tomto případě podporován aplikací, která na základě zvolených požadavků (a popř. jejich váhy) doporučí softwarové řešení. Jde o tzv. RFP (Request For Proposals), kdy potenciální zákazník specifikuje požadavky a firma předloží nabídku. Výsledkem by mělo být ohodnocení požadavků zákazníka a/nebo podklady pro výběr. Tyto podklady se dále analyzují za použití rozhodovací matice.

Pro podporu volby vhodného softwaru existuje řada aplikací, které se obvykle orientují na relativně úzce vymezené oblasti související s řízením podniku nebo jeho části. Nejčastějšími oblastmi je plánování podnikových zdrojů (ERP), řízení vztahů se zákazníky (CRM), management lidských zdrojů, účetnictví a mzdy, popř. strojírenství. Ukázkou aplikace, která detailně zjišťuje potřeby zákazníka, může být Accounting Software Library… (2009). Popsaný přístup lze aplikovat nejen na software, používá se často i pro výrobky (např. Katalog mobilů, 2009).

Aplikace jsou obvykle dostupné online, s rozhraním v podobě webového formuláře. Ve většině případů není vyžadována registrace uživatele a aplikace pro výběr je přímo přístupná. Ve formuláři uživatel přiděluje ohodnocení jednotlivým parametrům programů, popř. odpovídá na otázky o firmě a o svých preferencích. Množina parametrů programů, ke kterým se může uživatel vyjádřit, je ovšem statická, bez možnosti přidat nové parametry. Při hodnocení přicházejí v úvahu, podle typu parametru, následující možnosti: přímé zadání údaje (čísla), výběr z množiny předdefinovaných voleb, určení míry důležitosti (váhy) daného parametru.

Aplikace pro výběr softwaru se vyznačují rozdílnou mírou podrobnosti (a tím i složitosti a časové náročnosti). Počet hodnocených parametrů se pohybuje v desítkách až stovkách. Proces výběru je proto rozčleněn do několika kroků nebo tematických kategorií. Uživatel má rovněž možnost nevyjádřit se k některému parametru, kategorii parametrů nebo kroku výběru, může také zvolit indiferentní možnost. Na výsledek pak mají vliv jen ohodnocené parametry.

Na základě vyhodnocení parametrů a preferencí zadaných uživatelem je zpracováno výsledné doporučení. Výsledek je prezentován v některé z těchto forem:

- seznam vyhovujících/nevyhovujících produktů,
- seznam produktů s určením, z kolika procent vyhovují,
- tabulka produktů a jejich parametrů, které může uživatel dále samostatně analyzovat a srovnávat a popř. odstranit z výběru ty produkty, které nepreferuje.

Algoritmus výběru vhodného softwaru však není zveřejňován, možný přístup popisuje např. Bandor (2006).

Pro hodnocení a výběr programového vybavení (ale i obecnější vícekriteriální rozhodovací úlohy) lze využít i metodu AHP (Analytic Hierarchy Process). Zásadní nevýhodou této metody je použití pevně stanovených kritérií, u nichž se (expertně) stanovují váhy. Určitá vylepšení fuzzyfikací popisují Wang a Chen (2007).

Popsaný způsob výběru softwaru je typický pro komerční aplikace. Vzhledem k tomuto zaměření bývá součástí výsledného přehledu také kontakt na prodejce příslušného softwaru.

Předpoklad komerčního využití ovlivňuje i způsob výběru. Předpokládá se, že výběr softwaru je jednorázová záležitost, jejímž výstupem je koupě a zavedení jednoho programového produktu,

který bude co nejlépe vyhovovat potřebám organizace. Jako možná řešení jsou tedy vždy doporučovány programy jednotlivě, bez možnosti kombinace nebo návazností na jiné programy.

Pokud však bude pozornost organizace soustředěna spíše na projektové úkoly než na dlouhodobou práci s jedním programem, nebude již popsaný způsob dostačující. V případě řešení určitého projektu (úkolu) může být vhodnější takové doporučení, které bude uživatele informovat o možných návaznostech programů a o komponentách programů vhodných pro řešený úkol.

Uvedená řešení rozhodovacího procesu při výběru vhodného programového vybavení jsou často jednoúčelová (pro jeden typ problému nebo izolovanou kategorii programů). Jejich společnou nevýhodou je rovněž poměrně složité nastavování vstupních hodnot (atributů) a stanovení jejich vah, i přesto nemusí být zahrnuto vše, co uživatel potřebuje. Z toho rovněž vyplývá, že rozhodovací metody nejsou v praxi příliš rozšířeny.

Metody výběru jsou založeny na pokud možno podrobné specifikaci technických parametrů, které ovšem uživatel většinou nezná, nebo je není schopen kvalifikovaně určit. Vhodnější je uživateli nabídnout možnost vyjádřit cílové potřeby a na jejich základě zkonstruovat vhodnou sestavu technických parametrů.

MATERIÁL A METODY

Při prvním přiblížení lze rozhodovací proces rozdělit na dva dílčí problémy:

- precizní specifikace požadavků na výsledná data,
- důkladná znalost použitelného programového vybavení.

Pro snížení mohutnosti množiny použitelného programového vybavení budeme v dalším textu předpokládat pouze vybranou aplikační oblast. Tato oblast by měla zahrnovat dostatečné množství různorodého programového vybavení, které je možné používat i ve vzájemných kombinacích s možnostmi přenosu dat v různých formátech. Aplikační oblastí splňující uvedená kritéria může být oblast zpracování textů počítačem, přičemž výslednými daty budou požadované dokumenty. Téměř všichni uživatelé počítačů přicházejí s programy této oblasti do kontaktu, vzniká tedy velké množství rozdílných požadavků.

Množina požadavků na dokument, podobně jako množina dostupného programového vybavení pro jeho zpracování, má stále vzrůstající kardinalitu. Z toho automaticky plyne, že celý proces má stále větší množství variant, z nichž je stále komplikovanější vybírat optimální řešení.

Návrh obecného rozhodovacího systému řešícího výběr vhodného programového vybavení musí zahrnovat formalizaci požadavků, formalizaci vlastností zvolených programů a potřebné operace nad těmito strukturami. Formalizaci lze provést více způsoby, pro naše účely jsme zvolili algebraické struktury s odpovídajícími operacemi.

Princip činnosti rozhodovacího systému

Základní princip činnosti tohoto systému lze charakterizovat v následujících dvou bodech:

- Prvním krokem je specifikace relevantních požadavků na dokument a specifikace zvláštních požadavků uživatele. První množinu lze reprezentovat množinou obvyklých typů dokumentů, z nichž uživatel vybere ten, který se nejvíce blíží požadovanému dokumentu. Tím jsou do značné míry definovány instantní požadavky, o nichž už uživatel nemusí vůbec přemýšlet. K instantním požadavkům se uživatel může vyjádřit přidáním vlastních nároků nebo redukcí těch předdefinovaných. Tak lze s minimálním úsilím dospět k optimální množině požadavků, tvořících základ pro další rozhodování. Zvláštní potřeby uživatele týkající se uživatelského prostředí a vlastností programového vybavení jsou nezávislé na typu dokumentu a doplňují množinu vstupních požadavků.
- Druhý krok je do značné míry determinován výsledkem prvního kroku. Při rozhodování o použitém programovém systému je však potřeba detailně znát všechny vstupní požadavky a všechny relevantní vlastnosti dostupných programů. Automatizovaný systém je v tomto místě schopen množinu požadavků porovnat s množinami dostupných vlastností programových systémů a vybrat takový systém nebo takovou kombinaci systémů, která pokud možno beze zbytku nebo jen s minimálními nedostatky vyhovuje vstupním požadavkům.

Samotná informace o tom, který programový prostředek je pro požadovaný dokument optimální nebo jakou posloupností použití několika programů lze zadaný problém řešit, však uživateli nemusí přinést dostatečný užitek. Může se stát, že s takovými programy uživatel není dostatečně dobře seznámen a není schopen vstupní požadavky v daných programech realizovat. Výsledkem by tedy měl být i dostatečně podrobný návod, jak lze všechny požadované prvky ve vybraném programu vyřešit.

FORMÁLNÍ MODEL SYSTÉMU

Obecný postup činnosti rozhodovacího systému řešícího výběr vhodného programového vybavení lze formálně popsat modelem, v němž jsou zahrnuty oba uvedené kroky. Model je založen na principu propojení vstupních požadavků na dokument a odpovídajících vlastností systému programových komponent.

Dokumenty a vstupní požadavky

Pro reprezentaci požadavků na dokument a požadavků uživatele je zvolena pro optimální zpracování možnost pouze binárního ohodnocení. Ohodnocení 1 u daného požadavku dokumentu znamená, že tento požadavek je při zpracování vyžadován. Veškeré požadavky je tedy nutné atomizovat tak, aby bylo možné binární ohodnocení použít. Tím na jedné straně vzrůstá počet těchto požadavků a s tím jsou spojeny určité obtíže při jejich stanovo-

vání, na druhé straně je však zajištěna jednoznačnost a srozumitelnost.

Nechť $\Pi = \{\Pi_1, \ldots, \Pi_m\}$ je množina identifikátorů vlastností a $B = \{0, 1\}$ jsou možná ohodnocení. Pak množina všech možných požadavků a jejich ohodnocení je

$$E = \Pi \times B. \tag{1}$$

Předpokládáme, že systém je pro snadnější použití vybaven některými typy dokumentů, u nichž jsou již běžně požadované vlastnosti doplněny. Těchto vzorových dokumentů je možné využít pro odvození jednotlivých dokumentních instancí. Uživatel specifikuje pouze zvláštnosti konkrétní instance dokumentního typu. Předpokládejme tedy, že pojmem dokument je dále myšlen jak předdefinovaný dokumentní typ, tak i instance z tohoto typu odvozená a modifikovaná uživatelem.

Nechť $D = \{D_1, \ldots, D_c\}$ je množina dokumentů. Každý dokument $D_i \in D$ je popsán množinou

$$E_{di} \subset E = \{e_{i1}, \ldots, e_{ig}\}. \tag{2}$$

Hodnota g je pro všechny $D_i \in D$ stejná a představuje kardinalitu sjednocení všech vlastností všech dokumentů.

Požadavky uživatele na zpracovávající systém a jeho rozhraní můžeme vyjádřit množinou

$$E_{ui} \subset E = \{e_{i1}, \ldots, e_{ih}\}. \tag{3}$$

Přitom platí, že $E_{di} \cup E_{ui} = E$ a $g + h = m = \text{card}\, E$.

Bez újmy na obecnosti můžeme předpokládat, že množina E je uspořádaná. Kritérium uspořádání není z hlediska modelu relevantní, při implementaci lze uvažovat lexikografické uspořádání podle identifikátorů (případně vzestupné uspořádání podle indexů, jsou-li použity pro identifikaci jednotlivých požadavků).[1]

Systém programového vybavení

Nechť P je množina veškerého dostupného programového vybavení $P = \{p_1, \ldots, p_q\}$. Pro účely rozhodování o tom, jaký dokument lze nějakým programem zpracovávat, musí existovat systém korespondujících vlastností programového vybavení a požadavků na dokument. Konstrukce tedy bude obdobná jako v případě požadavků na dokument, je však rozšířena o povinný prvek tzv. anotace[2]:

Pro každé $p_i \in P$ existuje množina anotovaných vlastností $\Gamma_{p_i} = S_{p_i} \times B \times A_{p_i}$, (4)

kde S_{p_i} je množina identifikátorů vlastností, $S_{p_i} = \{\sigma_{p_i 1}, \ldots, \sigma_{p_i m}\}$, B je ohodnocení $B = \{0, 1\}$ a A_{p_i} je odpovídající anotace vlastností, $A_{p_i} = \{\alpha_{p_i 1}, \ldots, \alpha_{p_i m}\}$. Anotaci lze chápat jako text reprezentující postup realizace dané vlastnosti v daném programovém systému. Tvoří vodítko zobrazované ve výsledném souhrnu doporučení ke zpracování požadovaného dokumentu.

Korespondence s požadavky na dokument je zajištěna množinou identifikátorů, jejíž kardinalita je m a existuje bijektivní zobrazení množiny S na množinu Π.[3]

Vlastnost, kterou lze u daného programu p_i vypnout, má v množině Γ_{p_i} prvek s ohodnocením 1 pro zapnutý stav, tatáž vlastnost ve vypnutém stavu je realizována jako další prvek rovněž s ohodnocením 1. Stejně je realizován i požadavek dokumentu.

Stejně jako u množiny požadavků na dokument lze i u množiny anotovaných vlastností při implementaci uvažovat uspořádání. Z implementačního hlediska je však vhodné toto uspořádání volit shodně podle bijekce S a Π.

Dokumentní formáty

Dokumentní formát je jeden z klíčových parametrů každého programového systému. Výsledný formát je rovněž často primárním požadavkem uživatele na daný dokument. Obě uvedené vlastnosti jsou doplněny ještě třetím, zásadním aspektem – dokumentní formát determinuje možnou návaznost dvou programových systémů: export do daného formátu v jednom programu a následný import tohoto formátu ve druhém programu.

Podpora dokumentních formátů programovým vybavením je tedy modelována jednak jako vlastnost příslušného programu, ale také jako struktura umožňující detekovat možné návaznosti ve zpracování dokumentu rozdílnými programy.

Dokumentní formáty jsou často popisovány jednotlivými vlastnostmi. Podle nich lze každý dokumentní formát přesně specifikovat. Jeden a tentýž typ formátu může mít různé verze, které mohou mít rozdílné vlastnosti a jsou jedním a tímtéž programem podporovány různě, proto každá verze formátu funguje jako samostatný formát. Lze také předpokládat, že dva programy generující „stejný“ formát dávají v praxi rozdílné výsledky.

Možnost vstupu nebo výstupu příslušného formátu u daného programu je popsána binárně – předpokládáme, že daný program je schopen daný formát bezezbytku zpracovat na vstupu, resp. zcela správně generovat na výstupu. Tento stav lze pova-

1 Na způsobu identifikace jednotlivých požadavků či vlastností není model závislý. Jsou zde zmíněny dva nejčastěji volené způsoby – textové identifikátory a číselné indexy, lze však zvolit i způsob jiný, bude-li možnost jej vhodně využít při implementaci.

2 Pojem „anotace“ je zvolen zejména proto, že položka obvykle nevyjadřuje vyčerpávajícím způsobem potřebnou skutečnost, ale slouží jako odkaz na podrobnější vysvětlení.

3 Podmínka bijektivního zobrazení množin S a Π musí být splněna při volbě jakékoliv identifikace požadavků na dokument a vlastností programových systémů. Bez újmy na obecnosti lze dokonce stanovit $S \equiv \Pi$, čímž je bijektivní zobrazení automaticky zajištěno.

žovat za ideální, stoprocentně bývá v praxi bohužel splněn málokdy. Jako akceptovatelný formát tedy pravděpodobně může být zahrnut i takový formát, u něhož stoprocentní zpracování nebo generování není zajištěno, rozdíly od ideálního stavu však nejsou pro uživatele patrné.

Nechť existuje množina dokumentních formátů $F = \{f_1, \ldots, f_n\}$. Dále uvažujme dvě binární relace M_v a M_w na množině $P \times F$. Prvek relace $M_v(p_i, f_j)$ existuje právě tehdy, když existuje anotovaná vlastnost

$$(\text{import } f_j, 1, \ldots) \in \Gamma_{p_i}, \tag{5}$$

analogicky prvek relace $M_v(p_i, f_j)$ existuje právě tehdy, když existuje anotovaná vlastnost

$$(\text{export } f_j, 1, \ldots) \in \Gamma_{p_i}. \tag{6}$$

Relace M_v tedy vyjadřuje schopnosti importu a relace M_w schopnosti exportu dokumentních formátů jednotlivými programy.

Návaznosti programových systémů

Můžeme-li dokument zpracovaný v jednom programu přenést do jiného programu, je tato skutečnost realizována exportem a následným importem vhodného dokumentního formátu. Pomocí relací M_v a M_w můžeme tedy pro vyjádření možných návazností zpracování dokumentů v jednotlivých programech definovat na množině P binární relaci R, kde platí, že

$$R(p_i, p_j) \equiv M_w(p_i, f_k) = M_v(p_j, f_k) = 1$$

$$\text{pro nějaké } f_k \in \{f_1, \ldots, f_n\}. \tag{7}$$

Relace R umožňuje vytvořit posloupnosti programů, mezi nimiž existují vazby prostřednictvím nějakého dokumentního formátu. Na systém (P, R) lze nahlížet jako na orientovaný graf, kde množinu uzlů reprezentují jednotlivé programové systémy a množinu orientovaných hran mezi nimi relace R.

Z tohoto grafu lze zkonstruovat systém množin programů nad P, který označíme $\Psi = \{\rho_1, \ldots, \rho_t\}$, podle následujících pravidel:

1. Množina

   $$\rho = \{p_i\} \; \forall \; i \in \{1, \ldots, q\} \tag{8}$$

   je prvkem Ψ.

2. Je-li množina $\rho = \{p_i, \ldots, p_j\} \in \Psi$ a zároveň existuje $p_k \notin \rho$, pro něž platí $R(p_j, p_k)$, pak

   $$\rho \cup \{p_k\} \in \Psi. \tag{9}$$

Systém množin Ψ obsahuje všechny jednoprvkové množiny s jednotlivými programovými systémy (vztah 8) a dále všechny posloupné řetězce (dvojice, trojice, …) programů, které mají vazbu prostřednictvím nějakého dokumentního formátu (vztah 9), tj. množiny programů ležících ve zmíněném orientovaném grafu na nějaké cestě.

Zaveďme funkci ϕ: $2^P \to I$ definovanou vztahem $\phi(\rho) = \mathrm{N}$ právě tehdy, je-li $\rho = \{p_1, \ldots, p_N\} \in \Psi$.

Každý prvek ρ množiny Ψ je ohodnocen množinou anotovaných vlastností Γ_ρ vzniklou zobecněným sjednocením vlastností všech zahrnutých programů, včetně anotací týkajících se přenosu dat exportem a importem mezi jednotlivými systémy.

Je-li tedy $\rho = \{p_i, \ldots, p_j\} \in \Psi$, pak můžeme symbolicky psát

$$\Gamma_\rho = \bigcup_{x=i}^{j} \Gamma_{p_x}. \tag{10}$$

Nechť existuje funkce U: $\Gamma \to \vec{\alpha}$ získávající z množiny anotovaných vlastností uspořádaný vektor anotací, funkce V: $\Gamma \to \vec{b}$ získávající z množiny anotovaných vlastností uspořádaný vektor jejich ohodnocení a funkce W: $\mathrm{E} \to \vec{b}$ získávající z požadavků na dokument uspořádaný vektor jejich ohodnocení.

Dále nechť existuje operace $\circ$: $\vec{b} \times \vec{b} \to \vec{b}$ vytvářející ze dvou stejně uspořádaných vektorů ohodnocení výsledný vektor ohodnocení získaný jako logický součin stejnolehlých prvků vstupních vektorů, operace $\bullet$: $\vec{b} \times \vec{b} \to \vec{b}$ vytvářející ze dvou stejně uspořádaných vektorů ohodnocení výsledný vektor získaný jako logický součet stejnolehlých prvků vstupních vektorů a operace $\odot$: $\vec{\alpha} \times \vec{\alpha} \to \vec{\alpha}$ vytvářející ze dvou stejně uspořádaných vektorů anotací výsledný vektor anotací získaný jako zřetězení stejnolehlých prvků vstupních vektorů.

Funkce V, resp. W mohou využívat uspořádanosti množin Γ, resp. E_i. Výsledné vektory pak sledují totéž uspořádání.

Operaci zobecněného sjednocení vlastností všech zahrnutých programů v množině $\rho \in \Psi$ můžeme na základě uvedených operací definovat jako po dvojicích aplikovanou operaci $\bullet$ na vektorech ohodnocení a zřetězení anotací na vektorech anotací:

$$V(\Gamma) = V(\Gamma_{p_i}) \bullet V(\Gamma_{p_i}) \tag{11}$$

$$U(\Gamma) = U(\Gamma_{p_i}) \odot U(\Gamma_{p_i}) \tag{12}$$

Spojení požadavků na dokument a vlastností programů

Nechť $E_i \subseteq E = \{e_{i1}, \ldots, e_{im}\}$ je množina požadavků dokumentu D_i na zpracování. Dále nechť Γ_ρ je množina vzniklá sjednocením vlastností Γ_i pro $i \in \{j, \ldots, k\}$ skupiny programových systémů $\rho = \{p_j, \ldots, p_k\} \subseteq \Psi$.

Označme symbolem ξ každý prvek množiny Ψ splňující podmínku

$$V(\Gamma_\xi) \circ W(E_i) = W(E_i). \tag{13}$$

Pak množina

$$\Xi = \{\xi_1, \ldots, \xi_n\} \tag{14}$$

představuje všechny vyhovující skupiny programových systémů. Vektor požadavků na dokument je

zde pokryt vektorem vlastností určité množiny programových systémů.

Za určité konstelace požadavků se však také může stát, že podmínka 13 nebude nikdy splněna a množina Ξ bude prázdná a hodnota *n* ve vztahu 14 bude rovna nule. V tom případě je nezbytné vyjádřit alespoň částečný výsledek, prezentovat míru shody požadavků a vymezit všechny nalezené problémy.

Podmínku 13 modifikujeme takto: Nejprve bude stanovena míra neshody Y_j pro všechna $\xi_j \in \Psi$, kde $j \in \{1, \ldots, t\}$:

$$\vec{Y}_j = [V(\Gamma_{\xi_j}) \circ W(E_i)] \sim W(E_i), \tag{15}$$

kde operace ~: $\vec{b} \times \vec{b} \rightarrow \vec{b}$ provede ekvivalenci binárních vektorů. Následně bude do množiny Ξ zařazen každý prvek ξ_j splňující podmínku

$$\max_{1\ldots t} \sum_{k=1}^{m} Y_{jk}. \tag{16}$$

Množina Ξ může mít více prvků, pro které bylo získáno stejné maximum.

Optimálním řešením výběru programového vybavení je skupina $X \in \Xi$ taková, že

$$\phi(X) = \min(\phi(\xi_l)) \text{ pro } l \in \{1, \ldots, r\}. \tag{17}$$

Preferujeme řešení, která využívají co nejmenší počet programových systémů. Za ideální (a také v praxi často dosažitelný) lze považovat stav, kdy výsledná množina programových systémů je jednoprvková. Funkci min lze implementovat i tak, že jejím výsledkem je více rovnocenných alternativ. Z nich pak uživatel už může vybírat z hlediska modelu nedeterministicky.

Výsledná množina doporučení pro zpracování jednotlivých požadavků vznikne jako množina anotací vlastností výsledné (příp. vybrané) skupiny

$$A_X = \{\alpha_{\rho X1}, \ldots, \alpha_{\rho Xm}\}. \tag{18}$$

Je-li množina Ξ získána aplikací vztahů 15 a 16, pak výsledná množina doporučení je doplněna množinou nesplněných požadavků:

$$\bar{E}_X = E_i - Y_X. \tag{19}$$

Uživatel v tomto výsledku dostává komplexní informaci o optimálním programovém vybavení řešícím zadané požadavky, současně dostává souhrn postupů, které v daném programu realizují požadované prvky.

DISKUSE

Z uvedeného přehledu vyplývá, že existuje řada parciálních řešení výběru programového vybavení (i jiných produktů), neexistuje však nikdy možnost uživatelského zásahu do množiny požadavků ani množiny nabízených programů; výsledek neumožňuje využívat různé kombinace produktů a nenabízí popis řešení konkrétních požadavků uživatele. Všechny tyto nedostatky námi navržený systém odstraňuje.

Návrh modelu rozhodovacího systému pro volbu optimálního programového vybavení je postaven na koncepci binárních vstupních požadavků a binárního popisu vlastností programů. Tento přístup umožňuje jednoznačnou definici a jednoznačné porovnání požadavků a možností. Vzhledem k charakteru vstupních dat jsme se nepřiklonili k použití vícestavových nebo fuzzy hodnot.

Při výběru vhodného softwaru vycházíme z myšlenky, že uživatel zamýšlí vytvářet určitý typ dokumentu, pro nějž je typická určitá sada vlastností. Tyto vlastnosti se obvykle zásadně nemění, proto lze uživateli nabídnout šablonu typu dokumentu s předdefinovanými vlastnostmi. Uživatel se pak vyjádří pouze k těm vlastnostem, o které má zájem, u ostatních bude ponecháno jejich standardní nastavení. Tento postup je vhodný zvláště v případě, že uživatel nemá o vlastnostech přesnou představu. Uživatel obvykle ani není schopen rozhodnout, do jaké míry danou vlastnost vyžaduje a na kolik procent musí být splněna. Dokáže však určit, který požadavek musí být splněn a který nikoli, využívá se tedy binární ohodnocení. Specifikace požadavků a rozhodování může být aplikováno rekurzivně. Uživatel může ohodnotit určitou kategorii vlastností, v případě hlubšího zájmu o danou problematiku může své požadavky detailněji konkretizovat. Tím se navrhovaný systém výrazně liší od současných řešení, která buď nutí uživatele do velmi detailního rozboru problému, o kterém nemá přesnou představu, nebo naopak do rozhodovacího procesu vkládá vlastnosti, které uživatele nezajímají.

Jedním vstupem systému tedy bude částečně upravená množina standardně ohodnocených požadavků (E). Druhým vstupem jsou relativně neměnné vlastnosti programů (Γ). Výsledkem je doporučení takových programů nebo jejich kombinací, které zcela splňují požadavky uživatele (Ξ). Vzhledem ke značnému množství vlastností i požadavků a vzhledem ke značné variabilitě může běžně nastat situace, kdy podmínky nebudou zcela splněny. V tomto případě budou doporučeny alespoň ty programy (nebo jejich kombinace), které v maximální možné míře splňují požadavky uživatele. Uživatel tak bude mít vždy k dispozici návrh alespoň částečného řešení a bude se moci rozhodovat mezi přijetím navrženého softwaru a mezi úpravou svých požadavků. Tento přístup je v procesu rozhodování také spíše výjimečný. Současné systémy v situaci, kdy nenaleznou přesné řešení, oznámí pouze negativní výsledek a nenabízejí další alternativy. Přístup podobný tomu, který nalezneme v navrhovaném systému, lze spatřit jen v omezené míře. Příkladem může být vyhledávání pomocí Google, kde je vedle přesně odpovídajících výsledků nabízena i skupina výsledků podobných.

Navrhovaný systém bude po úspěšné implementaci zpřístupněn uživatelům, jako nejvhodnější prostředek pro implementaci se proto jeví webová aplikace. Uživatelské rozhraní bude řešeno v podobě webového formuláře. Uživatel nejprve zvolí typ do-

kumentu. Na základě toho bude vybrána vhodná šablona dokumentu a uživatel bude moci předdefinované vlastnosti, rozdělené do kategorií, jen změnit. Oproti vyplňování rozsáhlých formulářů je tato metoda velmi rychlá.

Po vyhodnocení uživatel obdrží výstup v podobě doporučení určitého programu nebo jejich kombinace. Zároveň budou zobrazeny požadované vlastnosti a s nimi související anotace. Anotace, jakožto krátký text, bude obsahovat pouze základní informace o dané vlastnosti a o způsobu její realizace. Bude-li vyžadováno delší vysvětlení, bude součástí anotace také odkaz na podrobnější návod. To umožní zahrnout do zpracování i programy, které uživatel neovládá perfektně a které by mohl jinak ze zpracování předčasně vyloučit.

Systém definuje tyto typy uživatelů:

- běžný uživatel – uživatel, který chce jen využívat schopnosti systému a nehodlá aktivně přispívat k jeho rozšiřování; může pouze prohlížet údaje zanesené do systému,
- přispěvatel – uživatel, který se aktivně spolupodílí na rozvoji systému; má vytvořený vlastní uživatelský účet a pod svým jménem může údaje prohlížet, vkládat i modifikovat,
- administrátor – uživatel, který je zodpovědný za správnost údajů a funkčnost systému; údaje prohlíží, modifikuje, přidává nové a především schvaluje změny provedené přispěvateli.

Navržený systém je z hlediska počtu programů a vlastností poměrně rozsáhlý. Přesto je koncipován jako otevřený, protože rychlý vývoj vyžaduje možnosti přidávání dalších programů a dalších vlastností programů.

Vkládání dat je navrženo ve dvou krocích – vložení údajů a schválení. Ve chvíli, kdy se oprávněný uživatel rozhodne doplnit či upravit údaje o systému již evidovaném, použije aplikaci, která umožňuje jednoduchým způsobem pomocí formuláře vyhledat danou vlastnost a provést její modifikaci, případně definici včetně úpravy anotace. V případě přidávání nového programu je možné pro urychlení a usnadnění práce využít jako předlohu již existujícího programového zástupce. Odlišné údaje u nového systému a jejich anotace pak stačí jen modifikovat. Jakmile uživatel (přispěvatel) provede v systému úpravu údajů, jsou tyto v aktuálním znění ihned k dispozici rozhodovacímu algoritmu, potažmo běžným uživatelům. Modifikovaná vlastnost je však dočasně označena za nejistou až do chvíle, kdy ji pověřená osoba (administrátor systému) schválí, případně zamítne. Tento přístup umožňuje rychlou aktualizaci údajů o systémech s pomocí široké účasti zainteresovaných uživatelů. Tím je zajištěna flexibilita i aktuálnost.

ZÁVĚR

Cílem návrhu systému pro optimalizaci výběru programového vybavení bylo překonání některých zásadních nedostatků současných systémů, spočívajících v omezení vybíraných objektů, obtížné definici vstupních požadavků a nemožnosti proces výběru uživatelsky ovlivňovat.

Usnadnění vstupních požadavků je založeno na existenci předdefinovaných šablon, v nichž může uživatel provést jednoduché modifikace. Výsledkem uvedeným v doporučení srovnávacího procesu může být nejen jediný program vyhovující požadavkům, ale také kombinace programů s návrhem přenosu dat pomocí podporovaných datových formátů. Rovněž lze získat výsledek, který nepokrývá všechny vstupní požadavky, ale alespoň jejich největší možnou část. Základním vodítkem pro uživatele je nejen výsledek v podobě názvu programu, ale také dostatečně podrobný popis jednotlivých služeb, které jsou k dosažení požadovaného výsledku v jednotlivých programech potřeba.

Navrhovaný systém lze ovlivňovat vkládáním nebo modifikací požadavků nebo vlastností zahrnutých programů.

Na teoretický návrh bude navazovat praktická fáze implementace, jíž bude ověřen navržený model. Nedílnou součástí implementace bude i vytvoření databáze dostupných programů a jejich vlastností a také návrh způsobu přebírání vstupních požadavků od uživatele. Předmětem dalšího zkoumání pak bude způsob verifikace výsledků, kdy budou požadavky uživatele porovnávány s vlastnostmi programů a budou vydávána výsledná doporučení.

Po úspěšném provedení všech kroků bude k dispozici nástroj, který odpovídá na požadavky uživatele v oblasti volby (nejen komerčního) softwaru.

SOUHRN

Článek je věnován návrhu systému pro optimalizovaný proces výběru programového vybavení. Výběr programu použitelného pro zpracovávaný úkol řeší uživatelé často intuitivně, což nevede k optimálním výsledkům. Volně dostupné aplikace pro podporu výběru programového vybavení jsou obvykle zaměřené na velmi úzkou oblast, vyžadují od uživatele detailní a zdlouhavou specifikaci preferencí a orientují se na komerční programy, z nichž jako řešení doporučují pouze jeden.

Navržené řešení je zaměřeno na odstranění těchto nedostatků, přičemž na rozhodovací proces má vliv jak specifikace požadavků na výsledná data, tak detailní informace o použitelném programovém vybavení.

Uživatel volí typ požadovaného výstupu podle šablony, kde jsou již hodnoty typických vlastností předdefinovány. Uživatel může na základě svých preferencí hodnoty změnit, přičemž se používá pouze binární vyjádření. Tato část procesu je proto velmi rychlá a efektivní. Systém pro výběr programů dále dis-

ponuje detailním popisem vlastností různých programových systémů. Porovnáním množiny ohodnocených požadavků uživatele s množinou ohodnocených vlastností programů vzniká množina vhodných programů, přičemž se může jednat jak o samostatné programy, tak jejich kombinace. Není-li žádné řešení zcela vyhovující, je vybráno nejbližší odpovídající řešení. Součástí řešení je také vždy množina anotací, která uživateli poskytuje návod, které vlastnosti má použít pro splnění svého úkolu a jak.
Vzhledem k rychlým změnám v programovém vybavení je nezbytná možnost rozšiřování systému. Kromě běžných uživatelů figuruje v systému také skupina přispěvatelů, kteří mají právo přidávat a měnit vlastnosti programových systémů i přidávat nové programy. Proces schvalují administrátoři. Systém bude implementován jako webová aplikace.
Návrh založený na těchto vlastnostech vyúsťuje v otevřený, snadno dostupný a flexibilní systém orientovaný na uživatele a jejich potřeby.

dokument, optimalizace, požadavky uživatele, vlastnosti programů, výběr software

SUMMARY

This article is aimed at a system design dealing with the optimized software selection process. Users often select programs for processing of their tasks intuitively, but the obtained results are not optimal. Free applications for software selection support are usually focused on a very narrow area, they demand the user's detailed and time consuming specification of preferences and are oriented to commercial programs, from which they recommend the only one as a resulting solution.
The aim of our suggested solution is to remove this insufficiency whereas both requirements specification related to the resulting data and detailed information about utilizable program equipment affect the decision process.
The user defines a demanded output type according to the document template, where values of typical properties are predifined. The user can modify these values on the basis of his/her preferences. Only binary expression is used. Therefore this process part is very quick and effective. The system for program selection also disposes of a detailed description of various programs properties. Comparing the set of evaluated user's requierements with the set of evaluated program properties generates a set of acceptable programs whereas the result could be either a standalone program or a combination of programs. If no result is acceptable, the nearest one is selected. A part of the solution is also an annotation set of what to use and how to gain the user's objective.
A possibility of system enhancing is a necessity owing to quickly changing program equipment. Besides of common users there is also a group of contributors. They can add and modify program properties and also add new programs to the system. Administrators of the system approve all changes. The system will be implemented as a web application.
The design based on these properties leads to an open, easily accesible and flexible system concentrated on users and their needs.

Práce vznikla v rámci výzkumného záměru MSM 6215648904/03/03/06.

LITERATURA

Accounting Software Library – Online Software Selection [online]. [cit. 30. 11. 2009]. Dostupné z http: //www.accountinglibrary.com/online/.

BANDOR, M., 2006: Quantitative Methods for Software Selection and Evaluation [online]. [cit. 30. 11. 2009]. Dostupné z http://www.sei.cmu.edu/reports/06tn026.pdf.

DRDLA, S., 2006: Využití systému Linux na pracovních stanicích. Bakalářská práce. Brno: MZLU v Brně, 36 s.

Ekonomické softwary. Jak vybrat správně ekonomický software [online]. 2009 [cit. 30. 11. 2009]. Dostupné z http://ekonomicke-softwary.cz/cz/novinky-13-jak-vybrat-spravne-ekonomickysoftware.

Katalog mobilů: rozšířené vyhledávání [online]. 2009 [cit. 30. 11. 2009]. Dostupné z http://www.mobilmania.cz/katalog-mobilu/sc-63-c-1/default.aspx?advsrch=1.

KOMOSKI, P. K., PLOTNICK, E., 1995: Seven Steps to Responsible Software Selection [online]. ERIC Digest [cit. 30. 11. 2009] Dostupné z http://www.ericdigests.org/1996-1/seven.htm.

KROB, J., 2009: Editory pro TEX [online]. [cit. 12. 12. 2009]. Dostupné na http://www.phil.muni. cz/~j-okr/tex/tex-editory.html.

OBČANSKÉ SDRUŽENÍ SPOTŘEBITELŮ TEST, 2009: *TEST* [online]. [cit. 12. 12. 2009]. Dostupné na http://www.dtest.cz.

SKOVAJSOVÁ, M., 2001: Srovnávání programových systémů. Diplomová práce. Brno: MZLU v Brně, 77 s.

WANG, T.-C., CHEN, Y.-H., 2007: Applying the fuzzy preference relation to the software selection [online]. [cit. 30. 11. 2009]. Dostupné na http://www.isu.edu. tw/upload/28/3/29520/paper/9601/960101.pdf.

Adresa

doc. Ing. Jiří Rybička, Dr., Ing. Petra Talandová, Ph.D., Ing. Jan Přichystal, Ph.D., Ústav informatiky, Mendelova univerzita v Brně, Zemědělská 1, 613 00 Brno, Česká republika, e-mail: jiri.rybicka@mendelu.cz, petra.talandova@mendelu.cz, jan.prichystal@mendelu.cz