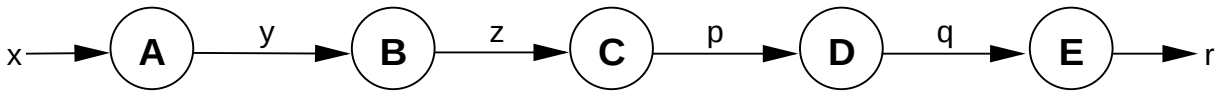
Strukturovaný návrh systému

- také funkčně orientovaný návrh (function-oriented design)
- vstupem jsou:
  - model kontextu systému resp. DFD úrovně 0
  - množina diagramů datových toků (DFD)
  - specifikace činnosti elementárních procesů (pseudokódy, Nassi-Shneidermanovy diagramy, rozhodovací tabulky a stromy)
  - datový slovník
  - ERA diagramy

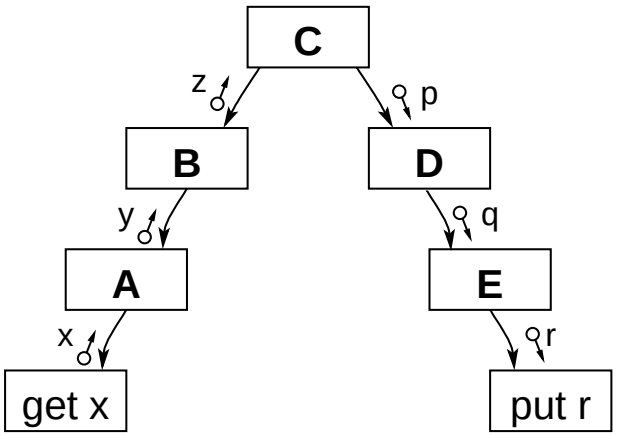
- produktem návrhu architektury je rozhodnutí, které části esenciálního modelu budou přiřazeny jednotlivým podsystémům, případně na které počítače nebo procesory budou alokovány (např. celý systém bude implementován na jednom počítači)

Podrobný návrh aplikace

- v rámci podsystémů transformujeme DFD na hierarchii podprogramů (každá bublina se stane podprogramem)
- např. z DFD



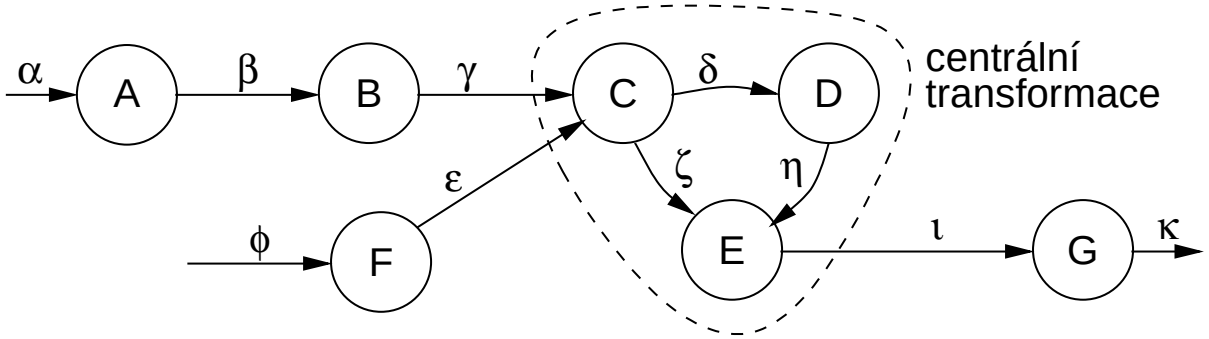
vzniknou podprogramy s následující hierarchií:



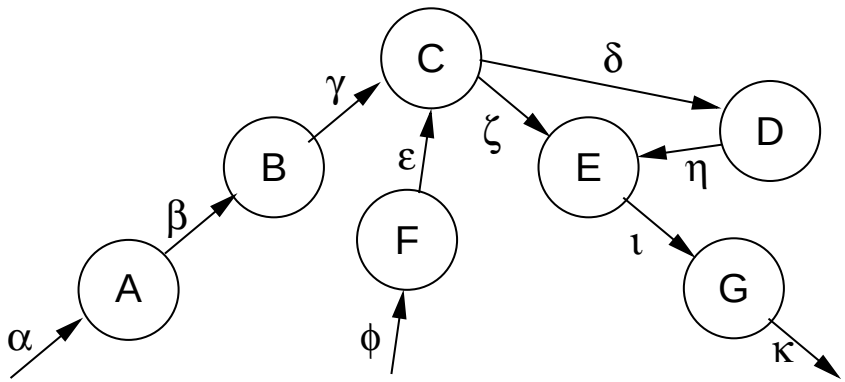
Transformace DFD na hierarchii podprogramů

v obecném případě není převod DFD tak přímočarý, používá se následující postup:

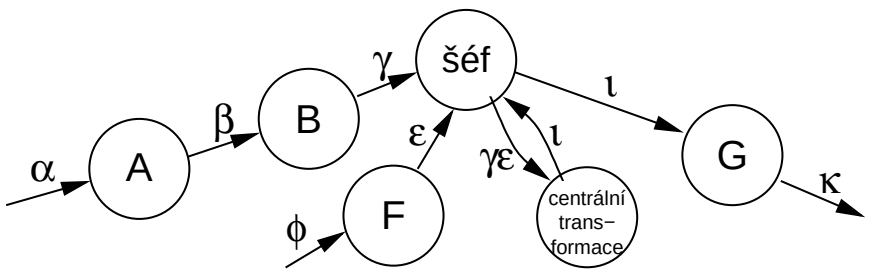
- *vycházíme z DFD nejnižší úrovně (bubliny = elementární procesy); pro každý DFD provedeme:*
  - identifikujeme centrální transformaci = část DFD popisující základní fce systému, nezávislá na implementaci vstupů a výstupů
  - dva způsoby hledání centrální transformace:
    - předpokládáme "ideální svět", kde vstupy neobsahují chyby, výstupy nemusí být formátovány apod.; odsekáváme všechny nepotřebné vstupní a výstupní proudy, zbytek je centrální transformace
    - najdeme "střed" DFD odhadem (to je horší varianta, ale nic jiného nám nezbývá, pokud nedokážeme centrální transformaci identifikovat jinak)
  - kolem centrální transformace nakreslíme obláček



- *vytvoříme hierarchii procesů*
  - vyjadřuje vztah nadřízenosti a podřízenosti
  - nejprve musíme najít kandidáta na šéfa
    - pokud je dobrý kandidát (v centrální transformaci), zvedneme ho a necháme ostatní bubliny viset dolů, viz obrázek (a)
    - pokud je potenciálních kandidátů více, vyzkoušíme, který nám poskytne nejlepší návrh



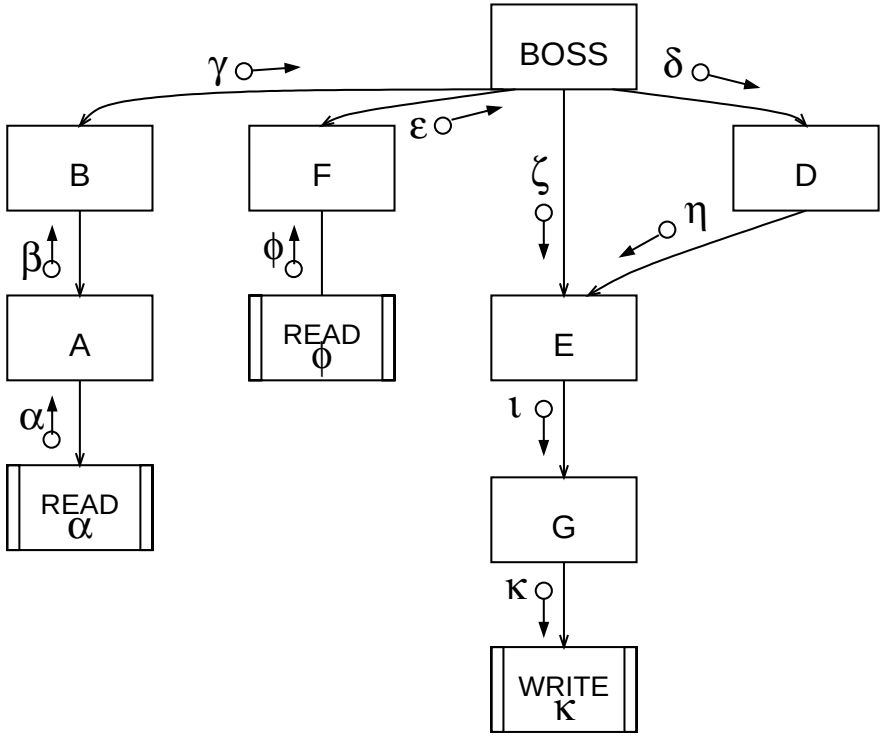
a) vyzvednutí šéfa



b) vytvo ení nového šéfa

- pokud není dobrý kandidát, vytvoříme nového šéfa, centrální transformaci nahradíme šéfem a původní transformaci pod šéfa zavěsíme (jako jednu bublinu, vstupy a výstupy budou proudit přes šéfa; viz obrázek (b))

- *transformace DFD na hierarchii bloků (každá bublina se stane blokem)*
  - "šéf" se stane řídícím blokem, ostatní budou jeho podřízení
  - vytvoříme počáteční strukturogram

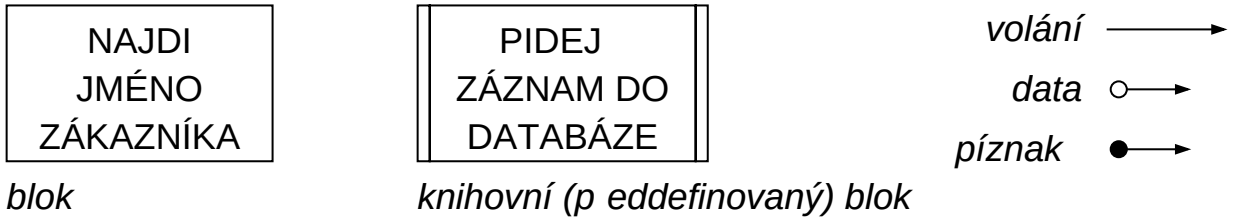


- šipky ukazují směr volání, přidali jsme bloky pro vstup a výstup
- jména bloků by měla odpovídat jejich rolím v hierarchii, tj. nemusejí odpovídat názvům bublin - jméno by mělo sumarizovat i činnost podřízených bloků
    - např. názvy bloků - postupně: "získej položku" (blok pro čtení dat), "získej transakci" (pomocí nižšího bloku vytvoří transakci), "získej platnou transakci" (ověří platnost transakce)

- *strukturovaný návrh se znázorňuje pomocí tzv. strukturogramů*

    - ukazují rozdělení systému do hierarchie bloků a jejich vzájemnou komunikaci
    - blok představuje podprogram programovacího jazyka; blok se znázorňuje jako obdélník obsahující název bloku
    - knihovní procedury a funkce (předdefinované bloky) se znázorňují pomocí zdvojení svislých čar obdélníka

- volání podprogramu je znázorněno obyčejnou šipkou (od volajícího k volanému)
- předávání zpracovávaných dat je znázorněno šipkou s prázdným kroužkem (šipka směřuje od odesilatele k příjemci dat)
- předávání příznaků je znázorněno šipkou s plným kroužkem



- *úprava strukturogramu*

  - přidáme čtecí a zápisové bloky, bloky pro čtení z databáze apod.
  - centrální transformaci můžeme rozdělit podle DFD
  - přidáme bloky pro obsluhu chyb
  - pokud jsou potřebné, přidáme bloky pro inicializaci a ukončení programu
  - pokud je nesprávný výběr šéfa, změníme ho (nesprávný výběr šéfa se obvykle projeví tak, že pravý šéf signalizuje svému nadřízenému, co má dělat)

- *ověření funkčnosti návrhu*

  - implementuje strukturogram požadavky vyjádřené v DFD?
  - pokud si nějaký blok potřebuje vyžádat data, může to udělat? pokud ne, můžeme změnit hierarchii volání

- *výše uvedené kroky provedeme pro jednotlivé DFD, máme k dispozici množinu nezávislých strukturogramů*

  - vytvoříme nadšéfa, který bude volat jednotlivé šéfy
  - optimální je např., pokud nadšéf obsahuje konstrukci case, která vyvolá některého podřízeného (např. výběrem položky v menu)

- *výsledný produkt strukturovaného návrhu*

  - rozdělení systému do bloků
  - jeden nebo více strukturogramů, z bloků vzniknou podprogramy
  - neřeší sdružení podprogramů do modulů nebo návrh vnitřku podprogramů (zde můžeme vycházet z pseudokódu)

Vytváření modulů

- modul = množina dat a podprogramů, například zdrojový soubor v C - bývá obvykle definován jako nejmenší jednotka zdrojového textu programu, kterou můžeme samostatně přeložit

Vylepšení modularity podsystému

- viz provázanost, soudržnost …později