CASE SYSTEMS ENGINEER

Kanisová Hana

Úvodem svého příspěvku o praktickém použití CASE prostředku Systems Engineer od britské firmy LBMS chci říci pro nezasvěcené několik slov o tom co je to CASE a nač je to dobré. Když jsem tvořila tuto stať nemohla jsem samozřejmě tušit, jestli některý předřečník již neobjasňoval tuto problematiku přede mnou, takže pouze stručně.

Takže CASE je určitý balík programů, které pomáhají analytikovi provádět funkční a datovou analýzu. Pouze pomáhají. Většinou podporují určitou metodologii a mohou běžet jak pod DOSem (resp. Windows) nebo pod Unixem nebo jeho mutacemi. Je naivní si myslet, že nákupem CASE prostředku se analýza urychlí, opak je pravdou. První analýza s CASE produktem bude spíše pomalejší. Ten malý zázrak tkví v následujících projektech, kdy už analytici umí tyto programy používat a ovládají metodologii. Protože hlavně metodologie je to nejdůležitější.

Již téměř před dvěma roky si naše firma Disam Systems Ostrava vybrala CASE prostředek Systems Engineer firmy LBMS. Nyní Vám mohu říci, že si vybrala dobře. Tento produkt běží pod Windows a podporuje funkční a datovou analýzu až do úrovně vytváření vlastních programů. Podporuje britskou státní metodologii SSADM a to tak důsledně, že analytika při práci vede, což je hlavně pro začínajícího projektanta software neocenitelné.

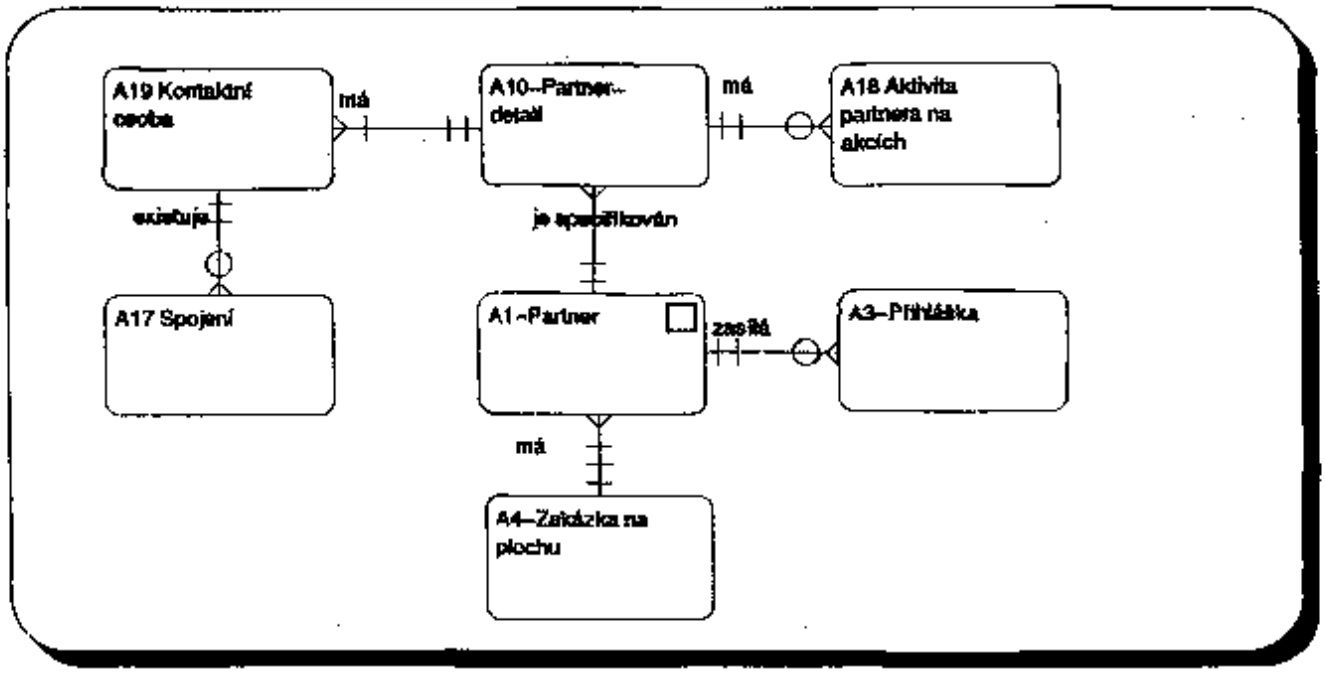
Z historického hlediska lze rozlišit dva přístupy při navrhování informačních systémů - funkční a datový. S tím souvisí různé typy metodologií a CASE produktů tyto podporující. Systems Engineer se s tímto problémem vyrovnal poměrně dobře.

Po uskutečnění úvodní studie, navrhne analytik kontextový diagram, který je vlastně startovací úrovní Diagramu toku dat (DFD) což je v podstatě začátek **funkční analýzy.** Souběžně s tímto krokem začne vytvářet prvotní diagram entit - to znamená start **datové analýzy.** Již na počáteční fázi diagramů toku dat jsou datová úložiště asociovány s entitami v datovém modelu. Tyto vazby, které jsou základem úspěšné analýzy, jsou programově hlídány Systems Engineerem, který Vám na požádání vypíše veškeré metodologické chyby, kterých jste se v průběhu navrhování a realizace informačního systému dopustili. DFD vystihují co vlastně informační systém bude dělat a diagram entit zase řeší jaké data a v jaké struktuře bude třeba v systému uchovávat.

Než budu pokračovat v popisu následujících kroků, chtěla bych se krátce zmínit o tom, na kterém projektu jsme tento CASE použili a proč. Jednalo se o velmi rozsáhlou zakázku na funkční a datovou analýzu pro informační systém zákazníka. Aby jste si představili rozsah této analýzy, uvedu pouze jeden údaj. Výsledný počet navržených tabulek se blížil k číslu 300 a počet firem, které má zákazník v evidenci se pohybuje od 50 000 a výše. Proč o tomto hovořím? Protože na tomto projektu pracovalo 14 analytiků a bez podpory CASE prostředku by byla jejich koordinace velmi obtížná. Tím, že byli nuceni dodržovat přísně jednu metodologii, nemohlo dojít k nejednotnému přístupu. Každý programátor mívá sklon k určité anarchii a ke svým vlastním řešením. Tyto vlastnosti mohou být užitečné při vlastním programování, ale při analýze se musí metodologie dodržovat. I tak při návrhu tabulek a funkci zbyde pro tvůrčí nápady spoustu místa.

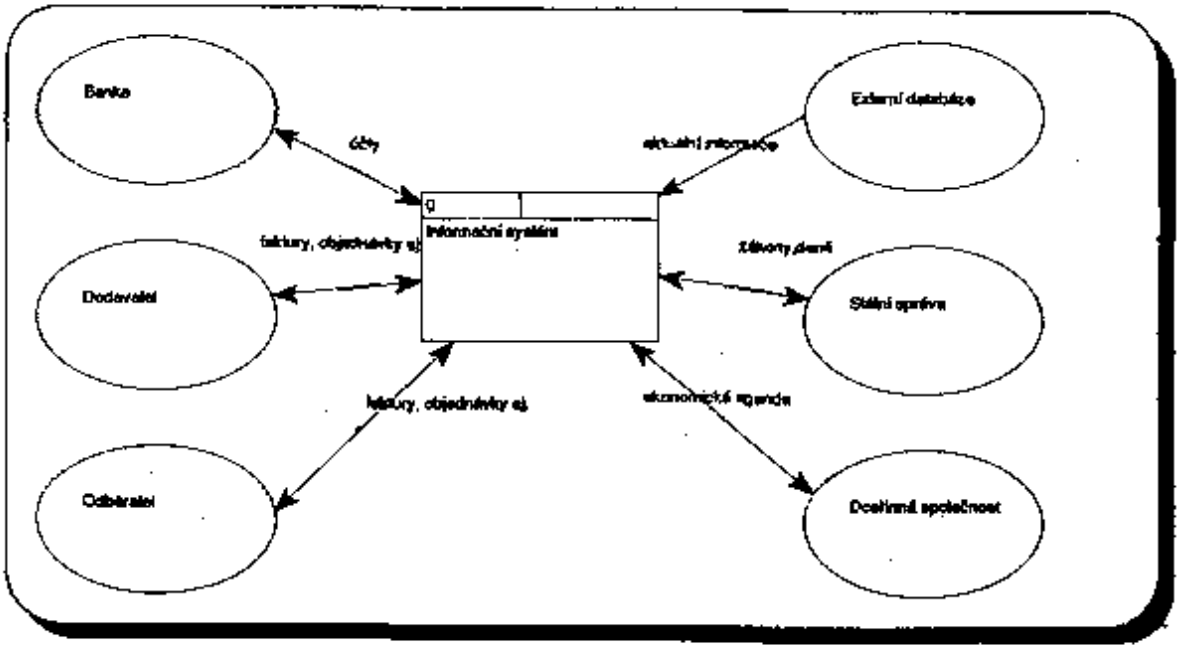
Proto je velmi důležité při nákupu CASE prostředku se podrobně seznámit s metodami, kterou tento produkt používá. Nežli naučit se používat CASE programy, bývá podstatně těžší osvojit si metodologii tohoto produktu.

Typický příklad diagramu entit



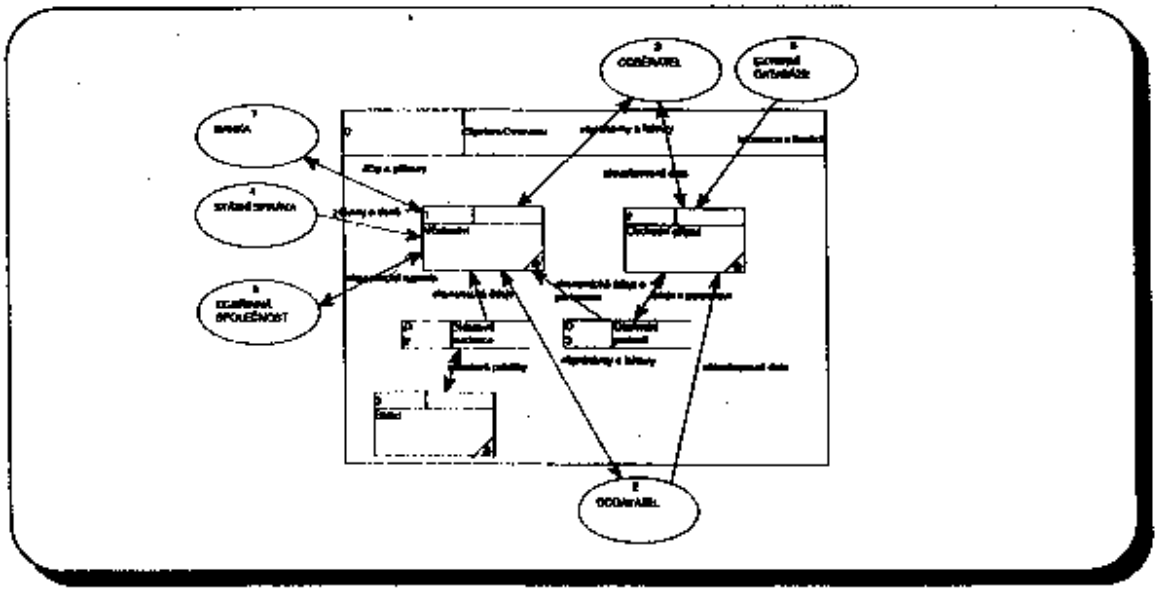
Nyní si tedy probereme krok za krokem techniky SSADM. Ke tvorbě kontextového diagramu je třeba si vytipovat veškeré externí objekty.

Příklad kontextového diagramu

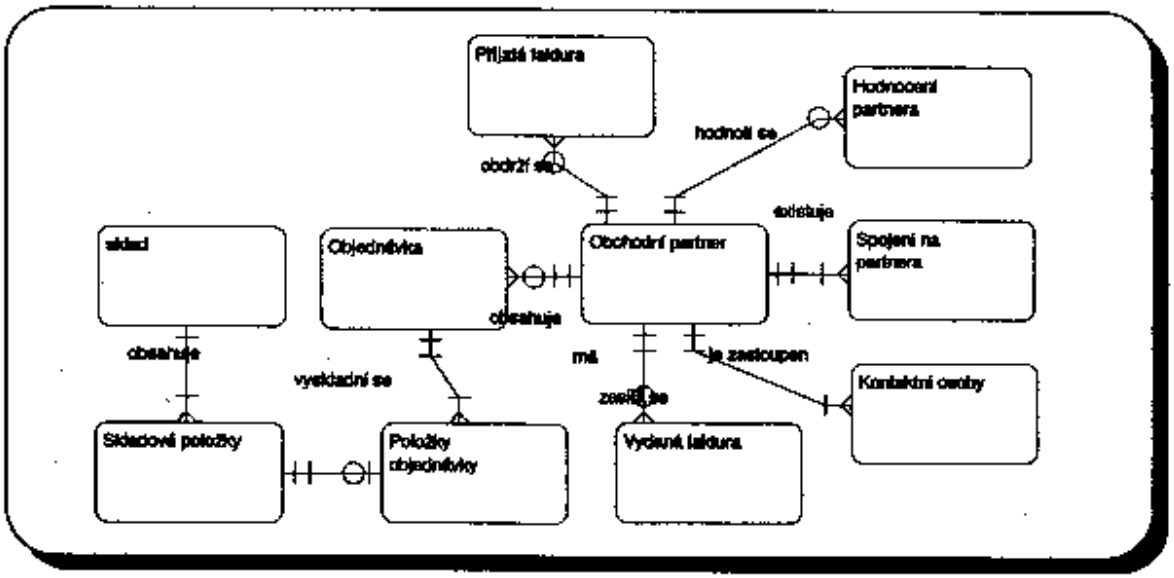


Externími objekty rozumíme např. banku nebo zákazníka apod., stručně řečeno zkoumáme tok dat přes rozhraní informačního systému. Dalším krokem je navržení subsystémů, čímž vytvoříme nultou úroveň DFD. Zde již uvažujeme logické rozdělení do budoucích procesů. Na obrázku můžete vidět velmi jednoduchý příklad - rozdělení na účetnictví, obchodní případ a skladovou evidenci. Je to

velmi zjednodušený pohled, který by se ve skutečnosti v takové formě asi těžko vyskytoval, avšak pro přiblížení problematiky je dostačující.



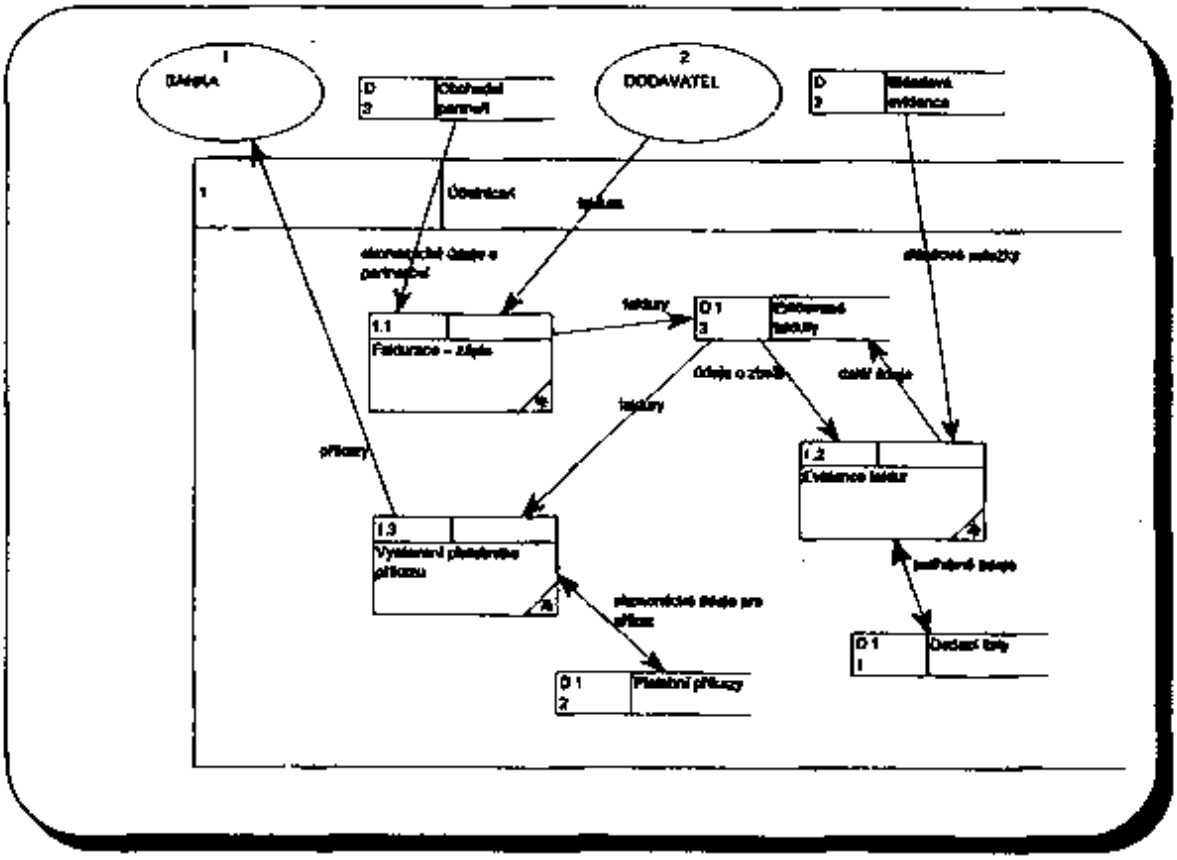
Již v tomto okamžiku je nutné tento diagram uvést do souladu s diagramem entit, a to tak, že jedna entita musí být obsažena vždy v jednom datovém úložišti diagramu toku dat, ale naopak jedno datové úložiště může obsahovat více entit. Entitám začínáme přiřazovat datové atributy, t.j. sloupce budoucích tabulek a tyto atributy se zase zpětně promítnou jako datové položky úložišť v DFD diagramech.



Zkoumají se vazby mezi jednotlivými entitami, navrhují se vazební věty, což znamená - jaký vztah má master entita k detail entitě. Např. vazební věta může být: obchodnímu partnerovi se zasílá faktura. Neustále se dbá na propojení diagramu entit a diagramu toku dat přes asociace. Asociace je určitý druh spojení, který za Vás udržuje CASE prostředek, a který Vám v každém okamžiku Vaší práce na projektu může poskytnout informace o vazbách v návrhu informačního systému.

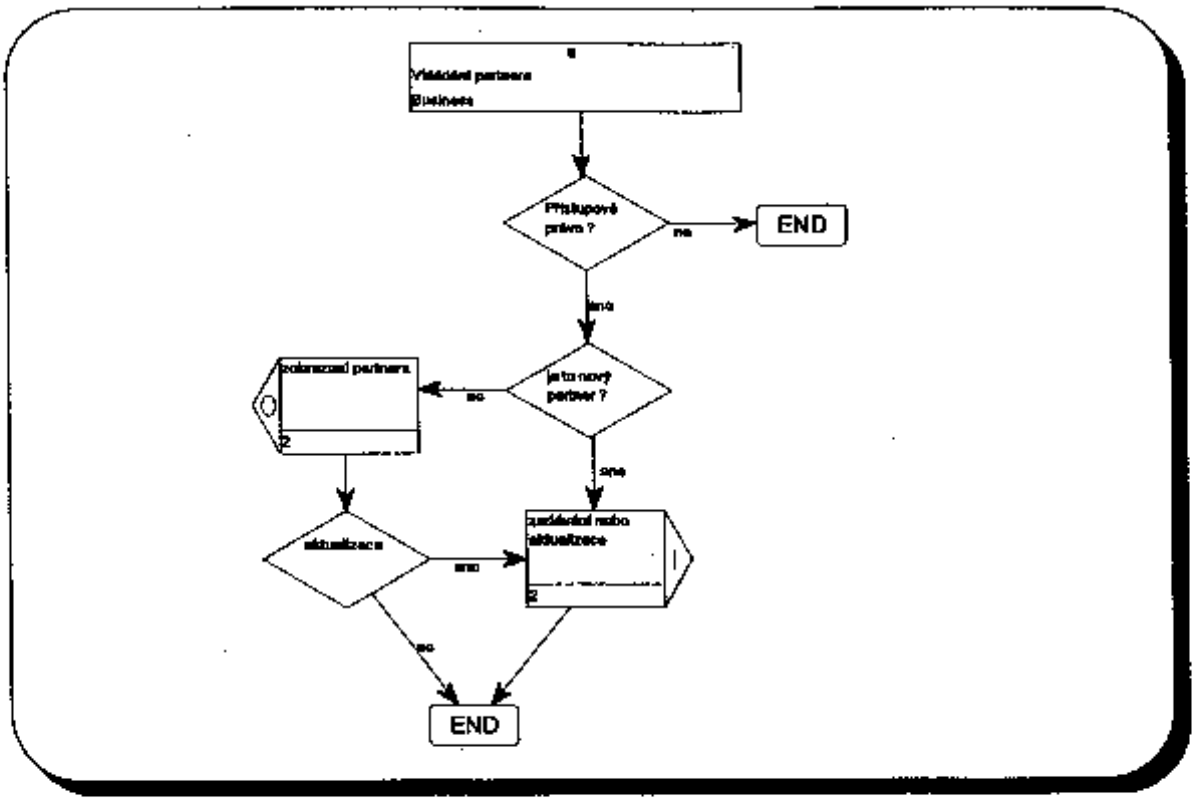
Souběžně probíhá dekompozice DFD a to až do úrovně, kdy jeden proces odpovídá jedné funkci. V této fázi je důležité mít sestaven katalog událostí. Událost je cokoliv co vyvolá určitý efekt v informačním systému. Řečeno programovacím jazykem SQL je efekt insert, modify a delete. To

znamená, že obecně je událost změna v datech. Podle počtu událostí můžeme usuzovat počet funkcí protože událost vyvolá funkci. Jako příklad události můžeme uvést třeba "příchod objednávky" nebo "vystavení platebního příkazu". Na diagramu toku dat poznáme událost jako protnutí hranice informačního systému s okolím tohoto systému. Událost však může nastat i uvnitř informačního systému.



Můžeme tedy z katalogu událostí a z diagramu toku dat odvodit jednotlivé funkce. Funkce se obvykle skládá z několika transakcí. Co rozumíme pod pojmem transakce? Transakce je ta část zpracování, která mi zanechá databázi v konzistentním stavu. Z toho pak vyplývá, že funkce jako taková spouští sekvenci transakcí. Funkce je nutno slovně popsat, ale nejdůležitější je vyznačit vztahy (asociace) pomocí CASE prostředku na procesy v diagramech toku dat (DFD), na události zachycené v katalogu událostí a nakonec na transakce. Transakce můžeme také slovně popsat, ale v CASE Systems Engineer je prostředek pro grafické znázornění vlastní práce transakce.

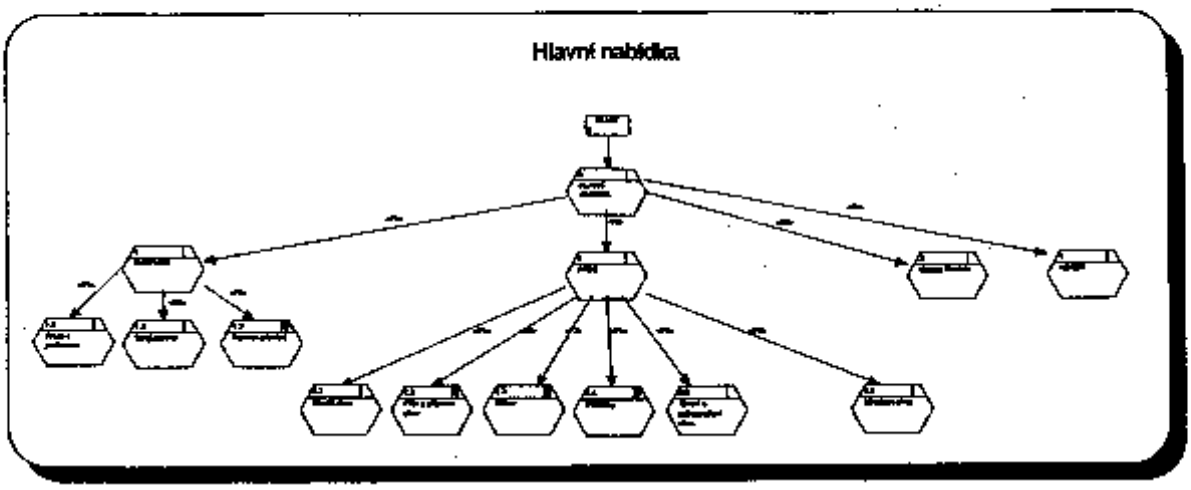
Při rozsáhlejších projektech není možné graficky zachytit v diagramech všechny transakce, transakční dialogy budeme dělat tedy pouze pro kritická místa informačního systému. Avšak katalog transakcí musí být kompletní, neboť slouží jako významný podklad pro budoucí programování. Jako příklad uvádím transakci, která bude umožňovat zadání nového obchodního partnera. Je to velmi zjednodušený elementární diagram.



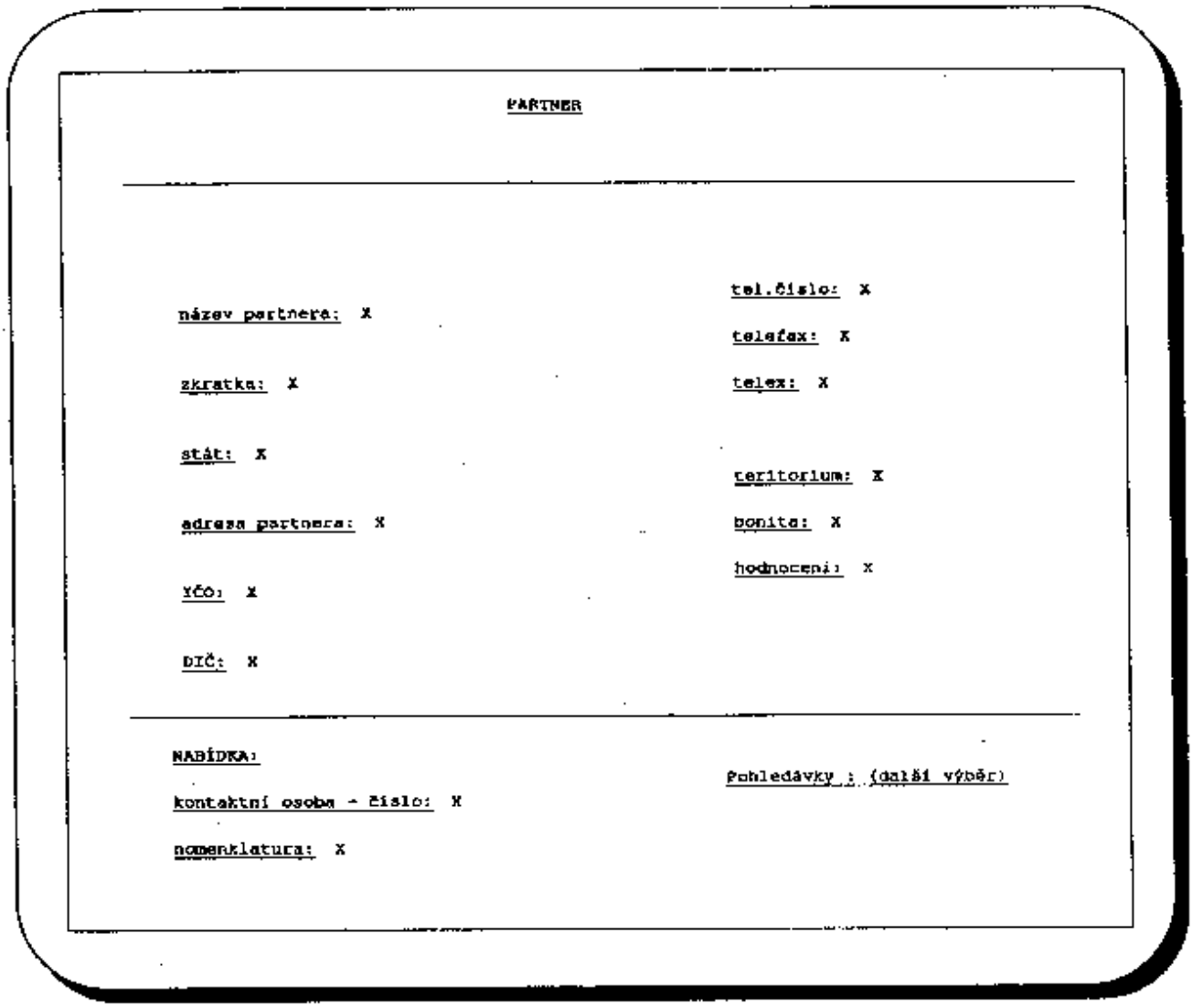
Příklad transakčního dialogu

Na obrázku vidíte dva typy obdélníků - vlevo se písmenem "O" znamená výstupní operaci (output) a vpravo se písmenem "I" určuje vstupní operaci (input). Tyto operace udržuje CASE Systems Engineer v asociaci s navrženými obrazovkami. Obrazovky můžete navrhnout buď jako klasické znakové obrazovky pracující v terminálové síti anebo je vytvořit pomocí prostředku Client - Builder jako klientovské aplikace běžící pod Windows. V druhém případě se již jedná tedy o spustitelné klient - programy.

Příklad návrhu struktury hlavního menu



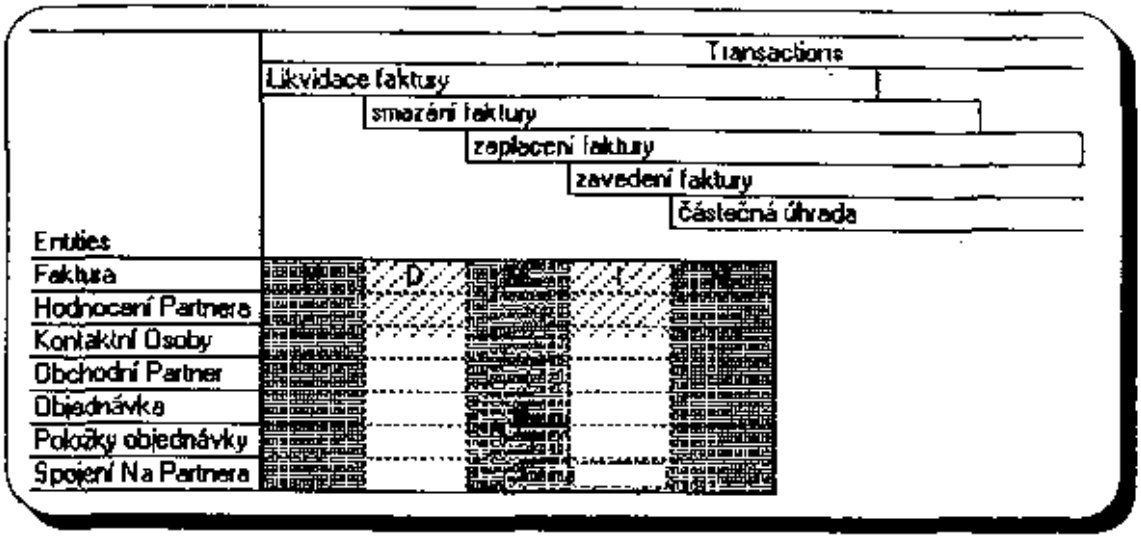
Příklad návrhu klasické znakové obrazovky



Další užívanou technikou je tzv. prototyping, který slouží k dialogu s uživatelem. Vychází ze tří komponent Systems Engineer - návrhu struktury menu, návrhu obrazovek a návrhu transakcí. Budoucímu uživateli je možno předvést jak bude požadovaný software vypadat, tzn. že CASE umožňuje spustit sekvenci obrazovek, které uživatel při běhu programu uvidí, což je velmi užitečné, protože často až v této fázi zákazník přijde na to, co vlastně od informačního systému může požadovat. Jako příklad zde uvádím klasický návrh znakové obrazovky vytvořené v Systems Engineer pomocí editoru obrazovek a editoru pro návrh uživatelských menu.

Nyní se ještě vrátím ke datové analýze. V okamžiku, kdy je hotov seznam transakcí a s tím související katalog funkcí, je nutné vytvořit matici entit a transakcí. Tato technika slouží k ověření toho, zda každá entita má transakce, které ji založí, eventuálně s ní pracují a ruší ji. Jestliže v průběhu vytváření křížového přiřazení entit a transakcí v matici zjistíme, že některá entita tento vztah k transakcím postrádá, je nutné provést změny v katalogu transakcí a následně i seznamu funkcí a v diagramu toku dat. Jako příklad zde uvádím výřez matice entit a transakcí, přičemž písmeno R znamená read, písmeno I - input a písmeno D - delete.

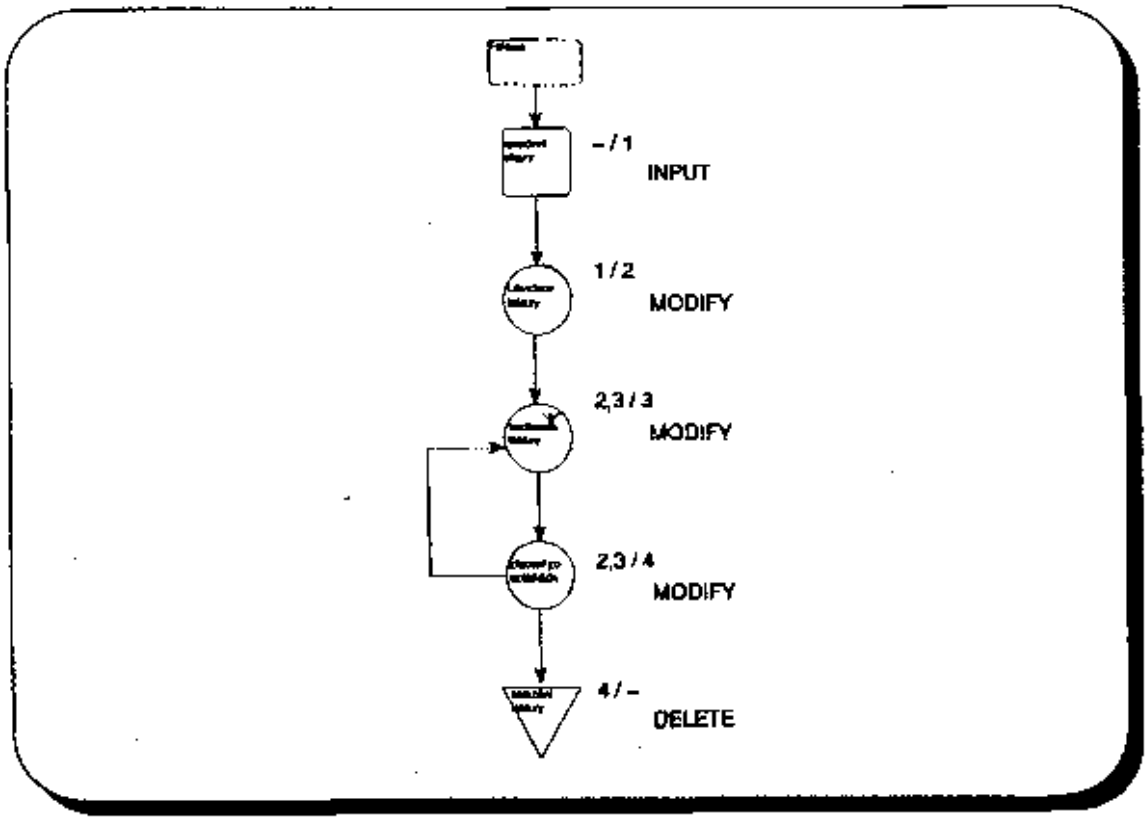
Matice entit a transakcí



Zatím všechny zde probírané techniky nedokázaly vyjádřit co se děje s entitou (tzn. s budoucí tabulkou) v průběhu existence informačního systému.

Příklad životního cyklu entity faktura

Vliv času je zachycen v tzv. životním cyklu entity. Tento postup zachycuje sekvenci zpracování. Jako příklad je zde zachycen životní cyklus entity (ELH) - faktura.



ELH podchycuje vliv transakčního zpracování na existenci entity (t.j. vlastně budoucího záznamu v tabulce). Vznik entity je symbolizován obdélníčkem, modifikace entity kolečkem a její výmaz trojúhelníčkem. Čísla vpravo vyznačují stavy, ve kterých se entita nachází, přičemž před

lomítkem je zaznamenán seznam stavů předchozích a po lomítku stav, který vzniká, pomlčka pak stav neexistence. Objeví-li se v grafickém symbolu transakce hvězdička, pak z toho vyplývá možnost cyklického opakování dané transakce. Můžeme tedy vlastně z tohoto stanovit definici transakce: transakce je převod z jednoho stavu entity do jiného.

Velkou výhodou CASE Systems Engineer je, že pokud máte vyplněnou matici entit a transakcí - tzn., že každá entita má transakce, které s ní provádí operace INPUT, MODIFY a DELETE, tak tento produkt vygeneruje diagram životního cyklu entit sám. Analytik se pak může po projití těchto diagramů lehce přesvědčit, jestli opravdu má pro každou entitu nadefinované transakce, které ji zakládají, mažou a eventuálně s ní pracují. Pokud zjistí, že například chybí výmaz určité entity, pak zařadí nově tuto potřebnou transakci do seznamu transakcí a následně pak se toto promítne v seznamu událostí i funkcí. Přes asociace funkcí na diagram toku dat bude zaveden potřebný proces. Z toho vyplývá že v metodologii, kterou používá Systems Engineer všechno souvisí se vším.

Jedním z důležitých výsledků analýzy je katalog událostí, funkcí a transakcí. Pokud byla analýza provedena správně, tak zjistíme, že každá funkce musí být vyvolána nějakou buď vnější nebo vnitřní událostí. Následně tomu musí být funkce jak v asociaci s procesem zakresleným v diagramu toku dat tak i v asociaci se transakcemi, které tato funkce vyvolá.

Vztah mezi diagramem entit a diagramem toku dat je jednoznačně definován vztahem mezi entitami a datovými úložišti v DFD. Z toho vyplývá, že každá entita musí být obsažena v datovém úložišti a opačně datové úložiště musí obsahovat jednu nebo více entit zakreslených v diagramu entit.

Velkou výhodou Systems Engineer je programová kontrola správnosti analýzy z hlediska použité metodologie firmy LBMS. Můžeme si tedy nechat vypsat asociační vazby mezi jednotlivými technikami, např. na které procesy v DFD, události a transakce má daná funkce vytvořené vazby apod. Při kreslení diagramů toku dat CASE upozorní na metodologické chyby. Bylo by velmi rozsáhlé a nudné zde uvádět příklady těchto chyb, protože metodologie SSADM jde velmi do hloubky a v každém okamžiku analytik ví, jaký další krok má provést.

Chtěla bych Vás nyní seznámit s tím, co je asi zajímavější, a to sice co vlastně dostane programátor od analytika do ruky po skončení analýzy. Jsou to následující věci:

1. Kompletní datový znormalizovaný model entit
2. Datový slovník s vyznačenými klíči
3. Diagramy toku dat s provedenou dekompozicí až na úroveň funkcí
4. Katalog funkcí se slovním popisem těchto funkcí
5. Seznam událostí, které způsobí změnu v datech informačního systému
6. Seznam transakcí, které jsou spouštěny funkcemi
7. Tabulku vztahů mezi událostmi-funkcemi-transakcemi
8. Matici entit a transakcí
9. Životní cykly entit
10. Návrhy obrazovek a menu
11. Diagramy struktur transakcí pro úzká místa informačního systému

Existuje několik odlišných názorů na další úlohu analytika v průběhu projektu. Zda má po ukončení analýzy pokračovat jako programátor anebo přejít na analýzu případného dalšího projektu. Myslím, že to závisí na velikosti firmy a projektu, avšak můj soukromý názor je, že by měl u původního projektu zůstat a působit spíše jako koordinátor několika programátorů, protože v CASE produktu se dá zachytit hodně, ale úplně všechno ne. Dosti často je určitě zapotřebí vysvětlit, co tím básník myslel.

Na závěr ještě pár slov o propojení CASE Systems Engineer na jiné produkty.
Existuje prostředek Server Builder pro převod datového modelu do cílové databáze. V současné době jsou pokryty všechny významné relační databáze vyskytující se na trhu.

Jednou z největších výhod CASE Systems Engineer je jeho otevřenost vůči vývojovým nástrojům jiných firem. Jako příklad můžeme uvést vazbu na dva nejvýznamnější dodavatele software pro tvorbu klient - aplikací pod MS - Windows.

Firma Gupta zajišťuje přenos datového modelu vytvořeného v Systems Engineer do prostředí své relační databáze. Dále pak existuje SE/Open for PowerBuilder, který integruje CASE Systems Engineer s vizuálním vývojovým prostředkem PowerBuilder
firmy PowerSoft. Je tedy možné převést pomocí SE/Open výsledky analýzy vytvořené v CASE produktu Systems Engineer do vývojového prostředí PowerBuilder a to sice:

1. Kompletní přenos datových atributů
2. Obousměrný přenos GUI objektů (grafic user interface) navržených v CASE SE a realizovaných PowerBuilder.
3. Využití SQL příkazů vygenerovaných v Systems Engineer jako objekty Query v aplikacích PowerBuilder.
4. Zpětný přenos objektů z PowerBuilder do repository CASE Systems Engineer.

Takže závěrem by jsem případné zájemce o zhlédnutí CASE Systems Engineer chtěla pozvat do naší firmy Disam Systems Ostrava, kde je možné tento produkt i jeho praktické použití vidět.

Autor :

ing. Hana Kanisová
Disam Systems a.s.
Výstavní 9
Ostrava 1
Telefon - 069 / 576 26
069 / 662 65 41 / 487
Fax - 069 / 661 17 57