Systém optimalizující zpracování požadovaného dokumentu

Jiří Rybička, Petra Talandová, Jan Přichystal

Výběr vhodného postupu zpracování

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

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

1. precizní specifikace požadavků na výsledný dokument (krok 1 – co chci),
2. důkladná znalost použitelného programového vybavení (krok 2 – čím a jak to udělám).

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.

V reálných podmínkách při intuitivním rozhodování uživatele je často množina požadavků na dokument dokonce irelevantní, protože rozhodnutí o postupu zpracování je dáno pouze použitým programovým vybavením. V extrémním případě (nikoliv ovšem v praxi vzácném) je redukováno na jednoprvkovou množinu systémů, s nimiž je daný uživatel schopen nebo ochoten pracovat.

Možným řešením optimálního zpracování daného dokumentu počítačem je tedy přesunout v obou krocích pokud možno co největší část celého rozhodovacího procesu na automatizovaný systém vybavený dostatečnými obecně dostupnými informacemi soustředěnými do vhodných struktur.

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

- První krok – specifikace relevantních požadavků na dokument a specifikace zvláštních požadavků uživatele. Vstupní požadavky na dokumenty lze reprezentovat množinou obvyklých typů dokumentů, z nichž uživatel vybere typ nejvíce se blížící 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 požadavků nebo redukcí předdefinovaných požadavků. Tak lze s minimálním úsilím dospět k optimální množině požadavků, tvořících základ pro další rozhodování. Zvláštní požadavky uživatele, týkající se uživatelského prostředí a požadovaných vlastností programového vybavení, jsou nezávislé na typu dokumentu a doplňují množinu vstupních požadavků. Z implementačního hlediska je podstatné, že zvláštní požadavky uživatele mohou tvořit uživatelský profil, který rovněž může být předdefinován.

- 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, tedy jinak řečeno, vstupní požadavky není schopen 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 řešení zvoleného typu dokumentu počítačem 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 stanovování, na druhé straně je však zajištěna jednoznačnost a srozumitelnost.

Každý požadavek musí být jednoznačně specifikován identifikátorem (pro tentýž účel lze zvolit i číselné indexy), volitelně je vhodné přidat vysvětlující komentář, který však do formálního modelu nezahrneme. Přestože je z hlediska teoretického modelu nepodstatný, můžeme s jeho existencí počítat při implementaci.

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 určitého uživatele U_i z množiny uživatelů $U = \{U_1, \ldots, U_u\}$ na zpracovávající systém a jeho rozhraní můžeme vyjádřit množinou

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

Podobně jako u dokumentů platí i zde, že hodnota h je pro všechny uživatele stejná a představuje kardinalitu sjednocení všech požadavků zahrnutých uživatelů.

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

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

$$\Gamma_{p_i} = S_{p_i} \times B \times A_{p_i}, \tag{4}$$

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]

[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.

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 považovat 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 relevantní.

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

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

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

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

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 $\phi : 2^P \rightarrow I$ definovanou vztahem $\phi(\rho) = N$ 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 $U : \Gamma \rightarrow \vec{\alpha}$ získávající z množiny anotovaných vlastností uspořádaný vektor anotací, funkce $V : \Gamma \rightarrow \vec{b}$ získávající z množiny anotovaných vlastností uspořádaný vektor jejich ohodnocení a funkce $W : E \rightarrow \vec{b}$ získávající z požadavků na dokument uspořádaný vektor jejich ohodnocení.

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

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

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

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

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

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:

$$\vec{E_X} = E_i - Y_X. \tag{19}$$

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.