SOUČASNÁ OBJEKTOVĚ ORIENTOVANÁ VÝVOJOVÁ PROSTŘEDÍ ZALOŽENÁ NA JAZYCE SMALLTALK

Ing. Vojtěch Merunka, Ph. D.

Katedra informačního inženýrství, PEF ČZU Praha

> *"In essence, Smalltalk is a programming language focused on human beings rather than the computer."*
>
> — Alan Knight
>
> *"I invented the term Object-Oriented, and I can tell you I did not have C++ in mind."*
>
> — Alan Kay

Příspěvek je prakticky orientován a slouží k prvnímu seznámení pro vývojáře a tvůrce softwaru. Článek si neklade za cíl podat úplný výklad jazyka Smalltalk.

Historie Smalltalku, vznik OOP

Smalltalk byl vyvíjen v Kalifornii v **Palo Alto Research Center** (PARC) kolektivem vědců vedených dr. Alanem Kayem (tým *Learning Research Group*) a dr. Adelou Goldbergovou (tým *System Concepts Laboratory*) v létech 1970-1980. Předmětem celého výzkumu, který byl financován v největší míře firmou Xerox, byl projekt "**Dynabook**" pro vývoj osobního počítače budoucnosti.

Počítač Dynabook se měl skládat z grafického displeje formátu listu papíru o velikosti přibližně A4 s jemnou bitovou grafikou, klávesnicí, v té době novou periférií - perem, později nahrazeným myší, a jeho součástí měl být i síťový interface. Pro vzhled systému byla poprvé na světě použita překryvná okna, vynořovací menu a ikony. V průběhu 70. let bylo dokonce vyrobeno několik prototypů takových počítačů. Předpokládalo se, že počítač bude obsahovat jednotné softwarové prostředí, které bude současně plnit úlohu operačního systému i programovacího jazyka s vývojovým prostředím. Právě tento software dostal název Smalltalk.

Ve Smalltalku, který byl jako projekt dokončen v roce 1980, se nejvíce odrazily prvky z jazyka LISP a z prvního objektově orientovaného jazyka Simula. Část týmu v PARC zůstala a založila pod vedením A. Goldbergové firmu ParcPlace Systems, která rozvíjí Smalltalk dodnes, jiní spolu s A. Kayem odešli do firmy Apple Computer, kde poté uvedli na trh první dostupný komerční osobní počítač Lisa s grafickým uživ. rozhraním (dále jen GUI). Smalltalk a jeho GUI byl v průběhu 80. let využíván zpočátku pouze na výkonných pracovních stanicích té doby, z nichž nejznámější byl Tektronix 4404[1] z roku 1982. V USA vzniklo několik firem (např. Knowledge Systems), které již okolo roku 1985 používaly Smalltalk pro náročné aplikace z oblasti expertních systémů, řízení výroby, řízení projektů apod. (např. program Analyst vytvořený na zakázku Texas Instruments). Smalltalk byl používán také ve výzkumu na vysokých školách. Myšlenka GUI Smalltalku dala během 80. let vznik systémům Macintosh OS, MS Windows, X-Window apod. Jazyk Smalltalku přímo ovlivnil vznik

[1] Jeden exemplář tohoto počítače je v osobním vlastnictví autora.

programovacích jazyků Objective-C, Actor, Eiffel, CLOS, Object Pascal, C++, Oberon a Java.

Smalltalk je dnes nenahraditelným pomocníkem ve výzkumu na univerzitních pracovištích, kde je často využíván i jako první vyučovaný programovací jazyk. Od počátku 90. let se objevuje nová oblast využití v oblasti tvorby rozsáhlých programů šitých na míru konkrétnímu zákazníkovi (*in-house software*) z ekonomické oblasti, řízení výroby apod. Smalltalk je jedním z mála podporovaných jazyků v moderních objektově orientovaných databázích (Gemstone, Versant, Ontos, Orion, ArtBase,...). Je patrný posun využití Smalltalku směrem od tvorby prototypů ke tvorbě "středních" a "vyšších" programových aplikací, kde se jeví jako vhodnější než například jazyk C++, jehož oblast využití se dnes posunuje směrem k "nižším" aplikacím (systémový software, doplňky do operačního systému apod.). V roce 1994 si Smalltalk zvolila firma IBM za jeden z podporovaných aplikačních programovacích jazyků (produkt „VisualAge“).

Ukázky syntaxe jazyka Smalltalk-80

Syntaxe jazyka Smalltalk je jednoduchá, ale jeho sémantika je natolik originální, že vyžaduje od začátečníka znajícího jiný programovací jazyk bohužel větší úsilí než je obvyklé např. při přechodu z Pascalu do C. Smalltalk elegantně využívá výhod třídně-instančního objektově orientovaného modelu, tj. skládání objektů, dědění, závislosti mezi objekty, polymorfnosti a vícenásobné použitelnosti kódu. Jazyk je integrován s programovacím prostředím (lze jej odstranit v hotové aplikaci), které je napsané taktéž v jazyce Smalltalk. Vše je přístupné včetně zdrojových kódů. Navenek se systém chová jako jediný rozsáhlý program, který je programátorem měněn (doplňován) za svého chodu. I když Smalltalk podléhá vývoji v oblasti OOP, kde stále udržuje náskok před jinými systémy, tak zde popsané vlastnosti jsou kromě modulů VisualWave a ObjectLens jeho součástí již od roku 1976.

Následující ukázka obsahuje jednoduchý program, který zjistí ze vstupního řetězce znak s největším kódem. Pro porovnání uvádíme i jeho zápis v jazyce C:

jazyk C

```
include <stdio.h>;
include <string.h>;
void main
        {
        char *s;
        int i;
        char c, temp;
        printf("enter text: ");
        scanf("%s",s);
        c = ' ';
        for (i = 0; i < strlen(s); i++)
                {
                temp = s[i];
                if (temp > c) c = temp;
                }
```

```
        putChar(c);
        }
```

jazyk Smalltalk

```
| s c |
s := Dialog request: 'enter text:'.
c := $  .
1 to: s size do:
                    [:i |
                     |temp|
                     temp := s at: i.
                     temp > c ifTrue: [c := temp]].
^c
```

V tomto textu se nebude zabývat syntaxí Smalltalku a omezíme se jen na základní informaci o zápisu objektů a zpráv, kterou si ukážeme na příkladu prvního výrazu ve Smalltalku za deklarací proměnných:

```
Dialog request: 'enter text:'.
```

V každém Smalltalkovém výrazu se nejprve píše objekt přijímající zprávu (Dialog), potom selektor zprávy (request:) oddělený mezerou a nakonec parametry zprávy ('enter text:') také oddělené mezerou.

Oba programy byly záměrně napsány tak, aby si byly velmi podobné. Není pochyb o tom, že lze v obou jazycích stejný algoritmus napsat úsporněji. Při dobré znalosti standardních tříd objektů a zpráv v knihovně Smalltalku je však možné výše uvedený program velmi zjednodušit[2]:

```
^(DialogView request: 'enter text:') asSortedCollection last
```

Důležitou součástí jazyka Smalltalk jsou **bloky výrazů**. Bloky výrazů představují vyčleněné sekvence výrazů. Blok výrazů je objektem, může být pojmenován, mohou mu být posílány příslušné zprávy, a může být použit v jiných výrazech (zprávách) jako příjemce nebo parametr[3]. Výrazy v blocích se vyhodnocují vždy až při příslušném požadavku na jejich vyhodnocení, a ne při vytvoření bloku. Tentýž blok proto může v různých situacích vracet různé výsledky. Získaná hodnota samozřejmě záleží na stavu systému v době spuštění bloku a ne na stavu v době vytvoření bloku. Následující příklad ukazuje blok kódu, který je uschován do objektu se jménem B. Blok obsahuje kód, který umocňuje vstupní parametr na druhou a přičítá k výsledku hodnotu objektu A:

[2] Tento příklad byl také záměrně vybrán proto, aby ukázal výhody OOP i v oblasti základní algoritmizace, která je někdy považována za doménu klasického procedurálního programování. OOP totiž není jen nástrojem pro počítačovou grafiku a programování uživatelských rozhraní.

[3] Bloky jsou objektová implementace lambda výrazu.

```
B := [:x | (x ** 2) + A].
```

Jestliže budeme mít v systému takto vytvořený blok, tak můžeme nastavit objekt A na jinou hodnotu, než měl v době vytvoření bloku, a blok postupně spouštět jako například:

```
A := 10.
B value: 3.        nám dá výsledek 19 (3^2 + 10) nebo

A := 5.
B value: 4.        nám dá hodnotu 21 (4^2 + 5).
```

Poslední ukázkou zajímavostí jazyka Smalltalk je zpráva perform:. Je to zpráva, která sama posílá další zprávu podle svého parametru.Ve Smalltalku je totiž i zpráva považována za objekt, který např. lze přiřadit do proměnné. Věc si ukážeme na jednoduchém příkladě aritmetických operací. Aritmetické operace jsou implementovány také pomocí zpráv posílaných číselným objektům:

```
20 sin.            nám dá hodnotu sin(20) nebo např.
20 cos.            nám dá hodnotu cos(20).
```

Použijeme-li zprávu perform:, tak můžeme selektor zprávy pro sinus nebo cosinus uložit pod objekt - v ukázce se jménem myOp - a aritmetickou operaci vyvolat následovně:

```
myOp := #sin.              a potom výraz
20 perform: myOp.     nám dá opět hodnotu sin(20).
```

Operací perform: tedy lze podobně jako pomocí bloků parametrizovat[4] algoritmus vyvíjené aplikace.

Architektura smalltalkového programu

Smalltalk nevyužívá přímo strojový kód počítače, na kterém běží. Namísto toho překládá programy do tzv. **byte kódu**, který je za běhu interpretován do hostitelského strojového kódu. Dříve byl pomalý chod programů pro Smalltalk velkou nevýhodou, protože první implementace SVM (Smalltalk Virtual Machine) pracovaly jako prosté interprety byte kódu. Přesto ale tato koncepce inspirovala architekturu Javy a nebo systému OS400 od IBM.

U nejnovějších verzí Smalltalku-80, který tvoří současný standard Smalltalkových systémů, je architektura SVM navržena jako tzv. **dynamic compiler**, kdy se za chodu aplikace přeložené části binárního kódu Smalltalku ukládají do cache paměti, čímž je dosahováno téměř výkonnosti klasicky přeloženého programu beze ztráty pružnosti systému. Účinnost tohoto řešení dosahuje podle typu úlohy hodnot přibližně od 0.5 do 0.95. Binární kód Smalltalku-80 je navíc asi 3x až 5x kompaktnější než odpovídající strojový kód a je nezávislý na použitém hardwaru počítače. Ve Smalltalku-80 je byte kód jednotný pro všechny podporované počítačové platformy od Apple a PC k mnoha typům pracovních stanic, což

[4] Kromě těchto prostředků je možné, jak je dále naznačeno v textu, do algoritmu programu zadat změny nebo dokonce tvorbu nových tříd a metod, které se budou provádět za chodu aplikace.

znamená že programy (hotové i rozpracované) mohou být okamžitě přenositelné z platformy na platformu. K tomuto přispívá i na OS nezávislý model grafických objektů[5] a diskových souborů v systémové knihovně.

Virtuální stroj Smalltalku-80 má na PC pro MS Windows i pro Linux cca 600KB. Samotný systém (překladač, prostředí, systémová knihovna, vizuální nástroje, …) má asi 6MB byte kódu[6]. Součástí základní instalace je ještě cca 15MB externích knihoven a dalších pro vývojové prostředí potřebných souborů. Systém uspokojivě běží i na 50MHz počítači s 16MB operační paměti.[7]

Charakteristické vlastnosti jazyka a prostředí

Smalltalk podporuje koncepci metatříd, paralelní programování (má podporu pro řízení procesů jako např. semafory nebo vyrovnávací buffery), z dalších vlastností např. ošetřování výjimek v programu, sledování verzí programového kódu během programování s možností návratu do libovolného z přechozích stavů aj. Programátor může do proměnných přiřazovat nejen data, ale i kód.

Čisté objektově orientované prostředí jazyka Smalltalk neobsahuje příkaz skoku, podprogramu ani funkce. V aplikaci dokonce nemusí být ani tzv. hlavní program, jak jsme zvyklí z procedurálních jazyků. Aplikaci tvoří množina objektů, kteří asynchronním prováděním svých metod určují, jak mají reagovat na došlé zprávy a posílají zprávy dalším objektům. Běh aplikace začíná posláním nějaké zprávy zvenčí (klávesnice, myš...), která způsobí posílání zpráv dalším objektům aplikace. Program se za chodu chová jako simulační systém řízený sledem vnějších událostí. Hlavní algoritmus nemusí být nikde explicitně popsán, neboť program se zjednodušeně řečeno řídí sám podle vnějších souvislostí a podle dynamicky utvářeného sledu zpráv z jednotlivých prováděných kódů metod zúčastněných objektů. Řízení toku výpočtu (podmíněné provádění bloků kódu, iterace, …) je implementováno také pomocí zpráv.

Smalltalk implementuje dynamický paměťový model, ale nenutí programátora používat ukazatele do operační paměti - všechny objektové proměnné jsou referencemi na jejich hodnoty ve virtuální paměti. Správu fyzické paměti řídí systémové paralelní procesy, které aplikační programátor nepotřebuje znát. Smalltalk používá algoritmus "garbage collection".

Smalltalk podporuje kreativní inkrementální programování. Ve Smalltalku je možné experimentovat s libovolně velkými částmi kódu vytvářeného programu, psát či překládat a odlaďovat programy po částech a měnit je za jejich chodu. Každá nově naprogramovaná metoda je po svém napsání okamžitě (tak, jak je její zdrojový kód zapisován nebo vizuálně sestaven programátorem) překládána a zapojována do systému. Ve Smalltalku se program

[5] Například je možné vyvíjet aplikaci pro MacOS na PC s Windows 98, protože vzhled grafického rozhraní i systém souborů je zajištěn vlastními systémovými objekty Smalltalku.

[6] Základní systém VisualWorks/Smalltalk-80 uložený v uvedených 6MB byte kódu obsahuje 1250 dvojic třída/metatřída a 27000 instančních a třídních metod.

[7] Tyto dnes skromné hardwarové nároky se však v 80. letech jevily zcela jinak, a tak Smalltalk získal pověst "velmi pomalého a hardwarově náročného systému".

nepíše jako nějaký text v textovém editoru, ale vytváří se po jednotlivých metodách nebo blocích kódu v nejrůznějších vizuálních nástrojích (viz. obr. 1.).

Program ani po své finalizaci neztrácí pružnost, neboť je možné ukládat na disk jeho tzv. "snímek" (angl. "image"). Tato technika umožňuje programátoru i uživateli pokračovat v systému (nebo i v hotovém programu) přesně od toho bodu, ve kterém program opustil. Kromě stavu pracovních souborů se uchová i obsah a vzhled jednotlivých oken, menu i běžících paralelních procesů.

Systém je natolik otevřený, že umožňuje nejen tvorbu vlastních programovacích a ladících nástrojů, ale i změny v samotném systému (syntaktický analyzátor, překladač, mechanismus výpočtu, debugger), což dovoluje pod Smalltalkem např. implementovat jiné programovací jazyky, doplňovat systém o vícenásobnou dědičnost, backtracking, ... Mezi systémovými a doplněnými vlastnostmi není formálního ani funkčního rozdílu.

V naznačené architektuře aplikačního programu je výpočet řízen kódy metod u jednotlivých objektů. Pro spouštění těchto operací nám slouží technika posílání zpráv objektům. Jednotlivé zprávy zde vystupují jako žádosti o operace. Kromě jednoduchého posílání zpráv však ve Smalltalku existuje i další mnohem rafinovanější řízení provádění operací. Tato technika využívá **závislosti** (dependency) objektů mezi sebou.

Ve vztazích závislosti mezi objekty rozeznáváme dva typy objektů: řídící objekty - tzv. klienty a řízené objekty - tzv. servery. Žádá-li nějaký klient provedení nějaké operace od serverů, tak posílá zprávu, neboť zpráva je i zde jedinou možností, jak spustit nějakou operaci. Na rozdíl od standardního poslání zprávy, kdy je třeba znát příjemce zprávy, v případě závislých objektů klient nepotřebuje znát svoje servery, protože oni sami jsou povinni svého klienta sledovat a zprávy od něj zachycovat. Zprávu, která je signálem pro servery, objekt klient posílá bez udání příjemce a systém ji automaticky rozšiřuje na příslušné servery, jejichž počet a vlastnosti se mohou v systému průběžně měnit. Vztah závislosti mezi objekty bývá úspěšně využíván při modelování grafických uživatelských rozhraní a při tvorbě nejrůznějších simulačních modelů.

Pro Smalltalk je také důležité vědět, že třídy jsou rovnocenné objekty, mohou za chodu programu být ukládány do proměnných, mohou vznikat, zanikat nebo měnit svůj datový obsah, kterým jsou například kódy metod nebo vazby dědičnosti. Aby to bylo možné, tak jsou v systému přítomny speciální objekty - nazývané **metatřídy**, které vystupují jako třídy tříd (a pro které jsou třídy jejich instance). Smalltalk tedy má systémovou podporu pro metamodelování.

Smalltalk v internetu

Poslední verzí systému VisualWorks/Smalltalk je verze 5i, která obsahuje silnou podporu pro práci v sítích. Systém obsahuje CORBA a DCOM rozhraní a má interface na objektové i relační databáze. Systém podporuje standard XML (používá ho i pro ukládání zdrojových kódů).

Zvláštní pozornost si zaslouží modul VisualWave[8], který je vlastně do Smalltalku integrovaným HTTP serverem, který dovoluje přidávat k aplikacím webové uživatelské rozhraní. Tento proces nevyžaduje od programátora psát žádné HTML nebo podobné kódy, protože konverze rozhraní aplikace to HTML a CGI probíhá zcela automaticky s využitím polymorfismu objektů v systémové knihovně. Na obrázku č. 2. je ukázka nástrojů VisualWave a ukázka malé aplikace, která běží jak přímo v operačním prostředí Smalltalku, tak i přes uživatelské rozhraní webového prohlížeče.

Součástí VisualWorks 5i jsou i doplňky (plug-in) do nejpoužívanějších webových prohlížečů, které umožňují práci s applety v binárním kódu Smalltalku.

Smalltalk v PDA – Pocket Smalltalk, Squeak Smalltalk

Velmi zajímavým využitím Smalltalku je oblast malé spotřební elektroniky. Zde se můžeme setkat se Smalltalkem Squeak, který je vlastně přímým pokračovatelem původního Dynabooku[9]. Squeak totiž stále obsahuje i vlastní operační systém a grafické uživatelské rozhraní. (viz. obrázek 4.) Pro spotřební elektroniku je dostupná jeho odlehčená verze, která se vejde do 1MB paměti.

Na nejmenších mobilních počítačích[10] se můžeme setkat s Pocket Smalltalkem. (viz. obrázek 5.) Jedná se opět o speciální verzi Smalltalku pro normální počítače, v tomto případě o Dolphin Smalltalk. Aplikace se programují na obyčejném PC a po sestavení se přenášejí na kapesní počítač.

Smalltalk a databáze - ObjectLens

Bylo by asi zbytečným nošením dříví do lesa psát, že Smalltalk dokáže pracovat s objektovými databázemi. Velká část objektových databází totiž ze Smalltalku vychází. Pro příklad[11] si uveďme systém Gemstone (http://smalltalk.gemstone.com) a nebo pro zajímavost slovenský systém ArtBase (http://www.artinapples.sk).

VisualWorks/Smalltalk má ale od roku 1992 velmi zajímavý prostředek pro práci s relačními databázovými servery. Jedná se o modul ObjectLens, který umožňuje propojení objektů v programovacím jazyce Smalltalk s daty uloženými na relačním databázovém serveru (viz. obr. 3). Relačním serverem může být například jádro RDBMS systému Oracle, Sybase, MySQL a nebo jakákoliv relační databáze s rozhraním ODBC. ObjectLens má následující vlastnosti:

1) Systém transformuje relační datový model na straně serveru na omezenou variantu objektového datového modelu na straně VisualWorks (klienta).
2) jazyk SQL se v systému ObjectLens používá především pro komunikaci s rozhraním relačního serveru. Na straně klienta lze s objekty pracovat výhradně pomocí programovacího jazyka Smalltalk, přičemž systém sám transformuje dotazy v jazyce Smalltalk na odpovídající příkazy SQL pro server.

[8] Další možností je modul ClassicBlend, který kombinuje do vzdáleného rozhraní Java aplety komunikující pomocí ORB.
[9] Na vývoji Squeaku se také podílí Alan Kay - autor Smalltalku ze 70. let.
[10] Například počítače Palm 3Com, Handspring Visor či IBM Workpad a nebo mobilní telefony Nokia Communicator.
[11] Podrobnou informaci o objektových databázích lze najít například na http://www.odmg.org.

3) Transparentnost - S objekty, které mají data uložena na relačním serveru se pracuje pomocí stejných příkazů a stejným způsobem jako s objekty, které jsou obsaženy pouze v paměti klienta jako běžné proměnné.
4) Systém umožňuje tvorbu relačních tabulek na serveru na základě existujících objektů, což znamená, že i hotové programy lze poměrně jednoduše přeměnit na databázové.

S pomocí ObjectLens může programátor svou pozornost soustředit na objektový návrh a nemusí se starat o problémy vzniklé použitím relační databáze do objektovém prostředí. Klient-server databázovou technologii ObjectLens lze kombinovat s technologií VisualWave a tak vytvářet třívrstvé klient-server aplikace. Pro aplikační programátory je zajímavé sdělení, že pro sestavení databázové aplikace spolupracující s relačním serverem **není třeba** používat pro konstrukci dotazů jazyk SQL. Dotazy lze totiž nejen sestavovat pomocí vizuálních prostředků, ale i samotný jazyk Smalltalk má dotazovací schopnosti. Uveďme si jednoduchý příklad SQL dotazu do nějaké relační tabulky:

```
SELECT *
        FROM Osoby O
        WHERE O.VEK > 18;
```

Stejný dotaz je možné napsat přímo v jazyce Smalltalk s využitím zprávy select: s parametrem bloku, který obsahuje selekční podmínku. Zpráva se posílá objektu Osoby, který vystupuje jako collection a má obsah pomocí ObjectLens svázán s příslušnou relační tabulkou:

```
Osoby select: [:o | o vek > 18].
```

Kromě toho, že ObjectLens pro každý atribut vytvoří příslušnou metodu a na atribut objektu lze potom přistupovat posíláním zpráv (což funguje nejen ve vztahu N:1 i ve vztahu 1:N, kdy se vrací collection hodnot), tak je samozřejmě možné přidávat libovolné další metody. Znamená to, že v naší ukázce dotazu ve Smalltalku může být věk osoby počítán z jiných hodnot (např. z data narození). V tomto případě by ale SQL příkaz vypadal složitěji.

Jaký Smalltalk si vybrat?

Současné systémy Smalltalk jsou následující:

1. **Smalltalk-80** firmy Cincom (www.cincom.com), která je pokračovatelem dřívější firmy Parc Place Systems. Je prodáván pod obchodním názvem **VisualWorks** (dříve ObjectWorks) a je koncipován pro širokou platformu počítačů od PC (Win, OS/2) přes počítače Macintosh až k řadě mnoha typů pracovních stanic Sun, HP, DEC, IBM atd. VisualWorks obsahuje vizuální programování GUI aplikací, CASE tool pro tvorbu klient-server aplikací pracujících s objektovými i relačními databázemi a tzv. Chameleon View, které umožňuje za chodu přepínat GUI vzhled programů (Win, OS/2, Motif, Apple, OpenLook). Smalltalk-80 má velkou podporu jak v možnostech rozšiřování základního systému (knihovny objektů, moduly obchodní grafiky, ...), tak i v napojení na relační a objektové databázové systémy. Existují i univerzitní projekty s volně dostupnými knihovnami na Internetu. Na uvedené adrese je možné získat zadarmo plně funkční nekomerční verzi (včetně ObjectLens) tohoto systému.

2. **Classic Blend** je rozšíření VisualWorks o možnost tvorby distribuovaných aplikací využívajících Java applety a prostředky WWW.
3. **VisualWave** je doplňkový modul VisualWorks, který zahrnuje uživatelská rozhraní Internetu (Web, HTML, CGI) mezi vlastní objektové komponenty sloužící k vývoji smalltalkových aplikací. Od verze 3.1 je standardní součástí VisualWorks (i nekomerční verze).
4. **Visual Smalltalk** Je také produktem firmy Cincom. Pracuje pouze pod operačními systémy MS Windows, OS/2 (existují i jeho starší verze označené Smalltalk/V pro samotný MSDOS) a pro operační systém počítačů Macintosh. Programovací jazyk je až na některá zjednodušení totožný s programovacím jazykem Smalltalk-80. Největší odlišnosti jsou v grafických knihovnách. Z tohoto důvodu není zdrojový kód jednoduše přenositelný do Smalltalku-80 a naopak. Vývojové prostředí je také poněkud jednodušší než ve Smalltalku-80. Na internetu je pro akademické instituce bezplatně dostupná verze označená **Smalltalk Express**.
5. **IBM Smalltalk** je systém odvozený ze Smalltalku/V a je velmi podobný Visual Smalltalku. Je součástí produktu **VisualAge** (CASE pro tvorbu GUI klient-server aplikací). Podporuje práci s relačními databázemi DB2, DB2/VSE, VM (SQL/DS), DB2/400, Microsoft SQL Server, ORACLE a SYBASE SQL Server. IBM Smalltalk alias VisualAge patří v zahraničí mezi nejprodávanější programovací nástroje v MS Windows a OS/2 a má podobně jako Smalltalk-80 mnoho možností k rozšiřování.
6. **Smalltalk/X** vznikl za podpory firmy Tomcat v Mnichově a nyní je dodáván firmou Exept Software AG (http://www.exept.de). Je do značné míry kompatibilní se Smalltalkem-80 a je zajímavý svou originálním způsobem vytvořenou vazbou na jazyk C s možnostmi překladu do strojového kódu. Je dostupný na jakémkoliv počítači s operačním systémem UNIX. Autor tohoto textu se v letech 1992-96 podílel ve firmě Tomcat na jeho vývoji a testování.
7. **Smalltalk DB** je jazykem objektově orientovaného databázového systému Gemstone. Nekomerční verzi tohoto jednoho z nejlepších objektových databázových systémů lze získat na adrese http://smalltalk.gemstone.com.
8. **Enfin** je produktem společnosti Easel Corporation z Burlingtonu v Massachusetts. Pracuje pod operačními systémy Windows, OS/2 a Unix. Použitá verze jazyka Smalltalk není kompatibilní s verzí Smalltalk-80. Obsahuje CASE pro tvorbu objektového modelu aplikace a pro generování obrazovek. Podobně jako VisualWorks umožňuje objektově orientovaným aplikacím pracovat s objekty, jejichž data jsou uložena v relační databázi.
9. **GNU Smalltalk** je produkt v rámci unixového projektu GNU.
10. **Little Smalltalk** je také produkt z dílny GNU. Jedná se o nejmenší smalltalkový systém (velikost jen 50kB), který je vhodný např. pro programování dávek operačního systému a jiné úsporné programy.
11. **Squeak** je původně projekt firmy Apple Computer pod vedením Alana Kaye. Dnes se jedná o velmi rozšířený a podporovaný systém především na zahraničních univerzitách, který je stále veden jako freeware (!) a je dostupný na adrese http://www.squeak.org.
12. **Smalltalk MT** je projekt firmy Object Connect pro podporu programování ve Windows, také dostupný na internetu. (http://www.objectconnect.com)
13. **Dolphin Smalltalk** je projekt firmy Object Arts Ltd. - London, England. se stejným zaměřením jako MT. Starší verze jsou také k dostání zadarmo na internetu (http://www.object-arts.com/Downloads). Pocket Smalltalk je na adrese http://www.pocketsmalltalk.org.

Na internetu na adrese www.smalltalk.org lze najít desítky úspěšných aplikací vesměs z oblasti velkých podnikových informačních systémů využívajících Smalltalk. Na této adrese je i seznam odkazů na instalace nejrůznejších verzí, jejichž počet je několikanásovbě větší než zde uvedený seznam, učební texty a na množství programů a diskuzních skupin zabývajících se Smalltalkem.

Závěr

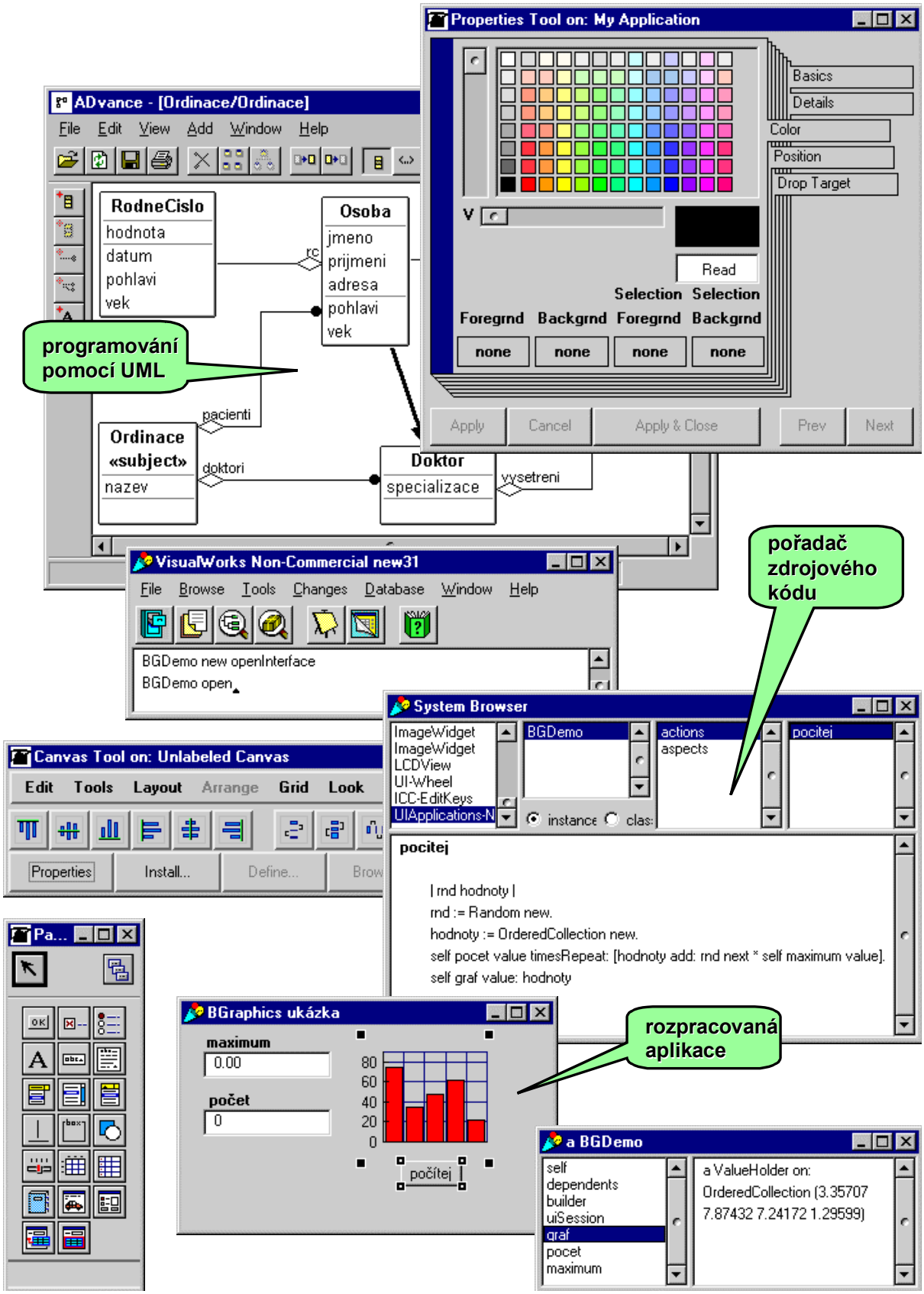
Smalltalk je typickým představitelem EPOL[12] jazyků a pro svoji syntaktickou čistotu a elegantní podporu všech důležitých objektových vlastností je již přes 15 let často používán v odborné literatuře a výzkumu. Role Smalltalku pro výuku objektových informačních technologií na vysokých školách je nezastupitelná.

Smalltalk je díky svojí koncepční čistotě a důslednému dodržování principů OOP velmi důležitý pro oblast objektově orientované analýzy a návrhu informačních systémů. Ve Smalltalku totiž můžeme na rozdíl od hybridních jazyků (C++, Java, Delphi Pascal, …) přímo nebo relativně jednoduše implementovat většinu pojmů a vazeb, jak je známe z objektových konceptuálních modelů. V této oblasti je Smalltalk dokonce používán i jako zdroj nápadů a myšlenek pro vývoj nebo vylepšování metod analýzy a návrhu. Zkušenosti se Smalltalkem měly například přímý vliv na vývoj metod Booche a Jacobsona (obě dnes součástí UML). Smalltalk je také používán jako testovací platforma pro výzkum nových programovacích paradigmat dále rozvíjejících objektový přístup. Jedná se například o instančně orientované programování, aspektové programování, agentové programování atp.

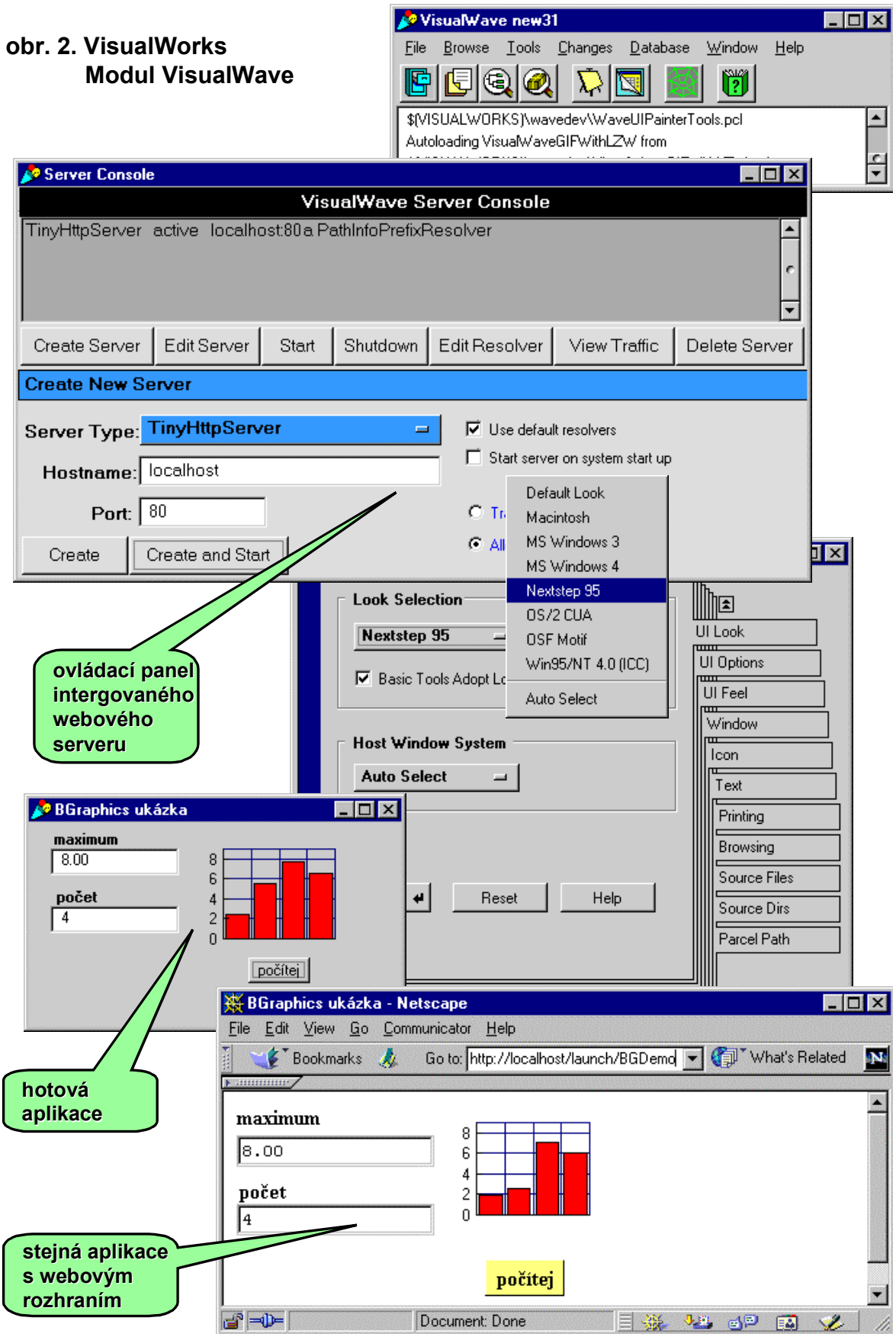
Smalltalk je od konce 80. let používán také pro tvorbu rozsáhlých podnikových informačních systémů[13]. V posledních dvou letech se nástup Smalltalku do praxe zpomalil jazykem Java, do kterého se vložilo mnoho (i neoprávněných) očekávání. Doménou Smalltalku však zůstává tvorba velkých softwarových aplikací spolupracujících s velkými relačními nebo objektovými databázemi. Mezi informačními technologiemi v USA od poloviny 90. let systémy využívající Smalltalk zaujímají[14] okolo 4%, přičemž ale například prvních 200 největších firem v USA používá z 50% technologii VisualWave. Autor článku se samozřejmě nedomnívá, že by se Smalltalk v blízké budoucnosti zařadil do hlavního proudu ve tvorbě softwaru a například nahradil Visual Basic nebo Visual C++. K tomuto nestačí jen být "lepší" a průkopnickou technologií. Tato zdánlivě nejdůležitější kritéria, jak nám ukazuje historie computer science, jsou až daleko za mnohem prozaičtějšími faktory, jako například podpora velkých firem nebo rozhodnutí státních institucí. Paradigma Smalltalku je příliš jiné - ani jeho syntaxe nevychází ze stereotypu Algol/Pascal/C, který tak oblíben u většiny těch, co o sobě říkají, že se zabývají tvorbou softwaru. Ale i pro softwarové specialisty v českých zemích je důležité technologii Smalltalku znát a umět ji použít. Autor článku se domnívá a dokonce si i prakticky ověřil, že to je možné.

[12] EPOL = Environment Pure Object Language.
[13] Autor článku se podílel na vývoji manažerského informačního systému pro BMW A.G. Mnichov.
[14] Podle každoročně publikovaných výsledků výzkumu prováděných firmou Deloitte&Touche.

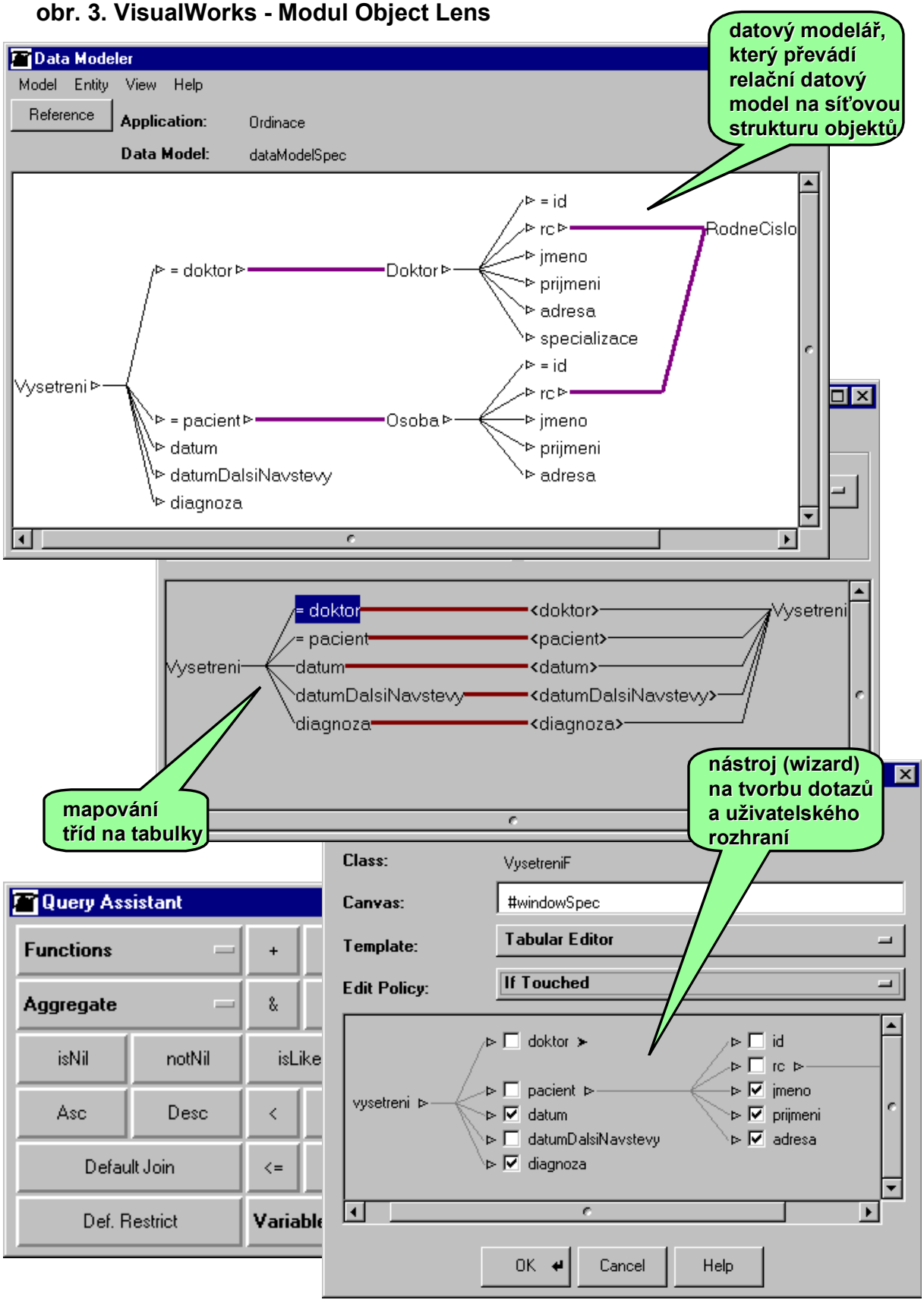
obr. 1. VisualWorks - některé nástroje vývojového prostředí



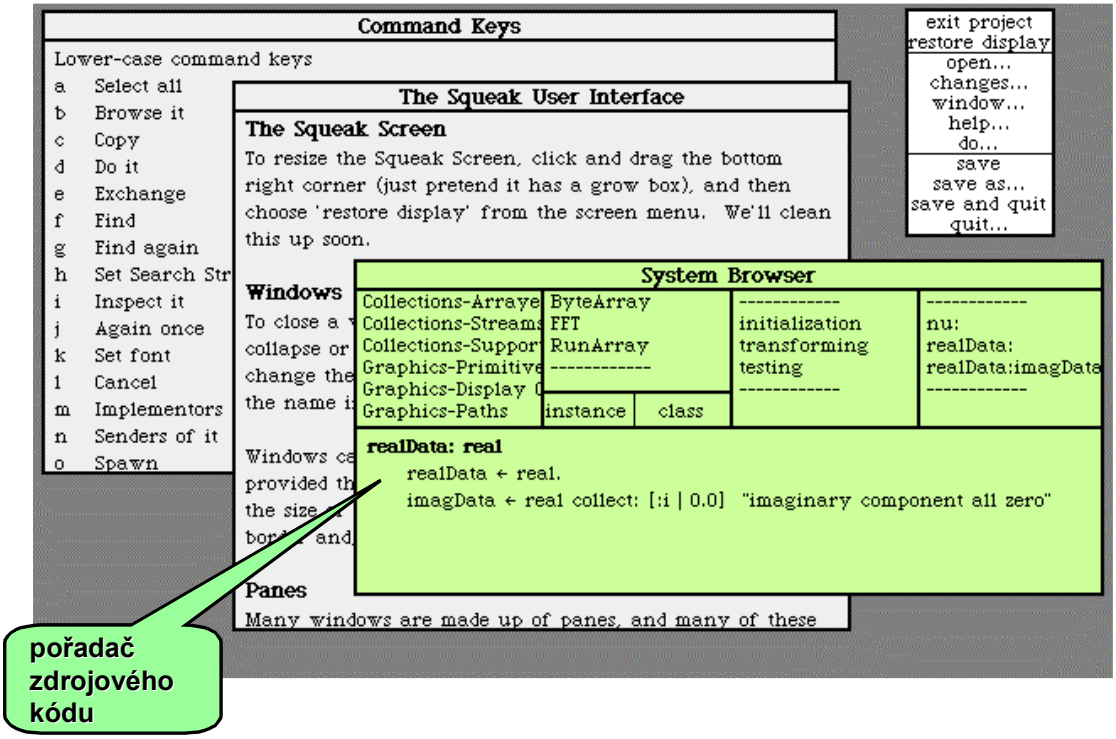
obr. 2. VisualWorks
Modul VisualWave



obr. 3. VisualWorks - Modul Object Lens



obr. 4. Squeak Smalltalk



obr. 5. Pocket Smalltalk