Structured Software Design a jeho podpora produktem CASE SDW

Helena Jilková

1. Yourdon Structured Method

V minulých dvaceti letech prošla metodologie projekce a programování ekonomických aplikací rychlým vývojem: od nových verzí jazyků 3. generace k jazykům 4. generace, od třídicích generátorů k aplikačním generátorům, od editorů k integrovaným prostředím, od konceptů strukturovaného programování k systemizovaným postupům pro celý životní cyklus projektu. Yourdonova strukturovaná metoda (**Yourdon Structured Method, YSM**) je v současné době v projektování nejvýznamnější představitelkou tzv. **funkčních přístupů**. Tyto přístupy své označení odvozují z toho, že za východisko analýzy považují **inventarizaci** externě viditelných **funkcí** projektovaného systému. **Fáze** projekčního postupu metodou **YSM** jsou zachyceny na **obr.**1.

Několik vysvětlujících poznámek k obr. 1:

a) Těžištěm zájmu tohoto příspěvku jsou aktivity ve fázích "Esenciální analýza" a "Design" (nejdůležitější z hlediska metodických specifik YSM).

b) Esenciální analýza zkoumá stávající systém a vytváří koncepci (model) systému projektovaného (jeho automatizovaných i neautomatizovaných složek). **Esenciální specifikace** je tedy model projektovaného systému jako celku. V metodě YSM **obsahuje** minimálně datový model (např. ERA-model a "slovník dat") a **systém DFD (Data Flow Diagrams).**

c) Východiskem esenciální analýzy je tzv. **kontextový diagram** (DFD úrovně 0). Kontextový diagram se na analyzovaný systém dívá jako na černou skříňku, která s významnými prvky okolí - **terminátory** - komunikuje datovými toky (**příklad viz obr.** 3). Analyzovaný systém je možné i v kontextovém diagramu rozdělit do více **transformací** (subsystémů) realizujících zpracování datových toků. Důvodem je snaha lépe zviditelnit to, co systém dělá - potenciálně se tak odhalí další potřebné datové toky a je rovněž vidět hrubý algoritmus zpracování představovaný posloupností transformací.

d) Každá projekční metoda řeší jako klíčový problém, jak rozdělit celý systém na části. Současná **YSM** používá tzv. **členění dle událostí** (viz "seznam událostí" jako vstup

do esenciální analýzy v obr. 1). YSM si vypůjčila pojem události a transakce ze systémů reálného času, zobecnila jej a nahradila inventarizaci funkcí soupisem událostí, které nastávají ve významném okolí analyzovaného (nebo projektovaného) systému.

Transakce je vymezena určením jejích 5 složek:

- událost (event) ve významném okolí systému,
- impuls (stimulus) do systému,
- činnost (activity) systému,
- reakce (response) systému,
- důsledek (effect) pro významné okolí systému.

Příklady událostí pro systém Školní knihovna:

- nová publikace do evidence,
- vyřazení publikace,
- nový výtisk existující publikace,
- čtenář chce vrátit výtisk určité publikace,
- čtenář chce rezervovat určitou publikaci,
- příchod nového čtenáře atd.

Příklad vymezení transakce pro systém Školní knihovna:

- událost: příchod nového čtenáře,
- impuls: knihovnice zadá volbu "nový čtenář",
- činnost: vytvoření záznamu "nový čtenář", vystavení čtenářského průkazu,
- reakce: hlášení o úspěšném založení nového čtenáře, vytištění průkazu,
- důsledek: terminátor "čtenář" má o jednu instanci více.

e) Při funkční inventarizaci se analytik nutně ve funkcích topil a to ho vedlo k tomu, aby do inventarizace vnášel sdružovací hlediska a hierarchii. Při soupisu událostí **hierarchizovat** a sdružovat **nesmí**. Mraky událostí, stejné množství transakcí, **jeden společný DFD obsahující pro každou transakci jednu transformaci (bublinu).**

A nyní teprve přichází úlevná chvíle, kdy analytik smí bubliny transakcí začít **sdružovat.** Doporučeným kritériem sdružování jsou **společná data.** Sdružením vzniknou nové transakce, které se budou dále **hierarchicky rozkládat.** Pro každou bublinu lze zkonstruovat **DFD nižší úrovně,** v němž je jedna transformace výchozí úrovně rozepsána jako síť procesů (transformací) nižší úrovně. **Příklad viz obr. 4 a 5** (dle společných dat jsme sdružili všechny transakce související s vypůjčováním a vracením výtisků publikací).

f) Až do nedávné doby byla "implementační nezávislost logického modelu systému" zaklinadlem většiny projekčních metodologií. YSM díky své pragmatičnosti tuto utopii zdařile opustila: čím dříve a čím více víme o implementačním prostředí, pro něž je projekt určen, tím spíše můžeme toto prostředí ovlivnit (nebo alespoň poz-

nat), a *tím* lepší projekt potom vytvořit. Úprava esenciální specifikace podle možností konkrétního implementačního prostředí je nejdůležitější součástí konfigurační analýzy.

2. Structured Software Design

Strukturovaný návrh programového systému (**Structured Software Design, SSD**) představuje tu fázi YSM, která z konfiguračně upravené esenciální specifikace vytváří **modulární strukturu cílového programového systému a specifikace jednotlivých modulů**. (Konfigurační analýza rozčlenila projektovaný systém podle časových, datových a všech možných jiných rozpoznatelných nároků transformací na části, které budou v etapě SSD řešeny jako samostatné programové systémy. Postup SSD budeme vysvětlovat pro jeden programový systém, v reálné projekci se tento postup samozřejmě uplatňuje opakovaně, pro každý z cílových programových systémů.)

Esenciální a konfigurační analýza poskytuje fázi designu DFD pro každou transakci a informaci o sdružení transakcí (např. též ve formě společného DFD). Pro tyto sdružené transakce se vytváří modulární struktura programového systému postupem, jehož hlavní **činnosti** jsou zachyceny na **obr.** 2.

SSD vyjadřuje modulární strukturu programového systému strukturním schématem (**structure chart, SCH, příklad viz obr.** 7). DFD je síť transformací, SCH zobrazuje **hierarchizovanou síť modulů**. DFD se na structure chart převádí pomocí **nalezení tzv. centrální transformace** v DFD. Centrální transformace představuje nejdůležitější část algoritmu vyjadřovaného DFD - algoritmus zpracování v ideálním světě, v němž vstupy jsou vždy správné a výstupy nevyžadují žádné formátování. Centrální transformace se získá oddělením aferentních a eferentních větví DFD (příklad pro DFD "Aktualizace kmenového souboru": aferentní větve - čtení, kontrola, příp. shromažďování změnových údajů, centrální transformace - vlastní aktualizace, eferentní větve - formátování a výpisy výstupů).

Po identifikaci centrální transformace lze vytvořit první, **hrubý SCH**, a to:

- buď **zvolením kořene** z transformací v rámci centrální transformace,
- nebo **dosazením nového kořene** nad centrální transformaci a aferentní a eferentní toky

(**příklad** pro DFD z obr. 4 a 5 je na **obr. 6 a na obr.** 7).

Hrubý SCH se dále zpodrobňuje a upravuje podle principů strukturovaného designu v následujících krocích:

a) Nejprve se využijí procesy uvnitř centrální transformace a procesy uvnitř nižších úrovní DFD jako vodítko k odhalení (vydělení) nižších úrovní modulů.

b) Přidají se moduly čtení, zápisu, přístupu do databáze....

c) Provede se tzv. **factoring** - reorganizují se moduly (spojí, či rozdělí) s cílem např. zmenšit velikost modulu (nebo naopak odstranit moduly malé a příliš jednoduché), minimalizovat duplicitu kódu, vytvořit užitečné opakovaně využitelné moduly, zjednodušit implementaci apod.
d) Přidají se moduly ošetření chyb.
e) Přidají se inicializační a závěrečné moduly.
f) Ověří se, zda názvy modulů korespondují s jejich rolí v celé hierarchii.
g) Přidají se nezbytné indikátory (flagy).
h) Ověří se vlastnosti modulů a jejich rozhraní a v případě potřeby se moduly opět reorganizují (**prověřované vlastnosti: soudržnost každého modulu, spřaženost modulů**, nutnost stavové paměti v modulu, korespondence programové struktury modulu s datovými strukturami, počet podřízených a počet nadřízených modulu, využití datových modulů, redundance kódu, "tvar systému" - systém by neměl být řízen fyzickými vstupy, způsob ošetření chyb, hierarchie kontrol, oddělení rozhodnutí od výkonu akce spjaté s rozhodnutím, oddělení práce od řízení).
i) Ověří se vhodnou simulací, zda systém specifikovaný SCH poskytuje funkce vyjádřené ve výchozích DFD.
j) Jestliže byly kroky a) - i) provedeny pro každou ze "sdružených transakcí", potom se těmto transakcím nadřadí **společný řídicí modul (nebo transakční monitor)**. Jeho funkcí je přijímat impuls, rozpoznat, které transakci patří, a tuto transakci spustit. Jak prosté.

3. Podpora SSD produktem CASE SDW

SDW (**System Development Workbench**) je produkt typu CASE (**Computer Aided Software Engineering**) holandské firmy Pandata. VŠE Praha získala tento produkt k výzkumným účelům. Za symbolickou cenu. Neúplný (bez "lower CASE" - bez generování aplikací ze specifikace). S povolením k instalaci na jedné pracovní stanici PC. Těmito výchozími podmínkami jsou jistě ovlivněny mé závěry, týkající se podpory strukturovaného designu produktem SDW. Nicméně troufám si za nimi dočasně stát:

- Každý **CASE je spjat s určitou metodologií**, kterou primárně podporuje a kterou je třeba **velmi dobře zvládnout**, má-li CASE skutečně pomoci vytvářet kvalitnější projekty. Jestliže CASE umožňuje např. malovat diagramy i z jiných metodik, znamená to skutečně pouze to, že s ním lze malovat i jiné obrázky.
  CASE SDW podporuje metodologii **SDM (System Development Methodology)** firmy Pandata, YSM (alespoň v její knižní podobě) nepodporuje.
- Přínos pro tvorbu kvalitní dokumentace projektu je značný (provázanost všech částí CASE **systémovou encyklopedií** nutí uživatele CASE, aby jeden objekt

označoval stále jedním jednoznačným identifikátorem. Kvalitu obrázků můžete zhodnotit na tomto příspěvku).

- Produkt CASE bez generování je spíše pěkná hračka.
- Dvě poznámky přímo ke strukturovanému designu v prostředí CASE SDW:
  - pro hodnocení kvality designu (soudržnost, spřaženost,...) nejsou v SDW speciální podpůrné prostředky. Každý modul však smí mít právě jedno rozhraní, a tudíž musí být volán stejně ve všech kontextech,
  - maximální počet podřízených modulů (7+ - 2) se v SDW velmi přirozeně dá dodržovat: zkuste na obrazovku namalovat vedle sebe více než 9 smysluplně pojmenovaných čtverečků. Jde to těžko.

4. Závěr

- YSM je pragmatická systemizace toho, co se v minulých 15 letech v projekci osvědčilo. Co je užitečné, je správné.
- Strukturovaný design vytvořil z moudrých Myersových myšlenek o vlastnostech modulů postup návrhu programového systému.
- CASE SDW je pěkný.

5. Literatura

[1] Yourdon, E.: Modern Structured Analysis, Prentice-Hall, 1989.

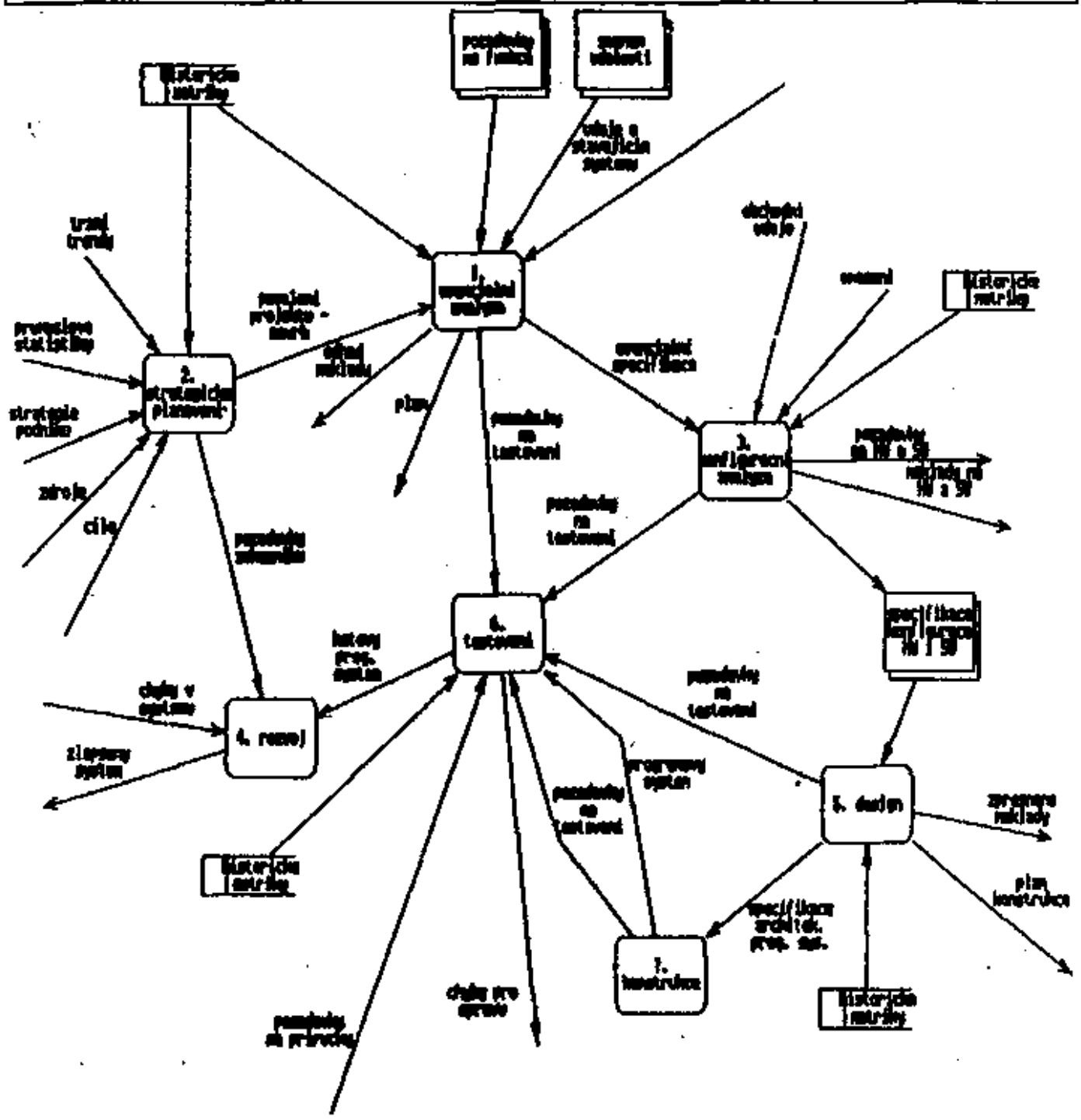
[2] Yourdon, E.: Essential Systems Analysis, Prentice-Hall, 1984.

[3] Page-Jones, M.: The Practical Guide to Structured Systems Design, Prentice-Hall, 1988.

[4] manuály CASE SDW fy Pandata

[5] Stanovská, Jilková: Případová studie SSD, interní materiál KIT VŠE, 1990.

Autor: Ing. Helena J i l k o v á, CSc., Katedra inf. technologií VŠE,
nám. W. Churchilla 4, 130 67 Praha 3, tel. 02-2125437

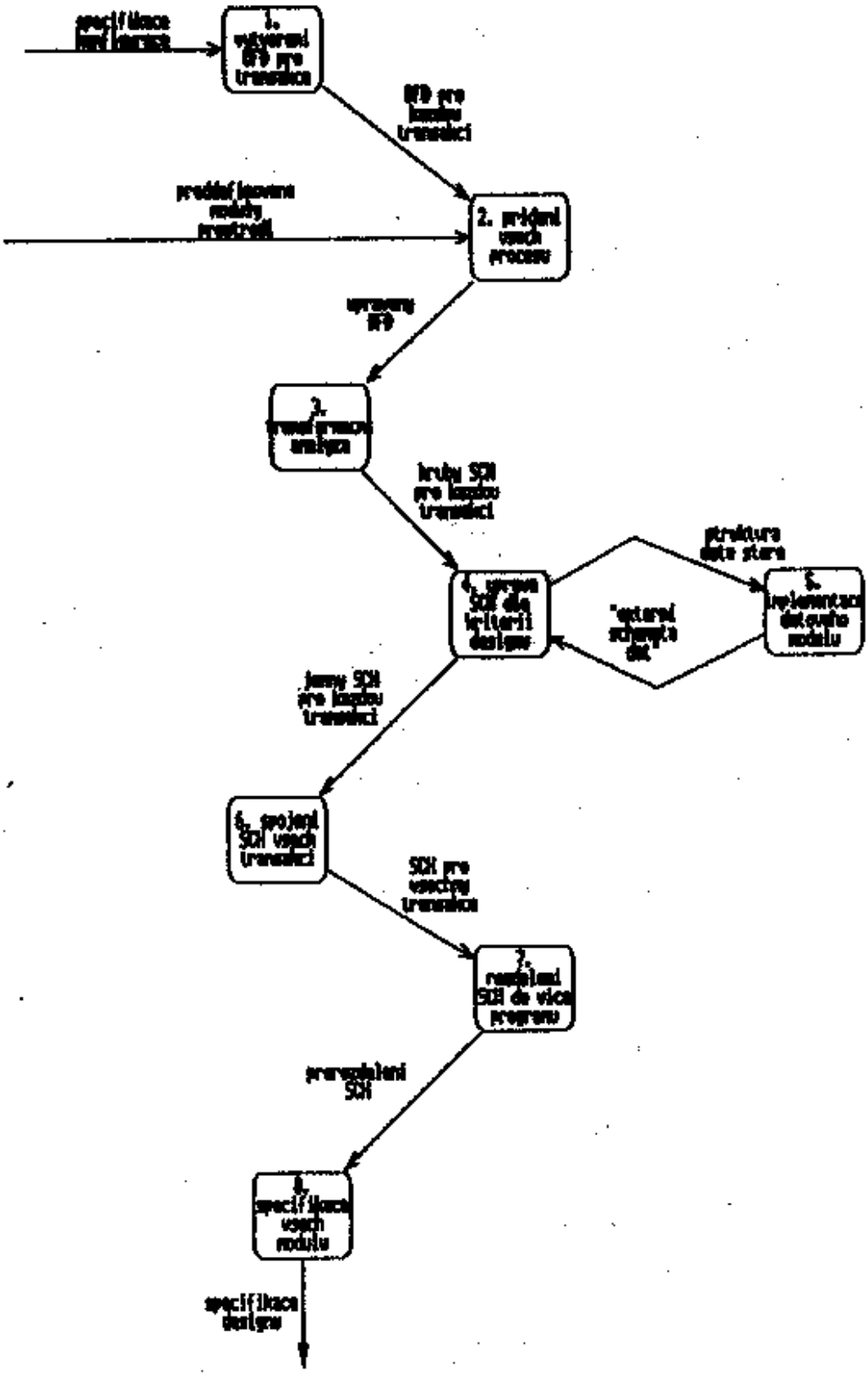


Vysvětlivky:

- činnost /transformace/
- datový tok
- datové toky významné z hlediska výkladu tzv. transakční analýzy

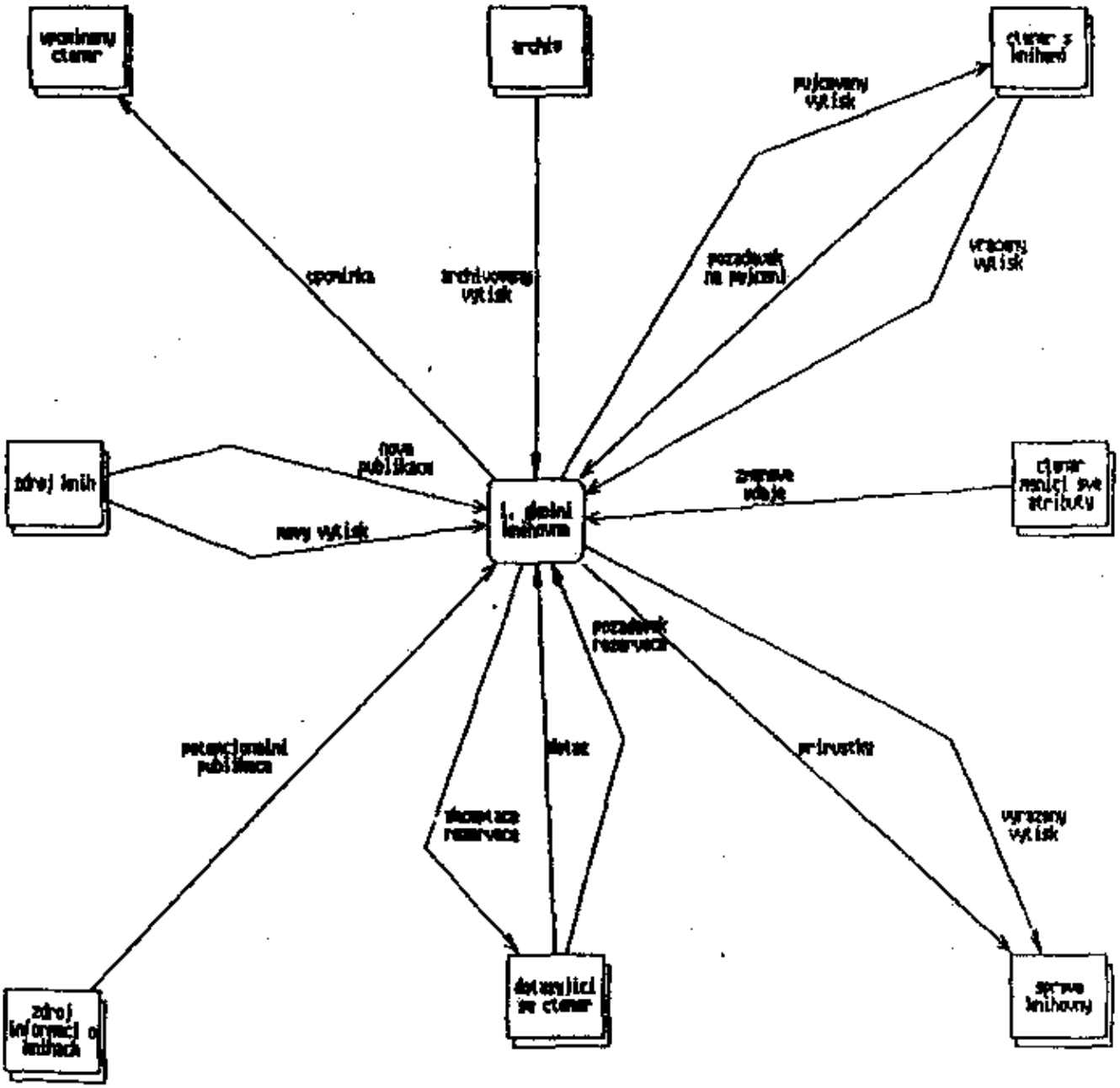
Obr. 1 YSM - činnosti a vazby projekčního postupu

System Development Workbench	Page : 1
System : OSTRAVA	Version :
Type : Data Flow Diagram	Author : Helena Jilkova
Name : SDI structure chart	Date : 27-03-1991



Obr. 2 Structured Software Design - schéma postupu

System Development Workbench		Page : 1
System : OSTROVA		Version :
Type : Data Flow Diagram		Author : Helena Jiskova
Name : DFD urovne 0		Date : 27-03-1991



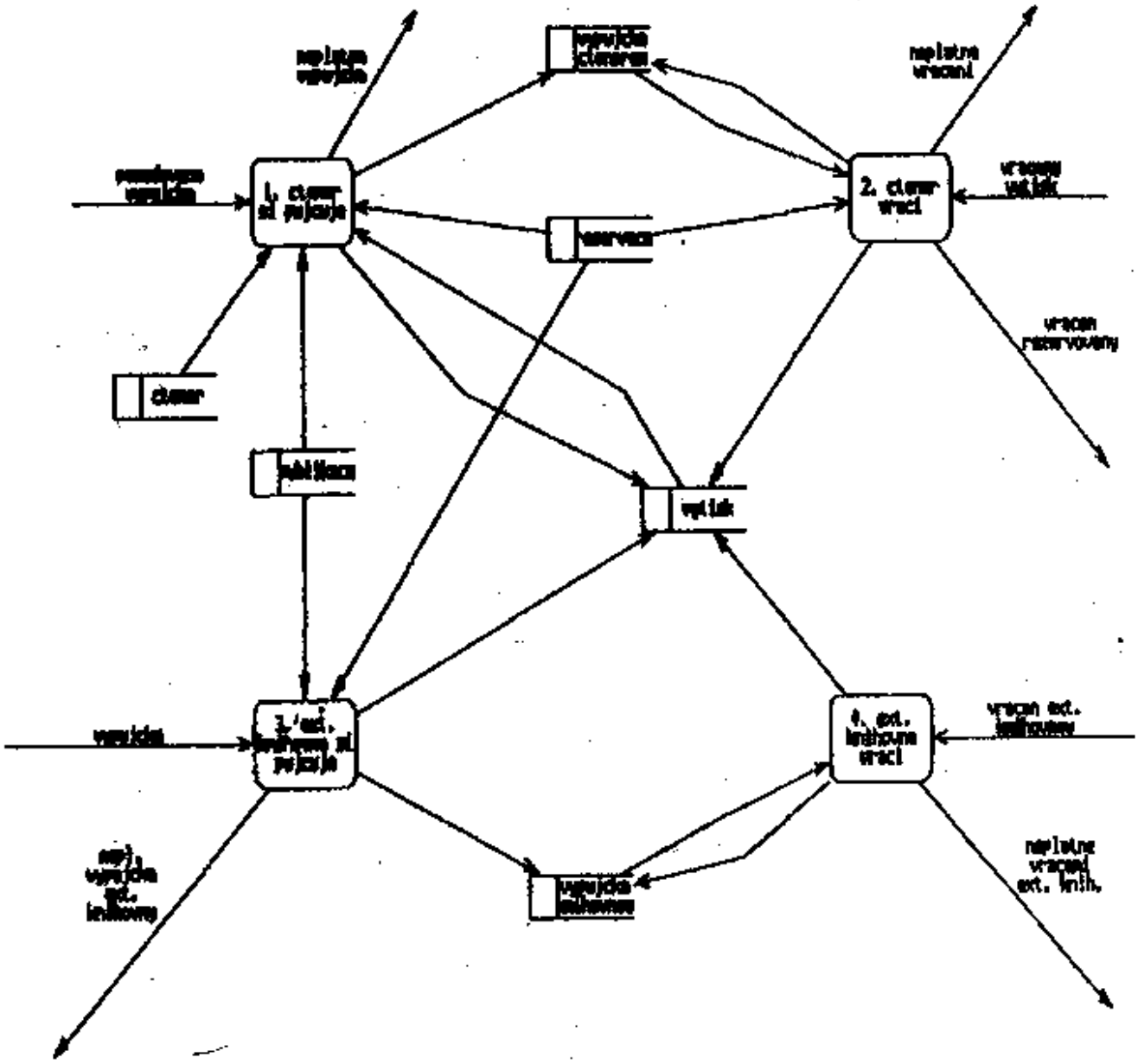
Vysvětlivky k symbolům

- terminátor /zdroj impulsů do systému Školní knihovna a/či příjemce výstupů ze systému/
- transformace
- → datový tok

Vysvětlivky k datovým tokům: každý datový tok musí být dále popsán ve slovníku dat. Pro popis se obvykle využívá zjednodušené BNF, např. Publikace=Identifikacepublikace + Autor + Název + Cena + Rokvydá

Obr. 3 DFD úrovně 0 /kontextový/ - "předprojektové fungování" systému

System Development Guidebook System : OSTRAVA Type : Data Flow Diagram Name : 1 DFD 1.urovne		Page : 1 Version : Author : Helena Jilkova Date : 27-03-1991



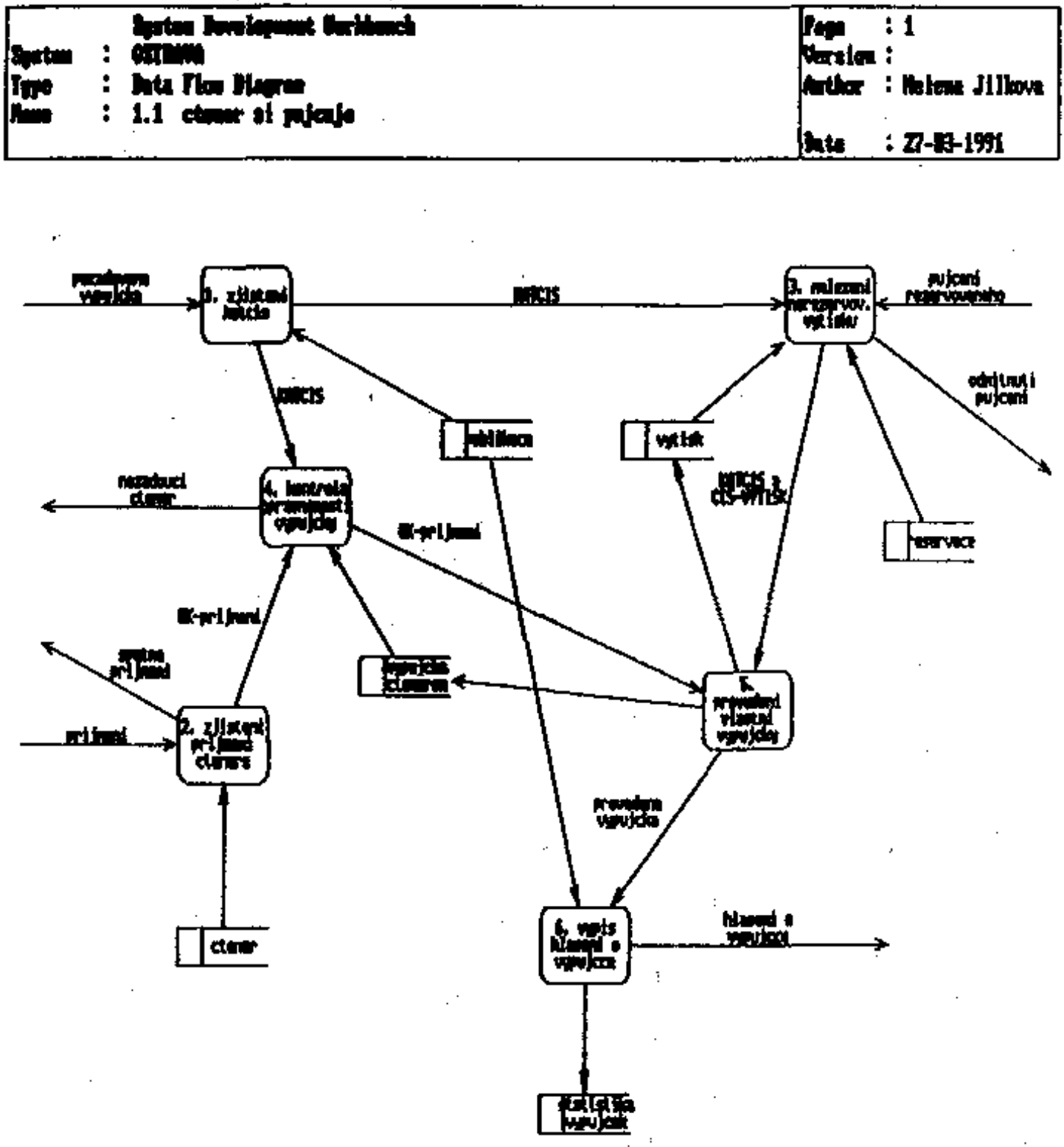
Vysvětlivky:

JMENO — tzv. data store /uložení dat, abstrahujeme od formy uložení - DB, soubor, způsob přístupu/

JMENO — čtení z data store

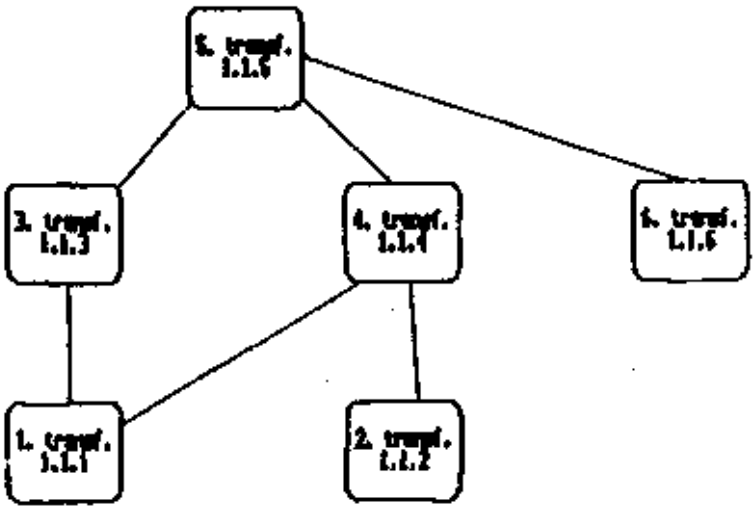
Pozn.: pokud datové toky z/do data store nejsou popsány, znamená to, že se přenáší celý "záznam" /popis data storů je součástí slovníku dat/

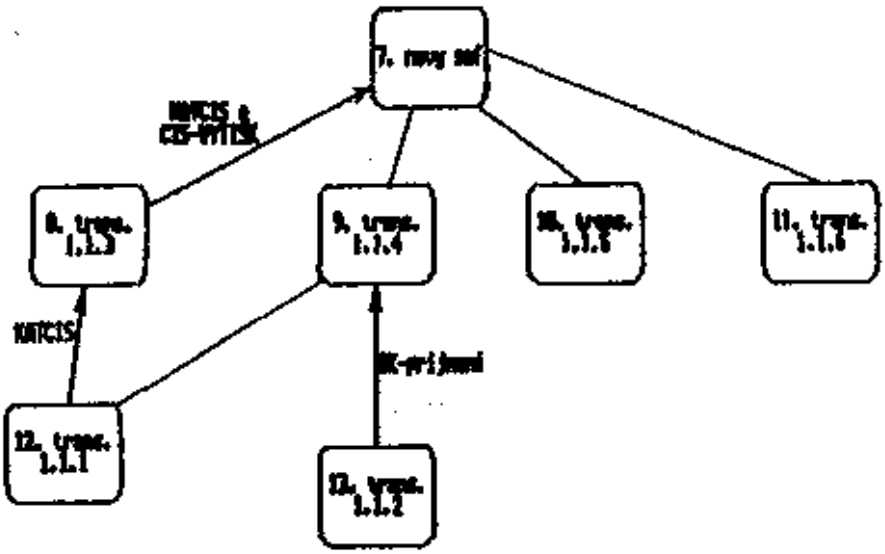
Obr. 4 DFD 1. úrovně pro transakce vypůjčování a vracení výtisků



Obr. 5 DFD 2. úrovně pro transakci 1.1 "čtenář si půjčuje"

System Development Workbench System : OSTROV Type : Data Flow Diagram Name : obr. 6		Page : 1 Version : Author : Helena Jilkova Date : 27-03-1991

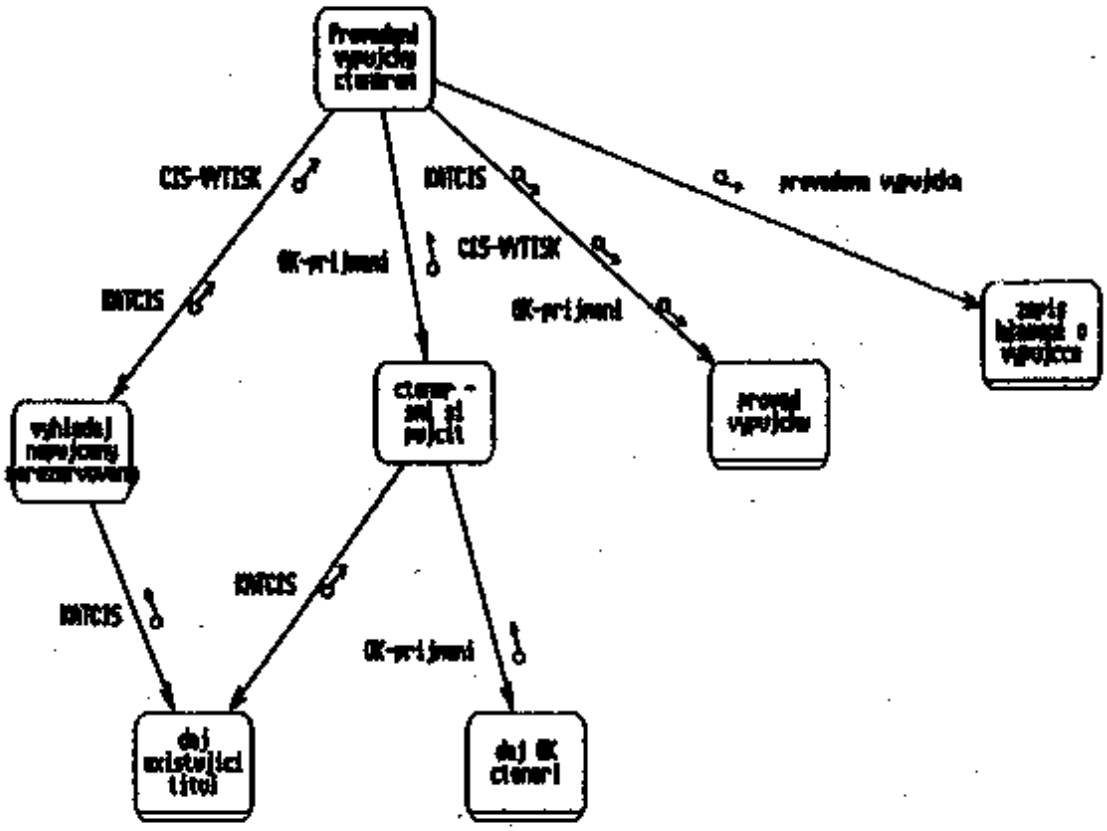




Vysvětlivky: - číselné označení v transformacích odpovídá očíslování v obr. 5
- KAICIS, KAICIS + CIS-VYTISK, OK-PRIJMENI jsou příkladem předávání parametrů /výběr centrální transformace tvoří předěl: transformace se stávají moduly progr. systému/

Obr. 6 Výběr centrální transformace v DFD

System Development Workbench System : OSTRAVA Type : Structure Chart Name : Provedení vypujčky čtenáře		Page : 1 Version : Author : Helena Jiřinová Date : 27-03-1991



Vysvětlivky: - názvy komponent vyjadřují funkci svoji a zároveň všech podřízených komponent
- SCH je odvozen s dosazením "nového šéfa"
o→ - předávaný parametr

Obr. 7 Hrubý structure chart transakce 1.1 "čtenář si půjčuje"