COBOL a objektové programování

Jiří Veselý

Standardní programování

Postupy standardního programování lze popsat následovně. Pro zadaný problém (úlohu) je provedena krok za krokem analýza vstupních a výstupních dat a volba jednoúčelových algoritmů pro splnění požadovaného zadání. Tyto popisy dat a specifické algoritmy jsou následně převedeny do příslušného jazyka a vytvoří se spustitelné moduly. Vznikají tak speciální řešení, která musí být oddělene od sebe dále udržována a rozvíjena. Hlavní nevýhodou je právě skladba řešení ze speciálně pro tento účel zhotovených jednoúčelových částí. V dosavadní praxi již byly vyvinuty prvky umožňující tvorbu opakovaně použitelných řešení. Jsou to moduly COPY a standardní podprogramy (např. SORT, STRING, INSPECT,...). Dopad jimi využitelných řešení je z hlediska obecnosti analytického řešení relativně nízký.

Principy objektově orientovaného programování OOP

Cílem OOP je vyvinout funkční řešení tvořené znovupoužitelnými vzory. Místo specifického projektu je třeba nejprve promyslet, ze kterých všeobecných problémů se naše úloha skládá. Potom se pro obecné problémy navrhne jedno všeobecně platné řešení jako vzor, který se dá použít i pro řešení specifického problému. Má–li být vzor řešení opravdu všeobecně použitelný, musí být tvořen tak, aby cesta jak problém řešit byla zvenku zcela skrytá. Smí se pouze vědět CO vzor řešení dělá, ale cesta dovnitř JAK zůstane utajena. Definice vzoru obsahuje definici datových struktur ovládaných vzorem a stanovení povolených operací na těchto datových strukturách. Vzniká tak abstraktní datový typ, který svůj obsah vůči okolí skrývá.

Třídy, objekty a zprávy

Vzor řešení obecného problému je v OOP označován jako třída. Synonyma pro třídu jsou model, makro nebo prototyp. Interní funkce třídy označujeme jako metody. Samozřejmě mohou být použity při vzniku a implementaci nových tříd již třídy existující. Místo vytvoření od základu nové třídy lze využít vlastnosti a tedy i funkce jiných tříd. Tento postup pojmenujeme jako odkaz (dědictví) nebo odvození od tříd.

Použití jedné třídy nebo třídy odvozené je jasné. Ve vzoru třídy musíme nahradit všeobecné parametry konkrétními datovými strukturami, které se vztahují k našemu

problému. Vznikají tak útvary označované v OOP jako objekty, někdy také výrazy, instance.

Všeobecně platná strategie řešení je použita na speciální případ. Mnohé objekty jsou odvozeny od týchž tříd, používají tedy stejné algoritmy třídy, tzn. části procedur jsou jen jednou k dispozici v hlavní paměti a jsou společně využívány. Při začátku zpracování objektu se formou zprávy sdělí objektu, které definované funkce se mají provést.

Celé tajemství OOP spočívá v tom, identifikovat správné řešení třídy, definovat je a uvést do vzájemného souladu. Když poznáme a realizujeme potřebné třídy se svými datovými strukturami, je značná část práce na řešení problému hotova. V průběhu času vznikají rozsáhlé knihovny tříd, které se dají víceúčelně používat. Programátor řeší tvorbu, správu a volbu vhodných tříd a stává se objektovým specialistou.

Volba programovacího jazyka

Programovací jazyk musí mít z hlediska využitelnosti OOP tři zásadní schopnosti:

1. Datové a procedurální části programu musí být od sebe jasně odděleny; jen tak lze vícekrát použít všeobecně platný algoritmus ve spolupráci s různými oblastmi definice dat.

2. Části algoritmu musí být vícenásobně použitelné; jen tak lze zajistit, že vždy stejná cesta řešení spadá pro všechny objekty do stejné třídy.

3. Části algoritmu musí být zásadně rekurzivně použitelné; jedna třída může být vytvořena kombinací jiných tříd a je tedy možné, že se v jednom okamžiku provozu úlohy třída sama nepřímo používá.

Další části jazyka jsou pro OOP pouze okrasou, která ulehčuje návrh a samo programování.

Vývoj Cobolu z hlediska OOP

V říjnu 1989 byla založena pracovní skupina CODASYLu, která definovala zadání pro objektový kompilátor. Samozřejmě nevznikne nový jazyk. Na realizaci prvků OOP v Cobolu a přípravě standardu objektového kompilátoru se většinově podílí britská nadnárodní firma Micro Focus. Micro Focus produkuje cobolská vývojová a provozní prostředí aplikací v operačních systémech DOS, OS/2, Windows, UNIX a AIX. Aplikace jsou v binární podobě přenositelné mezi těmito systémy a lze je provozovat v heterogenních sítích. Pracovní skupinou fy Micro Focus Palo Alto v Kalifornii jsou rozpracovávány již existující části jazyka a návrhy pro potřebné funkce OOP. Již dnešní verze MF COBOL v.3.0. obsahuje rozšíření jazyka ukazující způsoby aplikace OOP (příkaz FUNCTION).

Kompilátor COBOL fy Micro Focus vyhovuje výše uvedeným požadavkům na OO jazyk. Firma Micro Focus si uvědomuje, že technologie objektového programování je mezníkem ve vývoji aplikačního programování. Hlavní výhoda OO technologie spočívá ve zkrácení doby vývoje a modifikace aplikací. Metody objektově orientovaného programování umožňují profesionálnímu analytikovi popsat jeho zájem v termínech jemu známých objektů a operacemi na těchto objektech. Potom je tento specifikovaný zájem přenesen do symbolů počítače a následně zpracován.

Například, pracovník pojišťovny může popisovat průběh pojišťování; výpočetní systém, který automatizuje činnost pojišťovny obsahuje objekty pojišťovací politiky, které odpovídají charakteristikou a chováním reálnému světu tohoto popisu. Neschopnost modelovat prostředí, například obchodu, pojišťovnictví, v termínech srozumitelných pro obchodní profesionály vytvořila bariéru mezi těmito odborníky a specialisty výpočetní techniky. Obchodník potom může pouze doplňovat požadavky, místo aby rozhodoval, jak jeho požadavky mají být zpracovány.

Dochází často k tomu, že základní problém řešení je ztracen nebo nebyl vůbec zaznamenán, takže programátor neznalý podrobné analýzy ji musí namáhavě zkoumat a stanovovat kroky řešení. OO technologie poskytuje metodu a nástroje k tomu, jak základní analytický problém a jeho řešení zaznamenat způsobem, který není zpracovatelný pouze počítačem, ale je také pochopitelný oběma profesionály, obchodníkem i programátorem.

Tento proces od obchodníkova přání k jeho implementaci na počítači probíhá bez obtíží, je intuitivní a rychlý.

Cobolský program může pracovat rekurzivně a splňuje normy konvencí sdílení parametrů obecného rozhraní API (Aplication Programming Interface).

Object Oriented Option

Micro Focus produkt Object Oriented Option je tvořen třemi základními komponenty (prostředími) :

- OO sada vývojových nástrojů
- Smalltalk/V PM
- Reusable Code Manager (RCM)

Na začátku používání těchto prostředí jsou nezávislé prostory objektů. To znamená, že nejsou mechanismy pro vyvolání objektu jednoho prostředí přímo z druhého.

Existuje základní komunikační mechanismus mezi COBOLem a Smalltalk, který může být použit buď ve spojení s OO v COBOLu nebo s tradičními cobolskými programy. MF

vyvíjí způsoby, kterými může být tento mechanismus v budoucnu rozšířen a bude umožňovat přímé vyvolávání objektů mezi občma prostředími. Použitím technologie Co–LINK, která je součástí OO Option mohou spolu komunikovat prostředí OS/2 a DOS.

OO sada vývojových nástrojů (SN)

Je to softwarový balík, který lze zakoupit jako součást produktů SMALLTALK/V nebo MF Reusable Code Manager. SN poskytuje možnost vyvíjet OO aplikace v COBOLu právě tak, jako umožňuje používat technologie vývoje aplikací v kombinaci SMALLTALK/V s COBOLem. Jde o klíčový prvek OO Option, který obsahuje MF OO cobolské provozní a vývojové prostředí.

Hlavními nástroji SN jsou:

- OO Run Time Environment (OORTE)
- Makra

OO Ruu Time Environment (OORTE)

OORTE slouží jako rozšíření základního cobolského RTE. OORTE poskytuje služby jako například speciální ovládání paměti a funkce, které vyžaduje objektové programování:

- definice třídy objektu (DTO)
  definuje data a operace pro objekt ve formě cobolského programu
- dědičnost
  DTO může zdědit data a operace od rodičovské třídy objektu
- metoda vyvolání
  program může vyvolávat operace definované pro třídy objektu
  metoda vyvolání podporuje polymorfnost a dědičnost
- situačnost
  schopnost vytvářet nové objekty situací z popisu zadaného v definici třídy, používá se metoda dynamické alokace paměti
- stálost
  všechny objekty provozní jednotky mohou být uloženy a automaticky vyvolány

Makra

OORTE je zpřístupnitelné sadou maker, které se stanou součástí syntaxe Cobolu. Pomocí těchto maker lze definovat nové třídy, vytvářet a vyvolávat instance existujících tříd. Patří sem také sada systémových tříd, které obsahují komponenty popisující obecně používané datové typy jako seznamy, bytové streamy a sety. Sada systémových tříd je

určena pro uživatele, kteří chtějí dodávat další objektové schopnosti kombinovaným prostředím Smalltalk/V a Cobol Workbench.

OO programování v COBOLu je primárně vyvoláno užitím existujících cobolských příkazů. K dispozici je však i malý počet nových příkazů, které zpřístupňují služby OORTE. Tyto příkazy jsou uvedeny ve formě maker. Příklady těchto maker jsou:

- CALL-OBJECT
- CLASS-DATA
- CLASS-METHOD
- INSTANCE-DATA
- INSTANCE-METHOD
- EXIT-METHOD
- OBJECT-CLASS

Tato makra již byla definována a jsou součástí OO Option. Nástroj definice makra (musí být k dispozici při kompilaci) je REUSABLE CODE MANAGER (RCM). RCM je užitečný nástroj pro tvorbu a řízení knihoven opakovaně použitelných procedur. RCM má speciální výbavu, umožňující vytvoření cobolských maker. RCM makra, která jsou tvořena příkazy COBOLu ANS85 definují nové třídy, vytvářejí situace existujících tříd a vyvolávají je.

Příklad definice třídy OO COBOL

Pozn. Pořadové číslo vpravo zdrojového text'u popisuje jednotlivé body příkladu (makra).

```
IDENTIFICATION DIVISION.
CLASS-ID PhoneBK INHERITS FROM Object.                    1.
.
.
LINKAGE SECTION.
.
.
INSTANCE-DATA „PHONEBK.INS".                              2.
.
.
PROCEDURE DIVISION.
        CLASS-METHOD „NEW".                               3.
        CALL-OBJECT SUPER „NEW" RETURNING aBook
        CALL-OBJECT aBook „INITIALISE"
        EXIT-METHOD RETURNING aBook.
```

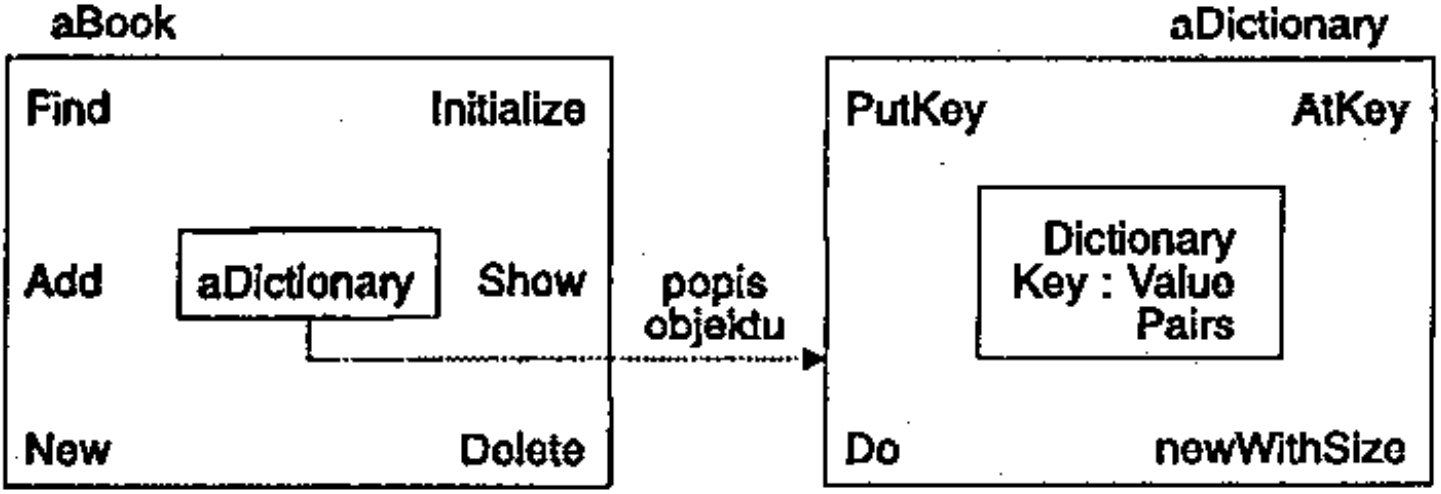
```
.
INSTANCE-METHOD „initialise".                                    4.
        CALL-OBJECT CDictionary „newWithsize"                    5.
                USING initialSize
                RETURNING aDictionary
        EXIT-METHOD.
        .
        .
INSTANCE-METHOD „add"
                USING entryName
                entryNumber.
                .
                .
```

Makro 1 deklaruje program definice třídy. Definované jméno třídy je PhoneBK a tato třída zdědila atributy a metody od třídy Object.

Makro 2 popisuje proměnné situace pro třídu PhoneBk. Tyto proměnné jsou přístupné všem metodám, definovaným v této definici třídy.

Makro 3 definuje metodu třídy „NEW". Tato metoda odpovídá za dynamickou alokaci paměti pro novou situaci třídy PhoneBk.

Makro 4 deklaruje metodu situace „Initialise". Tato metoda způsobí vytvoření situace třídy CDictionary a popis objektu nového slovníku, který bude umístěn do proměnné situace pro třídu PhoneBk. Všechny tyto činnosti jsou vyvolány makrem 5 CALL-OBJECT, které posílá zprávu: „newWithsize" do CDictionary.



Obr. 1

Obrázek 1 ukazuje vztah objektů aBook a aDictionary. Objekt aBook je situací třídy PhoneBk. Podobně objekt aDictionary je situací třídy Dictionary. Oba objekty jsou propojeny. Je ukázána schopnost jednoho objektu vytvořit popis do jiných objektů.

Smalltalk V/PM

Smalltalk je „čistý" objektově orientovaný jazyk specificky určený pro spolupráci s objekty a poskytující vhodné prostředí pro výuku a experimentování s objekty. MF řešení umožňuje programátorovi psát programy pomocí Smalltalk/V PM a používat OO sadu vývojových nástrojů MF, které rozšiřují knihovnu tříd Smalltalku/V pro vytváření komunikačních objektů. Tyto objekty dovolují programům ve Smalltalk/V a Cobolu spolupracovat v režimu klient / server. V současné době jsou prostředí Smalltalku a Cobolu vzájemně nezávislá, mohou si vyměňovat informace, ale neexistuje komunikační spojení mezi oběma typy objektů.

Schopnost výměny informací znamená, že aplikace Smalltalk mohou komunikovat s cobolskými programy. V cobolském programu je možno nahradit část produkující uživatelské rozhraní sekcí napsa– nou ve Smalltalku/V PM. Micro Focus nyní zpracovává komunikaci typu object–to–object mezi objekty Cobolu, Smalltalku/V PM a C++.

Micro Focus vyvinul rozšíření do verze DIGITALK SMALLTALK/V a je oficiálním distributorem tohoto produktu. Tato rozšíření umožňují komunikaci cobolského programu s aplikací SMALLTALK a nazývají se CO-LINK/V.

Prostředky CO–LINK/V zahrnují následující funkce:

- vytváření a údržba kanálů pro komunikaci CLIENT–SERVER
- zaznamenávání událostí (směrů těchto událostí) pro jejich potřebné uživatelské zpracování
- zpracování chybových podmínek
- umožnění asynchronního zpracování

CO-LINK/V využívá další nástroj produktu MF COBOL Workbench nazývaný CCI (Common Communication Interface). CCI zajišťuje základní rozhraní pro mnoho populárních síťových a dalších komunikačních protokolů.

Příklady OO

Součástí kombinovaného prostředí COBOL – SMALLTALK jsou dva příklady užití, jak systému SMALLTALK, tak CO–LINK/V a COBOLu. Jeden příklad se nazývá PROBE, druhý COBOL File Reader.

PROBE

PROBE je prostředek pro monitorování zdrojů aplikace a pro studování struktury aplikace na úrovni modulů. PROBE je složka MF COBOL Workbench.

Zdrojový kód SMALLTALK je dodáván spolu s OO Option a názorně ukazuje užití tříd, vytvořených pro řízení CCI. I tyto třídy jsou součástí OO Option.

COBOL File Reader

Zatímco PROBE je užitečný jako nástroj analýzy, tento příklad je značně jednodušší. Cobolský program uvede do provozu zapojení CCI a čeká. Aplikace SMALLTALK otevírá okno a ptá se obsluhy na jméno prohlíženého sekvenčního souboru. Potom SMALLTALK uvádí do provozu komunikaci s cobolským programem, posílá požadavek na čtení a přijímá zpět síťovou linkou věty ve standardním textovém okně SMALLTALK.

Zdrojový kód prvků SMALLTALK/V a COBOL pro COBOL File Reader jsou součástí OO Option.

Některé důležité prvky OO Option

Knihovna tříd

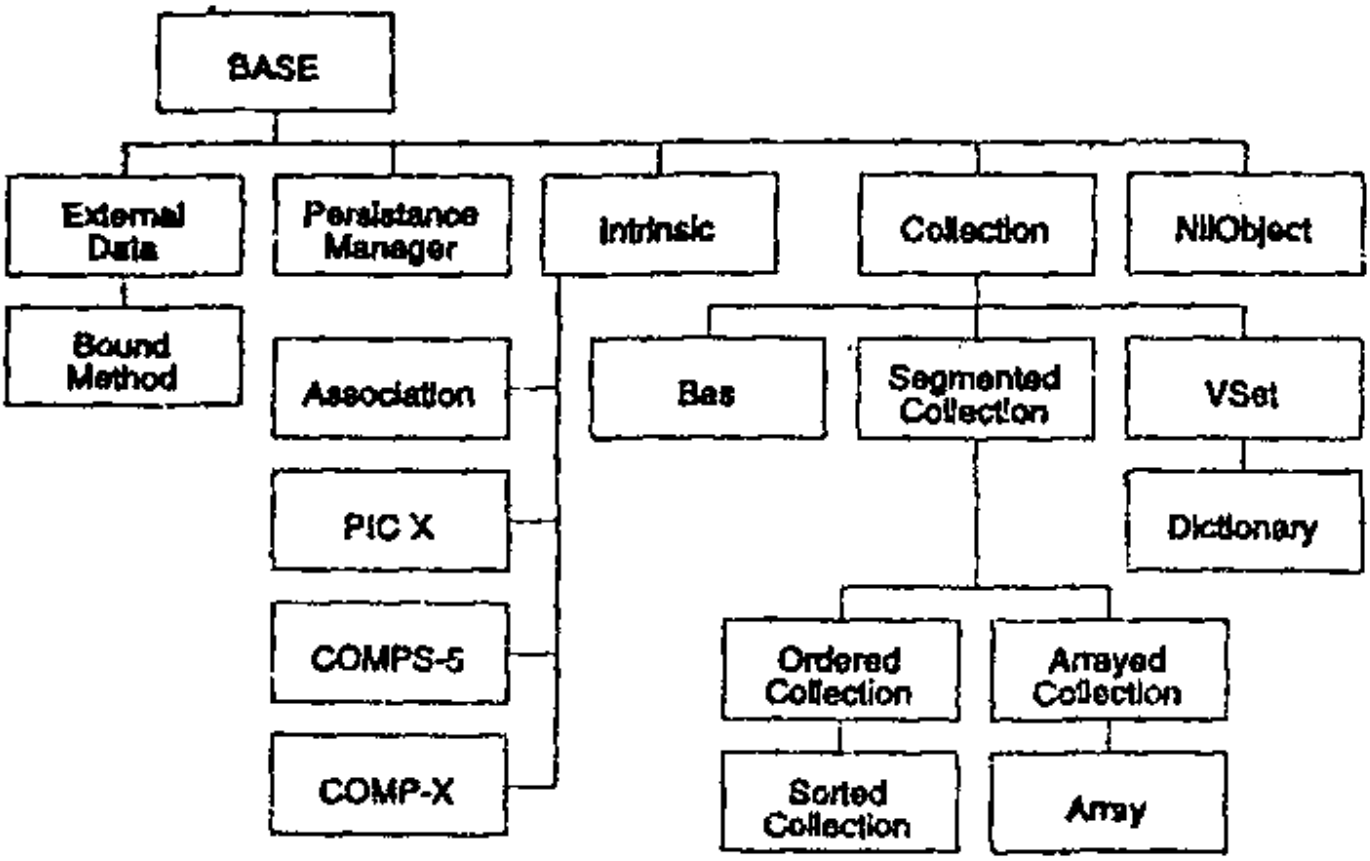
Schopnost definice tříd objektů je základem objektově orientovaného programování. Diagram knihovny tříd je uveden na obrázku 2.

- BASE – hlavní řídící a organizační třída
- External data – speciální systémy tříd
- Bound Method –
- Persistance Manager –
- Nil Object –
- Intrinsic – třídy popisující základní cobolské datové typy
- Collection – třídy pro vzájemné propojování objektů

Technické informace

Software OO programování pracuje v prostředí:

- DOS 3.3 a vyšší;
- OS/2 v.1.2 a vyšší;
- SMALLTALK/V v.1.3 nebo 1.4
- SMALLTALK/V WINDOWS v.1.1;



Obr. 2

- Reusable Code Manager v.1.5.21 a vyšší
- Při spuštění v DOS je vyžadován ovladač externí paměti XM a min. 2MB RAM.
- SMALLTALK/V WINDOWS pracuje s WINDOWS 3.0 a vyšší
- SMALLTALK/V PM pracuje s OS/2 v.1.2 a vyšší
- MF COBOL WORKBENCH je podmíněným produktem práce OO Option ve všech prostředích.

Hardware:

- IBM PC XT, AT, PS/2 a vyšší kompatibilní, doporučený procesor 80286 a vyšší,
- 640kB pro DOS, 2MB spolu s XM je doporučeno,
- 8MB pro OS/2, 4MB HD pro odkladové soubory,
- 40MB HD

Reusable Code Manager (RCM)

RCM, nadstavbový produkt MF COBOL WORKBENCH, je prostředek pro vývoj víceúčelových cobolských maker a programových modelů. Umožňuje vývojářům aplikací v COBOLu převádět odzkoušené existující cobolské procedury do flexibilních znovupoužitelných modulů. Tyto procedurální složky mohou být sestaveny do zcela nových aplikací nebo použity pro rozšíření existujících cobolských programů. Idea opakovaně použitelného kódu není nová. Již dříve bylo provedeno mnoho pokusů pro realizaci výhody znovupoužitelného kódu, například použitím členů COPY a podprogramů nebo sestavení důmyslných generátorů zdrojového textu. Tato řešení však nebyla kompletní, COPY členy nebyly příliš flexibilní, podprogramy mají často komplexní rozhraní, která jsou nedostatečně dokumentována a obtížně testovatelná. Generátory kódu mají zase limitovaný rozsah paměti. RCM řeší tyto problémy a nabízí programátorům maximum opakovaného použití.

Micro Focus využil RCM pro vývoj sady maker, které umožňují přístup k OO RTE. Pomocí těchto maker mohou programátoři vytvářet, inicializovat a vyvolávat objekty.

Makra jsou kompilována RCM preprocesorem do segmentů procedur, které vyvojávají OO RTE. Kombinování RCM a OO sady vývojových nástrojů dovoluje produkovat objekty obsahující aktuální cobolský kód a vyvolávat procedury v nových nebo existujících programech.

S RCM je možno:

- vytvářet nová flexibilní parametrizovatelná cobolská makra pro rutinní úlohy
- převádět ucelené části kódů (program v COBOLu, definice obrazovky, dávky JCL ...) do parametrizovaných modulů, které lze použít pro vytváření mnoha podobných, ale unifikovaných částí programů
- zpracovávat kompletní přehled opakovaně použitelných přínosů

S RCM lze odstranit mnoho nudných, únavných, k chybám náchylných programovacích operací použitím maker a modulů, které provádějí opakovanou činnost bez chyb.

Plně dokumentovatelné, znovupoužitelné moduly jsou přístupné okamžitě použitím funkčních kláves během editace programu. RCM byl vyvinut speciálně pro cobolské programátory, kteří používají MF COBOL WORKBENCH. RCM má rozhraní velmi podobné tomuto produktu a je integrován s Editorem, Checkerem a Animatorem. RCM pokrývá stejně široký okruh prostředí jako MF COBOL WB, aplikace střediskových počítačů i PC, vývoj nových a rozvoj stávajících programů.

Vývoj nových aplikací s RCM

Po pečlivém rozboru mohou být velké části nových aplikací vytvářeny automaticky pomocí maker RCM a modelových souborů. S pomocí RCM mohou být tvořena makra pro všechny běžné funkce. Tato makra lze skládat tak, že makra nižší tvoří makra vyšší úrovně. Modelové programy, ve kterých mohou být vložena makra, lze použít ke generování mnoha kompletních bezchybných programů, popisů vět, obrazových formátů atd. s podobnou funkčností, ale pracujících na různých datových strukturách.

Rozvoj existujících aplikací

Na rozdíl od jiných produktů, které poskytují znovupoužitelný kód, může být RCM použit v již existujících cobolských aplikacích nebo jako samostatný program v provozní jednotce. Libovolný cobolský program, udržovaný použitím MF COBOL WB, může čerpat z výhod těchto funkcí RCM:

- RCM dovoluje pokračovat v použití libovolné procedury užité programátorem ve zbytku programu
- RCM umožňuje pokračovat v použití volné formy cobolské syntaxe
- RCM umožňuje použít libovolný prostředek pro kreslení obrazovek
- RCM umožňuje rychlé zapojení existujících COPY členů a podprogramů do své kolekce znovupoužitelných prostředků

Konverze existujících aplikací za použití parametrizovatelných modelů

RCM lze používat pro transformaci aplikací následujících typů:

- z IMS (Information Manager System) do DB2 DDL
- z RPG do Cobolu
- z IMS nebo BMS definic obrazovek do Dialog Systému (produkt MF)
- při změně struktury vět souborů

Inventarizace prvků RCM

Prostředek LOCATOR/SELECTOR zajišťuje online inventarizaci všech prvků RCM (COPY členy, podprogramy, makra a modely) během editace programu. Jeho schopnosti jsou vyhledávání podle typu nebo klíčového slova, prohlížení dokumentace, vkládání a editování vstupních parametrů a vkládání kompletních zdrojových příkazů do programu. Prvky znovupoužitelného kódu lze vkládat do knihoven, kde mohou být sdíleny během práce programátorů v síti.

Vývoj modulárních znovupoužitelných cobolských maker

Prostředek COBOL MACRO umožňuje programátorům rozšiřovat syntaxi Cobolu vytvářením nových cobolských slov pro běžné programovací funkce. Například slovo

„CENTER" vystřeďuje text v datové položce nebo slovo „LOG-ERROR" je možno použít pro vyvolání podprogramu pro lokalizaci chyby. Tato makra dělají programování v COBOLu mnohem efektivnější, protože nabízejí programátorům schopnost vytvářet, kódovat a testovat aplikace na vyšší úrovni. Jediné makro může provádět práci mnoha řádek cobolského zdrojového textu. RCM obsahuje integrovaný makroprocesor, který umožňuje animovat program na úrovni maker. RCM rovněž obsahuje velkou sadu hotových maker, která umožňují:

- výhodnou definici datových struktur nebo COPY členů
- funkce zobrazování textových řetězců, které nejsou podporovány přímo standardní syntaxí
- operace s tabulkami
- rychlý přístup k MF systémovým podprogramům (call-by-name, call-by-number), které jsou součástí Run Time Systému
- příklady jiných užitečných maker

Zapojení COBOLu do znovupoužitelného parametrizovatelného kódu

Prostředek RCM Modeling poskytuje schopnost transformace standardně vytvořeného programu (nebo jiného textového souboru) do parametrizovaného modelu a následnou generaci mnoha nových programů z tohoto modelu. Tyto modelové soubory slouží jako základ pro konstrukci nových programů – jsou vytvořeny s maximální znovupoužitelností, která umožňuje tvorbu mnoha podobných, ale nikoli identických programů (nebo souborů) z jednoho modelu. Modely jsou zpřístupněny během editace prostředkem Locator/Selector. Modelové soubory mohou být využity pro generování ASCII textových souborů s maximální délkou věty 80 znaků. Dále jsou uvedeny příklady souborů, které lze generovat:

- COPY členy pro popisy vět formátů obrazovek
- soubory IMPORT pro DIALOG SYSTEM
- MFS kód pro definice obrazovek systému IMS
- kód SQL pro definici dat a jejich zpracování
- dávky JCL.

Modely mohou obsahovat výstupní body pro dodatečný unifikovaný kód. Následně může být model změněn a každá situace modelu může být regenerována libovolným unifikovaným kódem rozšiřujícím funkci modelu.

Modely používají jako vstup neomezené ASCII textové soubory. Z toho plyne, že jako vstup do RCM lze použít exportní soubory libovolného návrhového nástroje (i tvaru věty souboru).

Technické informace

Operační systém:

- RCM verze 1.5 pracuje pod:
- DOS 3.30 a vyšší (i s MS WINDOWS 3.0)
- OS/2 v.1.2 a vyšší (SE i EE)

Hardware:

- IBM AT, PS/2 nebo kompatibilní (min. procesor 80286)
- min. 2MB paměti pro DOS
- min. 6MB pro OS/2
- 2MB HD

Kompatibilní software:

- RCM vyžaduje buď:
  - MF COBOL WB verze 2.4 a vyšší nebo
  - MF TOOLSET verze 2.4 a vyšší nebo
  - MF Profesional COBOL
- RCM může plně využívat externí paměti pod OS/2 nebo XM (DOS)

Autor: Ing. Jiří Veselý
MF Servis
Veverkova 1631
500 02 Hradec Králové