CASE/4/0

Jiří Prokop

1. Úvod

Systém CASE/4/0 firmy Microtool je prostředek softwarového inženýrství, t.j. množina metod a nástrojů sloužících k vývoji a údržbě softwarových systémů. Metody jsou plánovitě aplikované, zdůvodněné způsoby jednání k dosažení stanovených cílů. Nástroje (tools) jsou softwarové produkty poskytující uživateli prostředky k realizaci těchto metod. Postupem rozumíme soulrn metod a nástrojů.

Metody systému CASE/4/0: analýza systému, návrh systému.

Analyza systému je metoda sloužící k rozboru současného stavu podnikové oblasti nebo stávajícího softwaru a ke specifikaci nového uživatelského systému zpracování dat. Výsledkem systémové analýzy je úplný popis všech manuálních a strojových funkcí, popis průběhu funkcí, zobrazení všech informačních toků a neredundantní struktura informací.

Návrh systému je metoda, která transformuje řešení vytvořené systémovou analýzou na návrh zpracování dat skládající se z modulů, databanky a dat. Slouží k detailní specifikaci algoritmů a k implementaci navržených modulů a databanky. Výsledkem je grafický návrh modulů a algoritmů, neredundantní struktura dat, zdrojový kód v cílovém jazyce (zatím to může být pouze jazyk C) a definice souborů.

Nástroje (tools) systému:

- system analysis tools
- system design tools
- system publishing tools
- system configuration tools

Výsledky systémové analyzy, návrhu systému i dokumentace systému spravuje CASE v jedné jednotce, označené jako systém a uložené vždy v jednom adresáři. Pod položkou Dictionary v hlavním menu je nabídka Select system, která umožňuje výběr systému, s nímž chceme právě pracovat. Potřebujeme-li části některého systému použít v jiném systému, použijeme nabídek Export (zvolené diagramy přesuneme do nového systému) a Import (diagramy jednoho systému přeneseme do jiného). Funkce Import je náročnější, protože CASE hlídá, aby při jejím provedení nedošlo k narušení pravidel konzistence. Funkce Backup a Restore dovolují zálohování a obnovení systému.

2. Analyza systému

Analyza systému používá následujících 5 metodických kroků:

- zjemňování funkcí pomocí modelu funkční struktury. Je to diagram typu strom s notací podle Jacksona. Představuje hierarchickou dekompozici aplikačního systému do funkcí. Funkce je v rámci systému jednoznačná.
- popis informačních vztahů funkcí síťovým diagramem podle de Marca.
- zobrazení funkčních toků. Prostředkem zobrazení je zde abstrakce Petriho sítí. Sítě typu funkce – událost dovolují modelování dynamických vlastností systému.
- zjemňování uložených dat a informací z informačních toků pomocí hierarchických datových struktur. Opět se používají stromy podle Jacksona.
- ERA model podle Codda.

Diagramy jsou vzájemně svázány pravidly konzistence, která systém v průběhu práce ověřuje.

Popsaných pět metodických kroků neprobíhá seriově. Představují spíše mnohastupňový iterativní proces k postupnému upřesňování výsledků vývoje. Tento způsob práce je aktivně podporován technikou oken CASE.

Nemusí nutně proběhnout všech pět kroků.

Výsledky systémové analýzy i návrhu spravuje CASE v jedné jednotce, označované jako systém a uložené vždy v jednom adresáři.

2.1. Funkční struktury

Funkční struktura je hierarchický rozklad uživatelského systému do funkcí a podmínek, řídících sekvenci provádění funkcí.

Funkce je množina logicky vzájemně svázaných činností, které způsobují změnu obsahu nebo struktury informací. Aby byla zaručena jednoznačnost funkcí, může funkce existovat jen v jedné funkční struktuře systému a jen na jednom místě v této struktuře. Funkce stejné hierarchické úrovně jsou ve funkční struktuře řazeny zleva doprava v závislosti na pořadí jejich provádění, je-li úroveň abstrakce už tak vysoká, že pořadí je možno definovat. Provádění funkcí není vždy přísně sekvenční, nýbrž často v závislosti na výskytu nějaké podmínky. Funkce i podmínky jsou ve funkční struktuře zobrazeny jako obdélníky, podmínka je zobrazena nad funkcí na ní závisející, druh podmínky je vyznačen nad pravým horním rohem jako symbol:

* Iterace (rejecting loop)

Opakování podřízených funkcí se provádí tak dlouho, dokud platí podmínka

+ Repeat (non–rejecting loop)

Opakování tak dlouho, dokud nenastane podmínka pro ukončení

Loop or controlled loop

Počet opakování specifikován podmínkou.

○ Exklusivní alternativa

Podřízené funkce představují vzájemně se vylučující alternativy.

■ Inklusivní alternativa

Může být provedena jedna z funkcí, více nebo všechny.

! IF nebo výjimka

Funkce se provedou jen při výskytu podmínky. Je to zvláštní případ exklusivní alternativy.

|| Paralelismus funkcí

Dvě nebo více funkcí probíhá paralelně.

• Break

Předčasné přerušení sekvence funkcí

Omezuje-li nás při vytváření funkční struktury velikost obrazovky, můžeme umístit podstromy na dalších stránkách. V diagramu je toto pokračování vyznačeno číslem stránky.

Jinou možností je určitou podstrukturu zobrazit jako samostatnou funkční strukturu. K tomu je v nabídce menu prvek PART.

K vytváření funkčních struktur:

Pod společnou rodičovskou funkcí by měly být jen takové funkce, které mají vysokou funkční vazbu, tedy popisují těsně související činnosti. Při zjemňování funkcí ve funkční struktuře jsou v zásadě dvě možnosti:

- objektově orientované zjemňování: od jedné úrovně k další je zjemňován zpracovávaný objekt. Činnost zůstává nezměněna.
- činnostně orientované: objekt zůstává, činnost se rozkládá do dílčích činností.

Pravidla konzistence:

- při přejmenování funkční struktury, na kterou se odkazuje pomocí PART, přejmenuje se automaticky i odkaz.
- funkční struktura může být vymazána jen tehdy, není-li na ni odkaz.
- odkaz PART nemůže být přejmenován.

2.2. Informační tok

Diagram informačního toku popisuje cesty informací mezi funkcemi, pamětmi (storages), interními a externími rozhraními a jejich transformace funkcemi.

Paměť (storage) je prostředek k pasivnímu uložení informace. V této roli mohou být také entity, pokud při vytváření informačního toku už jsou k dispozici ER modely.

Interní rozhraní je funkce z funkční struktury, která nemá stejnou rodičovskou funkci jako funkce informačního toku, ale je cílem nebo zdrojem informace v informačním toku vytvořené.

Externí rozhraní je funkce, organizační jednotka nebo osoba, která je zdrojem nebo cílem informace vytvořené v informačním toku.

Funkce se může vyskytovat jen v jednom informačním toku jako funkce (v diagramech dalších toků jako interní rozhraní). Storage nebo entita se může vyskytovat v několika informačních tocích.

Prvky diagramu mohou být spojeny třemi typy toku:

- jednoduchý – představuje přenos informace, která není v informačním zdroji měněna ani rušena,
- směrovaný (directed) představuje přenos informace, která může být změněna nebo zrušena v informačním zdroji,
- dvojsměrný (bidirected) – přenos informace s následným návratem zpět v pozměněné podobě.

Alespoň jeden ze spojovaných prvků musí být funkce. Není možný tok informace mezi pasivními prvky (storage, entita, interní nebo externí rozhraní). Spojení mezi funkcemi představuje synchronní přenos dat – informace se musí zpracovat v okamžiku svého vzniku.

Vztah mezi funkční strukturou a informačními toky: Informační tok sestává pouze z těch funkcí funkční struktury, které jsou na stejné hierarchické úrovni a mají společného rodiče. Informační tok je pojmenován po funkci – společném rodiči. Pomocí nabídky SIGHT v menu můžeme snadno vytvořit informační tok pro zvolenou funkci: informační tok bude automaticky obsahovat všechny „funkce–děti" označené funkce. Není efektivní ani nutné vytvářet informační toky pro všechny úrovně , smysluplné je to pro ty úrovně, které byly podrobeny objektově orientovanému zjemnění.

Diagramy informačních toků umožňují ověřit správnost navržení funkčních struktur. Komunikuje-li na příklad funkce s velkým počtem interních rozhraní, je to příznakem toho, že internímu rozhraní odpovídající funkce měla být zařazena v jiné větvi stromu. Příznakem vysoké datové vazby je t.zv. „sluneční efekt", t.zn. více než 7 šipek, začínajících nebo končících u jedné funkce, křižující se šipky a více než tři informace na jednu šipku.

Pravidla konzistence:

Je-li přidána funkce do funkční struktury a vytvořeno její spojení s rodičovskou funkcí, doplní se automaticky do odpovídajícího informačního toku. Je-li nová funkce vytvořena v informačním toku, je automaticky doplněna do funkční struktury a připojena k rodičovské funkci. Přejmenujeme-li nebo vymažeme funkci ve funkční struktuře (inf. toku), automaticky se přejmenuje nebo vymaže také v informačním toku (funkční struktuře). Přerušíme-li ve funkční struktuře spojení funkce k funkci rodičovské, vymaže se funkce v informačním toku. Smažeme-li funkci v informačním toku, je ve funkční struktuře přerušeno spojení k rodičovské funkci, funkce však smazána není. Funkci lze ve funkční struktuře nebo inf. toku smazat jen tehdy, není-li použita jako interní rozhraní v inf. toku. Interní rozhraní nelze přejmenovat. Přejmenování funkce změní automaticky jméno odpovídajícího interního rozhraní. Vkládání, výmaz nebo oprava informace, externího rozhraní, storage a entit jsou lokálními aktivitami, které jiné typy diagramů neovlivňují.

2.3. Řídící toky

Diagram řídícího toku je síťový graf, jehož uzly představují funkce a jehož orientované hrany jsou označeny jmény událostí.

Prvky diagramu jsou:

Funkce, která je stejně jako v informačním toku znázorněna kroužkem. Diagramy řídících toků jsou podobně jako informační toky vytvářeny pro ty funkce, které mají společného rodiče ve funkční struktuře.

Externí rozhraní je osoba, SW systém nebo zařízení, které nepatří ke zkoumanému systému, ale generuje událost, která je podmínkou pro funkci diagramu řídícího toku, nebo je iniciována událostí řídícího toku.

Link je funkce, viditelná v jiném diagramu (obdoba interního rozhraní inf. toku). Funkce buď generuje událost nebo je iniciována událostí v diagramu řídícího toku.

Pravidla konzistence jsou analogií pravidel pro informační toky.

2.4. Datové struktury

Struktura dat je hierarchickou dekompozicí informace. Diagramem je strom s notací podle Jacksona podobně jako u funkčních struktur. Prvků je ve srovnání s funkční strukturou méně:

\# Loop or controlled loop

Počet opakování je specifikován.

○ Exklusivní alternativa

Podřízená datová pole se vyskytují alternativně.

■ Inklusivní alternativa

Výskyt je možný i současně.

! IF nebo výjimka

Datové pole se vyskytuje při splnění podmínky. Je to zvláštní případ exklusivní alternativy.

Pravidla konzistence: Datová struktura se smí vymazat, jen když informace nebo storage se stejným jménem v žádném informačním toku neexistuje. Přejmenování datové struktury vede k přejmenováním stejnojmenných informací ve všech inf.tocích. Výmaz v inf.toku nemá za následek výmaz datové struktury. Přejmenování informace nebo storage vede k vytvoření reference na jinou nebo novou strukturu dat.

2.5. ERA modely

ERA modely zobrazují ve formě síťového diagramu množiny entit relevantní pro pozorovaný výsek reality a vztahy mezi entitami.

ERA model je dekomponován z funkčního hlediska do oblastí (areas) tak, aby v jedné oblasti byly všechny entity a vztahy nezbytné pro řešení operačního tasku. Operační tasky jsou odvozeny z funkční struktury.

Prvky diagramu jsou entity a vztahy mezi nimi. Entity jsou zobrazeny obdélníkem, vztahy jednoduchými nebo dvojitými (pro možnost zobrazit vztah 1:1, 1:M nebo M:N), otevřenými nebo uzavřenými šipkami (uzavřená znamená povinný výskyt).

Volitelně je možno doplnit diagram o atributy, klíče a semantický význam vztahu.

Pravidla konzistence: Entity a jejich vztahy se mohou vyskytovat ve více oblastech modelu. Při přejmenování entity nebo vztahu v jedné oblasti přejmenuje se automaticky v ostatních oblastech. Výmaz entity je lokální akcí a týká se jen jedné oblasti. Mezi identickými entitami mohou být v různých oblastech znázorněny různé vztahy.Množina atributů entity se může rozšiřovat z pohledu různých oblastí. Změny v ERA modelech se promítají automaticky do diagramů informačních toků.

3. Návrh systému

Návrh systému navazuje na analýzu systému. Představuje dvoustupňový proces:

- na základě specifikací, které jsou výsledkem systémové analýzy, následuje členění systému do implementovatelných jednotek (modulů), graficky reprezentovaných strukturami modulů.
- výsledkem dalšího stupně je grafický návrh algoritmu funkčních definic struktury modulu. Algoritmy se popisují ve formě implementačního stromu,

při čemž se používá metod strukturovaného programování, takže implementační stromy vykazují značnou podobnost s funkčními strukturami.

3.1. Struktura modulu

Prvkem struktury modulu mohou být funkce a datové struktury, definované v této struktuře modulu nebo převzaté z výsledků analýzy systému, menu nebo dialog realizující funkce, vytvořené pomocí Dialog designeru, a konečně odkazy na funkce a data jiných modulů.

K převzetí funkcí z funkčních struktur slouží nabídka Function structure pod nabídkou Element menu. Pak systém nabídne seznam funkčních struktur, vytvořených v rámci systémové analýzy, kde si vybereme diagram a v něm funkci, která musí být listem stromové struktury. Funkce dostane ve funkční struktuře označení „DESGN". Jiný postup dovoluje ve funkční struktuře zvolit nabídku Design, označit listovou funkci a přejít tak od analyzy systému k návrhu systému. Ve struktuře modulu je funkce znázorněna obdélníkem s označením „FUNC", stejně jako funkce, která byla ve struktuře modulu vytvořena pomocí nabídky FuncDef a nemá odkaz na diagramy funkčních struktur.

K převzetí datových struktur z výsledků systémové analýzy slouží nabídky Data structure a Entity type. Ke všem plně definovaným („plně" znamená, že všem datovým polím resp. atributům musí být přiřazen datový prvek se specifikovaným typem) datovým strukturám a entitám generuje Case v souladu s cílovým jazykem a gramatikou odpovídající deklaraci. Pro data bez vztahu k výsledkům systémové analýzy lze použít nabídky Typedef a Variable.

Pro menu nebo dialog se použije nabídek Dialog structure a Menu structure. V těchto případech se zároveň vytvářejí stejnojmenné funkce. Jméno bude použito jako identifikátor a musí proto splňovat příslušná pravidla.

Prvky struktury modulu, které se mají použít v jiném modulu, musí být označeny jako „public" (nabídka Public je v rámci menu Module). Nabídku Usage (pod Element) lze použít k převzetí funkcí a dat z jiných modulů. Pomocí Use Func nebo Use Data pod Module určíme, které funkce resp. data potřebujeme.

3.2. Dialog designer

Systém Case respektuje IBM standard CUA (Common User Access) konceptu SAA (System Application Architecture), který definuje uživatelské rozhraní, jednak tím, že sám komunikuje s uživatelem pomocí tohoto rozhraní, jednak tím, že pomocí do System designu integrované komponenty Dialog designer umožňuje uživateli, aby jím vytvořené produkty standard CUA respektovaly.

Po volbě nabídky Dialog designer (v submenu Sight) je nutno vybrat buď prvek MENU nebo DLG. V závislosti na tom bude vypadat další hlavní menu, z jehož nabídek nejdůležitější jsou tyto:

- Action Bar a Items (menu) určuje vzhled a položky menu
- Controls (dlg) obsahuje nabídku prvků komunikace podle CUA
- Callback spojuje prvky menu a dialogu s funkcemi modulu
- Animate umožňuje prohlédnout si výsledek návrhu na obrazovce

3.3. Implementace

Funkce ve struktuře modulu budou dále zjemňovány vytvořením implementačního stromu. Po zvolení nabídky Implementation v Sight se objeví nové menu, připomínající menu při vytváření funkčních struktur. Prvky v nabídce Element jsou rozšířeny o case, input, output, wait, call a empty. Nabídka Put/Get umožňuje kopii podstromu do implementačního stromu jiné funkční definice. Nabídka Listing zobrazí kód vygenerovaný na základě implementačního stromu. Po specifikaci algoritmu zbývá přiřadit jednotlivým listům stromu zdrojový kód v cílovém jazyce. K tomu je v submenu Text nabídka Code. Po jejím zvolení vybereme prvek stromu a zapíšeme zdrojový kód. K zadání zdrojového kódu máme ještě dvě další možnosti, jednak nabídku Listing, jednak při zpracování struktury modulu s využitím nabídky Code submenu Implement. Tímto způsobem přiřazujeme funkci zdrojový kód, aniž použijeme grafické reprezentace algoritmu implementačním stromem.

Po přiřazení zdrojového kódu všem prvkům lze volbou nabídky Make Module vygenerovat soubor se zdrojovým textem celého modulu. Cílovým jazykem může být zatím pouze jazyk C, standardně v syntaxi kompilátoru Microsoft C 6.0.

4. Dokumentace systému

Prvkům všech diagramů, o nichž jsme dosud hovořili, mohl být přiřazen vysvětlující komentář. Součástí Case systému je tedy textový editor. Používá se nejen pro komentáře, ale také pro opravy a vkládání zdrojového kódu v implementační fázi návrhu systému.

K automatizovanému vytváření dokumentace je v hlavním menu nabídka Publish. Umožňuje pomocí grafu stromového typu určit strukturu dokumentace, její členění do kapitol, odstavců, atd., při čemž každému uzlu grafu lze přiřadit text. Do tohoto textu mohou být vkládány kterékoli diagramy, vzniklé během systémové analýzy a systémového návrhu. Při požadavku vložení se vždy objeví dialog box, v němž můžeme určit, kterými dalšími informacemi má být vytištěný diagram doplněn. Např. se mohou vytisknout textové komentáře k jednotlivým prvkům diagramu, ERA model může být doplněn o seznam entit a vztahů a podobně. Do textu je vsunut jen požadavek vložení diagramu. K vlastnímu vkládání dojde až při tisku, a tedy je možno kdykoli měnit kterýkoli diagram a změna se promítne do dokumentace automaticky.

Nabídka Print Report umožňuje tisk celé dokumentace podle definovaného formátu. Jednotlivé kapitoly a odstavce jsou automaticky číslovány arabskými číslicemi, automa-

ticky se vytváří obsah a seznam obrázků. Nabídka Format dovoluje upravit formální vzhled dokumentace podle potřeb uživatele. Lze potlačit číslování stránek, obsah, seznam vyobrazení. Nadefinovaný formát lze uložit pro další použití.

Nabídka Reporting (v roletě pod Dictionary hlavního menu) Umožňuje jednak sestavit přehledné globální informace k jednotlivým typům diagramů, kontextově orientované výpisy, jednak otestovat vytvořené diagramy z různých hledisek. Submenu Diagram poskytuje na příklad pro funkční struktury (nabídka Function) tyto přehledy: seznam všech funkčních struktur, seznam všech funkcí, seznam funkcí včetně odkazů na jejich použití v jiných diagramech, test na úplnost funkčního zjemnění, test na chybějící spojení. Nabídka Check consistency otestuje konzistenci celého systému.

5. Zkušenosti s používáním systému Case

5.1. A.s. PragoData

Zkušenosti s používáním systému (libovolného) jsou vždy závislé nejen na vlastostech tohoto systému, ale i na vlastostech jeho uživatele. Je proto nutno na tomto místě blíže představit a.s. PragoData. PragoData je hardwarově nezávislý softwarehouse, specializovaný na dodávky softwarového vybavení a poradenské služby pro průmyslové i jiné organizace. V ČSFR je distributorem multiuživatelského relačního databázového systému Progress americké firmy Progress software corporation a systému Case/4/0 německé firmy Microtool GmbH. Vyvíjí a prodává aplikační systém ProFiS, napsaný v jazyce 4GL systému Progress. Zahrnuje moduly Technická příprava výroby, Plánování finální výroby, Operativní plánování výroby, Plánování kapacit, Dílenské řízení výroby, Skladová evidence zásob, Řízení nákupu, Prodej, Účetnictví, Personalistika, Mzdy, ZP, DKP, Správa systému. Vývoj systému ProFiS je podporován souborem vlastních utilit firmy PragoData (PDPacket), umožňujících jednotné a optimální používání vybraných programových funkcí.

Dokumentace systému ProFiS je vytvořena pomocí Case/4/0, ovšem dodatečně, protože ProFiS existoval už před zakoupením Case. Dokumentace používá pouze diagramů funkčních struktur, informačních toků a ERA modelu. Pro zákazníky, jimž PragoData zajišťuje komplexní projektovou přípravu automatizovaného systému řízení, je Case využíván už ve fázi úvodního projektu. Výsledky systémové analyzy, předané útvaru programování, obsahují funkční struktury, diagramy informačních toků a ERA modely. Ve všech diagramech jsou prvky diagramu doplněny komentářem.

Úpravy aplikačních programů podle specifických podmínek a požadavků uživatele však útvar programování provádí buď přímo v jazyce 4GL nebo, v náročnějších případech, použije PDBridge (bude popsán dále). System Design tak využit není, protože zatím lze generovat zdrojový kód jen v jazyce C. Do budoucna se počítá s rozšířením jazyků a mezi nimi bude i 4GL systému Progress.

V případě informačního systému pro půjčovnu CD disků byl CASE/4/0 použit ve všech fázích vývoje projektu. Z dokumentace tohoto programového systému byly vybrány ukázky pro obrazovou přílohu tohoto příspěvku.

5.2. PDBridge

PDBridge je nástrojem pro integraci Progressu a CASE/4/0. Vytvořila ho a.s. PragoData. Přenos informace z CASE/4/0 do Progressu zahrnuje následující kroky:

1) Navržení entit, atributů a vztahů v CASE/4/0

2) Kompletace definic, vložení metajazykových povelů do entity–textů, semantik atributů, domén atributů a semantiky relačních vztahů.

3) Dump této informace z CASE/4/0

4) Testování syntaktických pravidel.

5) Vytvoření povelů Progress DDL, jejich uložení do Data Dictionary

Jednotlivé kroky probíhají v různých prostředích:

1–2 CASE, 3 MS–DOS, 4–5 Progress

Program BSHELL.EXE je speciální program pro komunikaci s uživatelem. Dovoluje běh CASE/4/0, PDB–DUMP.EXE, Progress, PDB a 3 jiných uživatelských programů. Během celé bridge session lze pracovat v prostředí BSHELL.EXE.

5.3. Pokus o hodnocení

Hodnotící soudy je dnes možno vyslovovat jen opatrně, protože Case systémy jsou teprve na začátku intenzivního dynamického vývoje. Dalším důvodem je zatím omezená zkušenost: CASE/4/0 vlastníme sice téměř dva roky, ale srovnáme-li tuto dobu s celkovou dobou trvání všech fází složitého projektu (včetně pozdější údržby), je tato doba spíše krátká.

Domnívám se, že každému Case systému dal vzniknout jeden ze dvou možných počátečních impulsů:

1) Na základě studia metodiky projektování a programování vznikne myšlenka vytvořit vhodnou softwarovou podporu. Jde o přístup spíše teoretický, nezávislý proto na používaném programovacím prostředí a jazyku. Proto je množina podporovaných jazyků postupně rozšiřována.

2) Přístup praktický, kdy při programování v určitém prostředí a jazyku zjistíme, že určité činnosti s projektováním a programováním spojené by mohly být automatizovány. Výsledek pak bude pravděpodobně natrvalo spojen s tím prostředím, pro něž byl „ušit".

CASE/4/0 patří do první skupiny. Jeho největšími klady jsou proto zejména metodická hloubka, provázanost používaných metod, silná integrace uvnitř produktu (analýza –

implementace – dokumentace). Jeho používání ve stadiu analyzy by bylo přínosem i tehdy, kdyby výsledný programový produkt nebyl ani v jazyce C ani ve 4GL Progressu. Zmínku si jistě zaslouží i přínos pedagogický: všechny metody si lze snáze osvojit, máme-li prostředek k jejich snadnému využívání. Práci usnadňuje podrobný, kontextově orientovaný Help.

Slabinou systému je jeho textový editor, který neznal češtinu (tento nedostatek je právě nyní už odstraněn) a postrádá některé funkce, např. zarovnávání vpravo. Při našich jednáních s výrobcem přislíbila firma MicroTool, že v budoucnu bude CASE/4/0 možno provozovat pod MS–Windows a bude lze využívat služeb textového editoru MS–WORD. Kromě toho bude do systému integrován i MS–Project, který je v naší firmě už dnes používán pro plánování projekčních a programátorských prací.

Některé výtky, které bývají systému CASE/4/0 adresovány, jsou problematické: je mu např. vytýkáno, že trvání některých operací je dlouhé (funkce zahrnuté v nabídce Reporting, Export, Import). Je však třeba uvážit, kolik úsilí a času by bylo třeba vynaložit, kdybychom měli tyto informace získat ručně (pokud bychom si na to vůbec troufli). Ještě problematičtější jsou výtky, týkající se toho, že hlídáním konzistence v průběhu veškeré práce systém komplikuje uživateli provedení některých operací. Při bližším zkoumání se obvykle ukáže, že odstranění těchto „nedostatků" by vedlo naopak ke zhoršení práce.

Některé stesky byly také způsobeny využíváním nedostatečného hardwaru. Pro úspěšnou práci je vhodné mít procesor 386 (286 stačí taky, ale pak je provedení některých funkcí opravdu pomalé), kapacitu hard disku nejméně 80 Mbyte, (dokumentace systému ProFiS sama zabírá cca 12 MB) a nutností je myš.

Při rostoucích nárocích na rychlost a kvalitu projekčních a programátorských prací a konkurenčním tlaku je obtížné si představit, že výhod, které používání systému CASE přináší, bychom se někdy mohli vzdát.

Literatura:

[1] Gregor J., Chvalovský V.:Projektování datové základny, SNTL, Praha 1988

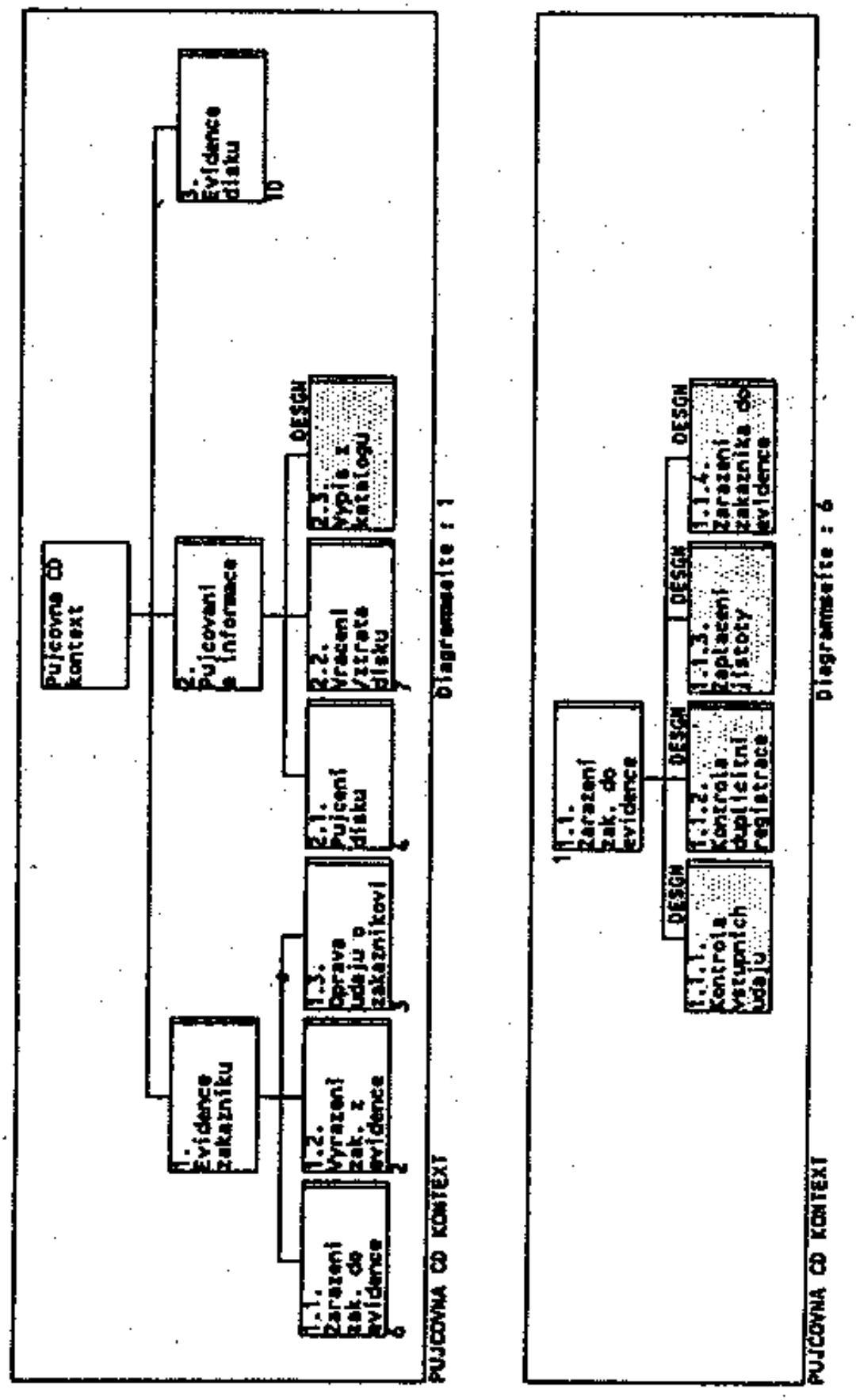
[2] Gregor J., Chlapek D., Chudějová E., Chvalovský V.: Datová základna, skripta VŠE, Praha 1991

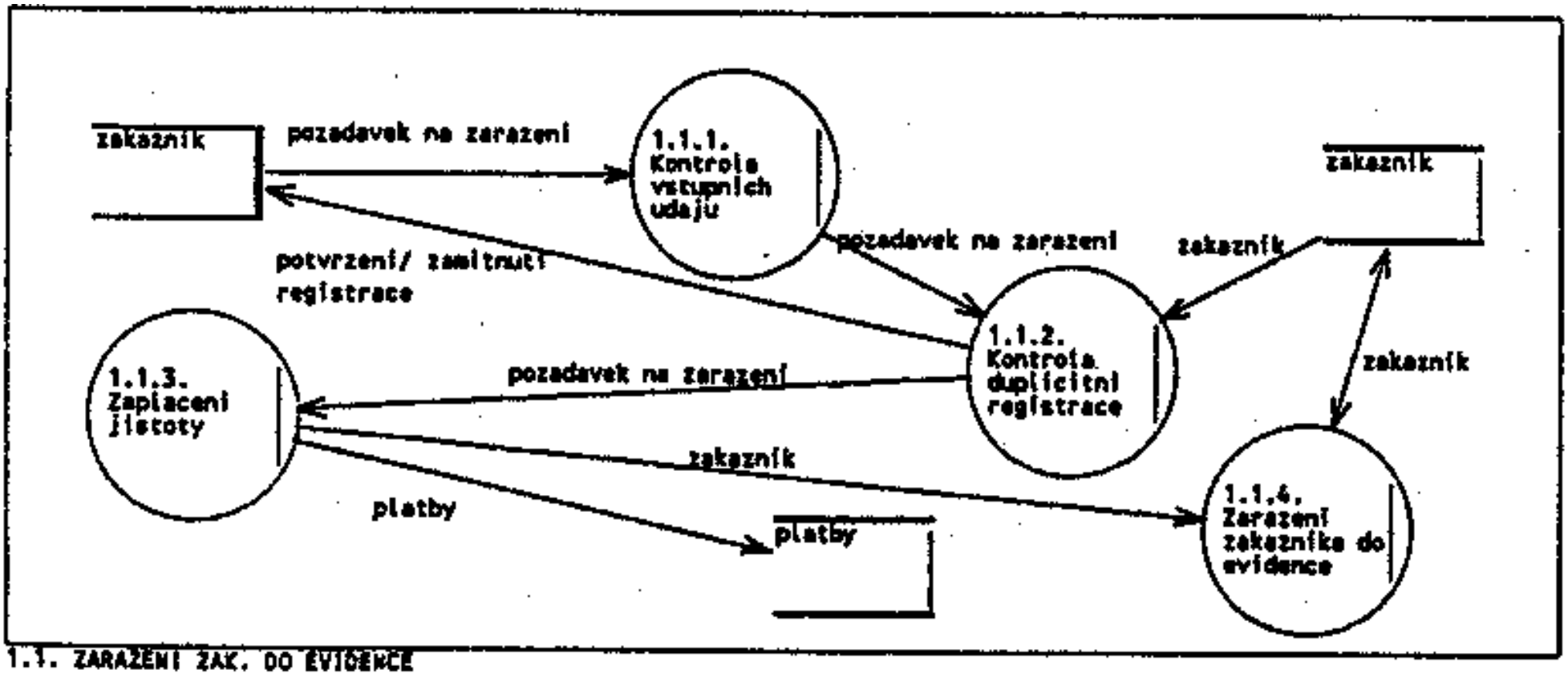
[3] Schmid P.: SAA, Die IBM System Anwendungs Architektur, IWT München 1990

[4] CASE/4/0, MethodenHandbuch, microTOOL GmbH Berlín 1991

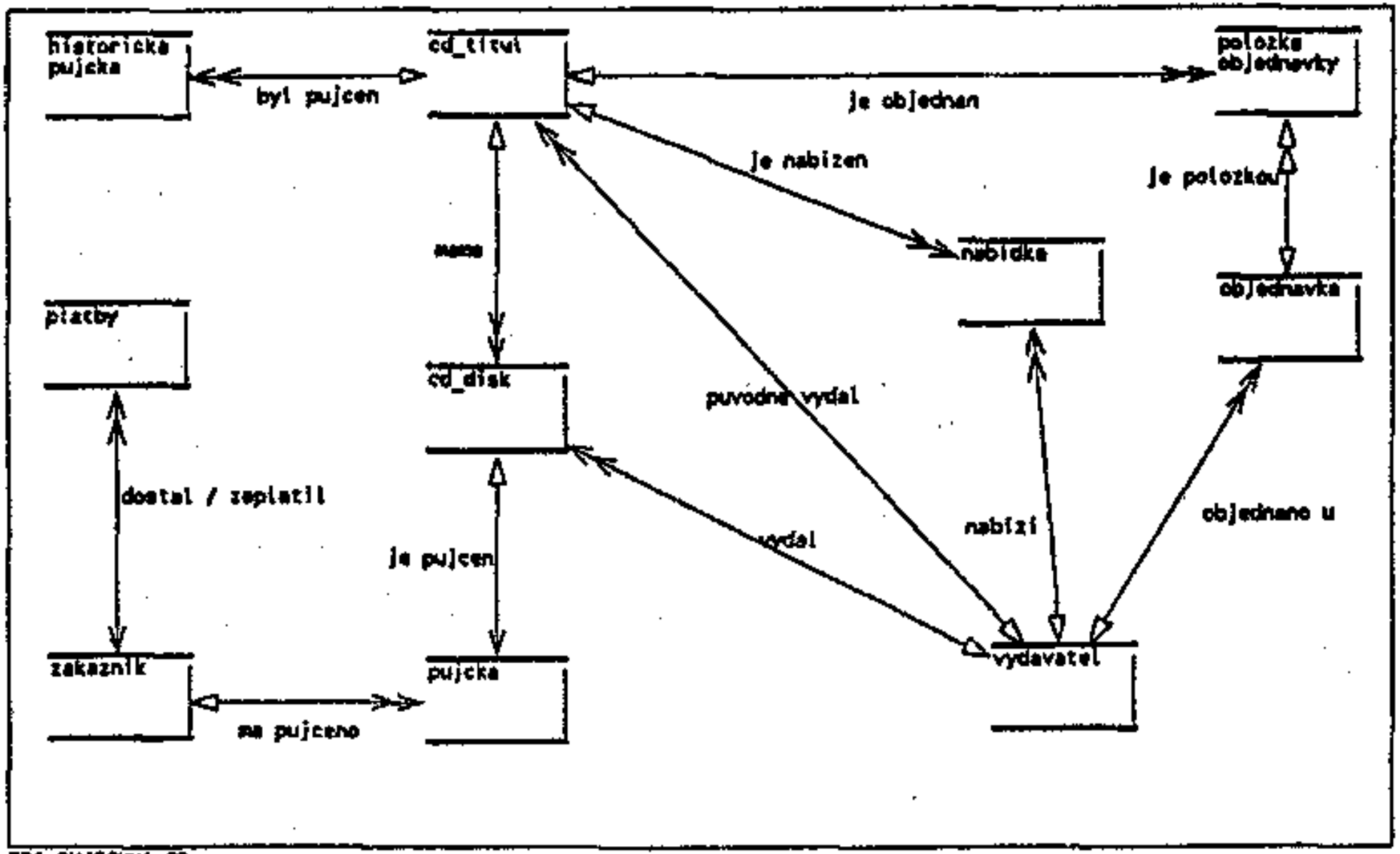
[5] CASE/4/0, BedienerHandbuch, microTOOL GmbH Berlín 1991

Ukazka FSD, IFD a ER diagramu



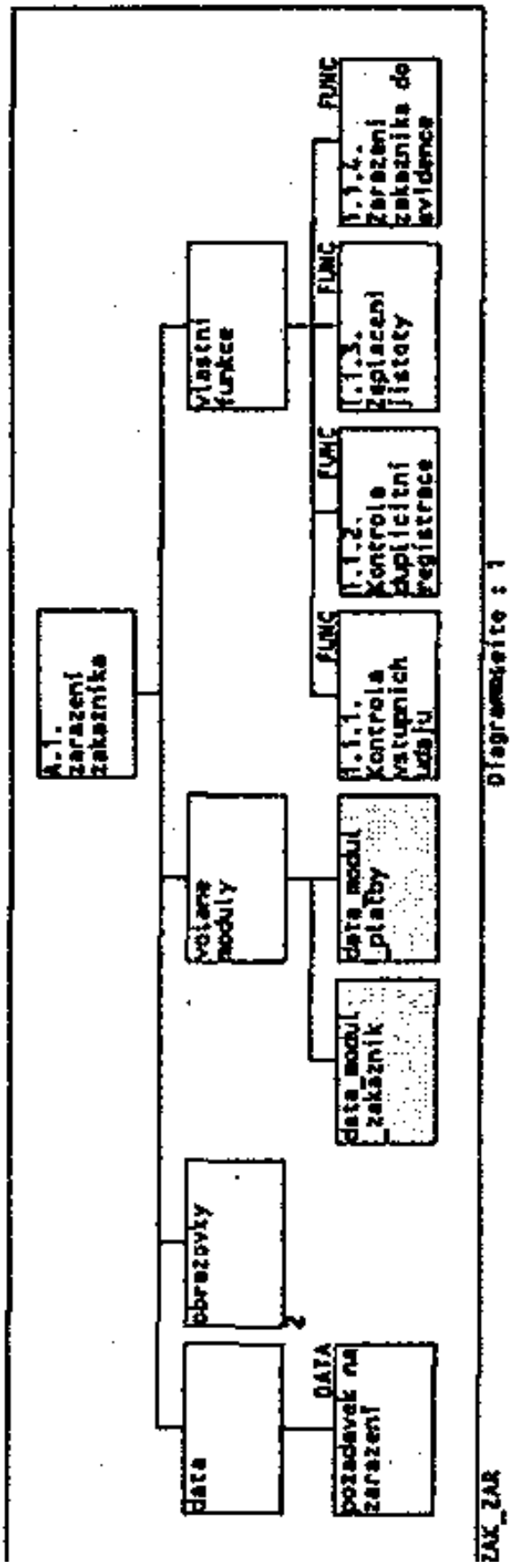


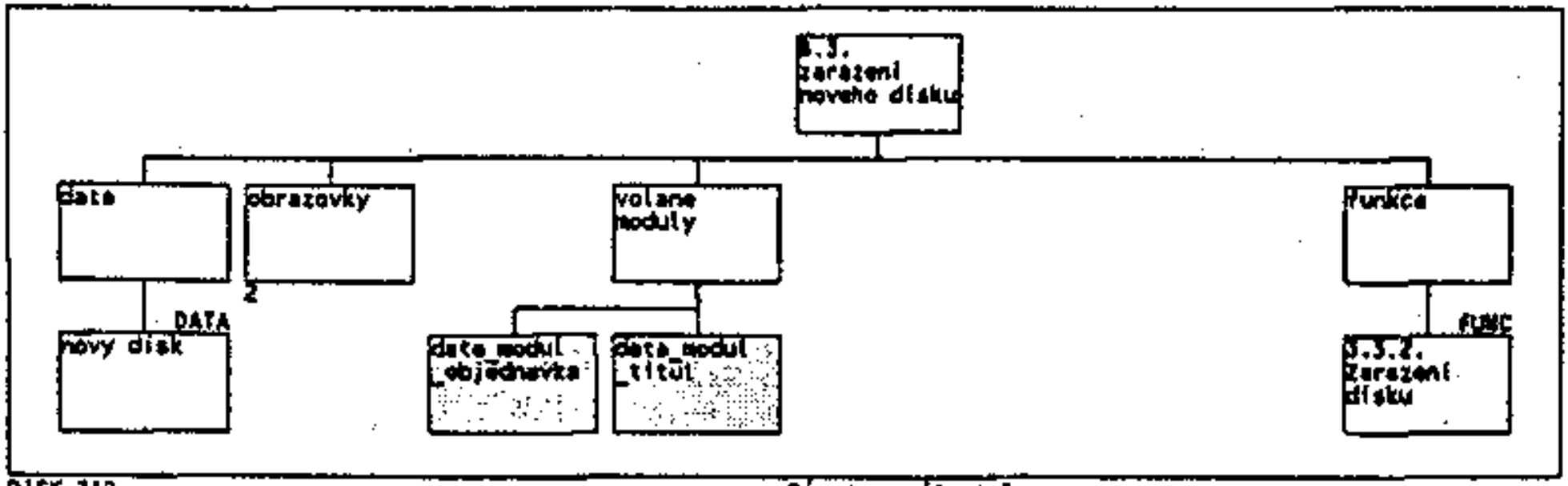
1.1. ZARAZENÍ ZÁK. DO EVIDENCE

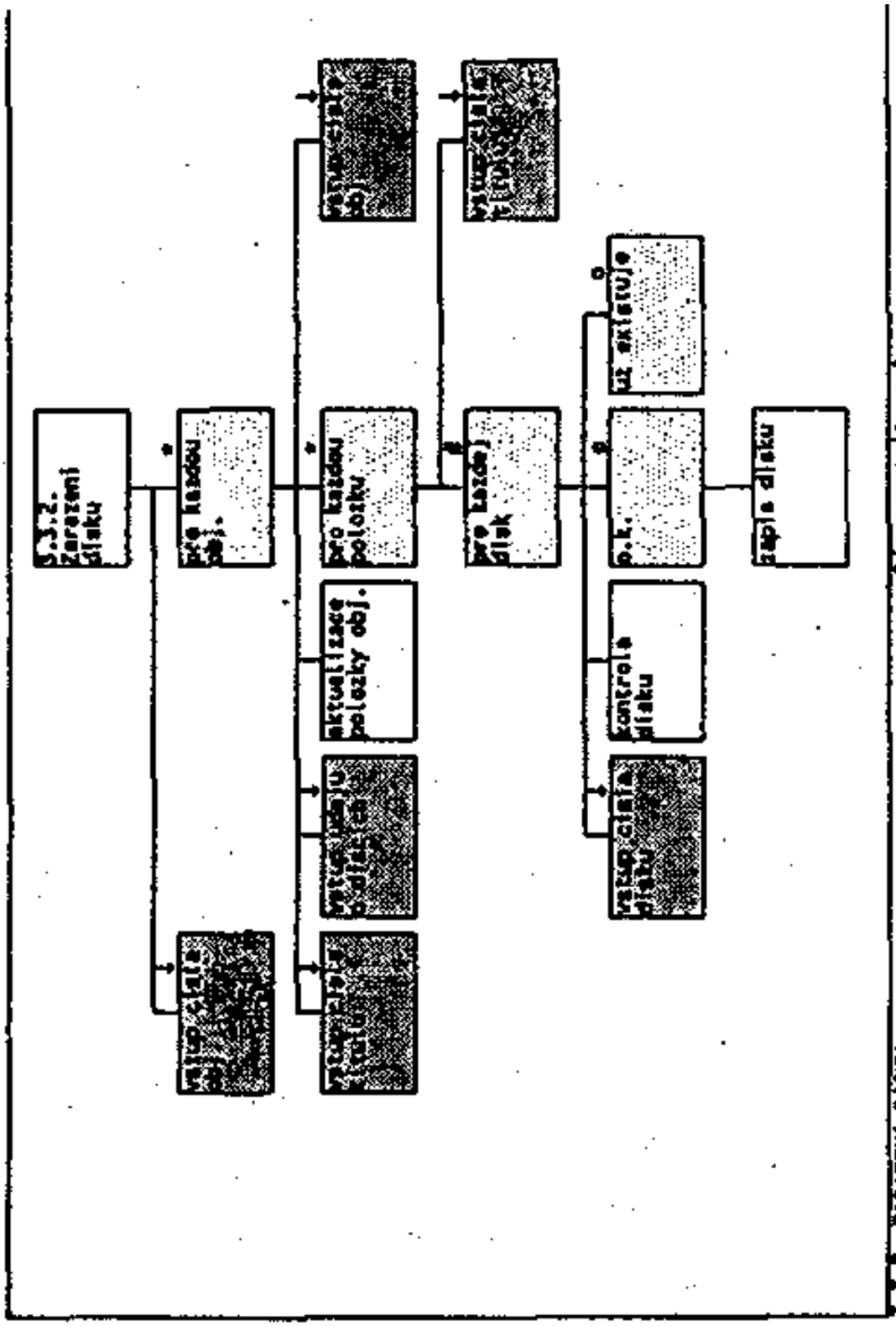


ERA PUJCOVNA CD

Ukázka návrhu modulu




5.3.
zarazeni
noveho disku
data
obrazovky
volane
moduly
funkce
DATA
2
novy disk
data_modul
_objednavka
data_modul
_titul
FUNC
5.3.2.
Zarazeni
disku
DISK_ZAR
Diagrammseite : 1



Autor: Ing. Jiří Prokop
PragoData a.s.
Ve smečkách 22
110 00 Praha 10
tel. (02) 23 52 348 až 51