PROCESI RAZVOJA SOFTVERA

Za razvoj softvera karakteristične su određene temeljne aktivnosti:
- Specificiranje potreba (identifikacija, transformacija potreba u zahtjeve, analiza, određivanje prioriteta …)
- Dizajniranje problema (oblikovanje problema grafičkom notacijom, oblikovanje procesa, analiza …)
- Implementacija (kodiranje, testiranje, uvođenje u rad, dokumentiranje, edukacija …)
- Validacija (testiranje softverskog sustava, procjena kvalitete …)
- Evolucija (održavanje sustava, reinženjering …)

Način razvoja softvera uvjetovan je karakteristikama konkretnog modela. Generički modeli procesa razvoja softvera su:
- ✓ Vodopadni model
- ✓ Evolucijski model
- ✓ Model formalne transformacije
- ✓ Model ponovne primjene postojećih resursa

Vodopadni model

Faze vodopadnog modela:

Definiranje zahtjeva
Oblikovanje sustava, oblikovanje softvera
Implementacija i testiranje dijelova sustava
Integracija i testiranje cijelog sustava
Uvođenje u rad i održavanje

Značajke modela:
- Proces se razvija slijedom koraka (faza po faza)
- Svaka faza mora rezultirati dokumentom
- Rezultat prethodne faze, polazište je u razvijanju slijedeće faze
- Uvažavanje naknadnih korisničkih zahtijeva stvara poteškoće
- Model se primjenjuje u situacijama kada su zahtjevi korisnika poznati u prvoj fazi modela
- Premda se radi o linearnom (slijednom) modelu on u određenim situacijama može prerasti u linearno-kružni model.

Evolucijski model

Faze evolucijskog modela:

Ulaz: okvirni (grubi) opis problema
Slijedne aktivnosti: specifikacija, razvoj, validacija
Izlaz: inicijalna verzija, međuverzije, konačna verzija

Značajke modela:
- Na temelju grube ideje dobivene od korisnika razvija se prva verzija budućeg proizvoda
- Proces razvoja uključuje: specificiranje, razvijanje, validiranje
- Inicijalna verzija upućuje se korisniku na testiranje
- Primjedbe korisnika utječu na promjenu specifikacije

- Inicijalna verzija se poboljšava primjedbama korisnika
- Poboljšana verzija ponovno se upućuje korisniku na testiranje
- Interakcija (projektant – korisnik) se ponavlja do izrade konačne verzije.

Dva načina evolucijskog razvoja:
- *Evolucijsko prototipiranje* (Istraživački razvoj) – proces razvoja počinje od najbolje razrađenih zahtjeva korisnika
- *Prototipiranje „učini – odbaci“* – proces razvoja počinje od najslabije razrađenih zahtjeva.

Model formalne transformacije

Faze modela:

Definiranje zahtjeva
Formalma specifikacija
Formalna transformacija
Integracija i testiranje sustava

Značajke modela:
- Model se temelji na transformaciji matematičke specifikacije u izvodljiv program
- Za primjenu ovog modela potrebna specijalistička znanja i vještine
- Model se primjenjuje u specifičnim situacijama, nema elemente opće primjene
- Neki korisnički zahtjevi ne mogu se razviti ovim modelom (npr: korisničko sučelje)

Model ponovne primjene postojećih komponenti

Faze modela:

Specifikacija zahtjeva
Analiza raspoloživih komponenti
Modifikacija zahtjeva
Oblikovanje sustava vodeći računa o komponetama koje je moguće ponovno upotrijebiti
Razvoj i integracija
Validacija sustava

Značajke modela:
- Model se temelji na planiranoj integraciji postojećih komponenti, dijelova sustava ili cjelovitog sustava u novi proizvod.
- Ideja je pozitivna, poznata pod nazivom COST (Commercial-Off-the-Shelf) sustavi

Iterativni (ponavljajući) procesi

Kod većine modela izražena je potreba povratka u ranije faze procesa razvoja proizvoda.

Dva pristupa su:
- Razvoj „dio-po-dio“ ili razvoj u koracima
- Spiralni razvoj

Razvoj „dio-po-dio“ ili razvoj u koracima

Faze modela

Definiranje grubih zahtjeva
Raslojavanje zahtjeva na dijelove
Oblikovanje strukture sustava
Razvoj sustava dio-po-dio (prema raslojenim zahtjevima)*
Validacija dijelova sustava*
Integracija dijelova u sustav*
Validacija sustava

Dijelovi modela označeni sa * ponavljaju se do razvoja cjelovitog sustava.

Značajke modela:

- Zahtjevi postavljeni pred budući proizvod se dekomponiraju (raslojavaju) na manje cjeline
- Proizvod se razvija i isporučuje prema prioritetima dio-po-dio
- Svaki isporučeni dio ima određenu funkcionalnost
- Ranije isporučeni dijelovi imaju ulogu prototipa u odnosu na kasnije segmente
- Jednom definirani zahtjevi „zamrzavaju“ se na početku razvoja konkretnog dijela (segmenta)
- Rizik validacije dijelova proizvoda manji u odnosu na validaciju gotovog proizvoda.

Spiralni razvoj – spiralni model

Model se razvija kroz četiri sektora:

1. Postavljanje cilja
   - identifikacija specifičnih ciljeva za svaku fazu
2. Procjena rizika i njegovo smanjenje
   - rizici su procijenjeni i provode se aktivnosti kako bi se smanjili ključni rizici
3. Razvoj i validacija
   - za sam razvoj sustava odabire se bilo koji generički model
1. Planiranje
   - dosadašnji tijek razvoja projekta se ispituje i planira se slijedeći „krug“ u spirali.

Značajke modela:

- Svaki „krug, petlja“ u spirali predstavlja jednu fazu procesa
- Nema fiksnih faza (koraka) u procesu razvoja (npr: nema specifikacije)
- Eksplicitno razmatranje (rasčlanjivanje, rješavanje) rizika u svakom krugu spirale.

3.6. Evolucijski razvoj – prototipiranje

Pod pojmom prototipa u slučaju razvoja softvera podrazumijeva se inicijalna verzija softverskog sustava čija je svrha (1) demonstrirati predodžbu kako bi sustav (program) trebao funkcionirati, (2) pronaći rješenje za rješavanje postavljenog problema.

Primjena sustava prototipiranja

Glavna primjena prototipiranja ogleda se kroz pomoć koju prototip pruža korisnicima i projektantima u procesu razumijevanju zahtjeva koje su korisnici postavili pred budući softverski sustava. Primjenom sustava prototipiranja korisnik može, eksperimentirajući s prototipom, utvrditi kako će prototip a kasnije i cijeli sustav, podržavati odnosno reagirati na njegove zahtjeve postavljene u ranijoj fazi razvoja prototipa, a korisnik i projektant mogu kroz prototip otkriti eventualne pogreške i propuste u odnosu na definirane zahtjeve.

Prototipiranje se također primjenjuje u funkciji smanjenja rizika. Detaljnom komparacijom postavljenih zahtjeva od strane korisnika i karakteristika prototipa moguće je utvrditi eventualna odstupanja na relaciji zahtjevi – prototip. U tom smislu smanjuju se rizici u izradi budućeg sustava koji bi se, kada to ne bi bilo učinjeno u fazi prototipiranja, mogli iskazati pri validaciji.

3.7. Prototipiranje u procesu razvoja softvera

Evolucijsko prototipiranja

Polazi od razvoja relativno jednostavnog sustava koji uvažava najvažnije zahtjeve korisnika. Stvara se inicijalni prototip. Taj se prototip oplemenjuje kroz primjedbe i sugestije korisnika i određeni broj iterativnih koraka do konačne verzije.

Prema tome, evolucijsko prototipiranje temelji se na ideji:

- Inicijalne implementacije
- Prikupljanja primjedbi korisnika
- Oplemenjivanju kroz korake do razvoja odgovarajućeg sustava

Osnovne prednosti ovakvog pristupa ogledaju se u uključivanju korisnika u sam razvojni proces prototipa i u mogućnosti brze isporuke dijela ili cijelog budućeg sustava.

Evolucijsko prototipiranje sastavni je dio metoda brzog razvoja aplikacije. Dvije poznate metode toga tipa su:

- Joint Application Development (JAD)
- Rapid Application Development (RAD).

Problemi pri isporuci prototipa

Jedan od mogućih problema pri izradi "učini i odbaci"prototipa jest situacija u kojoj proizvođači softvera iz različitih razloga (vrijeme isporuke, financijska sredstva) budu prisiljeni isporučiti «učini i odbaci» prototip kao finalnu verziju sustava. Razlozi zbog kojih se ne preporučuje isporuka «učini i odbaci» prototipa kao finalne verzije su:

- Nemogućnost prototipa da udovolji ne-funkcionalnim zahtjevima;
- Prototip je nedovoljno dokumentiran što umanje mogućnost dugoročnog održavanja;
- Promjene tijekom prototipiranja narušavaju strukturu;
- Uobičajeni organizacijski standardi kvalitete su ublaženi pri razvoju sustava prototipiranjem.

Prototipiranje povezivanjem postojećih (ranije stvorenih) dijelova u novu aplikaciju

Vrijeme razvoja prototipa (ili cijelog sustava) bitno se skraćuje ako se neke već ranije stvorene komponente mogu ponovno ugraditi u novi prototip. U novi sustav moguće j ugraditi ranije stvorene pojedine komponente (naredbe programa), cijele module ili kompletnu aplikaciju.

Joint Application Development (JAD)

Joint Application Development (JAD) ili Zajednički razvoj aplikacije primjenjuje se u situacijama gdje postoji potreba uključivanja korisnika u poslovni proces odnosno u razvoj projekta. Korisnici i projektanti zajednički surađuju putem JAD radionice.

Metodu su razvili Chuck Morris i Tony Crawford (IBM) kasnih 70-tih prošlog stoljeća. Njena šira primjena počinje u SAD krajem 80-tih godinama prošlog stoljeća. Metoda je tijekom godina doživjela niz promjena prateći promjene u razvoju informacijskih resursa i prilagođavajući se trendovima u primjeni informacijskih tehnologija.

Glavna ideja metode:
- uključivanje u rad predstavnika svih kategorija zainteresiranih sudionika,
- rad u timu,
- rad kroz sjednice (workshop-ove),
- temeljito pripremanje sjednica,
- dokumentiranje svih aktivnosti ...

Aktivnosti u JAD radionici razvijaju se kroz pet faza:
- *Definiranje projekta* (svrha, cilj, područje razvoja, formiranje JAD tima, plan sjednice)
- *Istraživanje* (upoznavanje sa sustavom, prikupljanje početnih informacija, modeliranje sustava, određivanje polazišta za planiranje daljnjeg rada)
- *Pripreme* (priprema sjednice, priprema potrebne opreme za sjednicu)
- *Sjednica* (slijedi se tijek projektnih zahtjeva, rezultati se dokumentiraju)
- *Završni dokument* (stvaranje završnog dokumenta na temelju informacija prikupljenih tijekom sjednice)

Svaka faza ima svoje ulaze, procese odnosno aktivnosti koje se obavljaju unutar faze, te izlaze.

Razlozi dugogodišnje aktualnosti JAD metode: transparentnost same metode,
mogućnost primijeniti u različitim poslovnim okruženjima i različitim segmentima poslovnog procesa, otklon od tehničke orijentiranosti rješavanja problema.

Rapid Application Development (RAD)

Rapid Application Development (RAD) ili Brzi razvoj aplikacije je metoda koja se javlja kao odgovor na modele razvoja sustava karakteristične tijekom ’70. g (npr: vodopadni model). Ideju o novoj (RAD) metodi postavili su Barry Boehm i Scott Shultz a samu metodu razvio je James Martin (’80. g.). IBM je formalizirao metodu izdavanjem knjige o RAD, 1991. g. Metoda se primjenjuje za razvoj softvera koji omogućava izradu uporabljivog softvera. Pojam uporabljivog softvera odnosi se na situaciju kada softver udovoljava temeljne zahtjeve i radi ali svi zahtjevi još nisu zadovoljeni. To je situacija kada se primjerice na tržište želi plasirati

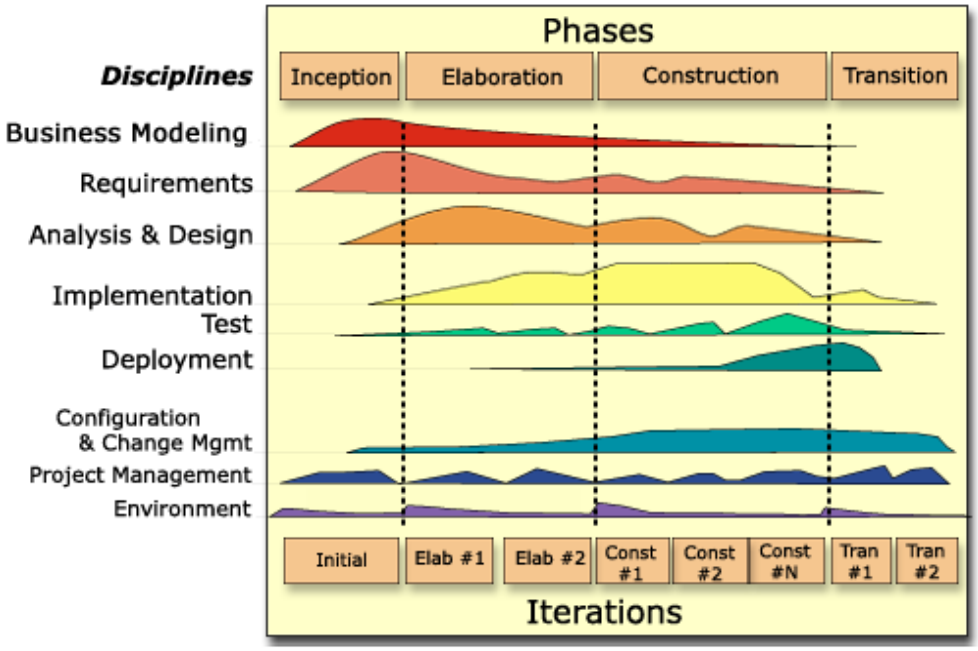
novi softverski proizvod što ranije (da bude prvi na tržištu) a zahtjevi koji nisu ključni za njegovu funkcionalnost dogradit će se (zadovoljiti) naknadno "u hodu".

Povećanje brzine moguće je kroz: brzo prototipiranje, vizualizaciju za sustav važnih rutina, primjenu CASE alata, primjenu drugih tehnika koje mogu ubrzati razvoj sustava.

RAD metodologiju moguće je primijeniti kod aplikacija koje se izvode samostalno, ako performanse nisu kritične, ako pouzdanost aplikacije nije kritična, kada je proizvod namijenjen visoko specijaliziranom tržištu i dr.

3.8. Rational Unified Process (RUP)

RUP = model iterativnog procesa razvoja softvera. Razvijen u Rational Software Corporation, danas u sastavu IBM-a.



Arhitektura RUP-a prikazana je kroz dvije perspektive, dvije dimenzije:

- Horizontalna dimenzija (horizontalna os) prikazuje dinamičku strukturu procesa (ciklusi, faze, iteracije). Pokazuje vrijeme.
- Vertikalna dimenzija (vertikalna os) prikazuje statičku strukturu procesa (aktivnosti koje se obavljaju u pojedinim fazama i iteracijama).

Horizontalna dimenzija

Životni ciklus razvoja softverskog proizvoda prema RUP-u čine četiri faze iterativnog procesa:

- Faza pripreme (Inception): definira se i razrađuje ideja projekta, odlučuje se da li se projekt može realizirati. Završetak faze je kontrolna točka pripreme (Lifecycle objectives milestone).
- Faza razrade (Elaboration): planiraju se nužne aktivnosti i resursi potrebni za realizaciju zahtjeva za pojedinu iteraciju, specificiraju se svojstva i dizajn arhitekture sustava.

Završetak faze je kontrolna točka životnog ciklusa arhitekture (Life cycle arhitecture milestone).

- Faza konstrukcije (Construction): gradi se softverski sustav. Završetak faze je kontrolna točka inicijalne operacijske mogućnosti (Initial operational capability milestone).

- Faza prijelaza (Transition): softverski sustav se kao proizvod predaje kupcu. Faza uključuje proizvodnju, isporuku, učenje korisnika, podršku i održavanje. Faza završava kontrolnom točkom isporuke proizvoda (Product release milestone). Ovom točkom završava pojedina iteracija.

Vertikalna dimenzija

Prikazuje osnovne (glavne) i dodatne discipline razvoja softvera.

Osnovne discipline su:
- poslovno modeliranje
- zahtjevi
- analiza i dizajn
- implementacija
- testiranje
- postavljanje.

Dodatne discipline čine:
- upravljanje konfiguracijom
- upravljanje projektom
- okruženje.

RUP opisuje: *Tko* radi *što*, *kako* i *kada*
- *Radnici* (Workers, iskazani kao Role) predstavljaju *tko*
- *Aktivnosti* predstavljaju *kako*
- *Artifakti* predstavljaju *što*
- *Tokovi rada* (workflow) definiraju *kada*

Uloga je apstraktna definicija skupa aktivnosti koje se izvode i artifakata koji pripadaju skupu aktivnosti. Ulogu obavlja jedan ili više članova razvojnog tima. Ona ne predstavlja osobe već opisuje kako se osobe ponašaju u poslu i koje su njihove odgovornosti. Osnovne uloge: analitičari, razvijatelji – projektanti, ispitivači, i voditelji.

Aktivnost je dio posla koji obavlja pojedina uloga. Aktivnosti su povezane s artifaktima.

Artifakti pružaju ulaz i izlaz za aktivnosti i mehanizme pomoću kojih se informacija razmjenjuje između aktivnosti.