

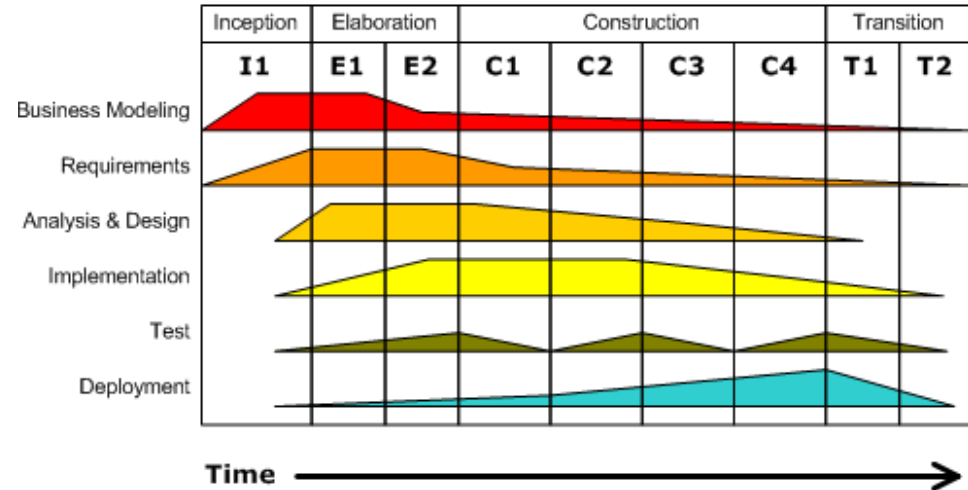
Projektovanje softvera

Uvod



Proces razvoja SW

- Faze razvoja SW po RUP-u:
 - započinjanje, razrada, konstrukcija, tranzicija
- U svakoj fazi se iterativno i inkrementalno prolazi kroz aktivnosti:
 - modeliranje poslovanja,
 - specifikacija zahteva,
 - analiza i **projektovanje**,
 - implementacija,
 - testiranje,
 - isporuka
- Različit je procenat aktivnosti po fazama



Pojmovi

- Objektno-orijentisana metodologija razvoja
 - dominantna u proizvodnji softvera danas
- Pojmovi
 - objektno-orijentisana analiza – OOA
 - **objektno-orijentisano projektovanje – OOD**
 - objektno-orijentisano programiranje – OOP
 - objektno-orijentisani jezik – OOL

Objektno-orijentisana analiza

- Tradicionalne tehnike strukturirane analize
 - fokus na toku podataka u sistemu
- **Booch** (1994): *Objektno-orijentisana analiza* je metod analize koji ispituje zahteve iz perspektive klasa i objekata pronađenih u rečniku iz domena problema
- Proizvod OOA
 - konceptualni model - ulaz u fazu OOD

Objektno-orijentisano projektovanje

- Tradicionalno strukturirano projektovanje
 - fokus na algoritamskim apstrakcijama
- **Booch** (1994): *Objektno-orijentisano projektovanje* je metod projektovanja koji obuhvata
 - proces OO dekompozicije
 - notaciju za predstavljanje
 - logičkih i fizičkih
 - statičkih i dinamičkihaspekata modela sistema koji se projektuje
- Proizvod OOD
 - model projektovane aplikacije ili sistema – ulaz u fazu OOP

Objektno-orijentisano programiranje

- Tradicionalno strukturirano programiranje
 - fokus na implementaciji algoritama
- **Booch** (1994): *Objektno-orijentisano programiranje* je metod implementacije po kojem su:
 - programi organizovani kao kolekcije objekata koji saraduju
 - svaki objekat predstavlja primerak neke klase i
 - sve klase su članovi neke hijerarhije klasa u kojoj su klase povezane relacijama nasleđivanja
- Proizvod OOP
 - izvršna aplikacija ili sistem

Objektno-orijentisani jezik

- **Cardelli & Wegner (1985):**
Jezik je objektno-orijentisan ako i samo ako ispunjava:
 - da podržava objekte koji su apstrakcije podataka
 - sa javnim interfejsom preko imenovanih operacija i
 - skrivenim lokalnim stanjem
 - da objekti imaju pridružen tip (klasu)
 - da tipovi (klase) mogu nasleđivati attribute nadtipa (natklase)
- Ako jezik ne podržava samo nasleđivanje naziva se objektno-baziranim jezikom
- Objektno-orijentisani jezici su:
 - Simula, Smalltalk, Object Pascal, Eiffel, Python, Ada95, C++, Java, C#, Visual Basic.NET, ...
- Objektno-bazirani jezici
 - Ada83, VisualBasic v6,...

Principi OO modela

- **Booch OOA&D (1994):**
- Osnovni (obavezni)
 - apstrakcija
 - kapsulacija
 - modularnost
 - hijerarhija
- Dodatni (neobavezni)
 - tipizacija
 - konkurentnost
 - perzistencija
- **Modifikacija:**
- Osnovni (obavezni)
 - apstrakcija
 - kapsulacija
 - modularnost
 - hijerarhija
 - polimorfizam
- Dodatni (neobavezni)
 - konkurentnost
 - perzistencija

Apstrakcija i kapsulacija

- **Shaw** (1984): Apstrakcija je uprošćeni opis ili specifikacija sistema koja naglašava neke od detalja ili osobina, dok potiskuje druge
- **Booch** (1994): Apstrakcija ističe esencijalne karakteristrike objekta koje ga razlikuju od drugih vrsta objekata i tako definiše jasne konceptualne granice iz perspektive posmatrača
- Kapsulacija je proces sakrivanja onih elemenata apstrakcije koji definišu strukturu i ponašanje
- Kapsulacija služi da razdvoji konceptualni interfejs (ugovor) od implementacije apstrakcije

Modularnost i hijerarhija

- Modularnost je osobina sistema da se razlaže na skup kohezivnih i slabo spregnutih modula
- Moduli su fizičke jedinice (nezavisno prevođenje)
 - predstavljaju komponente sistema
 - mogu se održavati nezavisno
- Hijerarhija je rangiranje ili uređivanje apstrakcija
- Nasleđivanje - “*is a*” hijerarhija
 - jednostruko/višestruko
 - potpuno (javno)/strukturno (privatno)
- Sadržanje - “*part of*” hijerarhija
 - po vrednosti/po referenci (relevantno u C++, ali ne u Javi)
 - agregacija/kompozicija

Tipizacija i polimorfizam

- Tipizacija je osobina da se objekti različitih klasa ne mogu uopšte ili se mogu zamenjivati na ograničene načine
 - stroga i slaba tipizacija
 - statička i dinamička tipizacija (vezivanje)
- Dinamička tipizacija i dinamičko vezivanje
 - tehnički preduslov za ispoljavanje polimorfizma
- Polimorfizam je osobina da se objekat kojem se pristupa kao objektu osnovne klase ponaša različito:
 - kao objekat osnovne klase ili kao objekat izvedene klase
 - ponašanje zavisi od dinamičkog tipa objekta, ne statičkog tipa reference
- Polimorfizam objekta se zasniva na virtuelnim metodima, odnosno dinamičkom vezivanju

Konkurentnost i perzistencija

- Principi koji se dobro uklapaju u OO paradigmu
- Nisu suštinski principi koji određuju da li je softver OO
 - OO softver ih ne mora posedovati
 - softver koji nije OO ih može posedovati
- Konkurentnost je osobina koja razlikuje aktivne objekte od pasivnih
 - *proces* - ima vlastiti adresni prostor (tipično njime upravlja OS)
 - *nit* - deli isti adresni prostor sa drugim nitima
- Perzistencija je osobina po kojoj se postojanje objekta proteže
 - kroz *vreme* (obj. nastavlja da živi nakon nestanka njegovog stvaraoca)
 - kroz *prostor* (obj. se premešta iz adresnog prostora u kojem je stvoren)

Model i modeliranje

- Model je pojednostavljene realnosti
- Model nekog sistema je apstrakcija tog realnog sistema iz određenog ugla posmatranja
- Osnovna namena modela
 - da se sistem koji se razvija bolje razume
- Modeliranje je važnije što je sistem kompleksniji
 - kompleksnost je odlika današnjih SW sistema
- Savremena metodologija razvoja softvera
 - *Model Driven Development* (MDD)

Ciljevi modeliranja

- Model pomaže da se sistem vizuelizuje
- Model omogućava da se specificira
 - struktura sistema
 - ponašanje sistema
- Model daje šablon koji usmerava konstrukciju sistema
- Model dokumentuje projektne odluke koje se donose
- Model smanjuje cenu razvoja
 - omogućava ispitivanje projektnih odluka po nižoj ceni

OO model i pogledi na model

- Model OO analize i projektovanja obuhvata više pogleda na sistem koji se razvija
- Dve dimenzije pogleda na sistem:
 - logički/fizički aspekti
 - statički/dinamički aspekti
- AiP OO sistema se najčešće obavlja u terminima klasa, objekata, njihovih relacija i interakcija
- Tokom AiP koriste se različiti uglovi gledanja na model sistema u datom 2D prostoru

Dijagrami

- Za svaki pogled na model sistema može se definisati adekvatan dijagram
- Svaki dijagram predstavlja jednu projekciju modela
- Primer - aplikacija sa 100 klasa:
 - potrebno je više klasnih dijagrama (svaki prikazuje jedan pogled na model)
- Jedno ime na svakom dijagramu označava isti entitet (sa izuzetkom operacija zbog preklapanja imena)

Logički i fizički aspekti modela

- Logički model sistema
 - opisuje ključne apstrakcije (klase) i definiše
 - strukturu klasa (atribute i operacije)
 - relacije između klasa ili komponenata sistema
 - interakcije između uloga (prototipskih objekata i aktera)
- Fizički model sistema
 - opisuje konkretnu softversku i hardversku kompoziciju
 - definiše arhitekturu modula i arhitekturu procesa

Statički i dinamički aspekti modela

- Statički aspekti modela se fokusiraju na
 - strukturu sistema
- Dinamički aspekti modela se fokusiraju na
 - ponašanje sistema
- Realni sistemi uvek imaju dinamičko ponašanje:
 - objekti se kreiraju i uništavaju
 - objekti šalju poruke drugim objektima nekim redosledom
 - spoljašnji događaji izazivaju reakcije izvesnih objekata

Notacija za opis modela

- Nekoliko notacija zaslužuju posebnu pažnju:
 - Booch i OMT notacija (iz istorijskih razloga)
 - UML notacija (standard)
- Pogodnosti standardne formalne grafičke notacije:
 - olakšava se komunikacija između:
 - korisnika i sistem-analitičara
 - članova razvojnog tima
 - projektant se rasterećuje od nebitnih detalja i koncentriše se na bitne
 - omogućava se razvoj i upotreba automatizovanih alata za :
 - proveru konzistencije i korektnosti modela
 - izvršavanje modela

Upotreba notacije

- Nije neophodno koristiti celu notaciju
- Primeri:
 - *Booch Lite*,
 - *UML Basic* (UML User Guide)
- Notacija treba da omogućava različit stepen detaljnosti
 - ponekad su potrebne samo grube skice modela
 - skice je potrebno nekad crtati i ručno, u toku razgovora
 - na kraju procesa projektovanja potreban je detaljan model
- Notacija treba da bude nezavisna od programskog jezika
 - neki elementi notacije nemaju podršku u konkretnom jeziku
 - neki elementi nekih jezika nemaju podršku u notaciji

Alati za modeliranje (1)

- IBM Rational: Software Architect (Rose, Rose XDE Developer, Software Modeler)
 - <https://www.ibm.com/us-en/marketplace/rational-software-architect-designer>
- Borland: Together
 - <https://www.borland.com/en-GB/Products/Requirements-Management/Together>
- Gentleware: Poseidon for UML (?)
 - https://download.cnet.com/Poseidon-for-UML-Community-Edition/3000-2212_4-10071385.html
- **Open Source: StarUML**
 - <http://staruml.io/>
- Altova: Umodel
 - http://www.altova.com/download/umodel/uml_tool.html

Alati za modeliranje (2)

- Omondo: EclipseUML
 - <http://www.omondo.com>
- Sparx Systems: Enterprise Architect
 - <http://www.sparxsystems.com>
- Visual Paradigm: Visual Paradigm for UML
 - <https://www.visual-paradigm.com/features/>
- Pregled alata:
 - https://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools