Objektově orientovaný návrh

- Kdy je objektově orientovaný návrh dobrý a kdy špatný, jak se to pozná?

- Zapouzdření a spojitost
- Různé abstrakce návrhu – jiná použitelnost, domény tříd, zatížení tříd
- Soudržnost tříd
- Stavový prostor, neměnné podmínky třídy, předběžné a následné podmínky operací
- Přizpůsobení typů a uzavřené chování – robustní hierarchie tříd
- Nebezpečí mechanismů dědičnosti a mnohotvarosti
- Kvalita rozhraní třídy

Zapouzdření a spojitost (vzájemná závislost)

- Existují už na strukturované úrovni, v oop další význam
- Důležité vlastnosti pro pochopitelnost a zpracovatelnost oo softwaru

Struktura a úrovně zapouzdření

- První princip zapouzdření (o úroveň výše než samotná struktura kódu) = podprogram = úroveň 1 (kód úroveň 0)
- Třída (objekt) – seskupení podprogramů a vlastností = úroveň zapouzdření 2
- Úrovně 3 a 4 - balíčky a komponenty (viditelné jen některé třídy nebo části jejich rozhraní)
- Úroveň 3
    - horizontální seskupení tříd – podle subjektu (třídy spolu nemusejí spolupracovat) – odlišné balíčky pro odlišné subjekty
    - vertikální seskupení tříd – vzájemně spolupracující třídy – obvykle z několika domén - komponenty

Od:/K:	Úroveň 0 (řádek kódu)	Úroveň 1 (procedura)	Úroveň 2 (třída)
Úroveň 0 (řádek kódu)	Strukturované programování	Vzdušnost	-
Úroveň 1 (procedura)	soudržnost	Vázání	-
Úroveň 2 (třída)	-	Soudržnost tříd	Vázání tříd

- vzdušnost (fan out) – míra počtu odkazů na další procedury (podle počtu řádků v dané proceduře)
- soudržnost (cohesion) – míra naplnění jednoho smysluplného chování dané procedury
- vázání (coupling) – míra počtu a síly spojení mezi procedurami
- soudržnost tříd - míra jednoho smysluplného chování dané třídy (soustředěnost operací a atributů na naplňování smyslu třídy)

- vázání tříd – míra počtu a síly spojení mezi třídami

A možnosti dalších vztahů mezi jednotlivými úrovni – hledání universa – kandidát: spojitost

Spojitost

- spojité sw prvky jsou zrozeny ze související potřeby (během analýzy, návrhu, programování)
- definice spojitosti pro sw: Spojitost mezi dvěma sw elementy A a B znamená:
    - lze postulovat nějakou změnu A, která bude vyžadovat i změnu B (nebo alespoň důkladné prověření), aby byla zachována celková správnost
    - lze postulovat změnu, která bude vyžadovat společnou změnu A i B, aby byla zachována celková správnost

Varianty spojitosti

A: int i;
B: i:=5;

- spojitost typu (změna int např. na char)
- spojitost názvu (změna i na j)

- explicitní spojitost – přímo vidíme
- implicitní spojitost

I: JUMP J+15
...
J: LR ...
...
K: (sem se skáče z I)

- čím implicitnější spojitost, tím hůře se hledá.
- sw elementy spolu nemusejí komunikovat, aby byly spojité

Spojitosti jsou

- směrové
  - jednosměrně spojité (spojitosti názvu – např. dědění)
  - obousměrně spojité
- nesměrové – explicitně na sebe neukazují (používají např. stejný algoritmus)

Statická spojitost – platí na úrovni tříd¨

- názvu - proměnné mají stejný název, aby se odkazovaly na stejnou věc

- typu (třídy) – např. proměnnou typu int naplníme celočíselnou hodnotou

- konvence – např. osobní čísla začínající A patří studentům aplikovaných věd

- algoritmu – např. používáte kódovací tabulku, do které vkládáte a z které vybíráte prvky, musíte používat stejný kódovací algoritmus

- pozice – sekvence řádků kódu, pořadí argumentů

Dynamická spojitost

- vykonávání – ekvivalent spojitosti pozice (inicializace proměnné před použitím, nastavení a testování semaforu, apod.)
- časování – systémy reálného času
- hodnoty – většinou omezení hodnot (rohy obdélníka musí i při posunu zachovat obdélník – nelze posunout jeden roh)
- identity – dva objekty odkazují na tentýž objekt

- „nespojitost“ (spojitost rozdílu) - dvě různé proměnné se nemohou jmenovat stejně, problém u vícenásobné dědičnosti

Spojitost a hranice zapouzdření

- bez existence zapouzdření – problémy
    - množství spojitosti (nespojitosti)
    - co je spojitost a co jej jen náhodná podobnost

- Spojitost – důvod funkčnosti objektové orientace

Spojitost a správa kódu

- pro lepší správu kódu:
    - minimalizace spojitosti (a také nespojitosti) – tj. rozdělení celku do zapouzdřených částí
    - minimalizace spojitosti překračující hranice zapouzdření
    - maximalizace spojitosti v hranicích zapouzdření

- jinými slovy: Podobné věci patří k sobě, odlišné věci je třeba oddělit… a podobné věci jsou ty, které vykazují vzájemnou spojitost - platí nejen pro úroveň 2, ale i pro vyšší úrovně

Jak se spojitost hledá?

- explicitní lehce – stačí editor
- implicitní špatně – zkušenost, nadhled

Zneužití spojitosti

- funkce přítele (friend) v C++ - má přímo účel narušovat hranice zapouzdření
- neomezená dědičnost – třída využívá „neomezeně“ interně viditelné prvky nadřízené třídy – spojitost názvu, třídy,… -> dědičnost abstraktního chování se nesmí rovnat dědění interní implementace
- spoléhání se na obecně stejné pochopení nedokumentované implementace

Domény tříd

Doména aplikace – třídy použitelné pro jednu aplikaci

- třídy správy událostí – vykonávání činnosti pro vzniklou událost
- třídy rozpoznávání událostí – detekce událostí

Doména činnosti – třídy použitelné pro jeden obor či organizaci

- třídy vztahů – asociace mezi objekty daného oboru – např. VlastnictvíPsa
- třídy rolí – role objektů v daném oboru např. Pes, Klient, Pacient
- třídy atributů – vlastností objektů v dané oblasti – např. ZůstatekNaÚčtu

Doména architektury – třídy použitelné v rámci jedné počítačové architektury (nikoli architektury návrhu viz dříve)

- třídy uživatelského rozhraní - např. Okno
- třída pro práci s databází – např. Transakce
- třídy pro komunikaci s přístroji – např. Port

Doména základu – třídy obecně použitelné

- sémantické třídy – např. Datum
- strukturální třídy – např. Fronta
- základní třídy – např. Int

Třídy vyšších domén jsou utvářeny třídami z nižších domén

Domény a znovupoužitelnost

- knihovny tříd základu – se kupují (součást standardních knihoven), největší znovupoužitelnost
- knihovny tříd architektury – se většinou kupují, ale nemusejí být kompatibilní se základem, vyžadují úpravy a rozšíření, „střední“ znovupoužitelnost
- knihovny tříd činnosti – se dají koupit, ale často vyhovují jen obecným aspektům oboru, „střední“ znovupoužitelnost
- knihovny tříd aplikace – píšete si sami, prakticky nemožné dosáhnout znovupoužitelnost

Zatížení

- jak vysoko se nachází třída nad základem
- zahrnuje další třídy, na které se třída musí spoléhat
- celkový počet přímých a nepřímých odkazů ze třídy T – měřítko složitosti třídy
- vyšší doména – vysoké nepřímé zatížení, nižší doména naopak
- neočekávané vysoké nepřímé zatížení v nízké doméně naznačuje chybný návrh třídy, nízké nepřímé zatížení ve vyšší doméně – pravděpodobně třída je tvořena přímo třídami základu

Zákon Demeter – princip omezení přímého zatížení třídy omezením velikosti jejích přímých odkazů (dvě verze –silný a slabý)

Pro objekt **obj** třídy T a pro libovolnou operaci **op** definovanou pro **obj** musí být každým cílem zprávy v implementaci **op** jeden z následujících objektů

- samotný objekt **obj**
- objekt odkazovaný argumentem v podpisu **op**
- objekt odkazovaný nějakou proměnnou **obj** (včetně objektů v kolekcích, na které se **obj** odkazuje)
- objekt vytvořený **op**
- objekt, na který se odkazuje globální proměnná

Zákon omezuje umělé odkazy na další třídy v rámci dané třídy, omezuje spojitost mezi hranicemi zapouzdření (silný zákon) – usnadnění správy a vývoje systémů

Soudržnost třídy

- míra vzájemné souvislosti prvků (atributů a operací) umístěných v externím rozhraní třídy
- problémy pozorovatelné v rozhraní třídy

Smíšená soudržnost instancí – třída má prvky, které jsou pro určité objekty třídy nedefinované – indikuje nedokonalou hierarchii tříd a nutnost zavést třídy na nižší abstraktní úrovni

Smíšená soudržnost domén - třída obsahuje prvek, který přímo zatěžuje danou třídu nevlastní třídou jiné domény (třída B je nevlastní třídě A, pokud lze A plně definovat bez odkazu na B) – indikuje, že jsme třídu zatížili třídou jiné než nižší domény (často se tam plete např. doména architektury) – třídy bychom měli zatěžovat pouze třídami z nižších domén

Smíšená soudržnost rolí - třída obsahuje prvek, který přímo zatěžuje danou třídu nevlastní třídou ležící v téže doméně

Závěrem

- největší nebezpečí tvoří smíšená soudržnost instancí, nejmenší smíšená soudržnost rolí

Stavový prostor a chování

= vlastnosti třídy

- stavový prostor – celek všech povolených stavů libovolného objektu třídy T
- dimenze stavového prostoru přibližně odpovídá atributům definovaným v dané třídě (nelze počítat atributy, které lze odvodit z jiných)

Stavový prostor podřízené třídy

- je-li třída B podřízenou třídou třídy A, pak stavový prostor třídy B musí být zcela obsažen ve stavovém prostoru třídy A.
- je-li třída B podřízenou třídou třídy A, pak stavový prostor podřízené třídy B musí zahrnovat přinejmenším dimenze třídy A, může jich však obsahovat i více

Chování podřízené třídy

- povolené chování třídy T je množina přechodů, které může třída T vykonat mezi stavy ve stavovém prostoru T

- je-li třída B podřízenou třídou třídy A, pak chování třídy B je omezeno chováním třídy A
- je-li třída B podřízenou třídou třídy A, pak chování třídy B rozšiřuje chování třídy A

Neměnná třídy – omezení stavového prostoru

- neměnná třídy je podmínka, kterou musí v každém okamžiku naplnit každý objekt dané třídy (když je objekt v rovnováze)

Předběžné a následující podmínky

- vztahují se k operacím
- předběžná podmínka musí být pravdivá na začátku operace, pokud není, může operace odmítnout vykonání a vyvolat stav výjimky
- následná podmínka musí být pravdivá na konci vykonávání operace, v opačném případě je implementace operace nesprávná je třeba ji opravit

- podmínky společně formují kontrakt s klientem

Přizpůsobení typu a uzavřené chování

- vytváření robustních hierarchií tříd
- typ a třída, typ a rozhraní
- hierarchie tříd, hierarchie typů

Přizpůsobení typu

- je-li P skutečným podtypem T, pak P musí být přizpůsobeno T. (Objekt typu P může být poskytnut v libovolném kontextu, kde je očekáván objekt typu T)

- hierarchie třída/podtřída musí odpovídat principu přizpůsobení typu

- jak zajistit přizpůsobení typu tedy polymorfismus?

Principy kontravariance a kovariance

1. Neměnná podřízené třídy je přinejmenším tak silná jako neměnná nadřízené třídy
2. Každá operace nadřízené třídy má odpovídající operaci v podřízené třídě se stejným názvem a podpisem
3. Předběžná podmínka každé operace není silnější než odpovídající podmínka operace v nadřízené třídě – princip kontravariance
4. Následná podmínka každé operace je přinejmenším tak silná jako odpovídající podmínka operace v nadřízené třídě – princip kovariance

Souhrn požadavků na přizpůsobení typu

- má-li být podřízená třída P podřízeným typem třídy T, pak musí být splněna následující omezení

1. Stavový prostor P musí vůči stavovému prostoru T splňovat podmínky uvedené výše

Pro každou operaci třídy T (T.op), kterou P přepisuje operací P.op, platí

2. P.op musí mít stejný název jako T.op
3. Seznam argumentů formálního podpisu operace P.op musí odpovídat seznamu argumentů formálního podpisu operace T.op
4. Předběžná podmínka P.op musí být rovna nebo slabší než předběžná podmínka T.op
5. Následná podmínka P.op musí být rovna nebo silnější než následná podnínka T.op

Princip uzavřeného chování

- přizpůsobení typu vede k dobrému návrhu v operacích přístupu, samotné však nestačí
- operace změny – princip uzavřeného chování – chování zděděné podřízenou třídou z nadřízené třídy musí respektovat neměnnou dané podřízené třídy
- k uzavření podřízené třídy do chování třídy nadřízené nedochází automaticky, musíme si navrhnout sami
- návrhář má povinnost zajistit uzavřenost chování své třídy

90% situací v objektovém návrhu je přímočarých, 10% problematických je popsáno výše