Zkušenosti z vývoje vlastního integrovaného DB-systému

Aleš Grim, prom.mat. a RNDr Lubomír Vašák, CSc
Výpočetní centrum Státní plánovací komise

Ve Výpočetním centru Státní plánovací komise skupina 5 až 6-ti pracovníků vyvíjí od roku 1983 interaktivní víceuživatelský distribuovaný databankový systém s názvem Systém Datových Základen (dále jen SDZ). SDZ je od roku 1987 provozován a pracuje s terminálovou a počítačovou sítí VC SPK a RVS INORGA Praha na počítačích IBM 4361 a IBM 370/148 pod operačním systémem VM/SP, alespoň Rel 4; byl též úspěšně odzkoušen na počítači EC 1057 pod operačním systémem SVM, Rel 3.4. SDZ je orientován na operační systém CMS (PTS), filosofii virtuálních strojů, využívá firemní produkty REXX, XEDIT, IUCV, RSCS, volitelně GDDM a APL, a CMS- nebo VSAM- soubory pro SDZ-databáze. Naprogramován je v jazyku ASSEMBLER a PASCAL. Při vývoji se pro prototyping užívá REXX a částečně i APL.

Při budování SDZ jsme získali určité zkušenosti z technologie tvorby software, které uplatňujeme při dalším efektivním vývoji systému a které zde chceme prezentovat. Pro základní představu o realizovaném objemu prací předkládáme stručnou charakteristiku našeho systému.

1. Stručná charakteristika SDZ

SDZ je obecné programové vybavení integrující pět složek:

a) práci s hierarchicky organizovanými objekty

SDZ pracuje se soustavou distribuovaných datových bází (datové báze mohou být v různých virtuálních strojích i v různých počítačích). Jednotlivá datová báze může být sdílená nebo individuální, s pouze čtecím nebo i zápisovým přístupem, uložená jako CMS- nebo VSAM- soubor. Je organizována jako hierarchická soustava objektů (podobně jako disk v MS DOSu na osobním počítači). V objektech lze ukládat identifikovaná data (viz bod b), vzory formátovaných obrazovek (viz bod d), zdrojové, přeložené i exekutabilní programy kompilovaných jazyků, utility v jazyku REXX, číselníky, obecné texty (např. příručky SDZ) a definice tabulek. SDZ dovoluje tyto objekty nejen ukládat, nýbrž je i aktivně využívat. Např. utility v jazyku REXX mohou sloužit jako další povely SDZ.

b) databanku

Databanka používá tzv. dimenzionální identifikaci dat a je to jiný princip, než relační nebo síťový nebo hierarchický. Společně identifikovaná data jsou uložena v základní logické a organizační jednotce - v datovém modelu. Údaj modelu je identifikován n-ticí hodnot souřadnic, tj. je prvkem v n-rozměrné souřadné soustavě. Výčet všech hodnot, které může jednotlivá souřadnice nabýt, se v rámci SDZ nazývá dimenze; je to vlastně nestrukturovaný číselník. Údaje jsou buď čísla (celá nebo desetinná), nebo texty; informace o délce a typu údaje je součásti údaje.

c) tabulkový editor (spreadsheet)

Tabulkový editor pracuje s 3-dim. tabulkou (má i stránky), údaje tabulky se vybírají z (ukládají do) modelu. Tabulka tedy není reálný objekt v DB, představuje jakési okno do modelu. Definice tabulek se mohou vytvářet dynamicky nebo ukládat do DB. V tabulkách lze provádět výpočty, tabulky lze agregovat, tisknout a zobrazovat i přes individuálně formátované obrazovky.

d) práci s formátovanou obrazovkou

SDZ umožňuje pracovat s formátovanou obrazovkou pomocí GDDM (produkt fy IBM) i bez něj (zlepšovací návrh). Vzory obrazovek (panely) lze vytvářet interaktivně a ukládat do DB nebo generovat v programu. Data do/z panelu se umísťují z DB nebo z dat programu. Je zajištěna plná komunikace mezi panely a firemním jazykem REXX.

e) rozhraní (interface) k hostitelským jazykům

Jako integrální součást SDZ je vazba na REXX. Využívá se jak pro budování aplikací v SDZ, tak i pro rozšiřování SDZ o nové funkce. Dále lze SDZ využívat z těchto programovacích jazyků: APL, PASCAL, PL/I, ASSEMBLER. Vazbu je možné rozšířit i na jazyky FORTRAN a COBOL.

2. Spolupráce v týmu

Při vývoji rozsáhlého programového systému nelze oddělit fázi vývoje od fáze užití. Obvykle po nasazení první verze probíhá dále vývoj a současně je třeba zajistit podporu uživatelům (školení, konzultace, vytváření vzorových aplikací apod.).

Jestliže na počátku vývoje mohou tvořit tým 2-3 analytici, v dalším období by měl počet členů růst v závislosti na velikosti projektu max. na 10 členů. Další rozšiřování počtu účastníků na projektu si vynutí rozdělení do více týmů se specializací práce. Vedoucí týmů pak tvoří řídící skupinu - s touto formou známe vlastní zkušenosti.

Podle podílu na vytváření koncepce systému a rozdělení práce rozlišujeme 3 skupiny členů:

a) ti, kteří vytvářejí koncepci celého systému tj. obecné principy, rozdělení do subsystémů, časový harmonogram. Odpovídají za jednotlivé subsystémy a obvykle jsou zakládajícími členy týmu.

b) ti, kteří vytvářejí koncepci subsystému tj. jeho funkce, společné datové struktury, postup implementace, rozdělení do programových jednotek.

c) ti, kteří vytvářejí jednotlivé skupiny programů a odpovídají za ně. Stávají se členy týmu postupně podle postupu prací. Je žádoucí, aby programováním se zabývaly i skupiny a) i b), oddělení analytické a programátorské práce vede k snížení efektivity práce a zájmu o konečný výsledek u skupiny c).

Vedoucí týmu nemusí být totožný s vedoucím po linii hospodařské (např. oddělení), musí ale mít přehled o celém systému, jeho centrálních datových strukturách a o funkci a vazbách subsystémů. Koordinuje činnost a zajišťuje komunikaci jak uvnitř týmu, tak i navenek. Všichni členové týmu by měli mít praktické zkušenosti s používáním budovaného systému. Tento požadavek je zásadní, tým snadněji nalézá nedostatky a úzká hrdla (např. zvýšené nároky na výpočetní systém) a inspiraci pro plánování dalšího rozvoje.

Podle naší praxe je nejvhodnější metodou pro vytváření koncepce celého systému, subsystému a klíčových programů tzv. řízený dialog. Je to vlastně "hádání" před tabulí, jehož iniciátorem je člen týmu odpovědný za danou problematiku; ten předloží problém a svůj návrh řešení. Účastní se všichni členové týmu. Pro každý problém jsou tyto porady nejméně dvě. Na první se tým seznámí s problémem a diskutuje se návrh - často se původní návrh "rozcupuje". Do druhé porady je třeba nechat dostatečný čas - minimálně týden - aby si členové týmu rozmyslelí vlastní protinávrhy (problém musí uzrát). Na další poradě probíhá výběr mezi navrhovanými variantami, obvykle dochází k syntéze více variant. Je třeba dospět do stavu, kdy celý tým a především iniciátor dialogu s výsledným řešením souhlasí. Diskuse bývá velmi rušná, je však třeba dodržet zásadu, že se neútočí na lidi, nýbrž na myšlenky.

3. Metodika budování systému

Představa, že rozsáhlé programové vybavení lze celé nejprve vymyslet a pak celé naprogramovat je podle našeho názoru nereálná. Zadání systému se vyvíjí v průběhu jeho budování a později v průběhu jeho využívání. Proto je vhodné navrhnout v prvním zadání především integrující složky systému doplněné o předběžný časový harmonogram:

a) základní vertikální členění systému (vrstvy)
b) základní horizontální členění (požadované funkce)
c) společné datové struktury a repertoár rutin
d) způsob komunikace a předávání řízení mezi jednotl. složkami
e) jednotné principy označování a dokumentace programů
f) formu uživatelské i systémové dokumentace
g) programovací prostředky a využitelný firemní software

Naše zkušenost ukázala, že všechny složky prvotního návrhu se vyvíjí tak, jak se realizují jednotlivé prvky systému - množina funkcí se rozrůstá, datové struktury se rozšiřují, harmonogram se mění. Společné principy jako celek však musí vydržet. Tomu napomáhá i vrstvená architektura systému, která konceptuálně odděluje jednotlivé složky, umožňuje udržet celkový přehled a usnadňuje přenositelnost na jiný operační systém. U složitých systémů nejnižší vrstva nutně navazuje na operační systém a je tedy nepřenosná. V SDZ rozlišujeme pět základních vrstev - vazba na operační systém, fyzická vrstva, logická vrstva, uživatelská vrstva a vrcholová řídící vrstva. Pokud je systém vyvíjen současně "shora" i "zdola", umožní se tím zahájit ladění celku v raných fázích programování a konstituovat funkčně ucelené verze, jejichž části jsou např. dočasně psané v interpretovaném jazyce. Domníváme se, že to je nejlepší způsob vývoje.

Využitelný firemní software musí být diskutován již v prvotním návrhu. Jeho výběr má vliv na přenositelnost systému, funkční možnosti, rozsah prací i časový hramonogram. Do SDZ je např. zakomponován REXX jako aplikační i systémový jazyk, XEDIT jako editor pro úpravu objektů, IUCV jako prostředek pro zajištění multipřístupu do sdílených DB, RSCS pro mezipočítačovou komunikaci, APL pro výpočetní funkce tabulek, GDDM pro práci s obrazovkou a VSAM jako jedna z přístupových metod. APL, GDDM a VSAM nejsou nutné pro provoz systému.

Pokud jde o programovací jazyky, osvědčila se nám kombinace tří typů:

a) kompilovaný jazyk blízký strojovému kódu (u nás ASSEMBLER, vhodný je i C) - pro nejnižší vrstvy navazující na operační systém a pro nejvíce zatížené programy.

b) vyšší kompilovaný jazyk (u nás PASCAL, možná ADA) - pro převážnou část vyšších vrstev.
c) interpretovaný jazyk (u nás REXX a částečně APL) - pro prototyping a vytváření uživatelské nadstavby. Používá se přechodně všude tam, kde je nutné vyzkoušet několik variant nebo kde nevadí zvýšené nároky na zdroje počítače.

Tyto prostředky je třeba systémově upravit tak, aby programy v nich vytvořené se daly navzájem vyvolávat bez narušení svých prostředí. Pokud má být systém víceuživatelský (náš případ), musí být navíc výsledný strojový kód reentrantní. Např. my jsme upravovali kompilátor i podpůrný systém (Run Time) jazyka PASCAL AAEC 8000 a vytvořili podpůrné procesory na APL, REXX a XEDIT.

Pokud jde o metodiku programování, používáme pojem "zřetelné programování". Neznamená to striktně strukturované programování, ale základní myšlenkou je vytvořit co nejpřehlednější programy "rozumnou" aplikací principů strukturovaného programování. To znamená, že principy lze porušit v případech kdy to nesníží přehlednost programu (např. odskoky po chybách). V SDZ nepoužíváme žadnou formalizovanou metodiku, základním kritériem je, aby každému programu mohl porozumnět kterýkoliv člen týmu.

4. Vztah k uživateli

Především jsme zamítli myšlenku, že uživatelé jsou nutné zlo a nechutní šťouralové. Naopak ta vlastnost, že jsou schopni obratem vymyslet takovou "nesmyslnou" kombinaci užití, s níž tvůrci nepočítali, je pro ladění systému nejužitečnější. Uživatele je třeba využít v celém procesu budování systému. Ale ne všechny ve stejné míře. Podle míry zapojení si dělíme uživatele do čtyř skupin:

a) **pasivní uživatelé**, kteří využívají aplikace vytvořené v rámci systému členy týmu nebo uživateli dalších skupin. Mimo rámec své aplikace systém neznají. S těmi se tým setkává vyjímečně a hovoří s nimi přátelsky.

b) **běžní uživatelé**, kteří umějí vytvářet nebo upravovat aplikace v systému. Od nich tým získává nejen informace o problematických místech systému, ale i řadu cenných námětů pro úpravy a další rozvoj. Používají právě platnou (rutinní) verzi systému. Na základě jejich připomínek se systém přibližuje uživateli a stává se tak pro něj "přítulnější" (user-friendly). Těmto uživatelům tým poskytuje stálou konzultační podporu.

c) **uživatelé** - externí spolupracovníci týmu. Jsou to vybraní uživatelé skupiny b), kteří natolik fandí práci týmu, že jsou ochotni stát se pokusnými králíky pro novinky v systému. Zkoušejí tedy vývojové verze systému po odladění členy týmu. Spočívá na nich hlavní tíha uživatelského ladění. S těmito uživateli tým konzultuje některé směry vývoje (někdy se účastní i "hádání") a jejich spolupráce si hluboce váží.

d) **uživatelé** - programátoři. Jsou to aplikační programátoři, kteří pomocí systému vytvářejí složité aplikace, často s využitím programovacích jazyků (nejen povelového režimu). Musejí se seznámit s některými složkami, které předchozí skupiny nemusí znát. Jejich přínos pro vyvíjený systém je kupodivu menší než u skupiny c), protože případné problémy řeší programovou nadstavbou a ne požadavky na úpravy v systému. Této skupině tým poskytuje interní školení a konzultační podporu.

Naším cílem je přenést co největší díl vývoje aplikací v systému na uživatele a co nejvíce podporovat jejich samostatnou práci. Pro tento cíl je třeba již od počátku budování systému vyvíjet prostředky - příručky, HELPy, menu, propracovaný povelový aparát, školení, konzultace a řadu vzorových aplikací. Samostatný uživatel nezdržuje trivialitami, ale je cenným zdrojem námětů.

5. Dokumentace a školení

Při vývoji systému je kvalitní dokumentace všeho naprostou nezbytností. Důležitá je především co nejvyšší užitná hodnota dokumentace, její aktuálnost a co nejnižší obtížnost jejího pořízení. Nejsnadněji tohoto cíle dosáhneme, když přiblížíme dokumentaci tomu, co popisuje, tzn. v SDZ je programová dokumentace přímo ve zdrojových programech, popis povelů je uvnitř povelové složky jako součást definice povelů, aktuální uživatelská dokumentace pro každou verzi je jako objekt v DB, HELP je příslušný výběr z příručky povelů. Popisy systému, maker, společných datových struktur, instalační návody, přehled chyb, provozní a uživatelská dokumentace jsou vytvářeny pomocí samotného systému, uchovávány v databázi a přes terminál okamžitě přístupny členům týmu a podle autorizace též uživatelům. Na okraj chceme podotknout, že máme negativní zkušenosti s HIPO-technikou, protože je pracná a odtržená od programu, který má popisovat.

Psát o nezbytnosti školení nemá smysl. V závislosti na tom, komu je školení určeno, dělíme je na několik druhů:

a) **interní** - členové týmu se navzájem informují o činnosti nově dokončených složek. Někdy se zúčastňují i uživatelé-programátoři.

b) **pro aplikační programátory** - uživatelé se seznamují se systémem hlouběji, především s vazbami na programovací jazyky a firemní produkty.

c) **rozdílová** - znalí uživatelé jsou informování o změnách v nové verzi.

d) **pro pokročilé uživatele** - seznamuje uživatele se systémem natolik, že jsou v něm schopni vytvářet vlastní aplikace.

e) **pro začátečníky** - poskytuje úvodní seznámení se systémem tak, aby uživatel mohl používat aplikace v něm vytvořené a provádět v nich jednoduché změny. Slouží také pro informaci případných zájemců o nákup systému.

Při všech školeních jsou významnou složkou praktická cvičení, která zabírají až polovinu doby školení. Při školeních využíváme připravených vzorových aplikací. Uživatele v nich mají často předlohu, podle které vytvářejí své první aplikace. Školení jsou dosti intenzivní a tedy náročná na přednášejícího i posluchače, a proto je třeba školení rozdělit dostatečně častými přestávkami (nejdéle po hodině).

6. Závěr

Na závěr chceme konstatovat to nejdůležitější - základním předpokladem úspěšné práce týmu je udržet aktivní zájem jeho členů a nejbližších uživatelů po celou dobu (v našem případě několika let) budování a implementace systému. Motivaci týmu udržují také jeho odpůrci a nedůvěřivci ("advocatus diaboli"). O ty jsme nikdy nouzi neměli.