Využití OOP v úvodních fázích analýzy informačních systémů.

Vojtěch Merunka, Katedra informatiky, PEF ČZU v Praze
e-mail: merunka@pef.czu.cz, http://omega.pef.czu.cz/pef/ki/merunka

Úvod

Objektově orientované paradigma (ve zkratce **OOP**) je jedním z nejprogresivnějších směrů v oblasti programování, analýzy, návrhu a realizace softwarových produktů. Netýká se však jen vlastního procesu programování, ale i také reprezentace světa v počítači, způsobu nazírání na uspořádání reálného světa, chápání zadání problému a metodiky jeho řešení (*tj. analýzy, návrhu* a *implementace*). Pomocí OOP lze také nazírat na organizaci architektury samotného počítače a celého informačního systému (IS).

V současné době se již OOP dostává v praxi maximální pozornosti. Nesmí se však zapomínat, že vznik OOP je především výsledkem **snahy řešit** již delší dobu trvající softwarovou krizi, která se projevuje rostoucí složitostí programů překonávajících rostoucí sémantickou mezeru, nebezpečně nízkou spolehlivostí programového vybavení a neúměrnými náklady na pracovní sílu pro programování a údržbu provozovaného softwaru. Roste obliba metod objektově orientované analýzy a designu a postupně se do popředí zájmu dostávají i objektově orientované databáze, které však zatím patří mezi nejméně známé (i využívané) aplikační oblasti objektově orientovaného paradigmatu.

Myšlenka OOP je důležitá pro rozvoj nových operačních systémů, databázových systémů, systémů pro reprezentaci znalostí a také expertních systémů. Uplatňuje se při vývoji velmi rozsáhlých aplikací; výsledný produkt je díky flexibilnosti objektově orientovaných programů lépe udržovatelný a také lépe umožňuje reagovat na změny v zadání.

Pro programátorskou obec je příznačné, že si pod pojmem OOP představují především jeho implementaci v konkrétním programovacím jazyce, nejčastěji v Object Pascalu nebo C++ nebo Java. Jazyků, které poskytují na různé úrovni možnost využití objektové orientace, je však od 70. let vyvinuto značné množství. Připomeňme jazyky Simula, Smalltalk, Actor, Eiffel, Objective C, Flavors, CLOS, Dragoon, Mainsail, ESP, Perl, Beta, ABCL, Actalk, Plasma etc.

Pro naivní uživatele výpočetní techniky je příznačné, že pojem „objektově orientovaný" ztotožňují s výhodami, které přinášejí grafická uživatelská rozhraní současných softwarových systémů.

Tento příspěvek je zaměřen na zkušenosti autora s využitím OOP v uvodních fázích analýzy informačních systémů a přímo navazuje na příspěvek stejného autora z loňského ročníku této konference.

Možnosti „klasických" objektových metod

Objektové metody, které dnes již lze označit za klasické, poskytují určité prostředky pro první fáze analýzy. Typickým představitelem je unifikovaná metoda (UML) [Sklenář, Rational], která ve verzi 1.1 obsahuje rozšíření směrem k „business" modelování. Většina takových rozšíření objektových metod (včetně uvedeného) vychází z Jacobsonovy „Use-Case" analýzy [Jacobson]. Tento přístup sice jde správným směrem, ale jednotlivé pojmy „business" modelu jsou na příliš abstraktní úrovni a celkově jsou značně podřízeny a svázány s následnými pojmy modelů ve fázi podrobné analýzy a návrhu.

Notace diagramu „business" modelu podle návrhu UML ukazuje, že se jedná o poměrně velmi jednoduchý nástroj, který dokáže popsat systém pouze na velmi vysoké úrovni zjednodušení. Kromě toho, že tento nástroj nedovoluje uspokojivě zachytit mnohé pro zadání důležité informace *(např. pravidla, návaznosti, vazby mezi entitami, ...)*, tak je jeho další nevýhodou, která dále snižuje jeho použitelnost pro náležitý popis „business" procesů, jeho samotná koncepce a pojmový aparát, protože dle názoru autora má přímou vazbu na programovou strukturu navrhovaného systému, tedy např. na implementaci jednotlivých modulů apod. V případě potřeby zachycení většího detailu o modelovaném systému je doporučováno použití následných nástrojů UML, jako např. object interaction diagram, object-class diagram etc.

Bohužel ani jeden z vyjmenovaných nástrojů nedosahuje takové míry srozumitelnosti, jednoduchosti a zároveň výstižnosti, jaké ve své době dosahoval dodnes známý a používaný ER diagram nebo DF diagram. Například notace UML sice pokrývá většinu situací, se kterými se při projektování může setkat tvůrce softwaru *(i když dle názoru autora i zde jsou ještě rezervy především směrem k návrhu objektových databází)*, ale je pro neprogramátora velmi komplikovaná a obtížně čitelná. Na rozdíl od dříve používaného ER či DF diagramu, který je laikovi čitelný po čtvrthodinovém výkladu *(zde je myšleno pouze postačující pasivní pochopení - ne schopnost takový model vytvářet)*, nelze prostředky UML nasadit například při jednáních se zadavateli a s pomocí nich ověřovat a korigovat správnost zadání, což by jistě bylo žádoucí, jenomže tyto prostředky jsou orientovány více na podporu softwarové implementace než na možnost podrobného modelování z pohledu vlastního zadání.

Proč provádět „business" analýzu

Samotný pojem „business" analýzy informačních systémů je poměrně novou a v současné době velmi rozvíjenou aplikací OOP v praxi. Tato oblast modelování informačních systémů má dvojí na sobě poměrně nezávislý původ:

1. Prvním je potřeba formálního podchycení prvních fází vývoje informačního systému nástroji používajícími jiný pojmový aparát, než nástroje klasického softwarového inženýrství právě z výše popsaných důvodů. Tato potřeba vznikla

na základě zkušenosti, že zahájení tvorby IS nějakou „klasickou" objektovou metodou zbavuje většinu zainteresovaných osob ze strany zadavatele schopnosti sledovat projekt již od samého počátku.

2 Jako druhá příčina rozvoje se považuje rozvoj tzv. „business process" reinženýringu (BPR) právě v souvislosti se zjištěním, že myšlenky OOP jsou velmi přínosné pro konstrukci nástrojů a technik pro potřeby BPR, jak dokazuje např. [Eagle] a [Objecta].

Využití myšlenek OOP a znalostí a postupů známých z „klasických" metod tvorby softwaru a aplikovaných ve fázi business analýzy z této fáze činí důležitou a novou poměrně ucelenou oblast nástrojů a technik, která je označována jako „business" modelování (*jiní autoři používají též označení „esenciální"*) v návrhu IS.

Je velmi zajímavé, že fáze „business" modelování je v procesu návrhu IS použitelná hned trojím možným způsobem:

1 Jako fáze, která předchází „klasické" konceptuální modelování. V tomto případě je využita jako **předstupeň pro tvorbu konceptuálních modelů** (např. v notaci UML), kdy plní úlohu spojovacího a komunikačního mezičlánku mezi zadavateli a tvůrci softwaru v úvodních fázích vývoje.

2. Jako fáze, která může být provedena jiným subjektem, než následná „klasická" analýza a návrh. V tomto případě je použita jako ucelená metoda pro **tvorbu strukturovaných podkladů a dokumentace**, která dále slouží jako výchozí bod tvorby IS prováděné jiným subjektem např. na základě následně provedeného výběrového řízení.

3. Jako samostatná metoda, která slouží pro tvorbu informačního modelu **pro potřeby BPR** a to jak zmapování stávajícího stavu, tak i pro popis návrhu budoucího stavu jednotlivých modelovaných procesů. Zde se objektové modelování dokonce jako metoda úplně osamostatňuje a vlastně nepřímo dokazuje větší uplatnitelnost myšlenek OOP než pouze v procesu tvorby programů, jak bylo naznačeno v úvodu článku.

Ze zkušeností autora vyplývá, že varianta 2) a především varianta 3) je dnes v praxi vyžadována, je součástí postupů např. velkých konzultačních firem a lze v blízké budoucnosti očekávat její další rozvoj.

Jak provádět „business" analýzu

Z výše uvedených informací vyplývá, že **není možné pracovat ve všech etapách tvorby IS se stejným pojmem objektu**. Možný způsob nazírání na celou problematiku zobrazuje následující schéma:

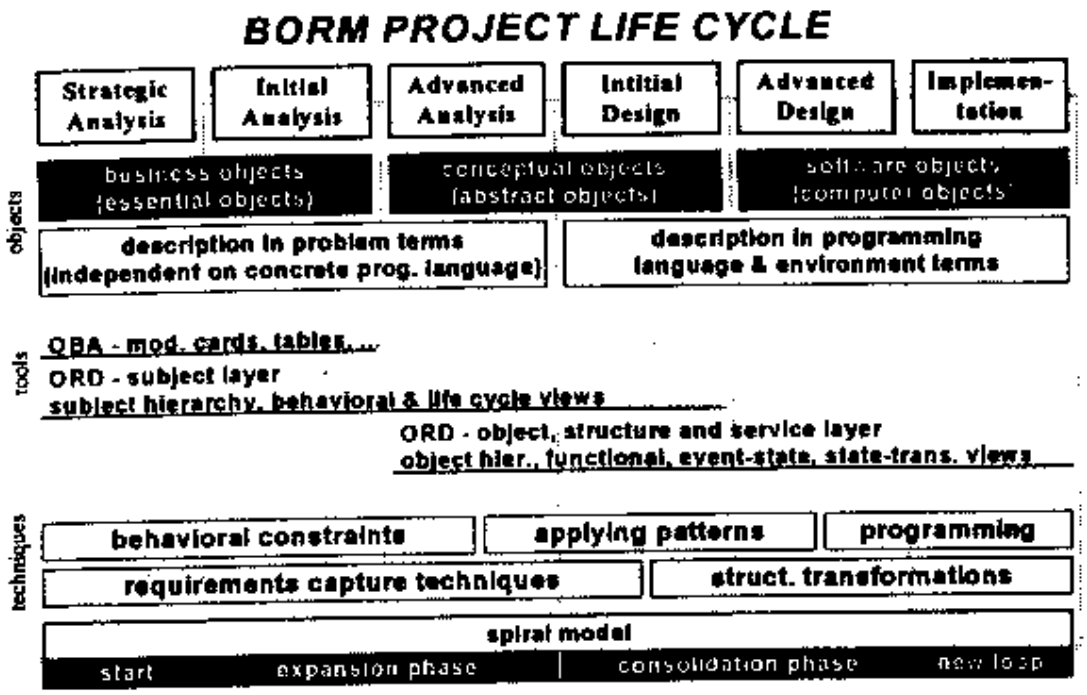
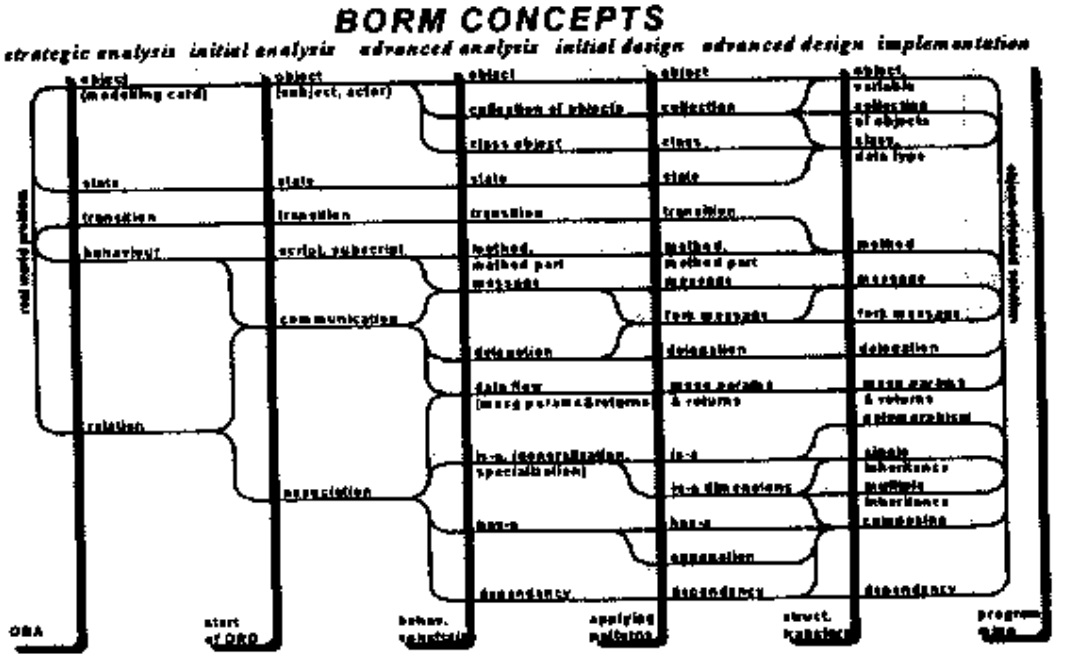


Schéma popisuje hlavní kroky metodologie Business Object Relation Modeling [BORM]. Podrobnějším výkladem metodologie se také zabýval článek [Merunka1] přednesený na této konferenci v minulém roce.

Na základě diskutovaných skutečností lze očekávat, že jednotlivé atributy a vazby mezi objekty i samotné nazírání na pojem objekt se v průběhu vývoje IS mění. Tuto skutečnost je možné v BORMu popsat následujícím schématem:



Nejedná se jen o to, že je na pojem objekt a na jeho vazby v úvodních fázích analýzy nahlíženo jednodušším způsobem na vyšší úrovni abstrakce. To by bylo příliš velké zjednodušení. Diagram ukazuje, že na proces návrhu IS lze také nahlížet jako na **postupné transformace modelu systému** skládajícího se z množiny objektů a jejich vazeb od vymezeného problému reálného světa až k softwarovému řešení.

[Merunka2] Tyto postupné trasformace jsou v obrázky vyznačeny jako svislé zaoblené obdélníky.

Samozřejmě, že softwarové objekty pokrývají vlastnosti konkrétních programovacích jazyků a vývojových prostředí a proto tyto objekty mají celou řadu vlastností specifických pouze pro fázi implementace bez smysluplné analogie v pojmech reálného světa a „business" objektů. O tomto samozřejmě není třeba pochybovat. Jde však i o to, že také mezi „business" objekty existují **pro model důležité vlastnosti**, které **nejsou přímo podporovány v současných** (*ani objektově orientovaných*) **programovacích jazycích a** musejí se v případě potřeby složitě implementovat pomocí dostupných konstruktů například s použitím určitých návrhových vzorů. Je to především:

- Schopnost zachycení objektu proměnlivého v čase do té míry, že objekt v různých stavech reaguje na stejné podněty (zprávy) různým způsobem.
- Modelování procesů v systému pomocí provázání stavů a přechodů několika vzájemně na sebe působících objektů mezi sebou.
- Business objekty nepředstavují pouze jakési „zárodky" budoucích datových objektů v softwarovém systému, ale také např. uživatele budoucího systému a další podobné entity, které nemusejí být ve fázi implementace softwarově přímo reprezentovány.

Metoda OBA

Metoda OBA (*Object Behavioural Analysis*) je typickým představitelem pokročilé techniky sloužící k získávání strukturovaných podkladů ze zadání pro potřeby konstrukce prvotního objektového modelu. Právě proto je velmi vhodná pro nasazení v počáteční fázi tvorby IS, kde výstupy OBA analýzy slouží ke konstrukci diagramů „business" objektů.

Metoda vnikla počátkem 90. let na základě zkušeností s aplikacemi různých technik JAD (*Joint Application Design*) a CRC (*Class-Responsibility-Collaborator*) pro potřeby objektové analýzy a návrhu a implementace v objektově orientovaných programovacích jazycích.

Jedná se o iterativní techniku začínající řízeným interview se zadavateli a pracující s různými typy formulářů, tabulek a modelových karet, ke kterým přísluší sada postupů a pravidel. Podrobnější popis OBA analýzy lze například nalézt v [OBA].

Jednotlivé kroky OBA analýzy jsou následující:

1. krok - **rozpoznání procesů** (plánování scénářů). V tomto kroku se na základě provedeného interview sestaví **seznam požadovaných funkcí systému** a **tabulka scénářů systému**. Jedná se vesměs o textové popisy, přičemž v nejjednodušší variantě se u každého scénáře rozlišuje původ procesu, vlastní popis procesu, participující entity a popis výsledku procesu.
2. krok - **definování objektů pomocí modelových karet**. V tomto kroku se pro každý rozpoznaný objekt z předchozího kroku vytvoří jeho modelová karta, která obsahuje jméno objektu, seznam aktivit objektu a s ním související seznam s modelovaným objektem spolupracujících objektů. Předpokládá se, že pro každý rozpoznaný spolupracující objekt je také vytvářena jeho modelová karta.

3. krok - **klasifikace objektů**. V tomto kroku dochází k přidání další informace k modelovým kartám jednotlivých objektů. Modelové karty jsou tříděny podle různých kriterií a podle určitých pravidel dochází ke vzniku nových modelových karet s novými objekty.
4. krok - **sestavení tabulky vztahů mezi objekty**. Tabulka vztahů v nejjednodušší podobě vyjadřuje jaký objekt má vztah s jiným objektem.
5. krok - **modelování životních cyklů objektů**. V tomto kroku se pro každý rozpoznaný objekt s pomocí informací v tabulce scénářů, modelových kartách a tabulkách vztahů sestaví životní cyklus objektu jako sled jeho stavů a přechodů mezi těmito stavy.

Metoda OBA je přímo založena na předpokladu iterativního přístupu k analýze. Například jednotlivé scénáře z 1. kroku jsou v 5. kroku příslušným předepsaným způsobem konfrontovány s životními cykly jednotlivých objektů a kontroluje se jejich vzájemná úplnost a souvislost. Následné kroky OBA tedy mohou posloužit i jako podklady pro dodatečné upřesňování informace v krocích předchozích. (*Pro varianty známých řešení se doporučuje provést 2 až 3 opakování všech kroků*).

OBA pomáhá získávat strukturovaným způsobem potřebné podklady k sestavení prvotních objektových diagramů. Má však i další zajímavé přínosy do procesu tvorby IS:

1. poskytuje prostředky pro dokumentování projektu od samého počátku,
2. modelové karty a další výstupy OBA jsou znovupoužitelné v dalších podobných projektech (například jako návrhové vzory) a
3. úsilí vynaložené při sestavování scénářů a životních cyklů objektů lze zužitkovat při návrhu optimální funkčnosti uživatelského rozhraní.

Metodu OBA lze provádět dokonce jen s tužkou v ruce a příslušnými předtištěnými formuláři a tabulkami na papíře. Samozřejmě lepším způsobem je použití CASE nástroje, který dokáže většinu rutinních operací (*například různé vzájemné kontroly, udržování projektových dat v konzistentním tvaru a možnost tisku tabulek a formulářů*) provádět automaticky. (*bude dále diskutováno v kapitole: softwarové nástroje „business" analýzy*)

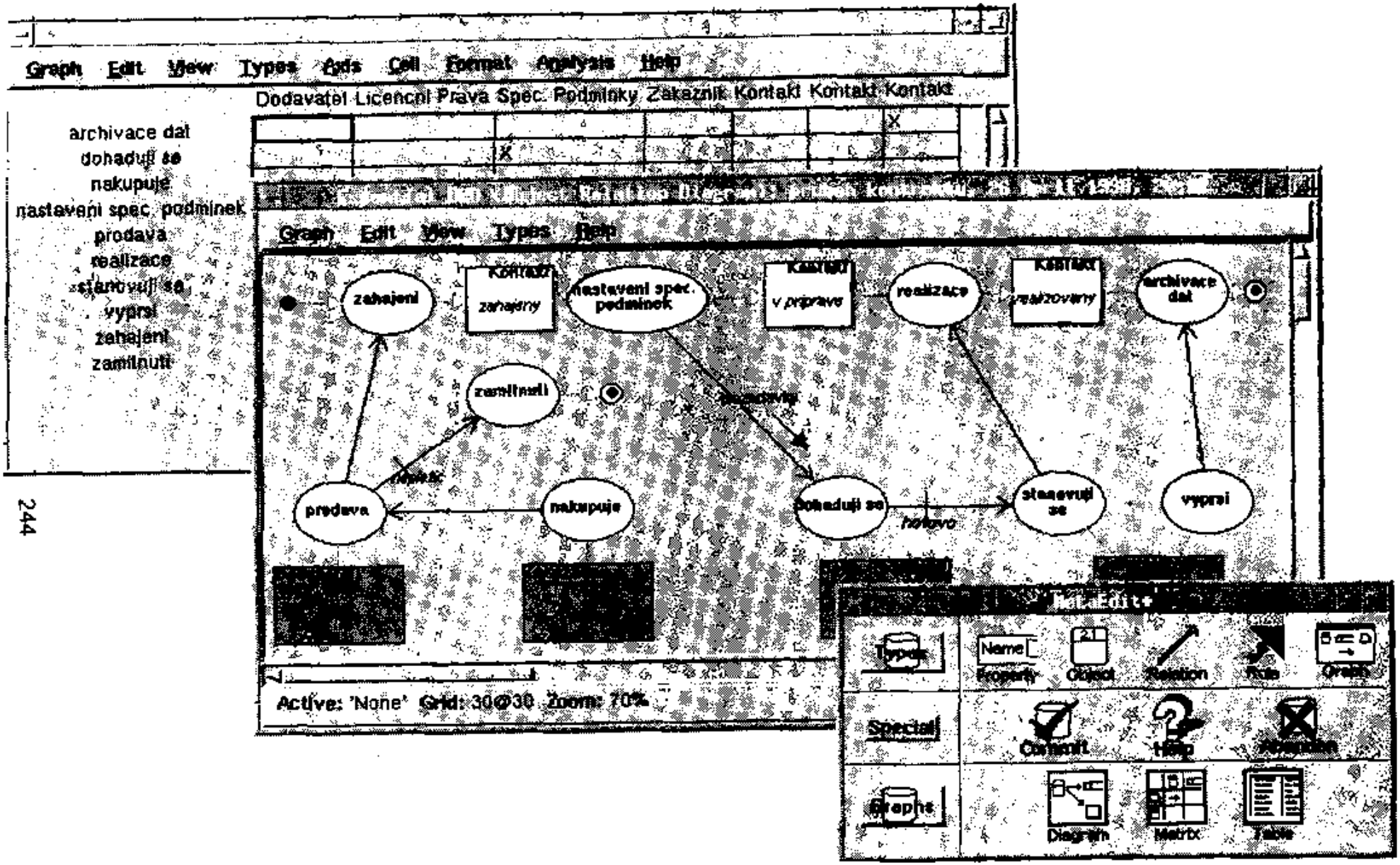
Diagramy pro „business" analýzu

Již bylo řečeno, že pro potřeby konstrukce modelu s „business" objekty je nevhodné použití „klasických" konceptuálních diagramů. Z tohoto důvodu autoři jednotlivých metod doplňují svoje metody o různá rozšíření. Diagram, který by byl vhodný pro konstrukci modelů „business" objektů musí splňovat následující podmínky:

1. Musí být podstatně jednodušší a srozumitelnější než konceptuální diagramy typu object-class diagram, jak je známe např. z OMT nebo UML. Diagram musí být čitelný i pro „neprogramátory".
2. Musí zobrazovat objekt, jeho jméno a aktivity (metody).
3. Musí zobrazovat vazby (asociace) mezi jednotlivými objekty i přímo mezi aktivitami jednotlivých objektů (komunikace = zprávy). Není potřebné vyjadřovat detailní podrobnosti vazeb jako např. různé varianty dědičnosti a nebo agregací, neboť se jedná až o podrobnosti potřebné až při modelování softwarových objektů.

4. Musí zobrazovat stavy a přechody jednotlivých objektů v čase včetně souvislostí, jaké aktivity a jaké vazby má příslušný objekt v příslušném stavu. Množina spolu souvisejících stavů a přechodů jednoho objektu je označována jako „role objektu" v systému. Předpokládá se, že jeden objekt může mít v systému více rolí.
5. Musí dovolovat propojení spolu souvisejících stavů a přechodů různých objektů. Právě tato propojení, která vlastně obsahují propojené role různých objektů, slouží k popisu jednotlivých „business" procesů v systému. Tyto modely procesů (agend, zákaznických procesů,...) podrobně popisují způsoby užití („use-cases") modelovaného systému.

Následující obrázek je ukázkou velmi jednoduchého zákaznického procesu pomyslného kontraktu na softwarový produkt v notaci BORM. Diagram ukazuje sled stavů a přechodů (operací) hlavního objektu Kontrakt a jeho souvislosti s dalšími spolupracujícími objekty. Obrázek byl pořízen pomocí softwarového nástroje Metaedit [Metaedit]


Graph Edit View Types Help
Dodavatel Licenční Práva Spec. Podmínky Zákazník Kontakt Kontakt Kontakt
archivace dat
dohaduji se
nakupuje
nastavení spec. podmínek
prodava
realizace
stanovuji se
vyprsi
zahajení
zamítnutí
Graph Edit View Types Help
zahajeni
Kontakt
zahajeny
nastaveni spec. podmínek
Kontakt
v přípravě
realizace
Kontakt
realizovány
archivace dat
zamítnutí
prodava
nakupuje
Dohaduji se
stanovuji se
vyprsi
Active: 'None' Grid: 30@30 Zoom: 70%
Name
Property
Object
Relation
Role
Graph
Special
Comment
Help
Attribute
Graphs
Diagram
Matrix
Table

Softwarové nástroje pro „business" analýzu

Softwarová podpora „business" analýzy je již dnes **částečně zahrnuta** ve většině profesionálních CASE objektových nástrojích, jako například Rational Rose, Select nebo Paradigm Plus a další.

Kromě toho existují i **specializované CASE nástroje** pro podrobnou „business" analýzu, které však nejsou snadno dostupné na běžném softwarovém trhu, protože kromě toho, že jsou velmi drahé a málo poptávané, tak jsou většinou součástí know-how příslušných firem - viz. např. [Eagle] a [Objecta].

Pokud nutně nepotřebujeme drahý profesionální CASE, je také možné použít některý z inteligentních programů pro tvorbu **prezentační grafiky** (např. Visio nebo Visual Though) a opatřit si nebo sám vyrobit šablonky pro podporu příslušné metody.

Vzhledem k rychlému rozvoji a změnám v této oblasti je především pro potřeby velké konzultační firmy a nebo na akademickou půdu vhodné použití některého z **meta CASE** nástrojů, které jsou někdy také označovány jako CAME - Computer Aided Method Engineering, jedná se o takové CASE nástroje, které kromě tvorby modelu (jako obyčejný CASE) také dovolují modelovat samotnou metodu - např. doplňovat symboly a vazby a kromě definování jejich grafické reprezentace i popisovat jejich chování např. pomocí programovatelného generátoru výstupních sestav a tím například měnit z modelu generovaný kód apod.

Je třeba zdůraznit, že použití meta CASE nástroje není vhodné jen kvůli možnosti úprav a doplňování podporované metody vlastní silou. Je tu i možnost **snadných úprav a změn používané metody na objednávku** dokonce v průběhu řešení projektu a samozřejmě bez nutnosti měnit celý softwarový produkt.

Autor sám má zkušenosti s programem Metaedit, který patří do kategorie meta CASE systémů. Jedná se o nástroj, který již podporuje 3 metody vhodné pro BPR (*Porter's Method, Business Systems Planning by IBM a Godkuhl's Activity Analysis*) a 6 metod OOA&D (*OMT, UML, Coad/Yourdon, Shlaer/Mellor, Fusion, Moses a Embley*). Generuje kód SQL, Smalltalku, C++, Delphi Pascalu a Javy. Pro ukládání dat a metadat používá víceuživatelskou třídně-instanční objektovou databázi s podporou transakcí. Kromě různých i uživatelsky definovaných výstupních sestav Metaedit dovoluje prezentovat informaci každého modelu trojím možným způsobem: 1) graficky jako diagram, 2) ve vztahových maticích a 3) v tabulkách, přičemž například změna informace v tabulce či vztahové matici může vyvolat automatickou změnu grafické podoby odpovídajícího objektu v diagramu.

V době psaní tohoto článku byla v tomto prostředku zahájena implementace metodologie BORM.

Závěr

OOP má v úvodních fázích analýzy informačních systémů nezastupitelnou úlohu. Dnes již neobstojí názor, že objektové prostředky jsou vhodné pouze jako lepší

implementační softwarové nástroje dovolující využívat například přednosti vizuálního programování a nebo používání hotových komponent.

Vzhledem k neustále rostoucí složitosti zadání informačních systémů a také vzhledem k souvislostem s reinženýringem „business" procesů je třeba úvodním fázim analýzy prováděné objektovým způsobem věnovat náležitou pozornost.

Literatura

[BORM] Knott R P, Polák J, Merunka V.: **Business Object Relation Modeling Methodology**, závěrečná zpráva k řešení projektu VAPPIENS, British Council - Know-How Fund, Dept. of Computer Studies, Loughborough University.

[Eagle] Andersen Consulting, **The Project Eagle**, http://www.ac.com/aboutus/tech/eagle/

[Jacobson] Jacobson I.: **Object-Oriented Software Engineering - A Use Case Driven Approach**, Addison Wesley 1992, ISBN 0-201-54435-0

[Merunka1] Merunka V.: **Zkušenosti s objektově orientovanou analýzou a návrhem**, ve sborník konference Tvorba softwaru 97, Ostrava 1997

[Merunka2] Merunka V.: **Výuka OOP**, ve sborníku sborniku konference Objekty 97, Praha, listopad 1997

[Metaedit] Metacase inc.: http://www.jsp.fi/metacase

[OBA] Rubin K.S., Goldberg A.: **Object Behavioural Analysis**, pp 48-62 Communications of the ACM 35(9) 1992; Goldberg Adele, Kenneth Rubin S.: **Succeeding with Objects - Decision Frameworks for Project Management**, Addison Wesley 1995, ISBN 0-201-62878-3

[Objecta] Objecta Ltd.: **Object-Oriented Business Process Reengineering**, http://www.objecta.demon.co.uk/home.html

[Rational] Rational inc : **Unified Modelling Language**, http://www.rational.com

[Sklenář] Sklenář V.: **Jazyk UML**, ve sborníku konference Objekty 97, Praha, listopad 1997