Verifikace funkční a datové analýzy

Doc. RNDr. Milan Mišovič, CSc.
Ing. Vlastimil Malý

Klíčová slova

Funkční a datová analýza, funkční model, datový model, klasická YOURDONOVA strukturovaná analýza řízená událostmi, DFD diagram, ERM diagram, cesta v DFD diagramu, funkční korektnost cesty, entita, životní cyklus entity, realizovatelnost cesty, dotaz, informační cesta a její topologie, informační dostatečnost ERM diagramu, semispolehlivost funkční a datové analýzy.

1. Úvod

Ověřování výsledků funkční datové analýzy zajímá informatiky již řadu let. Snaha odhalit závady na úrovni esenciálního modelu klade jistě verifikaci do velmi užitečné roviny. Kardinální otázkou takto situované verifikace je "co" porovnat s "čím". Je to situace poněkud složitější než obdobná problematika pro jednoduché programy, ačkoliv i tam byly a jsou problémy naformulovat to co bylo původně zamýšleno a porovnat to s výsledkem produkční práce programátora - textem programu.

Pochopitelně, funkční a datová analýza přináší hmatatelné výsledky ve formě DFD diagramu a datového modelu. Naformulovat však to, co bylo původně zamýšleno pro funkci stávajícího a modifikovaného systému zpracování informace je nejen obtížné, ale i těžko porovnatelné se získanými výsledky funkční a datové analýzy. Vyjádříme to jednoduše takto: "Je logika zpracování informace vyjádřená ve výsledcích funkční a datové analýzy podobná nebo stejná s tou, která byla zamýšlena v originálním popisu stávajícího a budoucího systému zpracování informace?"

Jelikož podstatu takovéto verifikace tvoří sémantika, je obtížné najít vhodný formálně-sémantický systém, který by podstatě kardinální otázky vyhověl. Kardinální otázku verifikace nelze obejít, můžeme jen najít pro nás její vhodnou interpretaci. Jednou z nich může být zavedení role analytika-projektanta a zástupce(ů) uživatele jako komparátorů mezi tím co bylo zamýšleno a tím čeho bylo analýzou dosaženo. Takový komparátor ale není schopen odhalit všechny nesrovnalosti - formální a logické nekonzistentnosti. V hledání formálních nekonzistentností ve výsledcích funkční a datové analýzy jistě a velmi užitečně pomáhají vývojová prostředí typu CASE. Zůstává ale mnoho logických nekonzistentností, které ve výsledcích funkční a datové analýzy není námi určený komparátor schopen odhalit. Na některé z nich bychom chtěli poukázat, vysvětlit jejich podstatu a navrhnout praktický prostředek - mocný nástroj pro analytika - projektanta a uživatele pro ověřování logiky výsledků funkční a datové analýzy.

2. Životní cykl entity

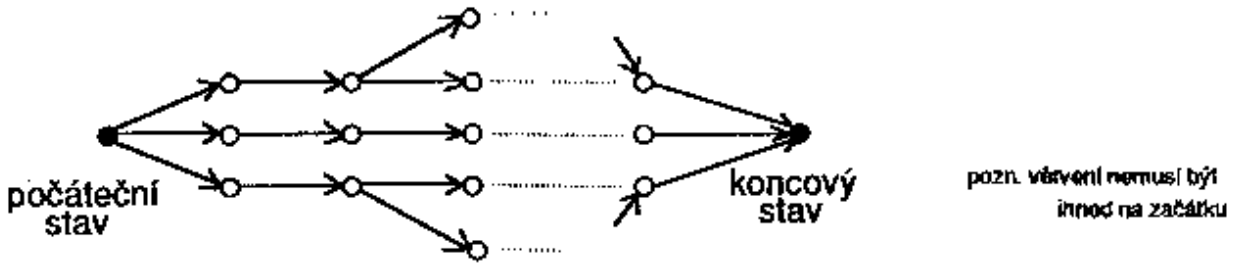
Entita $e=(a_1,a_2,\ldots,a_n)$, kde $\{a_1,a_2,\ldots,a_n\}=A$ vytváří atributy, je datovou strukturou, jenž je zpracovávána v řadě funkcí v DFD diagramu. Platí pro ni běžná definice datové struktury S ve tvaru

$S = [P, \Omega, \sigma]$, kde
P=A je množina atributů,
Ω je množina operací-funkcí, které lze nad e provést,
$\sigma = [P', \delta, h]$ je stav entity, kde
$P' \in P$ je vybraná množina operací pro operaci $\beta \in \Omega$,
δ je množina relací mezi atributy,
h je zobrazení, které vybrané množině P' pomocí operace β přiřadí nové hodnoty.

S entitou se váže její životní cykl. Termín a jeho obsah se používá v řadě metodologií založených na klasické strukturované analýze. Životní cykl začíná zřízením entity, potom entita přechází ze stavu do stavu pod vlivem zpracování funkcemi z DFD diagramu, nakonec její životní cykl končí a je zničena.

Z definice stavu entity vyplývá, že jestliže funkce f, která entitu zpracovává mění komponenty P', δ, h potom i vytváří nové stavy. Změna komponenty δ je ale diskutabilní. Jestliže entita e byla již podrobena převodům z 1NF až např. do 4NF, jsou již sémantické relace (hlavně silná závislost na definičním atributu) konsolidovány a jejich změna se nepředpokládá. Na druhé straně změna komponent P' a h je v praxi častá.

Vzhledem ke vlastnímu zpracování působí stavy v roli předchůdců, jiné v roli následovníků a vytváří se tak jisté historie - následnosti - řetězce stavů majících charakter START-STOP vazeb stavů, což je ukázáno na Obr.1.



Jaký je charakter těchto historií je jednoznačně dáno zpracováním entity. Funkce, patřící do jimiž se entity zpracovávají mají jednu z následujících charakteristik:
funkce f je ...

- aktualizační funkcí (mění hodnoty atributů) ... konstruktor měnící stav entity
- výpočtová funkce měnící jeden z atributů ... konstruktor měnící stav entity
- dotazová funkce ... je to selektor neměnící stav
- rozhodovací funkce (predikát) ... je to predikát (způsobuje větvení v DFD a nemění stav entity).

Z charakteristiky funkcí se potvrzuje, že aktualizační a výpočtové funkce jsou především tvůrci historií stavů entity. Potom možno ke každé historii stavů přiřadit řetězec funkcí např. $f_0f_1f_2\ldots f_n$, který

vlastně historii generuje. V uvedeném řetězci se mohou vyskytovat funkce všech typů, určující jsou ale konstruktory.

První velmi závažnou skutečností logiky zpracování entity v DFD diagramu je to, že stavy jichž je tímto zpracováním pro entitu dosaženo musí být jako řetězec součástí alespoň jedné z předem zjištěných historií stavů. Jinými slovy, že řetězec funkcí, který představuje operace nad ní provedené je povoleným - přípustným řetězcem. Teď vybudujeme aparát, kterým tuto skutečnost postihneme.

Každý ze stavů entity e lze zapsat výrazem v predikátovém kalkulu. Pro tyto účely by ale byl vhodný specifický jazyk L(G) predikátového kalkulu s gramatikou G a doplněný řadou specifických konstant a výroků (viz. příklad). Mnohé ze stavů je nutno zapsat specifickým výrazem, jiné výrazy ale mohou prezentovat celou třídu podobných stavů (např. neustálá změna téhož atributu,...).

Příklad

Buď null konstanta říkající, že "není hodnota", $b \in A$ speciální množinový výrok "b je z množiny A", kde b je individuální proměnná, b^N zápis výroku b=null, b^S zápis výroku "hodnota pro b se nemění". Potom verbálně vyslovené stavy jichž entita FAKTURA=(číslo,datum,zhotovitel,adresát,...) nabývá:

- faktura je v počátečním stavu, jestliže jsou vyplněny atributy číslo, zhotovitel, adresát a ostatní atributy jsou prázdné
- faktura je ve stavu, kdy byla změněna hodnota atributu cena v 10.řádku na hodnotu p, současně se změnila hodnota řadek_celkem a faktura_celkem na p', p".

lze zapsat těmito výrazy jazyka L(G):

- (číslo=031) a (zhotovitel="a.s.BVV Brno") a (adresát="a.s.Orag") a $\sqrt{}$ ($b \in$ A-{číslo,zhotovitel, adresát} $\Rightarrow b^N$)
- $Cena_{10}=p \Rightarrow$ ($radek\_celkem_{10}=p'$ a faktura_celkem=p") a $\sqrt{}$ ($b \in$ A- {$cena_{10}$, $radek\_celkem_{10}$, faktura_celkem} $\Rightarrow b^S$)

kde $\Rightarrow$ značí implikaci, A označuje množinu atributů entity FAKTURA (tj. detonáty pro individuální proměnnou b), p, p', p" jsou individuální proměnné s obory detonátů $D_p, D_{p'}, D_{p''}$; 031, "a.s.BVV Brno", "a.s.Orag" jsou konstanty.

Jestliže se teď opět podíváme na obrázek 1, můžeme potom pomocí podstatně složitějšího predikátového výrazu zaznamenat skutečnost, že po stavu σ, provede-li se nad e funkce f (typu konstruktor) následuje v historii stav σ' anebo jeden ze stavů z množiny $\{\sigma'_1, \sigma'_2, ..., \sigma'_n\}$. To vyjádříme jedním z dvojice výrazů

$$(\sigma_v = \sigma)_f \Rightarrow \sigma^v \in \{\sigma'_1, \sigma'_2, ..., \sigma'_n\} \qquad (1)$$
$$(\sigma_v = \sigma)_f \Rightarrow \sigma^v = \sigma'$$

Zápis můžeme číst takto:

"Jestliže je vstupní stav σ_v entity e roven stavu σ a provede se funkce f, potom to implikuje, že výstupní stav σ^v entity je buď roven stavu σ' anebo je jedním ze stavů z množiny $\{\sigma'_1, \sigma'_2, ..., \sigma'_n\}$".

Pochopitelně pod $\sigma, \sigma', \sigma'_1, \sigma'_2, ..., \sigma'_n$ rozumíme predikátové proměnné, jejichž hodnotami jsou konkrétní stavy zadané alespoň výrokovými formami s individuálními proměnnými.

Ke každé historii stavů pak existuje posloupnost interpretovaných výrazů typu (1), představující pravdivé výrazy, kde za $\sigma, \sigma', \sigma'_1, \sigma'_2, ...\sigma'_n$ jsou dosazeny konkrétní stavy entity e. Predikátové výrazy typu (1) jsou tedy prostředkem plně zachycujícím dvě skutečnosti:

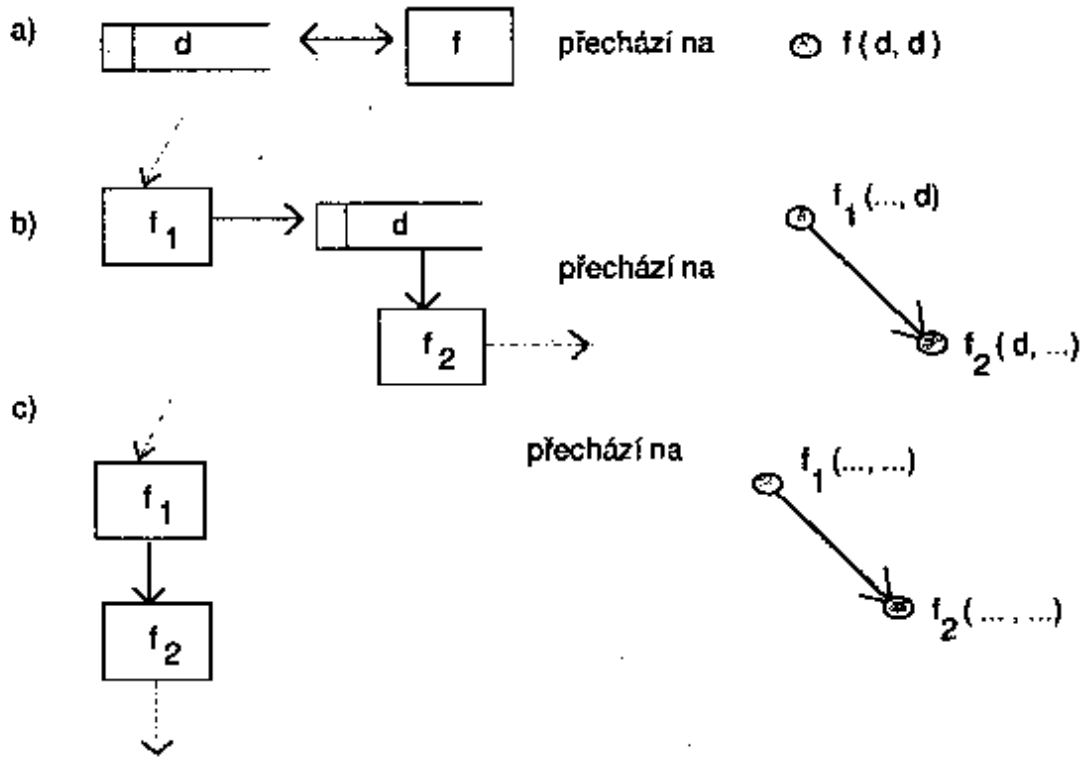
- složení celé historie stavů (předchůdce, následník jako uzly)

- tvar řetězce $f_0f_1f_2...f_n$, který náleží k dané historii stavů a který je tzv. přípustným řetězcem.

Tím se pomocí interpretovaných výrazů typu (1) získává celková představa o životním cyklu entity. Hledání životních cyklů entit je často součástí mnoha metodologií a stavy se tedy musí navrhovat a také popisovat.

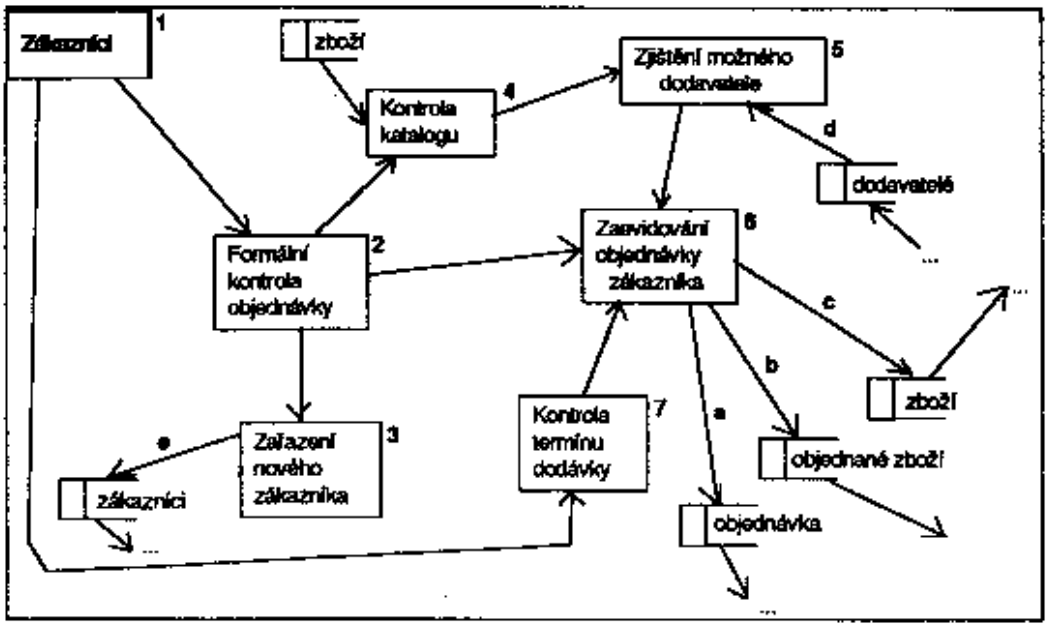
3. Funkční korektnost DFD diagramu

Pro další účely provedeme nad DFD diagramem dvě závažné operace. První je odstranění všech abstraktních úrovní a získání diagramu jen nad funkcemi. Operace musí plně respektovat aktuální konzistentnost jak v rozkladech procesů, tak ve využití Data Store. Druhá operace převádí DFD ve funkcích na jednodušší graf s uzly a hranami. Je založena na předpokladu, že DFD byl kreslen strukturovaně podle událostí. Vzniká tak orientovaný graf G=(U,H) řízení, kde za prvek množiny U je považována funkce, přičemž je pamatována její asociace se zpracovávanou entitou, hrany z množiny H jsou získány z potenciálních přechodů mezi funkcemi. Tedy

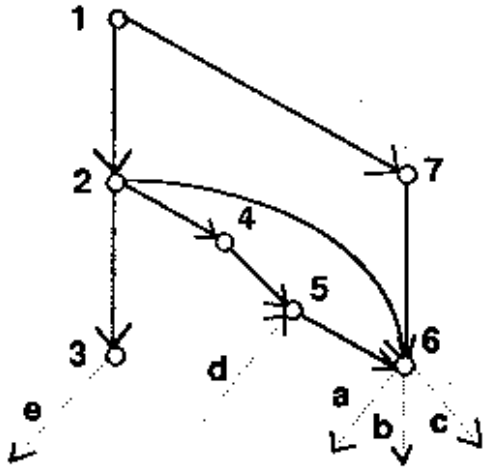


Provedení druhé transformace ilustrují obrázky 2 a 3.

Transformace na graf řízení, kterou jsme provedli, má stinná místa, protože ani v strukturování podle událostí nemusí být redukce případu b) korektní vzhledem k potenciálním přechodům mezi funkcemi. Jinými slovy, v grafu mohou existovat orientovaná spojení nemající reálnou podstatu. To ale pozdějším úvahám nebude vadit, protože výběr takového spojení pro verifikování by brzy vedlo k indikacím o špatných historiích stavů entit, resp. o nepřípustném řetězci $f_0f_1f_2...f_n$ funkcí prováděných nad entitou.



Obr.2 Fragment DFD diagramu



Obr.3 Graf řízení získaný z fragmentu Obr.2

V dalším chceme zavést základní pojmy pro korektnost DFD. Budou to především pojmy cesta v DFD, realizovatelnost cesty, funkční korektnost cesty a funkční korektnost DFD diagramu. Než k tomu ale přikročíme, uveďme několik postřehů z praktické projekční činnosti a závěrů z její diskuse.

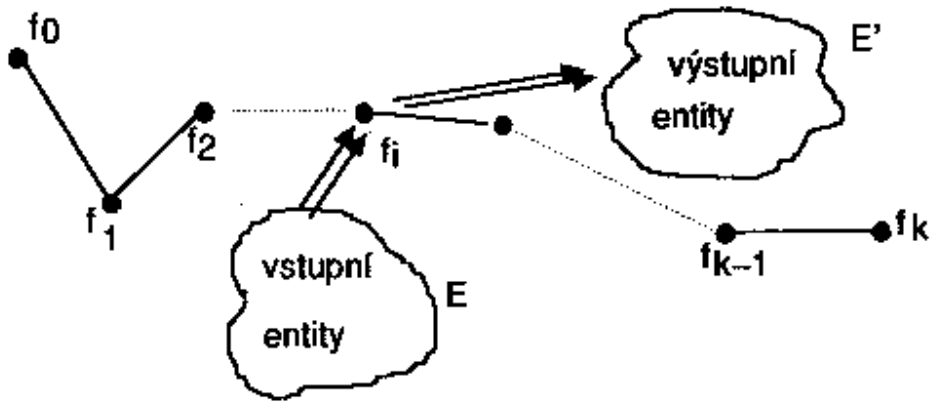
Již pro DFD běžných transakčních systémů je počet různých spojení - cest (s reálným opodstatněním) v grafu řízení značně velký. Nikterak nelze podceňovat schopnosti vzorového analytika - projektanta. Je však jisté, že plnou pozornost ověření těchto cest nemůže věnovat. Jde nejen o množství, ale i značnou pracnost vlastního ověření vybrané cesty. Praxe potvrzuje, že bývá vcelku prověřeno asi 40% cest. Závady ostatních se odhalí později, oprava může vyvolat velké náklady. Vzniká otázka - proč nevymyslet metodu podporovanou počítačem pro automatizaci verifikace cest DFD? Za vzor bychom mohli považovat takovou metodu, která by verifikaci provedla bez pomoci analytika pojektanta a poskytla mu až závěrečné výsledky. Tento vzor je ale těžko dostižitelný, což bude patrno i z dalšího textu. Tak proč nespojit inteligenci analytika projektanta s dílčí inteligencí metody a jejími vysokými mechanickými schopnostmi? To bude právě naší snahou. Výsledkem bude interaktivní systém usnadňující funkční verifikaci cest v DFD a poskytující analytikovi nezbytné informace k jeho rozhodování.

Přikročme k výstavbě základních pojmů.

Definice 1

Cestou v diagramu toků a zpracování dat nazveme posloupnost $(f_0,f_1,f_2,...,f_k)$, kde f_0 je zahajovací funkce asociovaná s událostí, f_k je ukončovací funkce a pro i=0,1,2,...,k-1 platí, že existují potenciální přechody $f_i \rightarrow f_{i+1}$

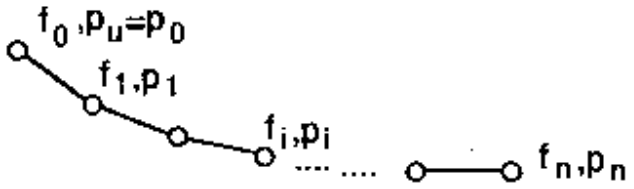
Cesty se označují malými řeckými písmeny a budeme používat zápis ve tvaru $\alpha = (f_0,f_1,f_2,...,f_k)$. Dále budeme předpokládat, že DFD je funkčně úplný vůči slovníku událostí jestliže ke každé události existuje zahajovací funkce f_0 určující počátek reakce na událost. Obrázek č.4 ukazuje vlastní zpracování entit cestou α.



Obr.4 Zpracování entit cestou - ukázáno pro funkci f_i

Vstupní entity z množiny E musí být v přípustných stavech, výstupní entity z E' taktéž. Nově zřízené entity v E' jsou obvykle v počátečních stavech. Je pochopitelné, že zpracování vstupních entit je funkčně korektní, jsou-li funkcí f převedeny na výstupu do přípustných stavů.

Jestliže vezmeme v úvahu výsledky analýzy stavů v životním cyklu entit, můžeme hovořit o tom, že ke každému uzlu cesty můžeme přidat vlastně predikátový výraz p_i, který bude zachycovat kvalitu zpracování entit. Pro jednoduchost předpokládejme, že funkce zpracovává jen jednu entitu.

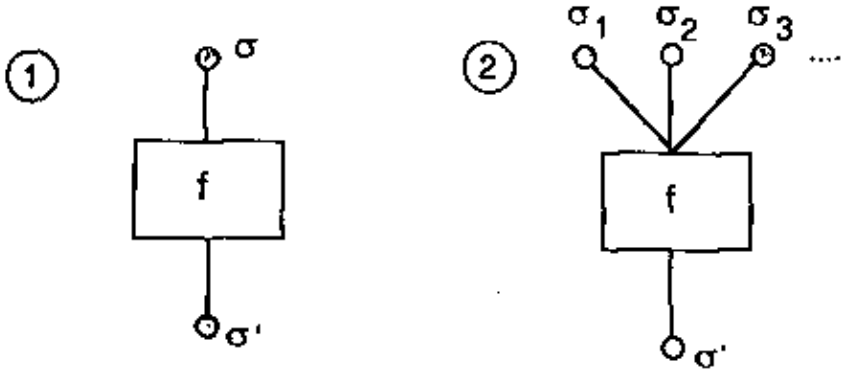


Obr.5 Přiřazení predikátových výrazů p_i funkcím

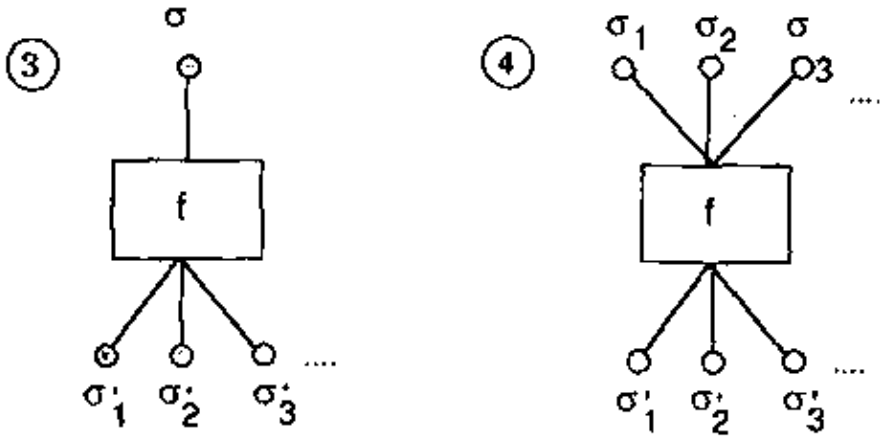
Obrázek 5 ukazuje, že f_0 je spojena s predikátem $p_n=p_0$, který vyjadřuje, že událost nastala, má-li hodnotu pravda. Dále platí:

- je-li f_i ... selektor, je p_i prázdný
  ... predikát, je p_i f_i
  ... konstruktor, vyjadřuje p_i kvalitu zpracování f_i, je jím konkrétní interpretace výrazů typu (1)

Toto ale platí pro konkrétní cestu. Vyšetřujeme-li cestu abstraktně, potom p_i u funkce f_i, je-li ona konstruktorem, jsou přímo výrazy typu (1). Jestliže jsme u životního cyklu entity e rozebrali jen případy



tak jistě chápeme, že existují i možnosti



Všechny tyto možnosti jen potvrzují různá větvení v historiích stavů entity e. K možnostem 3) a 4) náleží tyto predikátové výrazy:

$$\sigma_v \in \{\sigma_1, \sigma_2, \ldots, \sigma_n\} \Rightarrow \sigma^v = \sigma' \qquad (2)$$
$$\sigma_v \in \{\sigma_1, \sigma_2, \ldots, \sigma_n\} \Rightarrow \sigma^v \in \{\sigma'_1, \sigma'_2, \ldots, \sigma'_n\}$$

Všimněme si teď nejdříve predikátů p_i f_i. Tyto predikáty představují řízení potenciálních přechodů. Má-li se projít po cestě α, jsou-li v ní takové predikáty, tak jistě musí existovat stavy entit, které tyto predikáty převádí v pravdivé výrazy - tedy cesta je realizovatelná - proveditelná.

Věta 1

Cesta $\alpha=(f_0,f_1,\ldots,f_i)$ je realizovatelná tehdy a jen tehdy, jestliže:

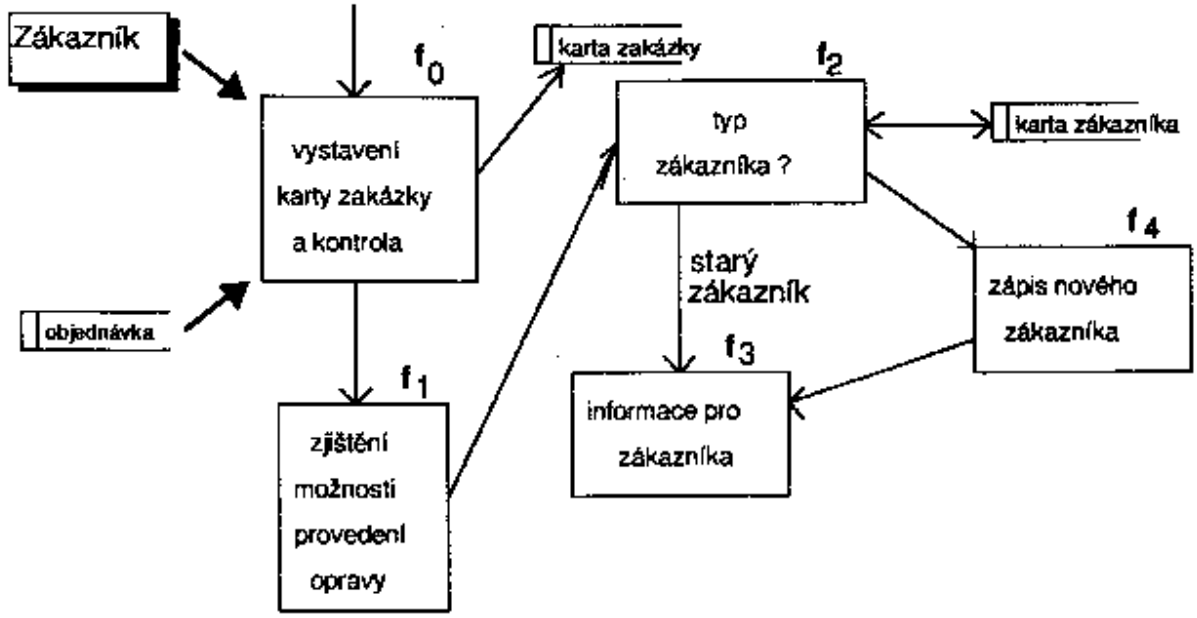
- nastala událost u asociovaná s funkcí f_0 (tj. je pravdivý predikát $p_u(\alpha) \equiv p_0$)
- systém $N_\alpha(p_i)$, i=0,... složený z predikátů typu $f_i \equiv p_i$ má řešení.

Slovo "řešení" je zde chápáno ve smyslu, že existují takové stavy entit, aby predikáty p_i po dosažení stavů byly pravdivými výrazy. Predikáty $p_i \equiv f_i$ nemají stejný charakter jako predikáty u funkcí jež jsou konstruktory.

Důkaz věty se provádí sporem.

Příklad

Uveďme fragment DFD zobrazující činnosti nutné pro zpracování zakázky pro opravu elektrického spotřebiče. Událostí je příchod zákazníka s porouchaným přístrojem a vyplněným listem OBJEDNÁVKA



Obr.6 Fragment DFD k události "Příchod zákazníka ..."

V uvedeném fragmentu jsou dvě cesty

$\alpha_1=(f_0,f_1,f_2,f_3)$ $\alpha_2=(f_0,f_1,f_4,f_3)$

Zabývejme se cestou α_2. Systém $N_{\alpha 2}(p_i)$ u $p_u(\alpha_2)$ může být naformulován např. takto:

E(číslo_zákazníka=x) a $\sqrt{}$ (b∈A-{číslo_zák} => b<>null)

k entitě KARTA ZÁKAZNÍKA, A je zde množ.atributů

($\sqrt{}$ b∈B => b<>null)

k entitě OBJEDNÁVKA, B je zde množ. atributů

kde E je kvantifikátor "existuje",

$\sqrt{}$ je kvantifikátor "pro všechna".

Po cestě se projde např. tehdy, jestliže sdělené číslo X přišlým zákazníkem odpovídá jedné uložené kartě a ostatní údaje v ní jsou rovněž vyplněné (X je řešení) a současně všechny kolonky OBJEDNÁVKY jsou vyplněné.

Dalšími významnými predikáty jsou ty, které jsou u funkcí f_i typu konstruktor. Tyto predikáty jsou z hlediska reálné cesty výrazy získané interpretací obecných výrazů typu (1) nebo (2). Nám jde o obecný přístup a proto takový systém $N'_\alpha(p_i)$ musí svým řešením zabezpečit korektnost zpracování entit. Tedy musíme hledat takové hodnoty individuálních proměnných a predikátových proměnných, aby po dosazení do výrazů systému $N'_\alpha(p_i)$ byly tyto výrazy pravdivé. Jen tak je zabezpečena korektnost funkčního zpracování.

Definice 2

Cesta $\alpha=(f_0,f_1,f_2,...,f_k)$ je funkčně korektní, jestliže všechny funkce cesty α převádí přípustné stavy entit opět do přípustných stavů a zřizují nové entity a uvádí je do počátečních stavů.

Věta 2

Je-li cesta $\alpha=(f_0,f_1,...,f_k)$ realizovatelná a systém $N'_\alpha(p_i)$ má řešení, potom je α funkčně korektní.

Realizovatelnost cesty na funkční korektnost cesty nestačí, je nutné zvažovat korektnost zpracování entit v každé funkci cesty. Realizovatelnost cesty je ovšem nutnou podmínkou její funkční korektnosti.

Věta 3 (důsledek definice 2)

Buď e libovolná vstupní entita zpracovávaná cestou $\alpha=(f_0, f_1,...,f_k)$, nabývající cestou historie stavů $\sigma f_0, \sigma f_1, ..., \sigma f_k$. Cesta α je funkčně nekorektní, jestliže tato historie není součástí žádné z historií stavů ze životního cyklu entity e.

Definice 3

Diagram toků a zpracování dat je funkčně korektní, když

- je funkčně úplný vůči slovníku událostí,
- každá jeho cesta je realizovatelná,
- každá jeho cesta je funkčně korektní.

Tato definice představuje závěrečný pohled na funkční korektnost DFD diagramu. Provést ale takovou prověrku celého DFD není snadná záležitost. Kardinální potíže jsou v metodách řešení systémů predikátů. Obecné metody (např. jedna uveřejněná v [1]) nedosahují úspěchu. Ale praktická stránka - kdy necháme systém řešit analytikem - projektantem bez výrazné podpory počítače nevede také k úspěchu. Chce-li analytik prověřit všechny cesty, musí mít nejdříve jejich seznam, poté stanovovat hodnoty atributů entit a zjišťovat, jsou-li realizovatelné. Kdyby bylo možné řešit systém $N_\alpha(p_i)$ bez přispění analytika - dostal by analytik všechny entity a hodnoty jejich atributů, které způsobí realizovatelnost cesty α. V dalším je navržen programový systém, který řadu těchto negativ zmírňuje.

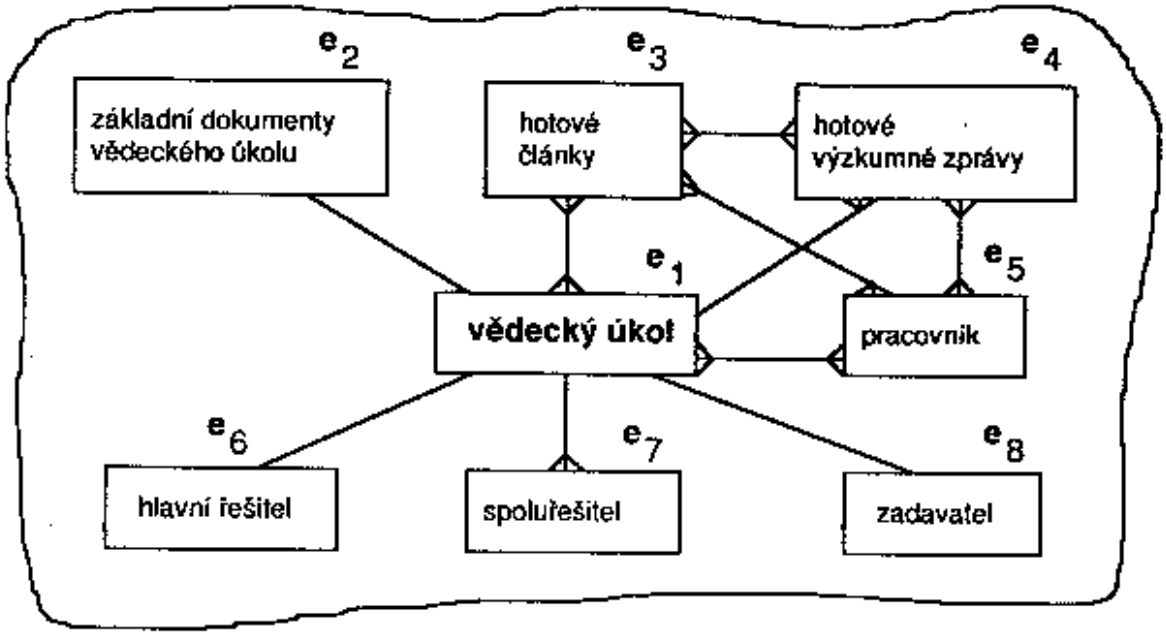
4. Informační dostatečnost ERM modelu

Podstatou myšlenky informační dostatečnosti datového modelu je skutečnost, že ke každému dotazu z úplné množiny dotazů stanovené uživatelem existuje v datovém modelu informační cesta (byť jen o jedné entitě) v níž je obsažena informace sloužící k odpovědi pro dotaz.

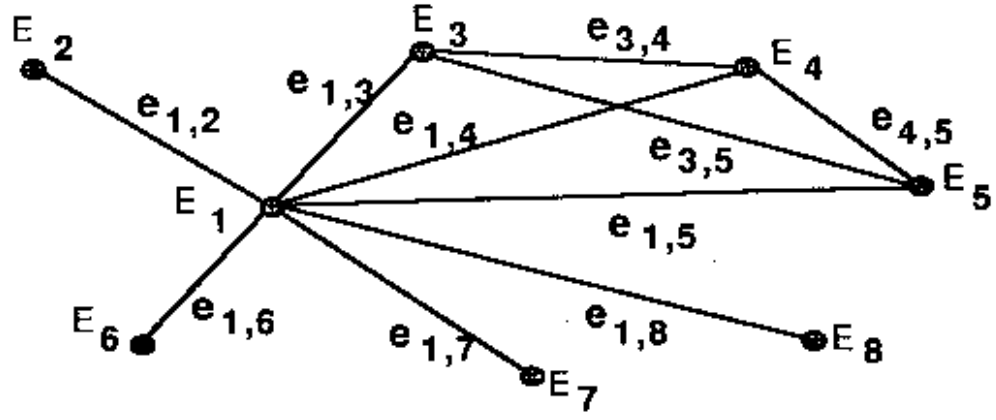
Nad datovým modelem provedeme operaci, která převede model na graf řízení G=(U,H), kde uzly jsou jednotlivé entity a hrany jsou vztahy mezi nimi. Graf je neorientovaný, jde jen o to, aby dal představu o možnosti nalezení spojení mezi zvolenými uzly. Předpokládejme, že všechny vztahy jsou realizovány pomocí vztahových entit.

Příklad

Následující fragment datového modelu je pak zmíněnou transformací převeden do neorientovaného grafu.



Obr.7 Fragment datového modelu



Obr.8 Výsledek převodu fragmentu datového modelu na graf. řízení

Prověrku informační dostatečnosti můžeme provést tenkrát, když umíme

- transformovat dotaz Q z úplné množiny dotazů na souvislý podgraf g, tj. udělat g:T(Q)
- zjistit, jaká výsledná informace I je v podgrafu g obsažena.

Záležitost transfomace g:T(Q) dotazu Q nebude jistě činit potíže, uvědomíme-li si, že v dotazu Q=Q(E,B,v) je právě veškerá potřebná informace:

E ... množina entit, které jsou uvedeny v dotazu,

B ... množina atributů entit z E, na které se má projektovat odpověď,

v ... predikát, na základě kterého se provede restrikce odpovědi.

Jestliže umíme I zjistit, potom pro dotaz Q stanovíme odpověď ve tvaru

$P[R(I)_v]_B$, kde

R je restrikce a P projekce. Informaci I získáme operací J spojení. Algoritmus, který umí získat informaci I z topologie grafu g objasníme na dvou typech topologií:

a) souvislá cesta,

b) hvězda.

Informace obsažená v elementu souvislé cesty tvaru

e_i —o— e_j

$e_{i,j}$ kde $e_{i,j}$ je vztahová entita, vypočteme výrazem:

$$I_1=J(e_j,J(e_i,e_{i,j})_{def\ ei})_{def\ ej}$$

Jestliže souvislá cesta pokračuje dál

e_i —o— e_j —o— e_k

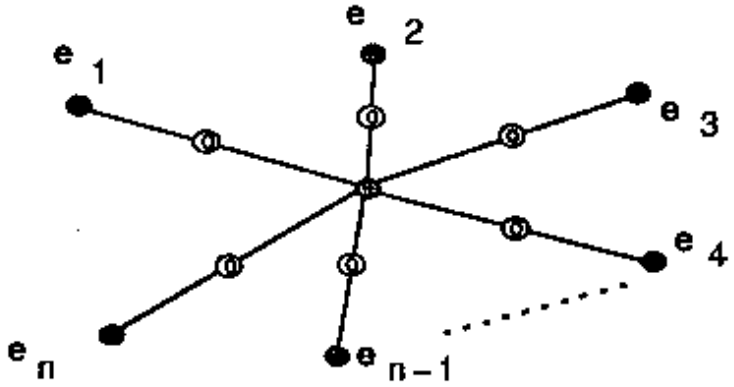
$e_{i,j}$ $e_{j,k}$ potom snadno vypočteme novou informaci I_2 výrazem:

$$I_2=J(e_k,J(I_1,e_{j,k})_{def\ ej})_{def\ ek}$$

Naznačeným způsobem můžeme pokračovat dál podle tvaru cesty

e_i —o— e_j —o— e_k —o— e_l ...

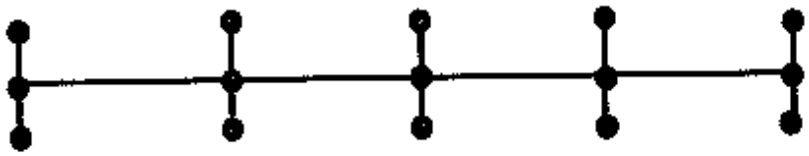
Poněkud jiný postup použijeme, má-li graf g hvězdicovou topologii



Algoritmus schématicky zakreslíme zápisem $I=\overset{n}{\underset{c=1}{J}} I$

kde $I_i=J(e_i,J(e_{0,}e_{0,i})_{def\ e0})_{def\ e0}$

Např. pro topologii "korál" ve tvaru



bychom již mohli použít obdobu postupu pro topologii hvězda.

Věta
Existuje algoritmus, který pro každý souvislý graf g zjistí informaci, kterou graf g reprezentuje

Tento algoritmus lze sestrojit v algebře [J,P,R] s respektováním postupu v g podle zásady shora - dolů, zleva - doprava. Takový algoritmus je obvykle součástí překladačů dotazů, protože je nutné přimět dotazový systém k jistým operacím nad bází dat. My použijeme takový algoritmus k prověrce informační dostatečnosti ERM modelu.

Na závěr této části již můžeme vyslovit cílovou definici pojmu semispolehlivost funkční a datové analýzy.

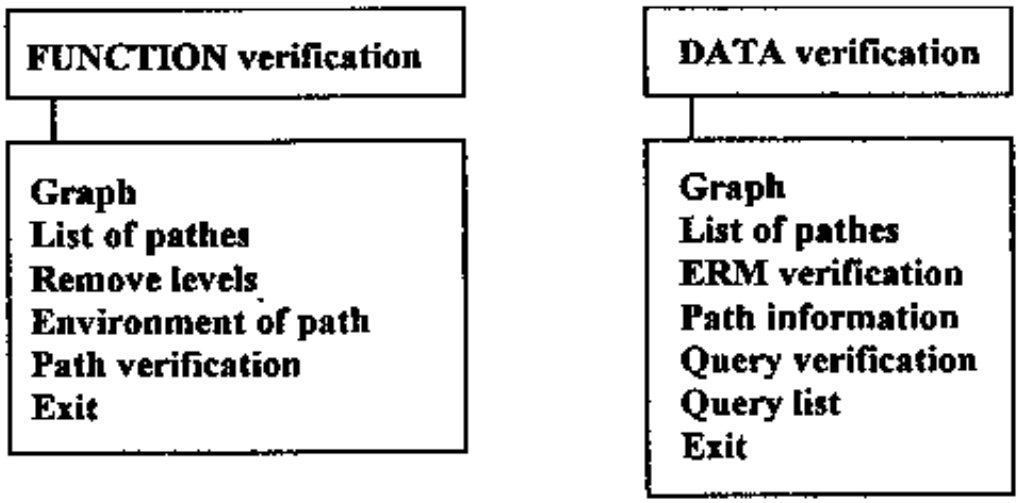
Definice 4
Funkční a datová analýza stávajícího a nového systému zpracování informace je semispolehlivá, jestliže je funkčně korektní a informačně dostatečná.

5. Praktické zabezpečení prověrky funkční korektnosti DFD diagramu a informační dostatečnosti ERM modelu

K podpoře prověrky funkční korektnosti a informační dostatečnosti navrhujeme systém, který se přidává k prostředkům typů CASE a plně využívá jejich prostředí. Takový systém plní dvě hlavní funkce:

- podpořit analytika projektanta při provádění prověrky funkční korektnosti DFD tak, že:
  - provede transformaci DFD na graf řízení
  - dává přehled cest
  - odstraňuje abstraktní úrovně v rozkladech
  - dá možnost vybrat si cestu, interaktivně ji modelovat a provádět prověrku funkční korektnosti v každém uzlu cesty,
- podpořit analytika projektanta při provádění prověrky informační dostatečnosti ERM modelu tím, že.
  - provede transformaci datového modelu na graf řízení
  - dává přehled o podgrafech různých topologií
  - má možnost doplňovat "úplnou" množinu dotazů
  - může provádět transformaci dotazu na graf, zjištění odpovědi.

K provádění obou hlavních činností jsou navrženy názvy menu, jejich spádová menu - viz Obr.9., řada oken (hlavních, pomocných) a postupy, které musí analytik zvládnout. Velmi náročný je postup prověrky funkční korektnosti, který vyžaduje od analytika podíl na řešení predikátů, systém mu ale nabízí k rozhodnutí všechny potřebné údaje. Jsou to stavy vstupujících entit, definované funkce, predikáty pro výsledné stavy, přehled výsledných stavů. Cestu v DFD může analytik vybrat dopředu, resp. může postupně vybírat uzly grafu řízení a využívat tak podpory systému.



Obr.9 Spádová menu pro systém interaktivní verifikace výsledků analýzy

Literatura:

[1] Manna, Z.: Mathematical theory of Computation. McGraw Hill 1974

[2] Jílková, H. Stanovská, I.: Strukturované postupy analýzy a návrhu informačních systémů. Programování 93 Ostrava

[3] Molnár, Z.: Moderní metody řízení informačních systémů. GRADA, Praha 1992

[4] Yourdon, E.: Managing the System Life Cycle. Yourdon Press 1988

[5] Yourdon, E.: Modern Structured Analysis. Prentice-Hall International, Inc. 1989