Objektový přístup k analýze a návrhu IS

Objektově orientovaný přístup je založen na objektech.

Objekt je struktura, která má definované

- **Vlastnosti** (tomu odpovídají atributy objektu)
- **Chování** (tomu odpovídají funkce, pro které se používá termín „metody“)

Vlastnosti a chování je zapouzdřené v jednotlivých objektech. Každý objekt je schopen reagovat na události.

Informační systém z pohledu objektově orientovaného přístupu je chápán jako množina spolupracujících objektů.

Objektový přístup lépe odpovídá chování reálného světa.

V oblasti programování je hlavní ideou objektového přístupu **znovupoužitelnost.**

Základní myšlenky objektového přístupu

- **Zapouzdření (encapsulation) -** objekt je pro nás „černou skříňkou – zajímá nás, co dělá a ne z čeho se skládá.
- **Dědičnost** - možnost vytvářet nové instance objektů s možností přidat nové prvky. Opakem dědičnosti je **generalizace** – z konkrétních dílčích částí sestavujeme obecnou společnou část
- **Polymorfismus (vícetvarost)** – různé chování objektů na stejný podnět – metoda stejného jména může mít trochu jinou funkcionalitu, přestože je tato funkcionalita pojmově blízká. Analogie je pojem „otevřít“ a rozdíl mezi funkcí „otevřít dveře“ a „otevřít láhev“. Příkladem polymorfismu je například funkce Read()/Write() v operačním systému – podle situace lze uplatnit na soubor nebo na nějaké zařízení.
- **Genericita** – možnost vytvářet parametrizovatelné programové moduly

Srovnání relačního a objektového modelu

Relační model – prvky reálného světa se snažíme zobrazit do pevných předem připravených struktur

Objektově orientovaný model – pro prvky reálného světa vytváříme objekty, které se jim podobají

Základní prvky objektového přístupu

Objekt
Konkrétní prvek vyskytující se v systému, který má definované vlastnosti a chování

Třída (Class)
Kategorie (skupina) objektů, které mají podobné vlastnosti a stejné nebo podobné chování

Atributy (Properties)
Slouží k popisu vlastností třídy (objektu).

Metody (Methods)
Algoritmy definující reakce objektu na vzniklé situace

K objektu můžeme přistupovat pouze využitím metod.

Instance třídy - objekt
Daný objekt je instancí určité třídy, tedy jejím konkrétním výskytem. Objekt ze strany třídy dědí atributy a metody. Třídu tedy můžeme chápat jako šablonu, která je použita k vytvoření objektu. Z jedné třídy lze vytvořit více instancí objektu, každá instance má alokovanou vlastní paměť. Třídu si můžeme představit například jako technický výkres, zatímco objekty jakožto instance třídy jsou pak výrobky vytvořené podle technického výkresu.

Hierarchie tříd
Třídy mají svou hierarchii, založené na dědičnosti.

Hierarchii tříd je možné budovat dvěma způsoby:

- **Směr specializace** – návrh obecných tříd a postupnému odvozování speciálních podtříd
- **Směr generalizace** – návrh objektů, jejich tříd a postupnému zobecňování

Abstraktní třída
Z abstraktních tříd se neodvozují konkrétní objekty; slouží k odvozování speciálnějších podtříd (potomků). Neexistuje konkrétní instance této třídy.

Rozhraní (interface)
Pojmenovaná množina operací, které třída používá ve vztahu k jiným třídám.
Vztah mezi třídou a jejím rozhraním říkáme **realizace.**

UML

UML neboli **Unified Modeling Language** je nástroj pro modelování informačních systémů založený na objektově orientovaném přístupu.

Je přijat sdružením OMG (Object Management Group) jako standard pro tvorbu informačních systémů.

Jedná se o nejrozšířenější objektovou notaci, která je podporována všemi CASE nástroji.

Jednou z důležitých charakteristik UML je jeho nezávislost na metodologiích. Možná i z toho pramení jeho široké rozšíření jakožto implementačního jazyka.

Použití UML je široké – od prostředku pro obecný popis systému po detailní návrh, který lze využit pro generování kódu v objektově orientovaném programovacím jazyce.

Diagramy UML

UML nabízí k popisu mnoho typů diagramů, rozdělených do tří skupin. První skupina diagramů popisuje statickou strukturu aplikace. Druhá skupina popisuje různé aspekty dynamického chování. Třetí slouží k organizaci a správě aplikačních modulů.

Statická struktura	Dynamické chování	Správa modulů
diagram tříd (Class Diagram)	use case diagram	balíčky (Packages)
objektový diagram (Object Diagram)	sekvenční diagram (Sequence Digram)	subsystémy (Subsystems)
komponentový diagram (Component Diagram)	diagram činností (aktivit) (Activity Diagram)	modely (Models)
diagram nasazení (Deployment Diagram)	diagram spolupráce (Collaboration Diagram)	
	stavový diagram (Statechart Diagram)	

Diagram komponent a diagram nasazení reprezentují **implementační model**.

Poznámka.

UML **nezahrnuje** DFD diagramy (Data Flow Diagramy), které slouží v strukturované analýze k popisu chování systému. Datové toky a jiné typy diagramů, které nebyly do UML zahrnuty, nezapadají čistě do konsistentního objektově orientovaného paradigmatu. Diagramy aktivit a diagramy spolupráce splňují mnoho z toho, co lidé chtějí od DFD. Diagramy aktivit jsou zároveň vhodné pro modelování workflow.

Pro další studium diagramů UML lze využít např. http://uml.czweb.org/index.html

Nedostatky UML

Za nedostatek UML můžeme považovat neschopnost UML poskytovat prostředky pro návrh uživatelského rozhraní a datových modelů. Například Activity diagram v UML neumožňuje zachytit místo pro uložení informací, podobně jako to umí diagram DFD ve strukturovaném návrhu.

Nedostatečností UML v oblasti datového modelování je to, že UML neposkytuje vhodné prostředky pro datové modelování, a proto se navrhuje rozšíření notace UML pomocí vhodných *stereotypů*, což jsou prostředky jazyka UML pro jeho vlastní rozšiřování. Soubor určitých *stereotypů*, které rozšiřují UML pro určitou oblast užití, se nazývá *UML extensions*.

Ostatní objektově orientované metodiky

Booch

Object Modelling Technique

Objectory/Object-Oriented Software Engineering

Object-oriented Process, Environment and Notation

Coad-Yourdon

Shlaer/Mellor

Class, Responsibility, Collaborators

Business Object Notation

Business Object Relation Modeling