VÁDEMÉKUM OBJEKTOVĚ ORIENTOVANÉ TECHNOLOGIE

Branislav Lacko

Object oriented...

Objektově orientované programování představuje dnes pojem, který doslova hýbe počítačovým světem. Jistě po zásluze!

V případě objektově orientovaného přístupu totiž výjimečně softwarové myšlenky předběhly hardwarové možnosti, když ve většině případů technické prostředky výpočetní techniky jsou hybnou silou vývoje.

Všeobecně se mluví o objektově orientovaném programování jako o moderní metodě návrhu programů a jejich realizaci. Je to způsobeno tím, že existence programovacích jazyků, které využívají myšlenek objektově orientovaných, byla impulzem k zavádění této nové metody a tím k její prezentaci na veřejnosti.

Je však třeba hned v úvodu zdůraznit, že je správnější hovořit o objektově orientované technologii, která v sobě zahrnuje celou škálu aplikací objektově orientovaného přístupu. Objektově orientovaná technologie zahrnuje především objektově orientovanou analýzu, která provádí analýzu problémů na základě objektově orientovaného přístupu. Dále objektově orientovaný návrh programů a programových systémů, které jsou pak realizovány objektově orientovaným programováním prostřednictvím objektově orientovaných jazyků. Výsledkem aplikací objektově orientované technologie jsou objektově orientované produkty, které mají své charakteristické vlastnosti. Obecné principy objektově orientovaného přístupu lze různým způsobem konkretizovat do určitých objektově orientovaných metod, které představují plánovitou a systematickou aplikaci těchto principů pro řešení určitých problémů. Rozborem metod, jejich porovnáváním a hodnocením se zabývá objektově orientovaná metodologie. V neposlední řadě můžeme hovořit o objektově orientovaných technikách, které představují určité dílčí vyřešení, ať již implementace programové nebo analytické činnosti, v rámci objektově orientovaných metod.

Uvedený přehled metod by měl postačit čtenářům k lepší orientaci v objektově orientované technologii a umožnit jim zařadit některá nová označení programových produktů (např. objektově orientované databáze), případně další pojmy, do souvislosti s touto technologií.

Poznamenejme, že široké přijetí objektově orientované technologie uživateli počítačů a bezprostřední nadšení pro tuto technologii způsobilo, že dnes řada producentů programových produktů a jejich dodavatelů využívá této skutečnosti a na prodejní artikly umísťuje nálepky „OBJECT ORIENTED" ve snaze zvýšit prodejnost svých výrobků bez opravdové souvislosti s principy objektově orientované technologie nebo jen s jejich povrchní implementací.

Trochu historie

Objektově orientovaná technologie zaznamenala své široké rozšíření právě v současné době, ale základní myšlenky byly obsaženy již v jazyku SIMULA 67, do kterého byly zabudovány v letech 1964–1966, i když v tomto případě nebyla použita dnes všeobecně přijímaná objektově orientovaná terminologie /1/. Bohužel se jazyk SIMULA 67 přes své nesporné přednosti nijak výrazně neprosadil, a tak i tyto progresivní myšlenky byly pozapomenuty.

Vrátili se k nim až tvůrci jazyka SMALLTALK, který vznikl v průběhu 70–tých let ve výzkumném středisku Palo Alto firmy XEROX v USA. Výsledkem jejich snažení byl jazyk SMALLTALK 80, který byl představen veřejnosti v letech 1981–1984 /2/. Kromě implementace objektově orientované technologie se autorům podařilo vytvořit velmi kompaktní integrované prostředí pro vývoj aplikací.

Všeobecnému rozšíření jazyka SMALLTALK však zabránily zřejmě dvě skutečnosti:

- zjevné ignorování otázky strojové efektivnosti v době značně vzdálené dnešní, která plně opodstatňuje správnost takového přístupu.
- odlišný přístup k vytváření programových produktů ve srovnání s klasickými procedurálními jazyky, ve kterých byli programátoři zvyklí programovat.

Proto až obohacení jazyka TURBO Pascal ve verzi 5.5, se kterým přišla firma Borland na jaře 1989, napomohlo opravdu širokému rozšíření objektově orientovanému programování a otevřelo cestu k prosazení objektově orientované technologie. Připojení jazykových prvků ke známému jazyku Pascal, které umožňují aplikovat principy objektově orientovaného přístupu, znamenalo pro mnoho programátorů potřebný most, po kterém by mohli přejít od současných programovacích technik strukturovaného programování do světa objektově orientované technologie. Vše bylo ještě více podpořeno skutečností, že se objevily další produkty, které využily stejného přístupu, např. rozšíření jazyka Quick Pascal firmou Microsoft a rozšíření jazyka C o prostředky umožňující objektově orientované programování od firmy Borland, čímž vznikl jazyk C++. Tím k pascalovsky orientovaným programátorům se přidala navíc dost široká rodina programátorů znalých jazyka C.

V roce 1990 na trh přišla firma Borland s verzí Turbo Pascal 6.0 obohacenou o unit Turbo Vision, který představuje standardizovanou knihovnu objektů, které programátor

může využívat ve své praxi. Současně firma Borland uvolnila svůj produkt Object Vision, což je prostředek, který umožňuje uživatelům neprogramátorům snadno vytvářet aplikace pro Windows orientované na sběr dat, jejich zpracování a ukládání do databází.

Velký význam pro další rozvoj objektově orientované technologie měly stále se zvyšující technické parametry mikropočítačů založených na výkonných mikroprocesorech i 80386 a i 80486, které odstranily technickou bariéru pro praktické nasazení jazyka SMALLTALK a dalších objektově orientovaných jazyků, např. Eiffel, Oberon, Actor a jiné.

Nebylo by však správné prezentovat rozšíření objektově orientované technologie jako pouhou odezvou na úspěšné produkty kompilátorů programovacích jazyků.

Daleko silnějším impulsem pro všeobecné přijetí a rozšíření objektově orientované technologie byla a je potřeba zvýšení programátorské produktivity a snížení potřeby na údržbu provozovaných programů.

Strukturované programování jako první ucelená metoda návrhu programů znamenala mnoho na cestě od intuitivních postupů programování k profesionálnímu programátorskému řemeslu. Přesto však nemohla odstranit vzniklou a dále se prohlubující softwarovou krizi.

Objektově orientovaná technologie je však v důsledku vlastností polymorfismu a virtuálních metod schopna tvořit stavebnicové moduly, které je možno doplňovat i bez znalosti jejich zdrojového kódu. Často se v této souvislosti hovoří o softwarových čipech. Ty dovolují daleko více zhodnotit a zužitkovat vynaloženou programátorskou práci lépe než kdy před tím.

Zvýšení opakované využitelnosti objektově orientovaně vytvořených produktů je jedním z nejvýznamnějších přínosů objektově orientované technologie.

Cíle objektově orientované technologie

Objektově orientovaná technologie umožňuje návrh a tvorbu programových produktů tak, aby řešitel se mohl věnovat odborné problematice řešeného problému. Přitom dává přednost využití již dosud vytvořených produktů před jejich návrhem od samého začátku.

Vytvořené produkty jsou nekomplikované, spolehlivé, snadno srozumitelné a jednoduše opakovaně použitelné.

Použitím objektově orientované technologie dochází ke zvýšení produktivity při návrhu a tvorbě programových produktů, a to cestou abstrakce a parametrizace. Zároveň se redukují náklady na opravy a na provádění změn, tj. na údržbu programových produktů.

Základní principy a pojmy

Objektově orientovaná technologie vychází z abstrakce reálného světa, ze kterého vyděluje objekty, které jí slouží k lepšímu pochopení reálného světa a k jeho modelování. Analogicky k objektům v reálném světě zavádí i vzájemné vztahy mezi abstraktními objekty.

Základním pojmem objektově orientované technologie je objekt.

Uživatel však musí porozumět i dalším odborným pojmům, na kterých je objektově orientovaná technologie založena, chce-li tuto technologii úspěšně používat. Je to nutná podmínka pro pochopení některých netradičních přístupů k programování i analýze a k překonání prvotního dojmu zdánlivé komplikovanosti.

Objekt (angl. OBJECT) v objektově orientované technologii je chápán, jak už bylo řečeno, jako prostředek k pochopení reálného světa a k jeho modelování. V počítači je realizován jako abstraktní datová struktura. Každý objekt má svůj název odlišný od jmen jiných objektů a atributy, které ho charakterizují (např. objekt PRACOVNÍK může mít atributy: osobní číslo, příjmení a jméno, datum narození, bydliště, stav, adresa posledního zaměstnavatele). Atributy objektu nejen charakterizují jeho vlastnosti, ale vypovídají o stavech objektu, případně popisují historii průběhu jeho aktivity.

Koncepce abstraktních datových struktur je však již delší dobu známa a používána ve strukturovaném programování a datové modelování s nimi běžně pracuje. Nový přístup objektově orientované technologie spočívá v tom, že pojem objekt zahrnuje také činnosti, které jsou s objektem svázány. V praxi je objekt vytvořen strukturou datových položek různých typů a činnosti jsou realizovány procedurami, resp. funkcemi. Toto spojení datových struktur s algoritmy nazýváme zapouzdření (angl. ENCAPSULATION) a činnosti zapouzdřené do objektu označujeme jako metody (angl. METHODS). Například k již zmíněnému objektu PRACOVNÍK bychom připojili procedury zajišťující jeho evidenci při nástupu do organizace, provádění změn v osobních údajích zajišťující administrativní vyřízení při ukončení pracovního poměru atd.

Objekty se společnými vlastnostmi sdružujeme do tzv. tříd (angl. CLASSES), což nám umožňuje přehledně uspořádat všechny objekty do určitých skupin.

Definice objektu, podobně jako definice typu proměnné, pouze popisuje druh objektu a jeho vlastnosti ve vztahu k jiným objektům. Konkrétní výskyt příslušného druhu objektu se nazývá instance. Například instance: 85324, Polák Josef, 20.6.1943, Praha, Nádražní 20, svobodný, GRAFIA Brno, představuje jednu instanci objektu PRACOVNÍK.

Instance mohou mezi sebou spolupracovat tím, že si vyměňují zprávy, kterými si sdělují různé žádosti na provedení potřebných akcí instancí jiného objektu. Zprávy figurují jako elementární kroky a zároveň zajišťují návaznost jednotlivých kroků, aby se dosáhlo požadovaných složitých činností. Například instance objektu VEDOUCÍ určitého úseku dá pokyn k ukončení pracovního poměru určitého pracovníka. Při ukončování pracovního poměru požádá instance tohoto pracovníka instanci objektu ÚČETNÍ o zúčtování zbývající dovolené atp.

Sdružování objektů, které mají podobné atributy a chování, do tříd nám však neslouží pouze k utřídění objektů. Existence tříd nám umožňuje zavést mechanismus dědictví (angl. INHERITANCE), kdy nový objekt určité třídy dědí všechny vlastnosti této třídy. Například při definici objektů ÚČETNÍ, VEDOUCÍ nemusíme znovu definovat, že každý z nich má své osobní číslo, příjmení a jméno atd. Využijeme objektu PRACOVNÍK a obohatíme ho o atributy a metody, které zde musí být navíc (označení útvaru, který je vedoucím veden, popisem prováděných činností atd.). Můžeme tedy hovořit o rodičovském objektu (angl. PARENT) a o odvozeném objektu (angl. DESCENDANT), který dědí (angl. INHERITS) všechny atributy a metody svého předka (angl. ANCESTOR). Odvozený potomek přitom může být rodičem pro další vztah.

Takto vybudovaná hierarchie vztahů, která připomíná příbuzenské vztahy v rodokmenech, nejen šetří výrazové prostředky (co se dědí je nutno znovu uvádět), ale slouží i jako prostředek postupného poznávání reálného světa a jeho modelování. Umožňuje od všeobecných vlastností a základních činností postupně zpřesňovat charakteristiku a působení jednotlivých objektů a instancí, jak to ostatně skutečně probíhá v praxi. Tato vlastnost činí objektově orientovanou technologii zvlášť vhodnou pro budování bází znalostí v expertních systémech.

Mechanismus zděděných vlastností můžeme i porušit a určitou instanci individualizovat. Stane se to tehdy, když jí a pouze jí některé atributy předáme nebo některé zděděné zrušíme.

Pro objektově orientovanou technologii je důležité, že máme možnost přiřadit jedno jméno akci, které je sdíleno celou hierarchií objektů, přičemž však každá úroveň má možnost přizpůsobit konkrétní provedení akce svým specifickým potřebám a podmínkám. Toto označujeme pojmem mnohotvárnost (POLYMORPHISM) – polymorfismus.

Polymorfismus přináší objektově orientované technologii právě onu možnost univerzálnosti v používání objektů, která dosud nebyla programátorům k dispozici. Pokud chtěli vytvořit univerzální proceduru pro široké použití, mohli využít jen myšlenky parametrizace. Ta však měla svá omezení. Volání procedury muselo respektovat počet, pořadí a druh formálních a skutečných parametrů. Proto programátor nemohl např. v jazyku Pascal vytvořit obdobu takových procedur jako write a read s proměnlivým počtem

parametrů, s různými typy dat a s jejich libovolným střídáním. S možnostmi, které dává polymorfismus programátorovi, snad může soutěžit jen koncepce generických modulů v jazyku Ada.

Překrývání metod, které objektově orientovaná technologie dovoluje, je realizováno dvojím způsobem: statickým a virtuálním.

Rozdíl mezi voláním statických a virtuálních metod je důsledkem mezi rozhodnutím okamžitým a rozhodnutím odloženým. Pokud použijeme volání statické metody, říkáme v podstatě překladači: „Víš, kterou metodu chci provést. Zavolej ji!" Volání virtuální metody však znamená: „Zatím nevíš, kterou metodu chci volat. Až přijde čas volání, dozvíš se od instance její adresu!".

Objektově orientovaná analýza a návrh

Všeobecné přijetí Turbo Pascalu ve verzích 5.5 a 6.0 a popularita jazyka C++ poskytly potenciální možnosti využití principů objektově orientované technologie v praxi. Přesto výsledky tohoto snažení nejsou nijak zjevné. Lze vyslovit domněnku, že důvodem je přecenění programátorského přístupu k realizaci aplikací.

Má-li objektově orientované programování přinést očekávaný efekt, musí mu předcházet objektově orientovaná analýza.

Klasický přístup používání jazyka Pascal nebo C spočíval v aplikaci metod strukturovaného programování. Existuje reálné nebezpečí, že uživatel těchto jazyků aplikuje principy strukturovaného programování na program, ve kterém chce využít prostředků objektově orientovaného programování. Výsledky takového postupu nebudou uspokojivé.

Přístupy objektově orientované analýzy jsou v mnoha směrech jiné, často protikladné ve srovnání se strukturovanou analýzou.

Připomeňme si základní myšlenky strukturované analýzy /viz 15, 16/, které mají vztah k diskutované problematice:

- Princip návrhu shora dolů /TOP DOWN/
- Co nejjednodušší kombinování tří základních struktur (sekvence, větvení, cyklus)
- Strukturalizace dat v návaznosti na jednotlivé úrovně dekompozice systému shora dolů (soubor, věta, skupina, položka)
- Postupné zjemňování funkcí navrhovaného systému

Proti tomu je objektově orientovaná analýza charakterizována pěti základními otázkami, jejichž řešení je výsledkem této fáze navrhování systému:

1. Jaké objekty z reálného světa vydělit?
2. Co za atributy těmto objektům přiřadit?
3. Do jakých tříd objekty uspořádat?
4. Které metody u objektů realizovat?
5. Čím zajistit opakovatelnou využitelnost objektů?

Proto objektově orientovaná analýza vychází spíše z inventarizace objektů a činností v reálném systému, aby relevantní z nich vyčlenila jako objekty dalších úvah a seskupila je následně do tříd. Aplikuje se zde tedy princip návrhu zdola nahoru (BOTTOM UP).

Zatímco strukturovaná analýza definuje nejprve globální funkce systému, které postupně zjemňuje dekompozicí na činnosti systému, ze kterých usuzuje na potřebná data, objektově orientovaná analýza vychází z datové analýzy, protože základem objektů jsou datové struktury. Vychází se z poznatku, že v reálných systémech jsou datové struktury daleko stabilnější než funkce, které často podléhají změnám.

Ve strukturovaném návrhu byla data strukturována do hierarchického stromu (typickým produktem byla struktura dat pro jazyk COBOL nebo PL/1). Atributy v objektech mají v podstatě lineární charakter a jsou od ostatních izolovány.

Hierarchická výstavba objektů uvnitř tříd v objektově orientovaném návrhu je narušena možností organizovat objekty také způsobem MASTER – SLAVE (resp. CLIENT – SERVER), který nemá analogii v principech strukturovaného návrhu.

Následující tabulka přehledně shrnuje základní rozdíly přístupů.

	Strukturovaná analýza	Objektově orientovaná analýza
Metoda postupu	TOP DOWN	BOTTOM UP
Způsob analýzy	Funkční analýza	Datová analýza
	Strukturovaný návrh	Objektově orientovaný návrh
Datové struktury	Hierarchické	Lineární
Řídící struktury	Rekurzivně řazené standardní řídící struktury	Interaktivně vázané moduly ve strukturách

V příspěvku na semináři PROGRAMOVÁNÍ 91 uvedl doc. Polák /17/ ukázku, jak se liší výsledky strukturované analýzy od objektově orientované analýzy jednoho a téhož problému.

Byl analyzován problém implementace jednoduché akční počítačové hry, kdy hráč odráží míček od stěny.

Zatímco strukturovaná analýza vedla k vytvoření hierarchické výstavby modulů zajišťujících postupně funkce: zahájení hry, údery hráče, odrážení míčku od stěny atd., výsledky objektově orientované analýzy byly objekty: stěna, míček, hráč, které měly své atributy a metody. Následně byla dokumentována pracnost úpravy této hry při její změně na dva hráče střídavě odrážející míček od stěny. Pracnost změn v objektově orientované analýze byla mnohem menší.

Konstatování nutnosti provedení objektově orientované analýzy a návrhu před objektově orientovaným programováním je nutno podpořit konkrétními metodami.

Paní Elizabet Gibson, pracovnice firmy Parc Place System (vedoucí firmy v USA v objektově orientované technologii), publikovala svoji metodu „Object Behavior Analysis" v časopise BYTE October 1990, která analýzu a návrh pro objektově orientované programování realizuje v pěti krocích:

1. Pochopení aplikace a identifikace chování systému
2. Vydělení objektů z hlediska chování systému
3. Klasifikace objektů
4. Identifikace vztahů mezi objekty
5. Modelování procesů

Podrobný popis použitých technik v jednotlivých krocích, detailní popis činností v jednotlivých krocích by přesáhl rámec tohoto příspěvku a zájemce je najde v originále nebo v příspěvku autora tohoto článku na semináři DATASEM 91 v Brně /14/.

Jinou metodu předvedl V. Railich z Wayne State University na mezinárodní konferenci o objektově orientované technologii

East Eur OOP'91 v září v Bratislavě. Metoda Decomposition Generalization Methodology je založena na iteračním procesu a skládá se ze dvou kroků:

1. Dekompozice
2. Generalizace

Je variantou metody Stepwise Refiement prof. N. Wirtha.

Objektově orientované programování

Je samozřejmé, že realizace principů objektově orientovaného přístupu není možná v podmínkách klasických programovacích jazyků.

Realizace závěrů objektově orientované analýzy je v podstatě možná třemi skupinami objektově orientovaných programovacích prostředků.

První skupinu tvoří programovací jazyky vyvinuté a realizované plně na principech objektově orientované technologie. Typickým představitelem takových jazyků je jazyk SMALLTALK. Jsou známé další jazyky, které byly vyvinuty v rámci experimentálních projektů zaměřených na vývoj a rozvoj objektově orientované technologie, jako např. Eiffel, EUCLID, BETA, ACTOR, LOGLAN, FLAVORS a jiné. Společnou charakteristikou těchto jazyků je, že v plném rozsahu jsou jejich principy práce, výstavba jazyka a jeho použití odvozeny z principů objektově orientovaného přístupu. Tyto jazyky bývají označovány jako „skutečné objektově orientované jazyky" (angl. true object oriented languages). Poskytují obvykle kompletní integrované prostředí pro vývoj aplikací. Jejich použití vyžaduje dokonalé zvládnutí objektově orientované technologie, důkladné seznámení se základy jazyka a jeho používáním. Praktické využití požaduje přiměřeně dostatečné technické prostředky (výpočetní mohutnost, velikost operační paměti).

Zájemcům o první seznámení s jazykem SMALLTALK lze doporučit publikaci J. Weintergera a kol. /11/, kde je popsán úvod do jazyka SMALLTALK. Zájemcům o použití jazyka SMALLTALK doporučujeme navázat kontakt s firmou ArtIn Apple S (Kremelská 13, 845 03 Bratislava), která nejen dodává překladače tohoto jazyka, ale sdružuje i jeho uživatele a vydává bulletin OBJEKTIV.

Pro naši programátorskou veřejnost je známější druhá skupina jazykových prostředků, které vycházejí z klasických procedurálních jazyků, např. Pascal a C, obohacených o prostředky nutné k realizaci objektově orientované technologie. Takto byl obohacen Turbo Pascal ve verzi 5.5 a 6.0 firmou Borland a Quick Pascal firmou Microsoft. Dále jazyk C++ firmou Borland a Objective C firmou Software – IC. Tyto jazyky se často označují jako hybridní, neboť představují směs klasického strukturovaného přístupu i objektově orientovaného přístupu,

Větší popularita těchto jazyků vyplývá ze skutečnosti, že programátoři se nemusí učit nový programovací jazyk, ale naučí se jen používat rozšíření, která vyplývají z nutnosti umožnění realizace objektově orientovaných principů. Konkrétně u jazyka Turbo Pascal jsou to klíčová slova:

object – pro definování objektu

constructor – pro zajištění propojení inicializační metody dynamicky deklarované instance

destructor – pro zrušení dynamicky deklarovaného objektu

virtual – pro definici virtuální metody

a forma zápisu dědičnosti, což se provede v závorkách za klíčovým slovem object při jeho deklaraci.

Další doplnění spočívá v rozšíření syntaxe procedur New a Dispose, které mohou volat inicializační metodu nebo destruktor jako druhý parametr. Prvním parametrem zůstává ukazatel na objektový typ.

Přibyla standardní procedura Fail, která umožňuje uvolnit již alokovanou paměť pro dynamický objekt.

Zásoba nových konstrukcí tedy není nijak rozsáhlá a programátor může zhodnotit dosavadní znalosti jazyka Pascal.

Na druhé straně tento přístup nemůže realizovat důsledně všechny principy objektově orientované technologie nebo je realizuje složitými konstrukcemi. V tom jsou omezené možnosti tohoto přístupu, který musí respektovat hostující jazyk.

Poznamenejme, že hybridní jazyky musí být rozšířeny nejen ve vlastním jazyku. Musí být odpovídajícím způsobem rozšířeny i integrované ladící prostředky, aby bylo možné vytvořené programy testovat. Tak např. Turbo Pascal má zavedeno inspekční okno pro typ objekt, které sumarizuje informaci o objektu a inspekční okno pro instanci, které poskytuje informace o konkrétní instanci. Dále umožňuje krokování volaných metod, zobrazení objektové hierarchie a další možnosti. Vzhledem k nutnosti zvládnout nové principy objektově orientovaného přístupu jsou pro programátory tyto prostředky velmi potřebné a hrají významnou roli v podpoře zvládnutí objektově orientované technologie.

Hybridní jazyky se většinou spokojí s menšími nároky na operační paměť a výpočetní mohutnost ve srovnání se skutečnými objektově orientovanými jazyky. Jsou také navrženy výhradně jako kompilátory, zatímco řada skutečných objektově orientovaných jazyků pracuje interpretačním způsobem. Proto hybridní jazyky jsou při skromnějších výkonech technických prostředků rychlejší.

Protože jak jazyk Pascal tak jazyk C jsou procedurálně orientované jazyky, poskytují dodavatelé kompilátorů podporu programátorům ve formě bohaté standardní knihovny objektů. Programová jednotka Turbo Vision pro jazyk Turbo Pascal 6.0 je toho klasickou ukázkou. Představuje pro programátora komplexní prostředek pro vytváření uživatelsky přívětivého rozhraní s myší, okny, pull – down menu, funkčních tlačítek atd. To vše ve formě objektů, které může programátor přizpůsobit svým vlastním potřebám. Kromě úspory práce přináší Turbo Vision i určitou standardizaci uživatelského rozhraní, což má velký význam pro aplikačního uživatele používajícího více různých aplikací, které se v minulosti ovládaly každá jinak. Zevrubný popis této programové jednotky zde nemůže být uveden a zájemce odkazujeme na originální firemní dokumentaci, případně její české překlady, které jsou již k dispozici, např. díky pražské firmě GRADA.

Třetí skupinu programových prostředků tvoří objektově orientované produkty, které nabízejí standardizovanou zásobu objektů a k nim prostředky manipulace s nimi. Na

rozdíl od Turbo Vision, což je knihovna orientovaná pro potřeby programátorů v jazyce Pascal, jsou objekty a prostředky orientovány na uživatele neprogramátory nebo problémově orientované aplikační programátory.

Příkladem může být objektově orientovaný produkt Object Vision firmy Borland, který firma představila v únoru 1991.

Object Vision je prostředek umožňující koncovým uživatelům, kteří nejsou profesionálními programátory, vytvářet aplikace na počítačích třídy PC pod grafickým rozhraním Windows. Aplikace se vytváří metodou tzv. vizuálního programování, která neklade nároky na znalosti procedurálních programovacích jazyků. Uživatel využívá specializovaných grafických editorů. Tam, kde nelze určité elementy, které se mají vybrat a modifikovat, graficky vyjádřit, se používá dvojrozměrných tabulek, které uživatel vyplňuje. Aplikace vytvořená pomocí Object Vision se skládá ze tří základních elementů :

- uživatelského rozhraní ve formě volně uživatelem definovaného formuláře
- rozhodovacích stromů pro vložení, kontrolu, zpracování dat a uložení dat
- komunikačních prostředků, které umožňují komunikovat s databázemi nebo jinými aplikacemi v prostředí Windows

Prostředky Object Vision dovolují uživateli navrhnout formát celé obrazovky i druhy jednotlivých údajových polí (čísla celá, reálná, datum, čas, textové řetězce) včetně edičních operací. Opatřit pole formuláře doprovodnými texty. Vytvářet seznamy pro výběr předdefinovaných hodnot a textové nápovědy pro každé políčko formuláře.

Pro popis výpočtu používá Object Vision rozhodovacích stromů, které sémanticky odpovídají příkazům IF – THEN nebo CASE. Tyto stromy jsou vytvářeny v grafické podobě, což zvětšuje přehlednost a usnadňuje tvorbu aplikace. K definování podmínek jsou k dispozici relační a aritmetické operace jak s numerickými hodnotami, tak textovými řetězci.

Object Vision podporuje spojní s databázemi Paradox, dBase a Btrieve jak souborem v ASCII, tak prostřednictvím výměnného protokolu systému Windows – Dynamic Data Exchange.

Jako další příklad třetí skupiny objektově orientovaných prostředků bychom mohli kromě Object Vision uvést systém SPACETALK, což je objektově orientovaný programovací systém vhodný pro tvorbu rozsáhlých expertních systémů, databázových systémů, CAD systémů a problémů z oblasti umělé inteligence. Systém je praktickou implementací v prostředí Smalltalk a prostřednictvím původní teorie p-objektů umožňuje vytvářet reprezentaci prostorových dynamických systémů /19/. Práce se systémem Spacetalk je rovněž založena na grafickém rozhraní, které poskytuje velmi efektivní způsob,

tvorbu, modifikaci a prohlížení objektů. Grafický způsob interakce dovoluje předat systému velké množství především grafických informací.

Uživatel systému Spacetalk může bez znalosti programování v jazyku Smalltalk jen s využitím znalostí objektově orientované technologie a komunikace se systémem Spacetalk definovat objekty, které mají vlastnosti „živých – inteligentních" objektů, tj.objektů spojených s popisem jejich chování odvozeného z podmínek, které musí objekty nebo jejich okolí splňovat a které mohou být vyvolané např. jejich změnou.

Koncepce systému předurčuje jeho použití především v oblastech ekologie, architektury, urbanismu, v CAD systémech strojního, stavebního a elektrotechnického zaměření, ale i v oblastech biologie a medicíny, plánovacích expertních systémů, simulačních systémů a grafických informačních systémů.

Systém Spacetalk je výsledkem práce autorského kolektivu firmy ArtInApple S (Artificial Intelligence Application Systems), která je zároveň jeho distributorem.

Objektově orientované databáze

Objektově orientovaná technologie změnila i pohled na realizaci datových bází.

Dosud se rozšířily dva datové modely pro realizaci bázi dat:

- síťový model DBTG CODASTYL, rozšířený v 70–tých letech pro realizaci bází dat zejména na externích pamětech sálových počítačů a minipočítačů
- relační model E. Codda rozšířený v 80–tých letech zejména pro realizaci bází dat v operačních pamětech osobních počítačů a minipočítačů

Oba dva modely dat hledaly řešení problému uzpůsobení lineární paměti von Neumanova počítače do struktury, která by odrážela co nejlépe struktury reálného světa, který má být modelován na počítači.

Objektově orientovaná technologie poskytuje návod ve svých principech jak strukturalizovat data podle objektů. Ve srovnání s dnes rozšířeným relačním modelem dat dovoluje objektově orientovaný přístup navrhovat datové báze strukturované složitěji v paměti využitím více ukazatelů.

V současné době pracují dvě skupiny na tvorbě návrhu objektově orientovaného databázového modelu, který by se mohl stát základem pro sjednocení dalšího postupu v této oblasti:

- Object Oriented Database Task Group u americké standardizační národní společnosti ANSI

– Object Management Group, což je nezávislé sdružení tvůrců software a uživatelů

V roce 1990 publikovala pracovní skupina firmy ALTAIR Group objektově orientovaný datový model O_2 plně založený na principech objektově orientované technologie.

O možnosti práce s objekty a jejich využití v databázovém prostředí byla obohacena i poslední verze databázového systému INGRES.

Ke konci roku 1991 byla počítačová veřejnost informována, že firmy Borland a Ashton–Tate spojily své síly pro vývoj nového databázového produktu Object dBase založeného na principech objektově orientované technologie. Tato databáze by měla prostřednictvím BLOB (Basic Large Object) rozšířit možnosti databázového zpracování kromě oblasti klasického hromadného zpracování dat i na oblasti souvisejícími s realizací rozsáhlých databází v systémech CAD, CAM, CIM.

Pokud bychom chtěli stručně vyjádřit rozdíly mezi relační bází dat a objektově orientovanou bází dat, může nám k tomu posloužit následující tabulka:

RDB	OODB
Strukturovaný databázový jazyk	Navigační dotazovací jazyk
Minimalizace závislosti dat	Minimalizace závislosti funkcí
Pomalejší než hierarchické DB	Pomalejší (zatím) než RDB
Krátké transakce s optimistickým souběhem	Dlouhé transakce získané optimistickým souběhem
Odpovídají jazykům 4. generace	Odpovídají objektově orientovaným jazykům
Implicitní relační vazby	Explicitní relační vazby
Víceznačné identifikátory	Jednoznačná identifikace objektů
Možnost reprezentovat objekty	Možnost reprezentovat relace
Platný mezinárodní standard	Dosud není žádný standard

Nově navržené objektově orientované databáze počítají s využitím nových paměťových prostředků (optické disky typu CD-ROM, WORM a RWM, paměti R–DAT, holografické paměti) pro ukládání rozsáhlých souborů numerických dat, ale i textů, obrazů a hudby (multimédia).

Používání objektově orientované technologie

Paní Adela Golbergová, manažer firmy Parc Place System, uvedla, že cobolští programátoři potřebují dva měsíce k základnímu zvládnutí objektově orientovaného přístupu a dalších 6 měsíců k tomu, aby se v tomto prostředí cítili jako doma. Zhruba dva roky je potřeba k tomu, aby firma, která zavede objektově orientovanou technologii, získala zpět do ní investované prostředky a aby začala pociťovat přínosy této nové technologie. Ty jsou později značné.

Není to však jen výchova analytiků a programátorů, jejichž dobrým školením se dají snížit počáteční náklady na zavedení objektově orientované technologie, protože dobrým zvládnutím objektově orientované analýzy a programování se sníží riziko chyb při zavádění. Programátoři a analytici se musí dobře naučit používat svoje nástroje (objektově orientované systémy CASE, objektově orientované jazyky a ladící prostředky, objektově orientované databázové systémy).

Nová technologie vyžaduje i poněkud odlišnou organizaci pracovních týmů.

Strukturovaný přístup dával přednost technologicky uspořádaným týmům (analytici, programátoři, provozní personál, realizační prac.). Příklad se viděl v symfonickém orchestru řízeném taktovkou dirigenta, kde převažují pojmy jako hudební kvalifikace, uniformita, organizovanost, subordinace, konformita. Zdá se, že pro objektově orientovaný tým lze nalézt analogii spíše v jazz bandu (improvizace, intuice, nadšení, impulsivní myšlení). Pro toto chápání práce se lépe hodí problémové uspořádání týmů orientované na automatizaci určité zadané předmětné oblasti.

Často slýcháme otázku : „Jak mám nejlépe postupovat, abych zvládl objektově orientovanou technologii, když u nás ještě nejsou pro to učebnice?"

Pro většinu zájemců je možno doporučit následující postup.

Přečtěte si některý všeobecný příspěvek o objektově orientované technologii pro získání základního přehledu.

Pak je ale potřeba začít „zdola"! Vyberte si jazyk, který znáte (Pascal nebo C) a seznamte se s jeho rozšířením pro objektově orientované programování.

Nemůžete však počítat s tím, že se této nové technologii naučíte jen z příruček! Přistupte k praktickému procvičování. Doporučuje se kromě vlastních pokusů s nedokonalými objekty vzít některé objekty z knihoven a pokusit se je rozšiřovat po jejich důkladném prostudování. Tento přístup je praktický z toho hlediska, že objektově orientovaná technologie dává přednost znovupoužití objektů před vytvářením nových objektů.

Praktické příklady nám dovolí nejen postupně rozšiřovat své znalosti, ale kromě získání rutiny, získáte i důkladnou znalost vývojového prostředí pro testování objektově orientovaných programů.

Profesionální programátoři mohou čím dál tím hlouběji vnikat postupně do dalších a dalších fines objektově orientovaného programování.

Analytikům a projektantům pomůže znalost objektově orientovaného programování k pochopení požadavků na výsledky objektově orientované analýzy a její metodické postupy.

Závěr

Závěrem zopakujme stručně výhody, které objektově orientovaná technologie přináší.

Zapouzdření dat a metod způsobuje, že jak data tak algoritmy tvoří jeden celek, který zabezpečuje, že správná data budou k dispozici pro příslušné algoritmy a že se zabrání aplikaci algoritmů na nepříslušná data. Navíc jsou zapouzdřením chráněna datová pole před zásahy zvenčí.

Polymorfismus je prostředek, který je z hlediska budoucích projektů daleko perspektivnější než knihovny procedur. Důmyslnou výstavbou objektů lze vytvořit základnu, ke které můžeme pohodlně přidávat doplňky podle zvyšujících se nároků uživatelů, aniž bychom byli nuceni od základů znovu něco měnit. To zjednodušuje a zrychluje práci a zhodnocuje dříve vynaloženou námahu.

Dědičnost a rozdělení do tříd poskytují prostředky pro vytvoření řádu a přehledu. Je rozdíl mezi tisícem druhů hmyzu uspořádaných do taxonometrického systému a mezi stejným tisícem hmyzích druhů, které vám právě krouží kolem hlavy.

To vše konec konců přináší zkrácení doby vývoje aplikací, snížení nákladů na jejich vývoj a zjednodušení údržby, ať už z hlediska oprav nebo z hlediska dodatečných změn a doplňků.

Přitom je zachována dostatečná srozumitelnost a přehlednost i při rozsáhlých projektech a velkých pracovních týmech.

Uvědomíme–li si, že objektově orientovaná technologie je teprve na začátku svého nástupu do praxe a čeká ji další rozvoj, pak stojí za to se s ní důkladně seznámit a začít ji prakticky používat.

Literatura:

1. O. Dahl, B. Myhrhang, K. Nygaard: Simula 67, Oslo 1968
2. A. Goldberg: Smalltalk – 80, Addison– Wesley 1983
3. V. Fiala: Objektově orientované programování. Seriál v časopise BAJT č. 1, 2, 3 a 4 roku 1990
4. E. Kindler, M. Brejcha: Objektově orientované programování v praxi. Seriál v příloze časopisu Automatizace č. 5–12/1989
5. R. Ralph, W. Konfel, D. Chalmers: Objekty pozornosti. Computer World č. 4/1990 z 29.6.1990
6. B. Meyer: From Structured Programming to Object – Oriented Design. Structured Programming č. 1/1989, str. 19–39
7. G. Blaschek, G. Pomberger, A. Stritzinger: A comparison of object – oriented languages. Structured Programming č. 4/1989, str. 187–197
8. E. Gibson: Object – Born and Bred. Byte č. 10/1990, str. 245–254
9. S. Atre: Výhradní zpráva o OOPS. PC World č. 2/1991, str. 66–68
10. J. Machholz: Turbo Vision. Chip č. 4/1991, str. 95–97
11. J. Weinberger a kol.: Objektově orientované programování a jeho aplikace v diskrétní simulaci. Inorga Praha 1990, ASŘ sešity str. 165–166
12. J. Honzík: Technologie programování. FE VUT v Brně, Brno 1991
13. Z. Benda, J. Staudek: Programování v jazyku SIMULA 67. SNTL Praha 1978
14. J. Hořejš: Principy strukturovaného programování. Informační systémy č.2 a 3, 1975
15. B. Lacko: Objektově orientovaná analýza v databázových systémech. Sborník referátů ze semináře DATASEM 91, CS–COMPEX, Brno 1991

16. B. Lacko: Analýza metod pro racionalizaci programování. VÝBĚR č.4/1976, str. 468–472

17. J. Polák: Jak objektově programovat? Sborník ze semináře Programování 91, Dům techniky ČSVTS, Ostrava 1991, str. 63–82

18. R Bušek: Objektově orientované databázové systémy. Sborník referátů ze semináře DATASEM 91, CS–COMPEX, Brno 1991

19. A. Mrázik, J. Kleinertová: Spacetalk. In 11, str. 125–184

20. K. Nenadál, D. Václavíková: Borland C++ – objektové programování a popis jazyka. GRADA, Praha 1992

21. O. Macko: Turbo Vision – praktický sprievodca. GRADA, Praha 1992

22. Š. Benyovský: Turbo Vision – výklad a programování. GRADA, Praha 1992

23. P. Štolcpart a kol: Object Professional pro Pascal. GRADA, Praha 1992

Autor: ing. Branislav Lacko CSc
VUT v Brně, Fakulta strojní
Katedra přístrojů a automatizace
Technická 2, 616 69 Brno
Tel: 05 – 714 2206