Softverski procesi

Programsko inženjerstvo

Ciljevi poglavlja

- Upoznati se s pojmom softverskih procesa;
- Objasniti tri osnovna procesna modela i kada se oni mogu koristiti;
- Razumjeti aktivnosti uključene u specifikaciju zahtjeva, razvoj softvera, testiranje i održavanje;
- Prikazati kako se može uvesti poboljšanja u proces razvoja.
- Softverski produkti i upravljanje softverskim produktima

Softverski procesi

- Niz aktivnosti koji vodi do razvoja softvera.
- Iako postoji mnogo softverskih procesa sljedeće aktivnosti su im svima zajedničke:
  - ***Specifikacija*** – funkcionalnosti softvera i njegova ograničenja;
  - ***Dizajn i implementacija*** – produkcija softvera koji odgovara zahtjevima;
  - ***Validacija i verifikacija*** – provjera radi li softver ispravno i ono što je kupac tražio;
  - ***Održavanje*** (evolucija) – izmjene softvera kako bi pratio promjene zahtjeva.

Planirani i agilni procesi

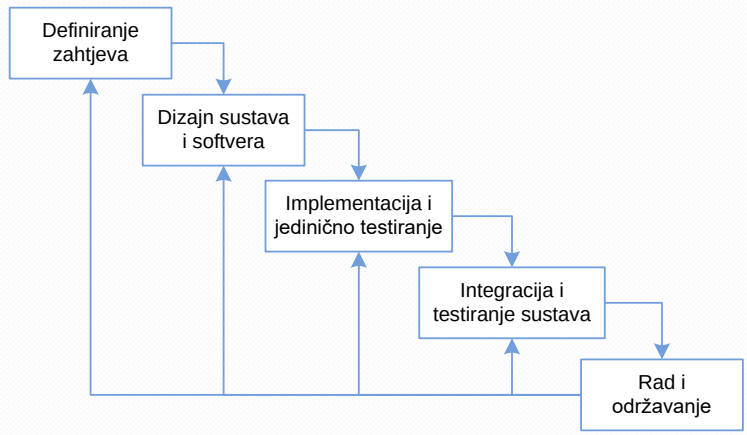
- Planirani procesi (engl. *plane based*) su procesi kod kojih su sve procesne aktivnosti planirane unaprijed i napredak se mjeri u odnosu na ovaj plan.
- Kod agilnih procesa planiranje je inkrementalno i jednostavnije je promijeniti proces kako bi zadovoljio promjenu korisničkih zahtjeva.
- U praksi, većina procesa uključuje elemente oba pristupa.
- Ne postoji pravi ili krivi softverski proces.

Modeli softverskih procesa

Modeli softverskih procesa

- **Vodopadni model**
  - Planirani model sa strogo odvojenim fazama specifikacije i razvoja.
- **Inkrementalni razvoj**
  - Specifikacija, razvoj i validacija se preklapaju. Može biti planiran ili agilan.
- **Integracija i konfiguriranje komponenti**
  - Sustav se sklapa od postojećih komponenti. Može biti planirano ili agilno.
- U praksi se većina velikih sustava razvija korištenjem procesa koji uključuju elemente iz sva tri modela.

Vodopadni model



Faze vodopadnog modela

- Prvi objavljeni model softverskih procesa, a objavljen je 1970. g.
- Kod ovog modela postoje strogo odvojene faze:
  - **Analiza i definiranje zahtjeva**
    - U suradnji s korisnicima određuju se usluge koje će nuditi sustav, njegova ograničenja i ciljevi.
  - **Dizajn sustava i softvera**
    - Uspostavlja se arhitektura sustava, identificiraju se osnovne apstrakcije u sustavu i njihove veze.
  - **Implementacija i jedinično testiranje**
    - Iz definirane arhitekture sustava se piše i testira kod.
  - **Integracija i testiranje sustava**
    - Individualni dijelovi koda se spajaju i testira se sustav kao cjelina kako bi se ustanovilo zadovoljava li korisničke zahtjeve.
  - **Rad i održavanje**
    - Najčešće najduža faza životnog ciklusa. Sustav se pušta u rad, ispravljaju greške koje nisu otkrivene u ranijim fazama, sustav se poboljšava i po potrebi dodaju nove funkcionalnosti.

Prednosti vodopadnog modela

- Velika prednost ovog modela je:
  - **Planiranje** - vrlo striktno planiranje se obavlja na početku procesa pa je moguće procijeniti kada će projekt biti gotov, koliki će biti trošak, ...
  - **Zahtjevi** – s obzirom da su svi zahtjevi poznati na početku moguće je pojednostavniti dizajn.
  - **Dokumentacija** – postoji detaljna dokumentacija , tj. rezultat svake faze je jedan ili više "potpisanih" dokumenata.
- Iako sljedeća faza ne bi trebala početi dok trenutna ne završi i praksi ipak postoji neko preklapanje jer susjedne faze dijele informacije (za vrijeme dizajna mogu se otkriti problemi sa specifikacijom ili za vrijeme kodiranja greške u dizajnu, ...).

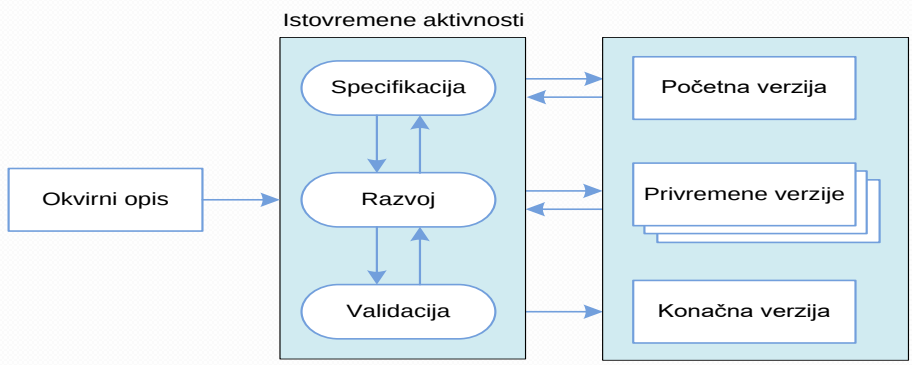
Problemi vodopadnog modela

- Glavni nedostatak ovog modela je nemogućnost da se prihvati promjena kada se započne s procesom. Faza treba biti gotova prije nego se pređe na iduću.
  - Zbog toga je prikladan samo kada su zahtjevi dobro definirani i jasni a promjene ograničene tijekom dizajna.
  - Jako malo poslovnih procesa ima tako stabilne zahtjeve.
- Vodopadni model bi se koristi kod razvoja:
  - Velikih sustava gdje jako puno ljudi radi na projektu i oni su fizički dislocirani (planiranje koje je osnova vodopadnog modela olakšava koordinaciju posla);
  - Medicinske opreme, ...

Inkrementalni razvoj

- Razvoj i isporuka sustava su podijeljeni u niz inkrementa od kojih svaki isporučuje neku od zadanih funkcionalnosti.
- Svi zahtjevi dobivaju prioritet i oni s najvećim se uključuju u početne inkremente.
- Jednom kada se krene u razvoj inkrementa ne smiju s mijenjati njegovi zahtjevi, ali se to može raditi za sljedeće inkremente.

Inkrementalni razvoj



Inkrementalni razvoj i isporuke

- Inkrementalni razvoj:
  - Standardan pristup kod agilnih metoda;
  - Sustav se razvija u inkrementima i svaki inkrement se procjenjuje prije nego se krene na sljedeći;
  - Procjenu svakog inkrementa radi kupac/korisnik sustava.
- Inkrementalne isporuke:
  - Isporuka i instalacija verzije korisnicima;
  - Realna procjena o praktičnoj koristi softvera;
  - Teško se može koristiti u slučajevima kad novi sustav mijenja stari, jer u tom slučaju stari nudi više funkcionalnosti.

Prednosti inkrementalnog razvoja

- Inkrementalni razvoj softvera vrlo često puno brže i efikasnije rezultira sustavom koji odgovara zahtjevima korisnika.
  - Brža isporuka i primjena najvažnijih funkcionalnosti.
  - Korisnici mogu koristiti (i zarađivati) softver prije nego bi to mogli korištenjem vodopadnog modela.
  - Korisnik može dati svoje mišljenje o sustavu na osnovu onoga što je napravljeno.
- Specifikacija zahtjeva se razvija inkrementalno, tj. uvidom u početne verzije sustava korisnicima je jasnije što još žele od sustava.
  - Početni inkrementi služe kao prototip koji iznosi na vidjelo zahtjeve za kasnije inkremente.
- Umanjuje se cijena promjene zahtjeva korisnika.
  - Potrebna analiza i dokumentacija koju treba ponovno napraviti ili prepraviti je znatno manja nego kod vodopadnog modela.
- Umanjuje se rizik od potpunog promašaja projekta.
- Zahtjevi s najvišim prioritetom se najviše testiraju.

Nedostaci inkrementalnih isporuka

- Inkrementi bi trebali biti mali (<20 000 linija koda), a nekada je nemoguće mapirati zahtjeve u tako male inkremente.
- Loša vidljivost procesa:
  - Menadžeri trebaju redovne isporuke kako bi mjerili projekt. Ukoliko se sustav razvija brzo ne isplati se stvarati dokumentaciju za svaku verziju sustava.
- U osnovi iterativnog procesa je da se specifikacija razvija zajedno sa softverom.
  - To je baš suprotno od klasičnog modela nabave u većini organizacija, gdje je potpuna specifikacija dio ugovora za razvoj sustava.
- Većina sustava zahtjeva niz osnovnih funkcionalnosti koje koriste različiti dijelovi sustava.
  - Kako zahtjevi nisu detaljno definirani dok određeni inkrement ne treba implementirati, može biti problem identificirati zajedničke elemente potrebne svim inkrementima.
- Struktura sustava opada kako se radi više izmjena, tj. dodaju novi inkrementi.
  - Zbog toga je potrebno trošiti vrijeme i novac na procjenjivanje koda (engl. *refactoring*) i neophodne izmjene kako bi daljnji rad bio moguć.

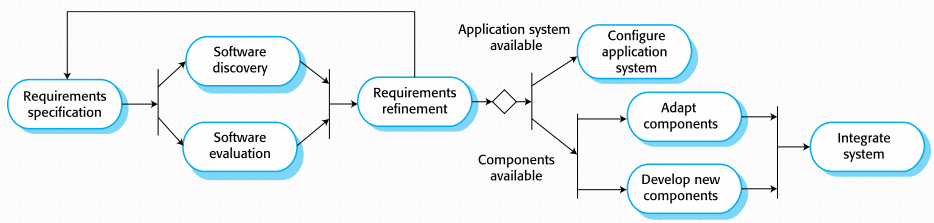
Integracija i konfiguriranje komponenti

- U većini softverskih projekata se događa ponovno korištenje na nekom nivou.
  - To se najčešće događa neformalno, kada ljudi koji rade na projektu znaju da postoji dizajn ili algoritam ili kod sličan onome što im treba. Oni ga pronađu i uključe u svoj sustav.
  - Ovakav način ponovnog korištenja ne ovisi o tome koji se razvojno proces koristi (vodopadni ili iterativni).
- Komponente koje se ponovno koriste s najčešće mogu konfigurirati tako da zadovolje potrebe različitih korisnika.
- Ovaj pristup se počeo sve više koristiti s pojavom standarda za izradu komponenti, a danas je standardan način za izradu različitih poslovnih sustava.

Integracija i konfiguriranje komponenti

- Bazira se na sistematskom ponovnom korištenju komponenti u slučajevima gdje su sustavi integrirani od postojećih komponenti ili COTS sustava.
- Kod razvoja ovakvih sustava postoje četiri osnovne faze:
  - Analiza komponenti (potraga za komponentama)
  - Modifikacija zahtjeva (kako bi se uskladili s komponentama)
  - Dizajn sustava uz korištenje komponenti
  - Razvoj i integracija

Integracija i konfiguriranje komponenti



Integracija i konfiguriranje komponenti

- ***Otkrivanje i procjena komponenti*** – Traže se komponente za implementaciju zahtjeva iz specifikacije i procjenjuju da se vidi u kojoj mjeri odgovaraju onome što se traži.
- ***Izmjena zahtjeva*** – Analiziraju se zahtjevi i informacije o pronađenim komponentama, te se zahtjevi modificiraju kako bi im odgovarali.
- ***Konfiguracija aplikacijskog sustava*** – Ukoliko je pronađen sličan sustav konfigurira se da odgovara korisničkim zahtjevima.
- ***Prilagodba postojećih i razvoj novih komponenti*** – Radi se prilagodba pronađenih komponenti, te se po potrebi razvijaju nove.
- ***Integracija sustava*** – Sve komponente se integriraju u funkcionalan sustav.

Integracija i konfiguriranje komponenti

- Prednosti:
  - Brža izrada softvera jer se ne troši toliko vremena na razvoj, a time se smanjuje cijena i rizik.
- Nedostatci:
  - Kompromisi na zahtjevima mogu uzrokovati da funkcionalnosti sustava ne odgovaraju kupcu.
  - Nove verzije komponenti ne ovise o onome ko ih koristi.

Integracija i konfiguriranje komponenti

- Postoje tri osnovne vrste softverskih komponenti koje se mogu koristiti u ovom procesu:
  - Web servisi koji se razvijaju prema određenim standardima i dostupni su za udaljene pozive.
  - Kolekcije objekata u obliku paketa koje se integriraju s platformom poput .NET ili J2EE.
  - Samostalni softverski sustavi (COTS) koji se podese za rad u određenoj okolini.

Procesne aktivnosti

Softverski procesi

- Osnovne aktivnosti:
  - Specifikacija zahtjeva
  - Dizajn i implementacija
  - Validacija i verifikacija
  - Održavanje

Specifikacija softverskih zahtjeva

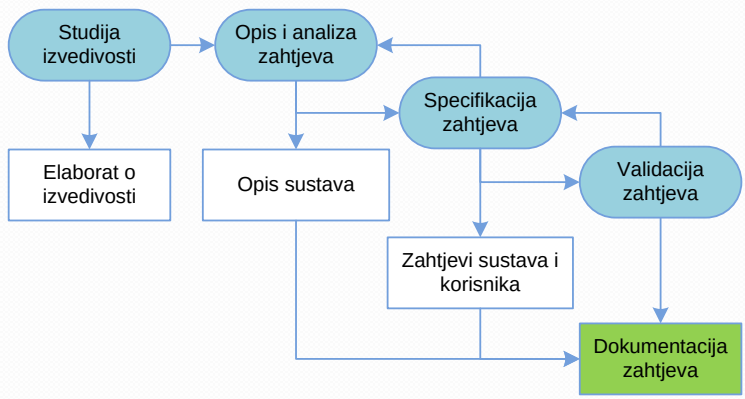
- Proces u kojem se utvrđuje:
  - koje usluge treba pružiti sustav;
  - koja su njegova ograničenja u radu (npr. broj korisnika koji mogu istovremeno pristupiti nekom podatku) i razvoju.
- Greška koja nastane u ovoj fazi jako je skupa pogotovo ako se pronađe tek kod verifikacije.

Cijena otkrivanja pogreške		Vrijeme otkrivanja pogreške				
		Specifikacija zahtjeva	Specifikacija dizajna	Implementacija	Testiranje	Puštanje u rad
Vrijeme uvođenja greške	Specifikacija zahtjeva	1x	3x	5-10x	10x	10-100x
	Specifikacija dizajna	-	1x	5-10x	15x	25-100x
	Implementacija	-	-	1x	10x	10-25x

Specifikacija softverskih zahtjeva

- Procesi specifikacije softvera:
  - ***Studija izvedivosti*** (engl. *feasibility study*) – Je li isplativo raditi sustav?
  - ***Opis i analiza zahtjeva*** – Što zainteresirane strane (engl. *steakholders*) zahtijevaju i traže od sustava?
  - ***Specifikacija zahtjeva*** – Detaljan opis zahtjeva.
  - ***Validacija zahtjeva*** – Provjera ispravnosti zahtjeva.

Proces specifikacije zahtjeva



Proces specifikacije zahtjeva

- ***Studija izvedivosti*** – Radi se procjena:
  - mogu li se zadovoljiti korisničke potrebe korištenjem dostupne tehnologije (hardvera i softvera);
  - hoće li sustav biti isplativ s poslovne strane i može li se napraviti u okviru dostupnog budžeta.

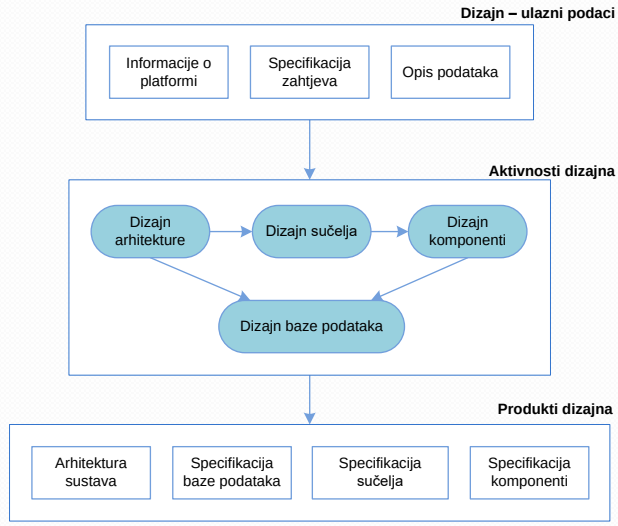
  Studija izvedivosti bi trebala biti brza i jeftina, a rezultat bi trebao odrediti ima li ići smisla dalje s detaljnijom analizom.
- ***Opis i analiza zahtjeva*** – "Otkrivaju" se zahtjevi sustava i to promatranjem postojećeg sustava, razgovorom s naručiteljem i korisnicima
  Može uključivati razvoj jednog ili više modela ili prototipova sustava.
- ***Specifikacija zahtjeva*** – Zapisivanje svih prikupljenih informacija o zahtjevima sustava u formalnom obliku .
- ***Validacija zahtjeva*** - Provjerava se koliko su zapisani zahtjevi realni, konzistentni i potpuni. Za vrijeme ove faze otkrivaju se i ispravljaju greške u specifikaciji.

- Sve ove aktivnosti se ne provode u predefiniranom redoslijedu.

Dizajn i implementacija softvera

- Proces konverzije specifikacije u sustav koji obavlja svoju funkciju.
- Dizajn softvera:
  - struktura programa koji će se realizirati, sučelja među komponenta, algoritama koji se koriste, organizacije podataka, ...
- Implementacija
  - Pretvorba ove strukture u izvršni program.
- Aktivnosti dizajna i implementacije su usko povezane i mogu se isprepletati ovisno koji se procesni model koristi.

Osnovni model procesa dizajna



Aktivnosti procesa dizajna

- *Dizajn arhitekture* – identifikacija osnovne strukture sustava, glavne komponente (pod-sustavi, moduli), njihove veze i distribucija.
- *Dizajn sučelja* –definiraju se sučelja među komponentama sustava.
- *Dizajn komponenti* – Analizira se svaka komponenta sustava i određuje na koji će način raditi.
- *Dizajn baze podataka* – detaljan dizajn organizacije podataka u sustavu, te na koji će način oni biti predstavljeni u bazi podataka.

Implementacija sustava

- Sustav se implementira razvojem ili konfiguriranjem složenog sustava kako bi odgovarao potrebama korisnika.
- Dizajn i implementacija se često preklapaju.
- U ovoj fazi se izvodi:
  - ***Programiranje*** - individualna aktivnost i ne postoje standardni procesi;
  - ***Uklanjanje grešaka*** – Aktivnost pronalaženja i ispravljanja grešaka u napisanom kodu.

Validacija softvera

- Verifikacija i validacija (V&V) se provodi s ciljem da se pokaže kako sustav odgovara specifikaciji i zahtjevima korisnika, te da radi bez grešaka.
- Uključuje proces provjere i testiranja.
- Testiranje sustava uključuje pokretanje sustava po testnim scenarijima koji se pišu iz specifikacije zahtjeva sa stvarnim podacima.
- Razine testiranja:

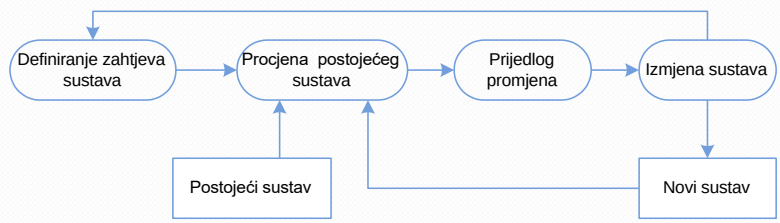
- Testiranje je najčešće korištena aktivnost V&V.

Razine testiranja

- Razvojno testiranje ili testiranje komponenti
  - Komponente se testiraju pojedinačno.
  - Komponente mogu biti funkcije ili objekti ili smislene grupe ovih entiteta.
- Testiranje sustava
  - Testiranje sustava kao cjeline. Posebno je važno testiranje izranjajućih (engl. *emergent*) svojstava.
- Test prihvata
  - Testiranje sustava s korisničkim podacima kako bi se provjerilo zadovoljava li sustav korisničke potrebe.

Održavanje (evolucija) softvera

- Softver je sam po sebi fleksibilan i može se lako mijenjati.
- Kako se zahtjevi mijenjaju kroz promjene poslovnih okolnosti, softver koji podržava poslovanje također treba evoluirati i mijenjati se.



Suočavanje s promjenama

Suočavanje s promjenama

- Promjene su neizbježne u svim velikim softverskim projektima.
  - Promjena poslovanja vodi prema novim ili promijenjenim zahtjevima.
  - Nove tehnologije otvaraju nove mogućnosti za poboljšavanje implementacije.
  - Promjena platforme zahtjeva promjene aplikacije.
- Promjene uključuju rad na analizi i prilagodbi postojećih zahtjeva, te implementaciji nove funkcionalnosti.

Smanjenje promjena

- ***Izbjegavanje promjena*** – u slučaju kada softverski procesu uključuju aktivnosti koje mogu predvidjeti moguće promjene treba ih primijeniti prije nego budu potrebne velike izmjene.
  - Npr. Razviti prototip sustava koji prikazuje osnovne karakteristike sustava korisniku.
- ***Toleriranje promjena*** – ukoliko proces dozvoljava omogućiti promjene na jednostavan način.
  - Najčešće podrazumijeva neki vid inkrementalnog razvoja. Predložene promjene se implementiraju u inkrementima koji još nisu razvijeni. Ukoliko to nije moguće onda se samo jedan inkrement (mali dio sustava) izmjeni kako bi se ostvarila promjena.

Suočavanje s promjenama

- Dva najčešća načina suočavanja s promjenama i izmjenama zahtjeva sustava su:
  1. ***Izrada prototipa sustava*** – Verzija ili dio sustava se brzo razvija za provjeru zahtjeva kupca i provjeru izvedivosti dizajna. Podržava izbjegavanje promjene jer omogućuje korisnicima da eksperimentiraju sa sustavom prije isporuke i tako poboljšaju svoje zahtjevi. Broj prijedloga promjene zahtjeva nakon isporuke će se stoga vjerojatno smanjiti.
  2. ***Inkrementalne isporuke*** – Inkrementi sustava se dostavljaju kupcu na eksperimentalni rad. Ovo podržava izbjegavanje i toleriranje promjene, jer se ne izrađuju odjednom sve funkcionalnosti sustava, već samo neke a onda se one nadograđuju u skladu s željama korisnika.

Izrada prototipa

- Prototip je inicijalna verzija sustava koja se koristi za demonstraciju osnovnih ideja, te za isprobavanje dizajna.
- Prototip se može koristit:
  - U procesu prikupljanja zahtjeva s ciljem pronalaska i potvrde zahtjeva.
  - U procesu dizajna s ciljem ispitivanja opcija i razvoj korisničkog sučelja.
  - U procesu testiranja za pokretanje "back-to-back" testova.

Prednosti izrade prototipa

- Veća uporabljivost sustava.
- Krajnji sustav bolje ogovara zahtjevima korisnika.
- Poboljšana kvaliteta dizajna.
- Lakše održavanje.
- Potrebno je uložiti manje truda u razvoj.

Odbacivanje prototipa

- Prototip treba odbaciti kada se krene u razvoj sustava, jer najčešće nisu dobra osnova za izradu pravog sustava:
  - Možda neće biti moguće ostvariti ne-funkcionalne zahtjeve;
  - Prototip nije dokumentiran;
  - Struktura prototipa je loša zbog velikog broja izmjena;
  - Prototip najčešće ne odgovara organizacijskim standardima kvalitete.

Poboljšavanje softverskih procesa

Poboljšavanje softverskih procesa

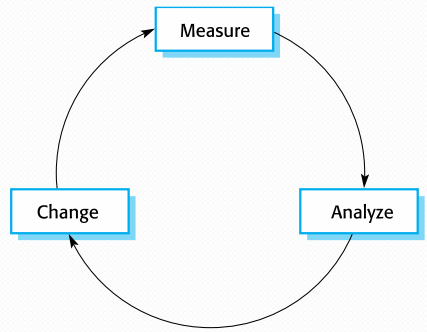
- Mnoge kompanije koje se bave razvojem softvera nastoje poboljšati svoje softverske procese kako bi poboljšali kvalitetu proizvedenog softvera, smanjili troškove ili ubrzali razvojni proces.

- Kako bi poboljšanja bila moguća važno je razumjeti postojeće procese i promijeniti ih kako bi se poboljšala kvaliteta i/ili smanjili troškovi ili vrijeme razvoja.

Pristup poboljšanju

- Dva su osnovna pristupa poboljšanju:
  - ***Zrelost procesa*** – fokusira se na poboljšanje procesa i načina upravljanja projektom, te uvođenje praksi programskog inženjerstva koje su pokazale dobre u praksi.
    - Nivo zrelosti procesa prikazuje mjeru u kojoj se dobra iskustva iz prakse primjenjuju u procesima razvoja softvera.
  - ***Agilni pristup*** – koji se zasniva na iterativnom razvoju i reduciranju svega što bi bio višak u procesu izrade softvera.
    - Osnovne karakteristike agilnih metoda su brza isporuka funkcionalnosti i mogućnost brze reakcije na promjenu korisničkih zahtjeva.

Ciklus procesa poboljšanja



Aktivnosti procesa poboljšanja

- ***Mjerenja procesa***
  - Mjeri se jedan ili više atributa softverskog procesa ili produkta. Na osnovu tih mjerenja može se donijeti odluka imaju li poboljšanja smisla.
- ***Analize procesa***
  - Trenutni proces se procjenjuje, te se identificiraju slabosti i kritične točke. Mogu se razviti modeli procesa kako bi se što bolje opisao.
- ***Promjena procesa***
  - Predlažu se određene promjene procesa koje se tiču glavnih nedostataka procesa. Uvode se promjene i nastavlja se daljnje mjerenje procesa kako bi se ustanovilo koliko su promjene imale utjecaja na proces.

Mjerenja procesa

- Kada je to god moguće dobro je mjeriti različite stvari vezane uz proces razvoja softver
  - Međutim u slučajevima kada organizacije nemaju jasno definirane standardne procese vrlo je teško odredit što mjeriti.

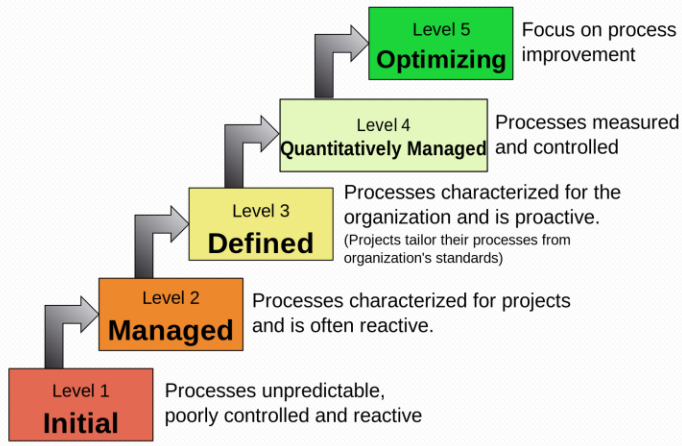
- Procesne metrike:
  - Vrijeme potrebno da se obave procesne aktivnosti
    - Npr. vrijeme ili napor potreban da se dovrši nekakva aktivnost ili proces.
  - Resursi potrebni za procese ili aktivnosti
    - Npr. ukupan napor u čovjek/danima.
  - Broj ponavljanja određenog događaja
    - Npr. broj otkrivenih problema

CMMI Model

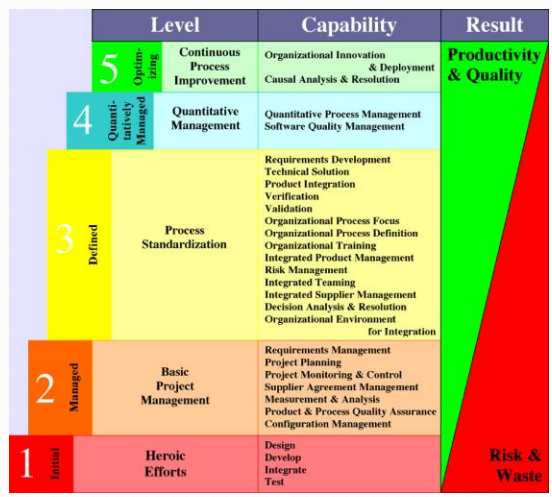
- **Capability Maturity Model Integration**(**CMMI**) je početno razvijen kao alat za objektivnu procjenu sposobnosti procesa izvođača kako bi se ustanovilo je li u stanju implementirati naručeni softverski projekt.
- Studiju korištenja ovog modela je financiralo Američko ministarstvo obrane i korišteni su podaci iz projekata koje je financiralo to ministarstvo.
- CMM model je razvio Institut programskog inženjerstva (engl. *Software Engineering Institute* - SEI)
- Pojam "zrelost" se odnosi na stupanj korištenja formalnih metoda i optimizacije procesa.

CMMI Model

Characteristics of the Maturity levels



CMMI model



CMMI model

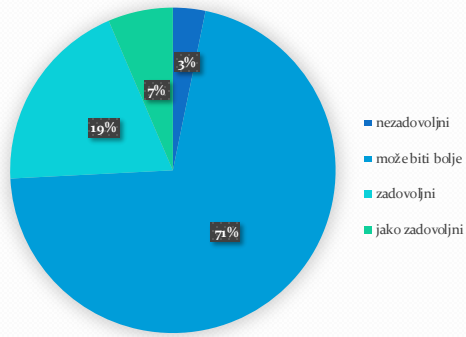
- Inicijalno stanje
  - U osnovi bez ikakve kontrole i procesi su nepredvidljivi. Uspjeh ovisi o individualnom naporu i ne može se garantirati da će se ponoviti.
- Ponavljajuće ili upravljivo stanje
  - Koriste se osnovne tehnike upravljanja projektom i uspjeh projekta se može ponoviti.
- Definirano stanje
  - Organizacija je razvila vlastite standarde softverskih procesa i vodi računa o dokumentaciji, standardizaciji i integraciji.
- Upravljano stanje
  - Organizacija nadgleda i upravlja vlastite procese kroz prikupljanje podataka i analizu.
- Optimizirano stanje
  - Procesi se stalno poboljšavaju korištenjem informacija koje se prikupljaju kroz trenutno korištene procese i uvode se novi procesi koji bolje služe određenim potrebama.

ISO 9001

- Jedan od ISO 9000 standarda koje je definirala Internacionalna organizacija za standarde (engl. *International Organization for Standardization, ISO)*.
- ISO 9000 standards specificiraju sustav kvalitete za proizvodnu i uslužnu industriju.
- ISO 9001 se bavi razvojem i održavanjem softvera i definira minimalnu prihvatljivu razinu kvalitete za softverske procese.
- Glavna razlika između ISO 9001 i CMMI je u tome što ISO 9001 definira minimalni prihvatljivi nivo kvalitete za softverske procese, dok CMMI uspostavlja okvir za neprestano poboljšavanje procesa i jasno definira kako to postići (za razliku od ISO 9001).

Primjer – anketa u soft. tvrtkama

- Svrha provede ankete je bilo prikupiti podatke o načinima na koje se provodi procjena napora potrebnog za razvoj softvera.
- Anketa je provedena u 31 tvrtki a sve se bave razvojem softvera u Hrvatskoj u periodu od lipnja do kolovoza 2018. godine.



Softverski produkti

- Softverski produkti su softverski sustavi koji pružaju određene funkcionalnosti korisnicima.
- Postoji veliki broj različitih produkata koje nude velike kompanije (npr. MS Office), preko aplikacija za osobnu upotrebu (npr. čitanje maila), pa do jednostavnih mobilnih aplikacija i igara.

Vizija softverskog produkta

- Početna točka razvoja softvera je tzv. „vizija produkta" (engl. *product vision*).
- Vizija produkta je skup jednostavnih izjava koje opisuju srž produkta koji se razvija.
- Vizija produkta bi trebala odgovoriti na tri osnovna pitanja:
  - Što se želi razviti?
  - Tko su ciljani klijenti i korisnici?
  - Zašto bi korisnici kupili ovaj produkt?

62

Moore-ov predložak vizije

- ZA KOGA
- KOME TO TREBA
- IME PRODUKTA i KATEGORIJA
- KLJUČNA PREDNOST i RAZLOG ZA KUPOVINU
- ZA RAZLIKU OD (slične aplikacije drugih proizvođača)
- OVAJ PROIZVOD (glavne razlike i prednosti)

Geoffrey Moore, *Crossing the Chasm: Marketing and selling technology products to mainstream customers* (Capstone Trade Press, 1998).

63

Primjer predloška vizije za iLearn sustav

FOR teachers and educators **WHO** need a way to help students use web-based learning resources and applications, **THE** iLearn system is an open learning environment **THAT** allows the set of resources used by classes and students to be easily configured for these students and classes by teachers themselves. **UNLIKE** Virtual Learning Environments, such as Moodle, the focus of iLearn is the learning process rather than the administration and management of materials, assessments and coursework. **OUR** product enables teachers to create subject and age-specific environments for their students using any web-based resources, such as videos, simulations and written materials that are appropriate.

Schools and universities are the target customers for the iLearn system as it will significantly improve the learning experience of students at relatively low cost. It will collect and process learner analytics that will reduce the costs of progress tracking and reporting.

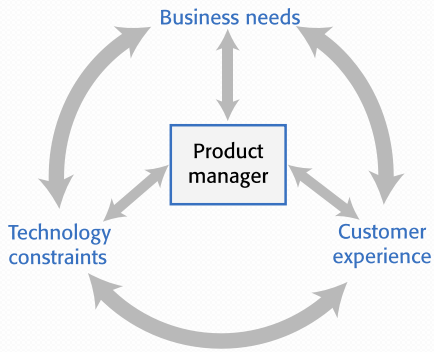
64

Upravljanje softverskim produktom

- Upravljanje softverskim produktom je poslovna aktivnost koja se bavi razvojem i prodajom softvera.
- Produkt menadžeri (engl. *Product Managers*, PM) su odgovorni za produkt i uključeni su u planiranje, razvoj i marketing.
- PM su sučelje između organizacije, tj. njenih klijenata i razvojnog tima. Uključeni su u sve faze života produkta, od početne ideje do povlačenja proizvoda sa tržišta.
- PM su fokusirani na korisnike i potencijalne korisnike, a ne na softver koji se razvija.

67

Zadaci PMa



- ***Poslovne potrebe*** - PM treba osigurati da softver koji se razvija odgovara poslovnim ciljevima kompanije koja ga razvija
- ***Tehnološka ograničenja*** - PM mora voditi računa da razvojni tim vodi računa o tehnološkim ograničenjima koja su važna klijentima.
- ***Doživljaj korisnika*** – PM treba biti u kontaktu s potencijalnim korisnicima kako bi shvatio što oni traže od produkta i na koji način se produkt može koristiti.

68

Technical interactions of product managers

- Product backlog creation and management
  - The product backlog is a prioritized 'to-do' list of what has to be developed. PMs should be involved in creating and refining the backlog and deciding on the priority of product features to be developed.
- Acceptance testing
  - Acceptance testing is the process of verifying that a software release meets the goals set out in the product roadmap and that the product is efficient and reliable. The PM should be involved in developing tests of the product features that reflect how customers use the product.
- Customer testing
  - Customer testing involves taking a release of a product to customers and getting feedback on the product's features, usability and business. PMs are involved in selecting customers to be involved in the customer testing process and working with them during that process.
- User interface design
  - Product managers should understand user limitations and act as surrogate users in their interactions with the development team. They should evaluate user interface features as they are developed to check that these features are not unnecessarily complex or force users to work in an unnatural way.

71

Softverski procesi

Kraj