Objektový versus strukturovaný přístup při analýze a návrhu informačních systémů

Helena Jiľková

„Objektový" životní cyklus projektu informačního systému se člení na analýzu (OOA – Object Oriented Analysis), návrh (OOD – Object Oriented Design) a implementaci (OOP – Object Oriented Programming). Pro větší nebo neznámé informační systémy je nejobtížnější fází analýza, v níž se vyhledávají objekty tzv. problémové domény a definuje postupně objektový model. Pojem **problémová doména** je jen jiným označením pro obsah, esenci informačního systému (např. problémová doména „knihovna", problémová doména „práce a mzdy"). **Objektový model** zachycuje problémovou doménu jako systém komunikujících a spolupracujících objektů.

Objektový model vytváříme postupně. Lit. [OOA] doporučuje postupovat po vrstvách (ne nutně; a ne striktně shora dolů!) a vidí v objektovém modelu **pět vrstev**:

- **vrstva subjektů** (pro problémovou doménu „knihovna" obsahuje například subjekty „evidence čtenářů", „výpůjční služby", „obnova knižního fondu");
- **vrstva tříd a objektů** (třídy mohou být abstraktní, sloužící jako zdroj děděných vlastností, avšak natolik obecné, že jejich konkrétní výskyty – objekty – nemají v modelu použití. Druhým typem jsou třídy, jejichž objekty v modelu žijí. Příkladem abstraktní třídy z „knihovny" může být TSvazek, který obsahuje několik společných atributů pro všechny typy publikací. Příkladem druhého typu je „čtenář");
- **vrstva struktur** (definuje vztahy mezi objekty. Vztahy mají charakter buď „generalizace–specializace", nebo „celek–část". Příkladem struktury typu „generalizace–specializace" je vztah mezi objekty „čtenář" a „čtenář–instituce", kde „čtenář" je generalizací a „čtenář–instituce" specializací, neboť má sice vlastnosti čtenáře, avšak vypůjčuje si a vrací publikace v odlišném režimu. Vztah „celek – část". může nastat např. mezi knihovnou a knihovníkem, jestliže konstruovaný objektový model vyjadřuje i to, z čeho se knihovna skládá);
- **vrstva atributů** (specifikuje atributy, tj. data zapouzdřená v jednotlivých třídách a objektech, a také vazby mezi instancemi objektů. Vazby mezi instancemi jsou do této vrstvy zařazeny proto, protože se předpokládá, že mají datový charakter. Příkladem atributu pro objekt „čtenář" je např. „číslo čtenářského průkazu");

– **vrstva služeb** (určuje služby, které každý objekt poskytuje, a spojení – „message connections", jimiž se ostatní objekty těchto služeb dožadují. Např. objekt „kartotéka čtenářů" poskytuje službu „najdi čtenáře dle čísla čtenářského průkazu". Tuto službu si dle své potřeby zasláním zprávy vyžádá objekt „výpůjční lístek".)

Máme navrhnout objektový modul. Můžeme využít jeho rozdělení do vrstev a víme zhruba, co jednotlivé vrstvy obsahují. Postupujeme s každou třídou/objektem průřezově po všech vrstvách, nebo postupujeme striktně shora dolů nejprve v rámci subjektů. Tento návrh považujeme za prvotní a provedeme jeho revizi podle intenzity mezisubjektové kooperace. Nebo postupujeme jakkoli. Podstatné je, že pro návrh vrstev (pro každou zvlášť) existují určitá rozumná vodítka, co dělat a na co nezapomenout. **Bohužel tato vodítka jsou nejslabší právě v klíčové vrstvě: jak nalézat třídy a objekty.** Srovnejme „návod" OOA např. pro vrstvu služeb versus vrstvu tříd a objektů.

Jak definovat služby – návod na odvození vrstvy.

■ identifikuj stavy objektu

(objekt žije a přechází tím ze stavu do stavu):

– urči možné hodnoty atributů
– urči, zda tyto hodnoty mají smysl: určení odvoď z toho, za co je navrhovaný systém jako celek odpovědný. Dále prověř, pro jaké skupiny hodnot se systém chová (v rámci svých odpovědností) různě: atributy nesou data potřebná pro poskytování služeb a zároveň v každém okamžiku vyjadřují, v jakém stavu se objekt nachází. To, že se systém „chová různě", bude znamenat, že i objekt bude muset mít několik rozlišených stavů;
– prověř výsledky dosavadní OOA a případně výsledky OOA pro podobné informační systémy. Tento bod se obvykle vyřídí velmi rychle, neboť není vcelku co prověřovat;
– popiš stavy a přechody diagramem;

■ identifikuj požadované služby:

– vytipuj algoritmicky jednoduché služby:

„create" – vytváří a inicializuje nový objekt ve třídě;
„connect" – připojuje objekt k jinému (resp. odpojuje objekt od jiného);
„access" – získává nebo nastavuje hodnoty atributu (atributů) objektu;
„release" – uvolňuje (odpojuje a maže) objekt;

– vytipuj algoritmicky složité služby:
  — prověř předchozí OOA;
  — předpokládej dvě základní kategorie služeb:

„calculate": počítá výsledek z hodnot atributů objektu;
„monitor": monitoruje externí systém nebo nařízení. Zpracovává externí vstupy a/či výstupy. Případně získává data ze zařízení nebo toto získávání řídí.

— pro odhalení konkrétních služeb se ptej:

„Za jaké výpočty na svých hodnotách objekt zodpovídá?"
„Co musí objekt sledovat, aby zachytil vnější změny a reagoval na ně?"

— pro označení nalezených služeb použij jména vyjadřující souvislost s problémovou doménou;

■ identifikuj spojení posíláním zpráv:

- pro každý objekt si polož dvě otázky:

  „Od jakých jiných objektů potřebuje služby?"

  (Nakresli ve schématu šipku ke každému z těchto objektů.)

  „Které jiné objekty potřebují služby tohoto objektu?" (Opět zakresli výsledek šipkou.)

- jdi po spojení k dalšímu objektu a otázky zopakuj;
- prověř výsledky předchozí OOA;
- zohledni požadavky, které jsou kritické pro provoz navrhovaného systému (např. čas pro real–time systémy). Zviditelni „ohrožení" u služeb a spojení zprávami, které jsou kompetentní se s ním vypořádat;

■ zkompletuj dokumentaci pro služby:

- prověř předchozí OOA;
- každou službu popiš standardizovaným způsobem (lit.[OOA] jeden z těchto způsobů zavádí a doporučuje);
- zviditelni další omezení (jsou–li);
- inventarizuj služby závislé na stavech.

Právě jsme přežili návod na odvození vrstvy služeb. Technologický postup to ovšem zdaleka není, ale když člověk uvidí alespoň jeden konkrétní příklad ke každému dílčímu kroku, dokáže si zhruba představit, co je třeba dělat. Srovnejme nyní odvozování služeb s odvozováním objektů a tříd.

Jak nalézat třídy a objekty

■ jak pojmenovávat:

- pojmenovává se jednotlivý objekt ve třídě, a to rozvitým či nerozvitým podstatným jménem v jednotném čísle;

- doporučuje se používat standardní terminologii problémové domény (např. tedy v knihovně „čtenář" a nikoli „žadatel");

■ kde hledat:

- pozoruj;
- poslouchej;
- projdi výsledky předchozí OOA (kde je však vzít?);
- prověř jiné systémy;
- čti;
- prototypuj;

■ co hledat:

- struktury;
- jiné systémy;
- zařízení;
- důležité věci nebo události;
- role;
- postupy;
- místa, kde se něco odehrává;
- organizační jednotky;

■ co zvažovat:

- co si systém musí pamatovat;
- jak je třeba, aby se systém choval;
- zda má objekt více než jeden atribut (je normální, aby objekty měly více atributů);
- zda je ve třídě více než jedna instance objektu (opět je normální, aby třída produkovala více instancí);
- výskyt stejných atributů u mnoha objektů;
- výskyt stejných služeb u mnoha objektů;
- požadavky problémové domény;
- co chybí, aby se daly vyprodukovat požadované výstupy.

Absolvovali jste návod, jak nalézat třídy a objekty. Už je vám jasné, jak ty objekty budete hledat? Hlavně nezapomeňte na požadavky problémové domény: čtenáře ani tak nezajímá, zda je pojímán objektově či strukturovaně. Avšak zajisté by si mnohdy rád půjčil knihu. Konec žertů. Tvorba objektového modelu a v rámci ní zvláště hledání tříd a objektů jsou nesmírně náročné. Mohou nám v této situaci pomoci strukturované přístupy?

Strukturovaný versus objektový přístup

Nejsou versus. Ve strukturovaném přístupu střídáme funkční a datovou analýzu a stále se strachujeme, aby se nám nerozjely funkce a data. V objektovém přístupu máme konečně data a funkce pohromadě a nemusíme je neustále provazovat. Těch starostí není míň, ale jsou více konsistentní. Objektový přístup představuje další historickou etapu rozvoje metodologií tvorby informačních systémů, avšak je dosud mlád. A zvláště v našich poměrech je málo podporován nástroji: kdo u nás má k dispozici objektově orientovaný CASE? Budeme si zoufat a/či oželíme objektový přístup? Nikoli! **Naším cílem je vytvořit dobrý informační systém.** Zhodnotíme cíl, možné postupy a nástroje a zvolíme strategii. **Nejpravděpodobnější výchozí situace:**

a) řešíme malý informační systém, jehož implementaci si lze představit jako jeden programový systém s jedním uživatelským rozhraním. (Informační systém knihovny by velmi pravděpodobně byl tímto případem.)

b) řešíme středně velký až velký informační systém a máme k dispozici prostředky CASE.

c) řešíme středně velký až velký informační systém a nemáme k dispozici prostředky CASE.

ad a) **Malý informační systém**

Jestliže máme k dispozici objektové implementační prostředí (např. Turbo Vision), postupujeme objektově a strukturovaným přístupem se zabýváme jen do té míry, jak se nám chce. Co lze např. využít při řešení dílčích problémů **hledání tříd a objektů:**

ad ■ **jak pojmenovávat:**

Data story, které máme v DFD, jsou pouze esenciální (to víme, protože jsme je tak navrhli). Jejich označení odpovídají entitám z datového modulu. Hle, máme některé objekty a zároveň i jejich vhodná jména.

ad ■ **kde hledat:**

Objektové implementační prostředí poskytuje stavební kameny pro uživatelský interface, správu dat i řízení událostmi. Jestliže navrhujeme objekty z těchto komponent, vycházíme ze zázemí existujících knihoven. Podle analogie se však nemůžeme řídit při výstavbě objektového modelu problémové domény. Tato neanticipuje knihovny objektů a ani anticipovat nemůže. Zde by bylo třeba projít výsledky OOA pro obsahově podobné informační systémy. Nemáme-li je, je možné a užitečné udělat jednoduchý esenciální model. Funkce v DFD ukazují potenciální služby, normalizovaná data story potenciální objekty, datové toky potenciální atributy, datové či řídicí toky typu signál ukazují na které externí události musí modelovaný systém reagovat atd. Je toho velmi mnoho, co lze

z rozumně provedené strukturované analýzy vytěžit. Třeba lze na funkčním modelu i dobře zjišťovat, zda systém dělá vše, co dělat má: simulace je jednoduchá, protože model je jednoduchý.

ad ■ **co hledat:**

Zvláště dobře se s pomocí funkčního a datového modelování inventarizují události a výstupy jakožto reakce na tyto události. Dále role a postupy, neboť DFD vyjadřující esenciální model není nic jiného, než popis rolí jednotlivých objektů a popis postupu, v němž se tyto role střídají.

ad b) **středně velký až velký informační systém s CASE**

Klíčovými body u velkých projektů je plánování a řízení (aby se to vůbec zvládlo udělat) a dokumentace (aby se to vůbec dalo provozovat a udržovat). Jestliže máte CASE, pak jej nutně musíte využít. Je-li objektově orientovaný, není o čem mluvit. Pravděpodobně však bude spíše podporovat některý z postupů strukturované analýzy a návrhu informačních systémů. Pak doporučuji postupovat strukturovaně až do chvíle, kdy bude celý systém rozdělen na části, které budou samostatně implementovány. (Nezapomenout na precizní vymezení rozhraní mezi částmi!). Tyto jednotlivé části pak již mohou být dále řešeny postupy OOA, OOD a OOP. Bez zábran a předsudků pak v těchto postupech využíváme, co již o informačním systému víme díky strukturovanému přístupu. Ilustrujme toto tvrzení na příkladu **odvození vrstvy služeb:**

ad ■ **identifikuj stavy objektu:**

Lze využít diagramů přechodu stavů, kde jsou již zformovány podmínky a akce: dle nich lze definovat atributy a služby.

ad ■ **identifikuj požadované služby:**

Pro tzv. algoritmicky jednoduché služby nám výsledky strukturované analýzy příliš nepomohou. Tyto služby jsou totiž typicky objektovou záležitostí. U tzv. algoritmicky složitých služeb můžeme v DFD přímo nacházet funkce, jež mají charakter služby „calculate", resp. „monitor" daného objektu. Nic nám nebrání, abychom již v okamžiku konstrukce DFD členili funkce – služby do samostatných procesů právě dle kriteria jejich typu. Otázky „Za jaké výpočty na svých hodnotách objekt zodpovídá?" a „Co musí objekt sledovat, aby zachytil vnější změny a reagoval na ně?" jsou DFD přímo zodpovězeny: funkce zpracovává své vstupní datové toky na výstupní, přijímá a vysílá řídicí signály a její „vnitřek" je popsán buď jazykem, nebo rozkladem na nižší úroveň;

ad ■ **identifikuj spojení posíláním zpráv:**

Pro objekt – data story lze přímo z DFD vidět, které „služby" s nimi manipulují. Lze tedy odvodit (přes funkce), které objekty „službu" potřebují.

ad b) středně velký až velký informační systém bez CASE

V této situaci je kritickým faktorem úspěchu projektu jeho brzké rozdělení na víc izolovaných, samostatně řízených částí, z nichž každá bude ve fázi analýzy a návrhu zvládnutelná v týmu 5–7 lidí. Často se v rámci celého informačního systému vybere pouze několik tzv. strategických úloh a tyto se řeší přednostně. **Volba strukturovaného či objektového přístupu by měla být zcela záležitostí řešitelského týmu.**

Literatura

[OOA] Coad,P.,Yourdon,E.:
Object Oriented Analysis, Yourdon Press, Prentice Hall, 1991

[OOD] Coad,P.,Yourdon,E.:
Object Oriented Design, Yourdon Press, Prentice Hall, 1991

[YSM] Jilková,H.,Stanovská,I.:
Strukturované postupy analýzy a návrhu informačních systémů, Programování'92, Ostrava, 1992

Autor: Ing. Helena Jilková, CSc.
VŠE, katedra IT,
nám. W.Churchilla 4,
13000 Pha 3
tel: 2125437