Praksa razvoja PO

Praksa razvoja PO je seznam konceptov, principov, metod in orodij, ki jih je treba upoštevati pri načrtovanju in razvoju PO.

Strategija reševanja problemov:

1) Razumevanje problema (komunikacija in analiza)
2) Načrtovanje rešitve (modeliranje in načrtovanje)
3) Izvedba načrta (kodiranje)
4) Preverjanje rezultatov (testiranje in evaluacija kvalitete)

Osnovni principi

- Programska oprema = dodana vrednost!
- Preprost načrt
- Jasna vizija
- Zavedanje, da bo nekdo drug uporabljal naš produkt
- Bodi odprt do prihodnosti
- Vnaprej načrtuj ponovno uporabo komponent
- Premisli!

Komunikacijske prakse

- Pozorno poslušaj!
- Priprava pred sestankom
- Moderator sestanka
- Pogovor iz oči v oči je najboljša metoda komunikacije
- Zapisnikar
- Sodelovanje
- Pozornost/drži se teme
- Če je kaj nejasno → skica
- Preidi na naslednjo temo, če glede trenutne teme: pride do sporazuma/ne more priti do sporazuma/če določenih zahtev ni mogoče razjasniti
- Pogajanja niso tekmovanje!

Prakse planiranja

- Razumevanje cilja projekta
- Vključevanje naročnikov
- Planiranje je iterativen proces
- Ocenjevanje na osnovi tega kar vemo
- Ocenjevanje tveganj
- Bodi realen!
- Prilagodi nivo detajlov
- Plan za ocenjevanje kakovosti
- Plan za upravljanje s spremembami zahtev
- Pogosto spremljaj izvajanje plana

Prakse modeliranja

Model mora biti sposoben prezentirati:

- informacije, ki jih PO procesira
- arhitekturo in funkcije, ki omogočajo procesiranje
- lastnosti sistema, ki jih želi naročnik
- obnašanje sistema ob procesiranju

- Modeli analize (naročniške zahteve: informacijska, funkcionalna in performančna domena)
- Modeli podrobnega načrta (karakteristike sistema: arhitektura, uporabniški vmesnik, komponente)

Prakse analize

- Informacijska domena problema mora biti jasna
- Funkcije sistema morajo biti jasne
- Obnašanje sistema kot posledica dogodkov mora biti definirano
- Modeli, ki predstavljajo informacije, funkcije in obnašanje naj bodo organizirani hierarhično
- Analiza naj povezuje zahteve z implementacijo

Prakse podrobnega načrta

- Model podrobnega načrta mora biti izpeljan iz modela analize
- Naredi splošno arhitekturo sistema
- Načrtovanje podatkovnih struktur je vsaj tako pomembno kot načrt funkcij, ki procesirajo podatke
- Zunanji in notranji vmesniki
- Uporabniški vmesnik prilagojen naročniškim zahtevam
- Funkcionalno neodvisne komponente
- Šibka medsebojna povezanost komponent (vmesniki, sporočila)
- Model podrobnega načrta naj bo lahko razumljiv
- Izdelava podrobnega načrta naj bo iterativen proces → večja enostavnost

Prakse kodiranja (priprava)

Preden napišemo eno vrstico kode, moramo:

- Razumeti problem
- Razumeti osnovne principe načrta
- Izbrati primerne programske jezike
- Izbrati primerno razvojno okolje z ustreznimi orodji
- Izdelati serijo unit testov za komponento, ki jo želimo izdelati

Prakse kodiranja (kodiranje)

- Uporabljeni algoritmi naj upoštevajo pravila strukturiranega programiranja
- Uporabi ustrezne podatkovne strukture
- Upoštevaj arhitekturo in naredi vmesnike za izmenjavo podatkov na osnovi le-te
- Pogojna struktura programov naj bo kar se da enostavna
- Vgnezdene zanke naj omogočajo kar se da enostavno testiranje
- Uporabi smiselna imena spremenljivk
- Koda naj bo samodokumentirana
- Uporaba zamikanja v kodi

Prakse kodiranja (validacija)

Po koncu vsake iteracije kodiranja:

- Preglej kodo, ki si jo napisal
- Izvedi unit teste
- Po potrebi predelaj kodo

Prakse testiranja

- Vsak test naj izhaja iz določene naročniške zahteve
- Testi naj bodo načrtovani veliko pred začetkom testiranja
- 80-20 princip
- Testiranje naj se začne pri komponentah in nadaljuje proti popolnoma integriranemu sistemu
- Popolno testiranje ni možno

Prakse inštalacije/dostave

- Upravljanje s pričakovanji naročnika
- PO mora biti testirana v obliki, v kakršni jo bomo dostavili naročnikom
- Pred dostavo moramo vzpostaviti režim podpore
- Pripravljena mora biti ustrezna podporna dokumentacija
- PO oprema z napakami naj bo najprej popravljena, šele nato dostavljena naročnikom

Zbiranje zahtev

- Pomaga razvijalcem bolje razumeti problem
- Sodelovanje softverskih inženirjev (analitikov) in naročnikov
- Rezultat: razumevanje projekta v pisni obliki → uporabniški scenariji/seznam operacij in funkcionalnosti/analitični modeli/specifikacije zahtev
- Preverjanje s strani analitikov, naročnikov in uporabnikov

Razlogi

- Programiranje je (običajno) zabavno in predstavlja programerjem največji izziv
- Zato se velikokrat lotimo programiranja preden so nam stvari jasne
- Velikokrat to utemeljujemo s tem:
  - da bodo stvari postale jasne tokom razvoja
  - da bodo naročniki lahko lažje opisali svoje potrebe po parih iteracijah
  - da se zahteve spreminjajo tako hitro, da je začetno zbiranje zahtev izguba časa
  - da je končni cilj projekta delujoča aplikacija

Faze zbiranja zahtev

- Iniciacija projekta (splošna vprašanja brez konteksta)
- Zbiranje osnovnih zahtev (kakšen je namen produkta, kako bo produkt zadovoljil poslovne potrebe, kako bo produkt uporabljen, razvoj uporabniških scenarijev, itd)
- Preciziranje zahtev (analiza uporabniških scenarijev za določitev analitičnih razredov → entitete poslovnih domen, ki so vidne uporabnikom, določitev atributov teh razredov in servisov, ki so potrebni za njihovo delovanje. Relacije in sodelovanje med razredi lahko opišemo z različnimi UML diagrami)
- Pogajanja (pogajanja glede obsežnosti zahtev, prioritet, razreševanje konfliktnih zahtev, itd)
- Specifikacija (pisni dokument, ki vsebuje množico diagramov, matematičnih modelov, uporabniških scenarijev, prototipov, specifikacijo funkcionalnosti in operacij, itd)
- Validacija (preverjanje specifikacije s pomočjo naročnikov)
- Upravljanje z zahtevami (vodenje evidence o zahtevah)

Iniciacija

- Identifikacija naročnikov
- Več pogledov na projekt
- Sodelovanje
- Postavljanje splošnih vprašanj (kdo so naročniki, kdo bo uporabljal produkt, kakšne bodo ekonomske koristi od uspešnega produkta, ali obstajajo podobni projekti, katere probleme bo produkt reševal, v kakšnem poslovnem okolju bo produkt deloval, kako bi okarakterizirali dober odziv produkta, so kakšne posebne performančne zahteve, ste prava oseba za odgovarjanje na ta vprašanja, ali bi morali vprašati še koga drugega za mnenje, itd)

Zbiranje osnovnih zahtev

- Sestanki med naročniki in sistemskimi analitiki
- Postavljena pravila za sodelovanje na sestanku
- Dnevni red
- Moderator
- Predstavitev informacij (tabla, projektor, forum, programi za komunikacijo, ipd)
- Identifikacija problema, predlog rešitve in izdelava začetnega seznama zahtev
- Izdelava uporabniških scenarijev
- Ocena funkcionalnosti QFD (normalne, pričakovane, izjemne)

Preciziranje zahtev

- Analiza uporabiških scenarijev
- Identifikacija elementov v uporabniških scenarijih: akcije, analitični razredi, stanja aplikacije, elementi pretoka informacij
- Različni nivoji podrobnosti

Pogajanja

- Stremenje k win-win situaciji
- Identifikacija ključnih naročnikov
- Identifikacija win situacije za naročnike
- Pogajanja z naročniki za dosego win situacije tudi za razvijalce

Validacija

- Ali je vsaka zahteva konsistentna s cilji projekta?
- Ali so vse zahteve bile dovolj podrobno specificirane?
- Ali je zahteva res nujno potrebna ali predstavlja samo dodatek?
- Je vsaka zahteva nedvoumna?
- Ali je kaka zahteve v protislovju z neko drugo?
- Je vsaki zahtevi možno zadostiti v okviru okolja, v katerem bo produkt deloval?
- Je vsako zahtevo možno testirati?
- So zahteve bile razdelane v smislu bolj podrobne specifikacije?
- ...