5. POSTUP TVORBY ESENCIÁLNÍHO MODELU

5.1. Druhy funkčních modelů vyjádřených DFD

Podle klasického strukturovaného přístupu návrhu systému lze modely vyjádřené pomocí DFD diagramů rozdělit na (DeMarco):

- fyzický model současného stavu,
- logický model současného stavu,
- logický model nového stavu,
- fyzický model nového stavu.

Fyzický model současného stavu slouží k poznání současného stavu systému (jak je nyní systém realizován - implementován). Vychází se z předpokladu, že analytik zpočátku nerozumí oblasti, kterou má navrhovat.

Při tvorbě logického modelu současného stavu se odstraňují implementační závislosti z předchozího modelu. Zobrazuje se, jak by stávající systém fungoval, kdyby byla k dispozici ideální technologie (funkce by probíhaly bez časového zpoždění atd.). Tento model však obsahuje i funkce a data, která nevyhovují, mají se změnit, nahradit nebo vypustit.

Logický model nového stavu vznikne přidáním nových požadovaných funkcí a změnami ve stávajících funkcích. Někdy může být úplně shodný s modelem předchozím, přecházíme-li např. jen na nový HW, nebo chceme změnit dobu odezvy, nebo nejsme schopni současný systém udržovat.

Fyzický model nového stavu ukazuje implementační omezení z hlediska uživatele, hranice automatizace systému ...

Všechny čtyři modely vyjadřují uživatelský (funkční) pohled.

Esenciální versus implementační model

Rozdělení na výše uvedené čtyři modely vychází z těchto předpokladů:

- analytik nezná zkoumaný systém,
- analytik není schopen hned vytvořit logický model nového stavu systému,
- transformace z logického modelu současného stavu do logického modelu nového stavu není zbytečná práce.

Tyto předpoklady však neplatí vždy a rozhodně neplatí pro všechny a pro celý modelovaný systém. Zároveň mnoho uživatelů pochopitelně pochybuje o smyslu pečlivého modelování systému, který přece bude odstraněn a nahrazen výsledkem návrhu nového systému.

Za normální situace 75% toho, co je ve fyzickém modelu současného stavu, je i v logickém modelu současného stavu. Pokud je procentní poměr příznivější, platíme za to jinak: fyzický model současného stavu je 3x až 4x rozsáhlejší než logický model současného stavu. Obsahuje redundance, verifikaci, validaci a odstraňování chyb (tj.to, co není v logickém modelu).

Je potřeba se snažit co nejrychleji vytvořit model nového systému - "esenciální model" a v některých případech část současného fyzického stavu systému. Tu část, která představuje esenciální procesy.

Esenciální model

Esenciální model - CO musí systém dělat, aby zajistil uživatelské požadavky. NESMÍ říkat, jak bude systém implementován. Tyto charakteristiky se odkládají na implementační modelování. Esenciální model předpokládá, že máme perfektní technologii a nulové náklady. Neuvažujeme, kdo co bude dělat (člověk, počítač...). Esenciální model je tedy implementačně nezávislý. Esence systému - podstata informačních procesů - žije cca 10 až 20 let. Za tuto dobu se většinou změní počítačová technologie. Změní se tudíž i implementace, ale esenciální model je stále pravdivý. Cum grano salis. Držme se ideálů, abychom nezahynuli.

Jak udělat esenciální model implementačně nezávislý

- Nesnažte se určovat pořadí funkcí v DFD modelech. Jediné pořadí mohou určovat směry toku dat nebo pořadí externích událostí s vlivem na systém.
- Esenciální model má obsahovat pouze nezbytné story. Data story, které nejsou potřeba při perfektní technologii, v esenciálním modelu nejsou.
- Dočasné (mezilehlé) soubory jsou potřeba v implementačním modelu z důvodu časového zpoždění funkcí (v noci batch, ve dne on-line) nebo

pro backup při chybě apod.
Esenciální data story jsou důsledkem externích asynchronních událostí: přijde faktura a informační systém musí zjistit, k jaké objednávce patří. Kdyby objednávky nebyly uchovávány v data storu, informační systém by nic nezjistil.

- Esenciální model nemá obsahovat odstraňování chyb a ověřování dat a procesů uvnitř systému. To je opět nutné v implementačním modelu, protože je třeba odstranit možnosti chyb. Některé funkce provádějí lidé - a lidé dělají chyby. Rovněž procesy si nemusejí v implementačním modelu předávat vždy jen správná data.
- Esenciální model nemá obsahovat redundantní nebo odvozená data - někdy se vyskytují v data storech pro urychlení přístupu. Ale toto doplnění se má provádět až ve fázi designu databáze podle konfigurační analýzy.

Části esenciálního modelu (dle YSM)

Esenciální model dle YSM se skládá ze dvou základních modelů: modelu prostředí a modelu chování. Každý z těchto modelů je tvořen z několika částí (dílčích modelů).

Model prostředí - popisuje hranice mezi systémem a okolím. Skládá se:

- z kontextového diagramu,
- ze seznamu událostí,
- ze stručného popisu účelu celého modelovaného informačního systému.

Model chování - popisuje požadované chování uvnitř systému, které zajistí správnou interakci s okolím. Skládá se:

- z DFD,
- z ERD,
- z DD,
- ze specifikace procesů,
- případně z STD.

Konverze části implementačního modelu současného stavu na esenciální model

Někdy je potřeba při tvorbě esenciálního modelu udělat i část implementačního modelu současného stavu (pro nejasné funkce, specifikaci a DD). Ale vždy jen nezbytnou část.

Při převádění implementačního modelu na esenciální musíme odstranit implementační závislosti:

- Oddělit datové toky sdružené na jednom médiu, které jdou do různých funkcí, nebo data, která spolu nesouvisí.
- Rozlišovat mezi podstatnou prací prováděnou "procesorem" a identifikací procesoru zakresleného v implementačním modelu (procesor = člověk, počítač ...). "Procesor" může dělat několik procesů nebo části. Sdružit dohromady ty esenciální procesy, které jsou spouštěny jednou externí událostí.
- Odstranit procesy, které pouze transportují data z jednoho místa systému do jiného (např. Print report, Čti data od uživatele ap.). Tyto procesy se přidávají až ve fázi designu.
- Odstranit procesy, které ověřují data, jež jsou systémem vytvářena a jsou v něm dále používána.
- Ověřit fyzické soubory (normalizace).
- Odstranit ze storů datové prvky, které nejsou potřeba.
- Odstranit implementační soubory (zajišťující časovou nesouvislost procesů - spooling soubory, report soubory apod.).

5.2. Postup při vytváření esenciálního modelu na základě analýzy událostí

Postup, kterým lze dospět ke konečnému funkčnímu modelu systému vyjádřenému hierarchií DFD, může být různý. V současné době se používají tři postupy:

- rozdělováním procesů (funkční dekompozice),
- postupem podle datových toků,
- na základě analýzy událostí.

Popíšeme nejprogresivnější z těchto postupů, vytváření esenciálního modelu na základě analýzy událostí.

Tato technika je poměrně nová (Yourdon: "Modern Structured Analysis" 1989). Analýza událostí byla objevena jako východisko z krize klasického hierarchického rozkladu. Čistý postup shora dolů se intuitivně jeví jako velice působivý (nejdříve kontextový diagram, potom DFD úrovně 0 a dále nižší a nižší úrovně rozkladu). Ale tento postup - zvláště funkční dekompozice - může být dost problematický. (Co je nižší úrovní? Podle čeho rozdělovat? Podle tradičních subsystémů nebo podle terminátorů? Všechny datové toky od a do jednoho terminátoru budou obslouženy jednou funkcí ...???)

Jako přirozenější cesta se jeví identifikovat všechny externí události, na které musí systém reagovat. Tento postup je vlastně rozšířením a zdokonalením postupu "od výstupních datových toků". Události se využijí k vytvoření prvního předběžného DFD. Funkce z tohoto předběžného DFD se sdružují (skládají) směrem nahoru (vytváří se vyšší úroveň DFD) a rozdělují směrem dolů (nižší úrovně DFD).

Esenciální model dle YSM se skládá z modelu prostředí a modelu chování. Nyní ukážeme, v jakých kontextech v těchto modelech události figurují.

Cílem modelu prostředí systému je:

- určit, co je částí systému a co ne,
- definovat interface mezi systémem a zbytkem světa (tj. prostředím, okolím systému),
- identifikovat události, které nastanou v okolí systému, a systém na ně musí reagovat.

Model chování systému zviditelňuje, co systém dělá, jaká data potřebuje: DFD, ERD, DD, specifikace procesů ap. Z hlediska návrhu funkcí se postupuje takto:

- nejdříve se vytvoří hrubý, předběžný model systému (pro každou událost jedna funkce - transakce),
- z předběžného modelu se vytvoří konvencím odpovídající hierarchie DFD. Doplní se specifikace elementárních funkcí a DD. Zároveň se provádí datová analýza. Tím se zajistí vzájemné ovlivňování funkční analýzy, datového modelování a datové analýzy.

Model prostředí se skládá ze:

a) stručného popisu účelu systému,

b) kontextového diagramu,

c) seznamu událostí.

ad a) Účel systému se vyjádří textem. Slouží pro řízení a pro každého, kdo není přímo zatažen do vývoje systému. Měl by být dlouhý maximálně jeden odstavec. Nejde o úplný popis systému.
Příklad: Účelem systému "Knižní zásilková služba" je zajistit zpracování všech údajů o objednávkách knih od zákazníků, dodávky knih zákazníkům a fakturaci, sledování plateb zákazníků a prošlých faktur. Poskytnout vedení informace o provedených prodejích.

ad b) Kontextový diagram zobrazuje (viz též předchozí výklad úrovní DFD):

- lidi, organizace, systémy, které s modelovaným systémem komunikují (terminátory),
- data, která systém dostává z okolí a která musí nějak zpracovat,
- data produkovaná systémem do okolí,
- data story sdílené systémem a terminátory (jinými systémy). Tyto externí story buď jsou vytvářeny vně systému a systém je využívá, nebo naopak je systém vytváří a poskytuje vně.

ad c) Seznam událostí obsahuje soupis stimulů z okolí, na které musí systém reagovat.

Příklad: Zákazník pošle objednávku (příchod dat)

Zákazník zruší (stornuje) objednávku (příchod dat)

Vedení požaduje informace o prodeji (časová událost)

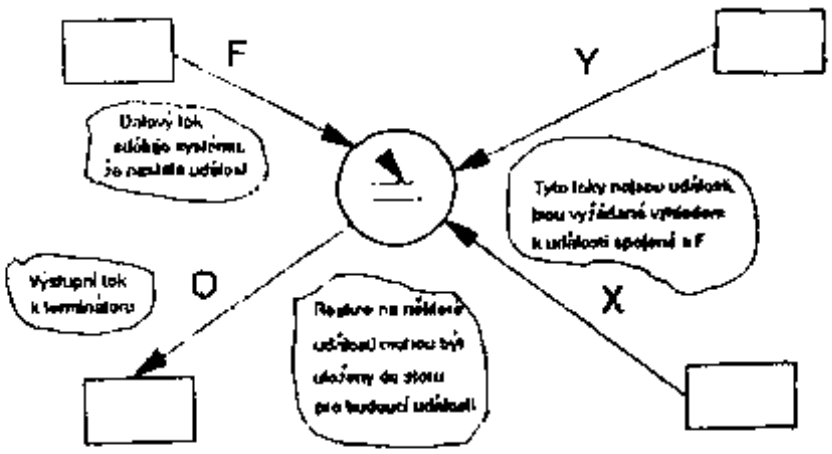
Typy událostí (stimulů).

V YSM se rozlišují tři typy událostí. Klasifikačním kritériem je druh stimulu, podle něhož modelovaný systém rozpozná, že událost nastala.

F: flow-oriented (událost spojená s příchodem dat do systému, v kontextovém diagramu se projeví datovým tokem).

T: temporal (časová událost - nastává v určitém časo vém okamžiku. Není spojená s datovým tokem. Před pokládají se jakési vnitřní hodiny systému. Např. "Výstupní sestava všech objednávek má být vytiště na každý den v 9.00").

C: control (řídící událost - spojená s řídícím tokem z okolí systému, vyžádání reakce něčím vně systému. U systémů běžného řízení podniku - ekonomického - se nevyskytují, patří k real-time systémům).



Obr. 18. Model prostředí - typy událostí

Prohlédněte si obr. 18 a uvědomte si, jaké toky mohou figurovat v kontextovém diagramu. Ne všechny vstupní datové toky v kontextovém diagramu musí být typu F (spojené s externí událostí). Některé mohou být systémem vyžádané za účelem toho, aby mohl systém na událost reagovat.

Model prostředí je v podstatě koncept systému. Navíc by tedy měl obsahovat DD všech externích datových toků (a případně i storů) v DD a hrubý datový model systému (viz datové modelování).

Postup tvorby modelu prostředí

Tento model se postupně upravuje, neustále se k němu vracíme - jeden člověk většinou nezná celý systém.

1. Celý systém nakreslíme jako jednu funkci a dáme jí jasné jméno, vypovídající o tom, co systém je.
2. Přidáme terminátory a dáme jim názvy, které vyjadřu jí jejich role vzhledem k systému.
3. Přidávají se datové (případně řídící toky).
   - Každý vstupní datový tok musí být potřeba k vyprodukování reakce nebo k zjištění, že nastala nějaká událost, nebo k obojímu.
   - Každý výstupní datový tok musí být reakcí na událost.
   - Pro každou "F" a "C" událost musí existovat vstup ní tok, aby systém rozpoznal, že nastala.
   - Reakcí systému na každou událost je buď vyprodukování výstupu do okolí, nebo uložení dat, která se použijí pro pozdější výstup, nebo změna stavu systému (u real-time systémů).
   - Při velkém množství vstupů a výstupů se použije jejich kompozice (u velkých systémů jich jsou stovky). Ale podle výše uvedených pravidel se kontrolují nesložené toky.
   - Seznam událostí se doplní i na základě ERD (potřeba vzniku dat a jejich aktualizace). Nové události se doplní do kontextového diagramu.

Model chování systému

Model chování systému je tvořen:

a) úplnou hierarchií DFD diagramů pro celý systém (jaké funkce systém plní),

b) úplným slovníkem dat obsahujícím popis složení a dalších charakteristik všech datových toků a storů,

c) minispecifikacemi všech elementárních funkcí systému,

d) úplným datovým modelem tvořeným normalizovanými entitami (konceptuální schéma dat),

e) u real-time systémů všemi diagramy přechodu stavů (STD).

Všechny tyto prvky musí být vzájemně konzistentní a provázané. Metodicky problematická a náročná je z bodů a) až e) tvorba hierarchie DFD (bod a), datový model (bod d) a diagramy přechodu stavů (bod e). Tvorbou hierarchie DFD s využitím konceptu událostí se nyní budeme zabývat podrobněji. U ostatních bodů odkazujeme čtenáře na příslušné kapitoly o nástrojích. Zde totiž koncept událostí nevnáší metodické změny.

Postup tvorby úplné hierarchie diagramů systému

Již jsme si vysvětlili možné postupy tvorby DFD. Možné problémy těchto postupů jsme si také naznačili:

- "ochrnutí analýzy": hledá se inspirace, jak systém rozdělit,
- úkaz počtu analytiků: pro každého analytika se vytvoří jedna funkce v DFD úrovně 0 (mám-li 6 analytiků, bude 6 funkcí ap.), ale co se stane, bude-li analytiků víc, nebo některý odejde?
- svévolné "fyzické" dělení: současný systém se rozdělí podle stávajících oddělení v organizaci, což opět není nejšťastnější způsob. Kdo zaručí, že to je to pravé? Návrh systému zahrnuje přece i organizační změny, tudíž stávající organizace nebude třeba v budoucnu platit.

Možným lepším řešením může být: postup "od středu nahoru a dolů" (middle-out), tvorba modelu podle událostí.

Vytvoří se a) úvodní předběžný DFD (množina DFD) - pro každou událost jedna funkce.

b) Spojováním těchto funkcí směrem nahoru a rozdělováním směrem dolů vznikne hierarchie DFD.

a) Postup tvorby úvodního (předběžného) DFD:

a1) Pro každou událost ze seznamu událostí se nakreslí jedna funkce (proces, transformace) a provedou se následující dva kroky. Každé takto vzniklé funkci se přidělí číslo shodné s číslem události, na kterou je reakcí.

a2) Funkce se pojmenuje tak, aby popisovala (vystihovala) reakci systému na související událost. Např. událost je - "zákazník platí za zboží", název funkce je - "Aktualizace příjmových účtů".

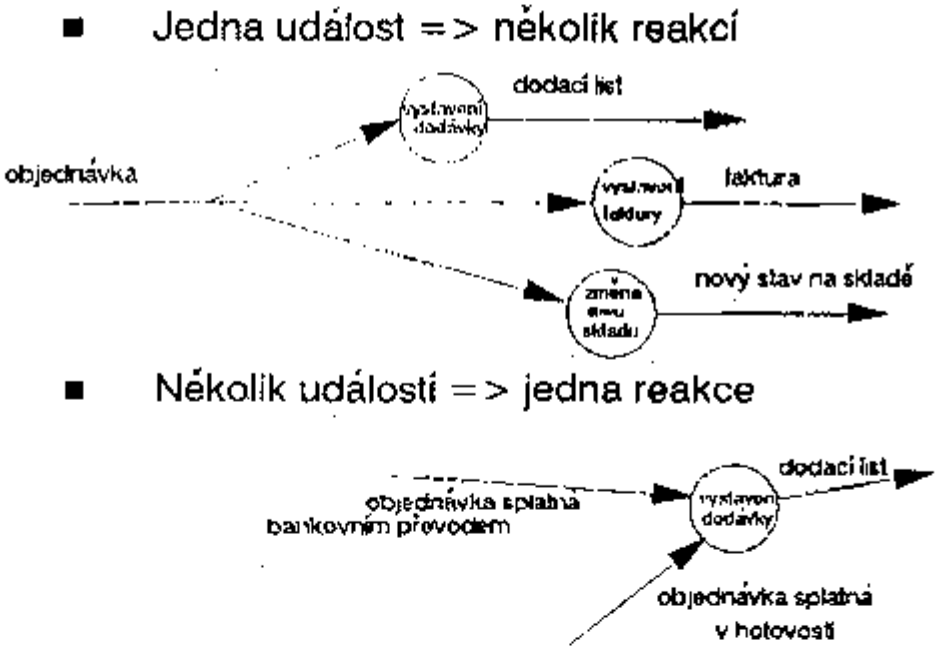
a3) Nakreslí se potřebné vstupy a výstupy, aby funkce mohla zajistit požadovanou reakci na událost, a nakreslí se data story, které jsou potřeba pro komunikaci mezi funkcemi.

a4) Výsledek tohoto předběžného DFD je ověřen s kontextovým DFD a soupisem událostí, zda je úplný a kompletní. Platí pravidla daná pro úrovně DFD:

- každý vstup z kontextového DFD musí být v předběžném DFD
- každý výstup z kontextového DFD musí být v předběžném DFD
- každý výstup produkovaný jednou funkcí předběžného DFD jde do okolí systému nebo do storu.

Na obr. 19 jsou uvedeny speciální případy vztahu "událost - reakce na událost".

Speciální případy: - jedna událost způsobí několik reakcí,
- několik událostí vede ke stejné reakci.



Obr. 19. Model chování - speciální vztahy vztahu "událost - reakce"

Propojení funkcí (= reakcí) na události je pouze přes data story.

V předběžném DFD nejsou funkce propojeny přímo datovými toky (ale pouze přes data story), protože události jsou asynchronní. Reakce na jednu událost ale může požadovat data produkovaná na základě jiné události. Žádným způsobem nemůžeme zjistit, kdy událost nastane. V esenciálním modelu předpokládáme dokonalou technologii (jak jsme o tom již také hovořili), tedy že každý proces bude svoji činnost provádět

nekonečně rychle a že každý datový tok působí jako trvalý přísun dat libovolnou rychlostí. Z toho vyplývá, že synchronizace několika nezávislých událostí je možná pouze přes data store.

Takový data store, "esenciální data store", není tedy storem nutným z důvodu nedokonalé technologie (kvůli zpoždění funkcí), ale kvůli časovým vztahům mezi procesy (jsou asynchronní).

Poznámka: Pokud máme nezávisle vytvořený ERD - provede se křížové prověření ERD a DFD. Story zkusmo definované v DFD mohou být použity jako východisko k entitám v předběžném ERD a naopak - vzájemná výpomoc datového a funkčního modelování, ani jedno neřídí druhé.

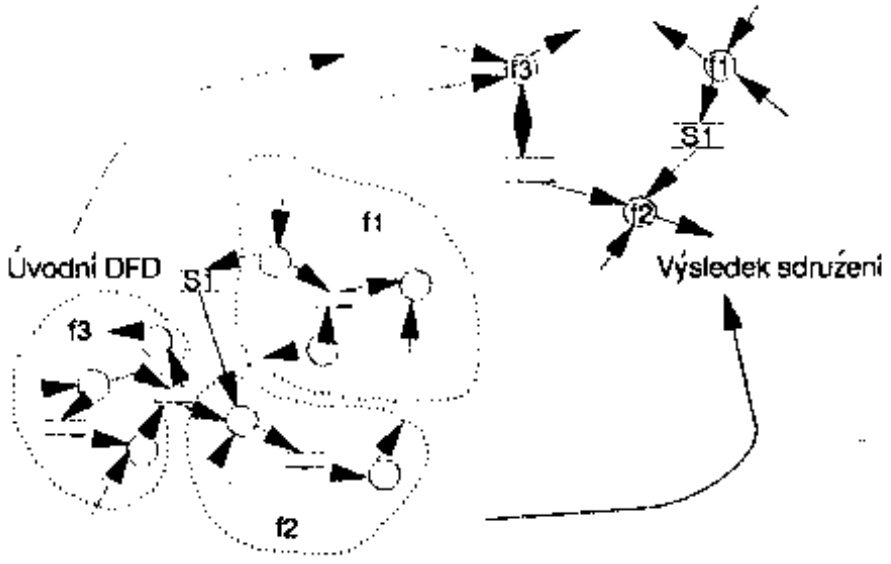
b) Spojování a rozdělování funkcí (dokončení funkční části modelu chování)

Předběžný model systému nelze prezentovat uživateli, ani použít pro následující činnosti, protože:

- je příliš složitý. Pokud systém musí reagovat na 100 událostí, je v modelu 100 procesů. Každý takový proces může být sám o sobě ještě hodně složitý (není elementární),
- je spíš grafický s málo textovými údaji. Musí být doplněn slovník dat (DD) a specifikace procesů.

Vytvoření úrovní DFD při spojování a rozdělování funkcí

b1) Procesy (funkce), které spolu souvisejí, se sdruží do smysluplných agregátů, z nichž každý reprezentuje 1 funkci DFD vyšší úrovně. Schéma viz obr. 20.



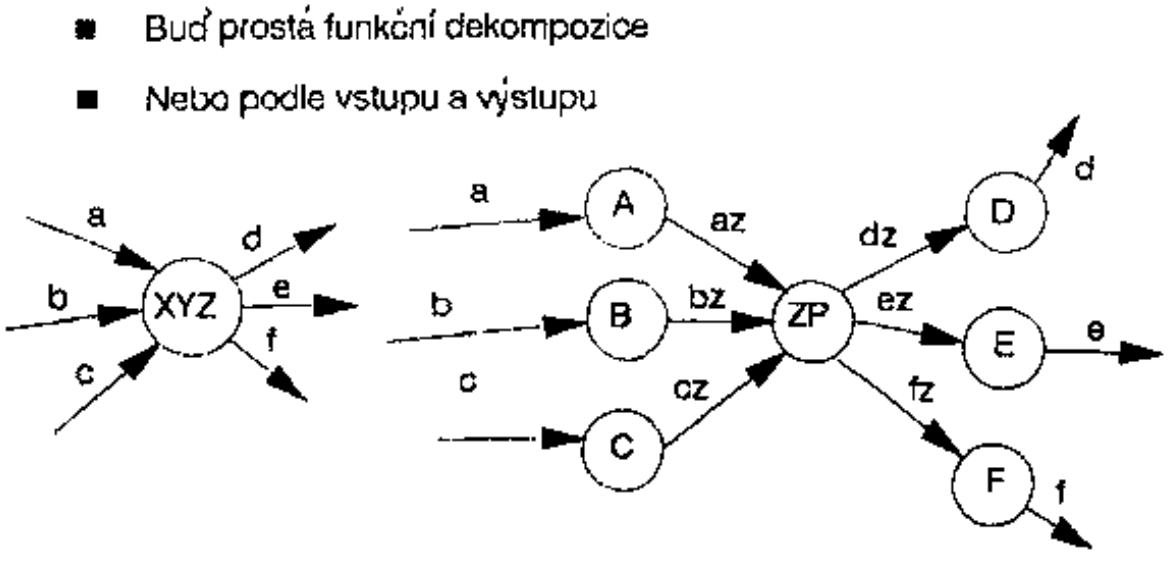
Obr. 20. Model chování - sdružování procesů

Vodítka pro sdružení:

- Každá skupina procesů by měla zahrnovat procesy, které se týkají spolu úzce souvisejících reakcí - to většinou znamená, že takové procesy pracují se stejnými nebo spolu souvisejícími daty.
- Zjistit možnost ukrytí uchovávaných dat, která se vys kytují na nižší úrovni. Jestliže se najde skupina pro cesů, které potřebují společný store, a ostatní proce sy jej nepoužívají, vytvořte vyšší úroveň tak, aby ta kový store skryla (princip information hiding pro ob jektový přístup).
- Pamatujte si, že ten, kdo DFD "čte", toho nechce vědět hodně najednou. Dodržujte magické číslo 7 (± 2).

b2) Pro rozdělení příliš složitých procesů potřebujeme rovněž pravidla. Cílem je dosáhnout takové úrovně rozkladu, kdy pro každou funkci lze napsat max. 1 strán ku A4 specifikace.
Vodítka pro rozdělení příliš složitých procesů (reakcí):

- V některých případech vystačíme s prostou funkční dekompozicí (problém již je jasný, pro každou vyčleněnou funkci nakreslíme jednu bublinu, viz postup funkční dekompozice),
- Nebo nakreslíme pro každý výstupní datový tok z transakce jednu bublinu, pro každý vstupní datový tok jednu bublinu a mezi ně vložíme funkce vlastního zpracování. Ilustrativní schéma je na obr. 21.



Obr. 21. Model chování - vodítka pro rozdělení transakcí

- Nebo vytváříme posloupnost funkcí "odzadu", "rozkladem" výstupů.

Tento krok je již hodně intuitivní.

Nezapomeňte na neustálé sledování konzistence.

Dále je třeba zajistit úplnost slovníku dat, doplnit všechny minispecifikace elementárních procesů (funkcí), dokončit datový model, (případně dokončit STD systému s real-time charakteristikami).

Závěr k modelu chování

Mnoho analytiků žije v představě, že DFD jsou to jediné, co potřebují, aby mohli dělat strukturovanou analýzu. Na jedné straně to vypovídá o "síle" DFD - často jsou jedinou věcí, kterou si lidé zapamatují po přečtení nějaké knihy nebo po absolvování školení o analýze a návrhu systému. Na druhé straně je to velice nebezpečné. DFD nelze použít bez dalších nástrojů modelování a bez techniky, jak je vytvářet. Dokonce i když jsou vztahy mezi daty a časové charakteristiky systému triviální (což bývá málokdy), je nezbytné DFD kombinovat s DD a minispecifikací.

V tomto okamžiku máte k dispozici nutné informace o tom, jak se vytváří esenciální model. Úlevný okamžik, v němž je možno začít programovat, však ještě nenastává. Nyní musíme překlenout propast mezi esencí a implementací systému.