**TEORIJA:**

**Dokumentacija** je bilo koja pisana ili slikovna informacija koja opisuje, definira, specificira, izvještava ili potvrđuje aktivnosti, zahtjeve, procedure ili rezultate. Omogućava koherentnu komunikaciju između članova razvojnog tima i sadrži informacije o programskom proizvodu namijenjene krajnjim korisnicima. Olakšava održavanje. Osigurava informacije namijenjene menadžmentu za planiranje i dodjelu budžeta

**Karakteristike dobre dokumentacije:** -Dobra dokumentacija doprinosi laganoj uporabi PP-a (povećana uporabljivost);-može utjecati na troškove održavanja PP-a;- može pomoći boljoj prodaji PP-a;-krajnji korisnici su zadovoljni s dokumentacijom prije će odabrati prateći PP unutar skupine više sličnih proizvoda.

**Životni ciklus dokumentacije:**Razvoj specifikacija,Izrada prototipa,Prva skica,Uređivanje,Pregled,Ispitivanje na uzorku stvarnih korisnika,Objava,Održavanje

**Plan(proces) razvoja dokumentacije**:Faze: prikupljanja podataka, prototipa, prve skice, druge skice,konačne verzije

**Dokumentacija sustava:** Zahtjevi, Opis arhitekture sustava,Opis arhitekture svake pojedine cjeline sustava,Opis funkcionalnosti i sučelja svake komponente sustava,Source – code,Dokumenti o validaciji,Upute za održavanje sustava(opis poznatih problema,opis SW/HW ovisnosti,predviđanja evolucije sustava)

**Pravila dobrog tehničkog pisanja:** RAZUMLJIVOST- Tehnički pisac-TP-treba prilagoditi pisanje tako da odgovara potrebama i zahtjevima krajnjih korisnika, Ne smije se izgubiti poruka koju TP želi prenijeti krajnjim korisnicima, JASNOĆA- Ispravno korištenje gramatike, konverzija, JEDNOSTAVNOST- Izbjegavati opisivanje nepotrb., JEZGROVITOST-Izbaciti dugačke rečenice bez značenja, PRECIZNOST- Navođenje izvora, Izbjegavati proturječne izjav, CJELOVITOST- određena misao koja se izražava potpuna

**Arhitektura:** je osnovna organizacija sustava koja sadrži komponente, njihove međusobne odnose i okolinu te principe dizajna i evoulcije.Ključni elementi arhitekture**:**Organizacija,komponente(moduli), odnosi komponenti(hijerarhija),okolina(sučelja),evolucija

**Modul** -ima specifičnu zadaću,funkcionira kao samostalni entitet(crna kutija), Odnosi između modula, Sučelja - nužna za komunikaciju (moduli su “crne kutije”), mora se znati koji su ulazi i izlazi modula (analogija s firewall-om),sučelje odražava funkcionalnost

**Česte greške kod definiranja arhitekture -** fokus isključivo na trenutno stanje i trenutne funkcionalnosti (zahtjeve), čim jednostavnija i brža za implementirati u početku,arhitektura se gradi s programskim rješenjem, zanemarivanje mogućih promjena u infrastrukturi, zanemarivanje mogućih promjena u sustavima s kojima je proizvod integriran (npr. rezervacijski sustav)

**Što se najčešće mijenja kod arhitekture?-**Algoritmi, Reprezentacija podataka, Promjene u infrastrukturi, Promjene iz društvene sfere, Evolucija proizvoda

**Uzorci i stilovi:** uzorci su općenitiji (3-slojna (3-tiered) , model-view-controller (MVC)) ,stilovi su specifičniji konkretnim problemima (višeslojna arhitektura (N-tiered),klijent – poslužitelj ,peer-to-peer , repozitorij , SOA (service-oriented architecture), objavi-pretplati (publish-subscribe)

**Višeslojna arhitektura:**tradicionalna, n-tiered, Svaki sloj komunicira isključivo sa slojem ispod sebe, 3-slojna arhitektura(prezentacijski sloj, sloj poslovne logike,podatkovni sloj), Iskoristivost svakog sloja (re-usability), Kombinacija n-slojeva s ostalim uzorcima(klijent-poslužitelj + MVC + 3-slojna arhitektura)

**Klijent-poslužitelj:**tradicionalna, Klijent(prezentacija,pred-obrada podataka,komunikacija s bazom podataka posredstvom poslužitelja), Poslužitelj(obrada zahtjeva,upravljanje klijentima,sigurnost,konzistentnost baze podataka)

**Peer-to-peer:**Generalizirani klijent-poslužitelj(klijent i poslužitelj u jednom), distribuiranost, Mreža ravnopravnih čvorova(svaki čvor obavlja dio procesiranja),PrimjerP2P:seti@home,skype

**MVC-Model-View-Controller:**C->M->V,odvaja model,akcije,prezentacije, Kontroler(presreće zahtjeve klijenta i prilagođava model,obavještava model o promjeni zahtjeva korisnika), Pogled(mijenja se prema modelu,ono što korisnik vidi),Model(logika,mijenja pogeld na zahtijev kontrolera, elementi su crne kutije

**Repozitorij**: centralni repozitorij kao resurs koji koriste ostali elementi, Ostali elementi arhitekture ne komuniciraju međusobno, sinkronizaciji pristupa – konkurentnost, Najčešće baza podataka sa dijelom poslovne logike(centralizirana,distribuirana)

**SOA:**usmjerena uslugama, klasičan princip-moduli i sučelja, usluge i komunikacija između usluga i klijenata, Opis usluge (servisa) -WSDL (Web Service Description Language), Komunikacijski protokol -SOAP (Simple Object Access Protocol), Zapis podataka-XML, JSON, Integracija -jednostavnija jer sve usluge “pričaju istim jezikom”. Međusobna neovisnost usluga - loose coupling

**Objavi-pretplati:**model komunikacije, Jedan entitet objavljuje, više entiteta je pretplaćeno na objavu, Entitet koji objavljuje ne zna tko su pretplatnici(loose coupling, ne šalje samo objavljuje, pretplatnici “uzimaju” objave), Obično se koristi broker poruka

**Pogledi na arhitekturu:**logički(raspored modula i odnosi između njih),procesni( komunikacija, konkurentnost),fizički(hardware),razvojni( softverskih komponenti ),slučajevi uporabe

**Parametri kvalitete arhitekture:**pouzdanost,raspoloživost,portabilnost,skalabilnost,performanse,sigurnost

**Odabir tehnologije:**Otvoreni kod (besplatan,naplaćuje se podrška,velika zajednica razvijatelja),Vlasnička rješenja(skupo na početku, godišnje licence, dobra podrška, stabilnija opcija), Razvoj vlastitog rješenja(dugotrajno, zna kako sve radi-lakše ispravljanje,za manje projekte),Radni okviri(ubrzava razvoj,crna kutija-problem kod greške)

**Uzroci promjena programskog proizvoda**nove verzije pp(različiti uređaji, tehnologije ,nove funkcionalnosti, ispravci neispravnosti), artefakti na koje utječu-zahtjevi,dizajn,kod,dokumentaciju

**Strategije promjene PP:**promjene funkcionalnosti(Unošenje promjena u PP zbog promjene zahtjeva, bez promjene arhitekture), Promjena arhitekture(iz centralizirane u distribuiranu), Re-inženjerstvo(Promjena strukture bez dodavanja novih funkcionalnosti)

**Aktivnosti održavanja:**Prije isporuke(planiranje akcija koje će se poduzeti nakon isporuke, pogodnost za održavanje,logistika održavanja), Nakon isporuke(modifikacija programa, obuka korisnika,organiziranje službeza podršku korisniku)

**Ciljevi održavanja programskih proizvoda:**ispravljanje grešaka,izrada sučelja prema drugim sustavima,ugradnja različitih poboljšanja,izvedba promjena sustava

**Osnovni aspekti na koje se održavanje fokusira :**održavanje kontrole nad svakodnevnim funkcioniranjem sustava,održavanje kontrole nad modifikacijama sustava, poboljšanje postojećih funkcija, prevencija performansi sustava od degradacije na neprihvatljivu razinu

**Problemi održavanja:**Ograničeno razumijevanje(loša dokumentacija, 40-60% troškova održavanja), Problemi testiranja(teško odrediti razinu napora testiranja, ažurnost regresijskih testova), Analiza utjecaja, Procjena troškova i napora, Degradacija arhitekture

**Napori održavanja:**Produktivne aktivnosti(analiza i procjena, modifikacije dizajna,kodiranje),Aktivnosti prikupljanja informacija(razumijevanje koda, interpretiranje struktura podataka, značajke sučelja)

**Troškovi održavanja:** , M-ukuni napori, -napori uloženi u produkvitne aktivnosti,K-konstanta,c-mjera složenosti zbog lošeg dizajna/dokumentacije,d-mjera bliskosti održavatelja i PP

**Razlozi troškova perfektivnog oržavanja:** Održavatelji - nedostatak iskustva s aplikacijom, Nestrukturirani razvoj, Uvođenje promjene - nove neispravnosti - novi zahtjevi za modifikacijom, tendencija degradacije strukture sustava, Gubitak veze između određene verzije i dokumentacije, Stabilnost tima

**Mjerenja i metrika:** Mjerenja:( Kvantitativne informacije o programskom procesu ,Procjena prednosti i nedostataka procesa,Kontekst odnosa procesa i njegovih rezultata), Metrika(Određuje u kojoj mjeri programski proces sadrži zahtijevana svojstva,), SW proces->Prikupljanje podataka->Mjerenja->Metrika->Procjena

**Metrika procesa održavanja:** Procesna metrika za procjenu održavanja(

Broj zahtjeva korektivnog održavanja,Prosječno vrijeme provođenja analize utjecaja,Prosječno vrijeme implementacija zahtjeva za promjenom,Prosječan broj neimplementiranih zahtjeva za promjenom),Ako jedan od faktora počne rasti – smanjena pogodnost za održavanje.

**Zahtjev za modifkacijom MR:** Standardizirana forma,Generiran od strane razvijatelja PP, Popunjava korisnik koji zahtijeva aktivnosti održavanja,Opis svih uvjeta koji su doveli do pojave neispravnosti,Specifikacija promjena,MR dokument služi kao osnova za planiranje aktivnosti održavanja

**Konsenzualni patent :**posebni oblik patentne zaštite u Hrvatskoj , priznaje se bez potpunog ispitivanja, tj. na temelju sporazuma (konsenzusa) javnosti, ukoliko protiv njega nema prigovora zainteresiranih strana ,u bilo kojem trenutku bilo koja zainteresirana osoba može pokrenuti potpuno ispitivanje

**Najčešći problemi patenta:** Opis u prijavi ne prikazuje dobro svojstva izuma, Izum nije nov - bio je otkriven javnosti ili je već poznat i objavljen Izum predstavlja tek ideju, načelo ili metodu, a ne konkretnu tehničku izvedbu, U postupku nije pravovremeno izvršena neka radnja, Prijavitelji nisu bili svjesni u što se upuštaju

**Autorsko pravo copyright:** Bernska konvecnija-automatizam autorskog prava, programski kod automatski postaje autorsko djelo u onom trenutku kada se postavi na “opipljiv” medij, autorsko pravo vrijedi samo za kod,ne i za ideju!

**Odricanje od autorskog prava – copyleft:** suprotno od autorskog prava,! uz navođenje autora i uvjet da sve izvedenice moraju biti copyleft, GNU GPL

**Vlasničke licence :**potencijalna zarada ,važni i tehnički mehanizmi zaštite ,“pravo” na pružanje podrške za programski proizvod ,patentibilnost

**Licence otvorenog koda :**dijeljenje i unaprjeđivanje rješenja ,atribucija (navođenje) , veća prodornost do korisnika,potencijal podrške korisnicima

**GNU General Purpose Licence (v3):** dozvoljeno komercijalizirati, modificirati i distribuirati ,nužno je objaviti cijeli izvorni kod novog rješenja, Lesser GPL: ako se LGPL rješenje ne modificira i koristi se kao zamjenjiva “crna kutija” nije uvjet da cijeli izvorni kod bude objavljen ,nije dozvoljeno statičko povezivanje LGPL komponenti, Aplikacija/usluga instalirana kod kupca GPL – kupac dobija cijeli izvorni kod aplikacije i ima ga pravo dijeliti s licencom GPL, Aplikacija/usluga na našoj opremi korisnici/kupci pristupaju putem preglednika weba , GPL – sve dok izvršni kod ne dostavimo kupcu ne moramo otkrivati izvorni kod

**Licence otvorene:** Apache, MIT, BSD :dopuštena modifikacija i komercijalizacija bez odavanja izvornog koda ,zahtjevaju navođenje (attribution) i navođenje modifikacija (Apache) ,bitna razlika – patenti: Apache regulira (uz GPL)!, “najtolerantnije” ,Mozilla, Eclipse.., kompromis između BSD i sličnih licenci i strožeg GPL-a , odavanje dijela izvornog koda ,Creative Commons :može se koristiti za kod iako nije preporučeno ,prikladna za dokumentaciju, upute, fotografije ili predavanja ,različite izvedenice osnovne licence

**Najvažniji parametri licenci:** Komercijalna primjena, Objava izvornog koda, Modificiranje, Distribucija, Statičko povezivanje(može li se licenciranu komponentu zamijeniti nekom drugom), Podlicenciranje, Odgovoronost

**Dvostruko licenciranje:** ako proizvod ostaje otvorenog koda onda je “besplatno” ,ako planirate prodavati proizvod a ne želite da ostane otvorenog koda onda morate kupiti komercijalnu licencu, enterprise verzija – vlasničko licenciranje

**Najčešći modeli licenciranja:** Licenca po korisniku: licenca je vezana za osobu – “imenovanog korisnika” ,web-aplikacije (jedan korisnički račun) , samostojeće aplikacije koje se instaliraju na jedan (ili više!) uređaja, Licenca po instalaciji/uređaju: licenca vezana za jednu instalaciju ili uređaj , može je koristiti više osoba, Licenca po klijentu: arhitekture klijent-poslužitelj , licenca vezana uz jednog klijenta – ne nužno i korisnika! , klijent može biti i programska komponenta, Licenca po razvijatelju: primjenjiva za licenciranje programskih okvira i knjižnica – alata za razvoj programskih rješenja , svaki razvijatelj koji koristi proizvod mora se moći autorizirati, Licenca po lokaciji: licenca vrijedi za sve osobe i uređaje na određenoj lokaciji , lokacija – fizička ili logička ,ured, odjel tvrtke, Prenosiva licenca: vrijedi za više korsinika na više uređaja, ali: može je koristiti samo jedan korisnik na jednom uređaju u određenom trenutku ,specijalzirani alati i proizvodi, Licenciranje otvorenog koda: puno donekle različitih licenci – najkorišteniji GNU GPL , odabir licence ovisi o dugoročnom planu proizvoda , “svi mogu koristiti uz navođenje” – MIT, BSD ,! “svi mogu koristiti uz navođenje bez patentnih prava” – Apache , “svi mogu koristiti uz navođenje i objavu koda bez patentnih prava” – GPL, uvjetno LGPL, Eclipse, Mozilla

**ISPITI:**

**4 pravila pisanja dobre dokumentacije:**Pisati dokumentaciju s gledišta krajnjeg korisnika; Ispravna gramatika; Izbjegavati nepotrebna ponavljanja; Izbjegavati dvosmislenost; Koristiti standarde organizacije; Bilježiti logički,Održavati dokumentaciju aktualnom

**4 vrste dokumentacije:** 1. PROJEKTNA-Planovi, troškovi, rokovi, norme; Projektna dokumentacija kreirana za uspješno upravljanje projektom razvoja PP,2. PROCESNA-Bilješke o procesu razvoja i održavanja PP-a; Relativno kratki životni ciklus; Važna u internom procesu razvoja,3. DOKUMENTACIJA O PP-u (Sistemska)-Opisuje PP koji se razvija/održava i način na koji radi ali ne i kako radi; Dokumentacija o sustavu; *Primjer*: specifikacije zahtjeva, opis arhitekture, komentirani izvorni kod, planovi; Testiranja s uključenim testnim uzorcima, izvještaj o poznatim neispravnostima,4. KORISNIČKA-kup dokumenata koji opisuju PP; postupak instalacije i puštanja u rad, administracija i održavanje; opis PP-a, opis rada i načina korištenja.

**Veza procesa razvoja PP-a i procesa kreiranja dokumentacije.** Dokumentacija PP-a mora odgovarati stvarnom opisu PP-a,Konzistentnost se osigurava uspostavljanjem dobro prilagođenog procesa razvoja dokumentacije ,unutar okvira osnovnog procesa razvoja PP-a.Proces treba imati :-dobro definirane spojne točke između razvoja PP-a, testnih aktivnosti i razvoja dokumentacije,-dobro definirane odgovornosti svake pojedine uloge koja sudjeluje u procesu razvoja PP i u procesu razvoja dokumentacije

**Navedite i obrazložite optimalnu i uobičajenu početnu točku za razvoj korisničke dokumentacije.-**Optimalna – blizu kraja faze specifikacije i dizajna,Uobičajena – nakon početka faze ispitivanja,Faze: početka( planiranje razvoja dokumentacije, optimalna točka projektne dok),specifikacije i dizajna(optimalna točka korisničke dok), konstrukcije, ispitivanja(uobičajena točka korisničke dok), isporuke(između, ažuriranje dok), održavanja

Izrada dokumentacije mukotrpan je i težak posao, a većina ljudi ga ne voli raditi. Zato se često radi što kasnije, jer su moguće brojne promjene u skladu s kojima je potrebno mijenjati i dokumentaciju. Zato se često korisnička dokumentacija piše tek nakon što se stabiliziraju korisnički zahtjevi i broj njihovih promjena se značajno smanji.

**Norme dokumentacije:**-procesna:definirati proces za razvoj kvalitetne dokumentacije,-norma proizvoda:definirati kojeinfo o PP-u dokumentacija treba sadržavati, kako će izgledati,-norma za promjene:koji alati će se koristiti za održavanje dokumentacije i osiguravanje praviila formatiranja

**Dobro strukturirana dokumentacija**-Naslovna stranica:naslov,naziv projekta, razvojni tim, datumi, čitatelji, povjerljivost podataka,-Podjela na poglavlja i podpoglavlja,-Sadržaj dokumenta,-Rječnik  
-Tablice,-Slike

**Dokumentacija važna za održavanje:** Dokumentacija o programskim zahtjevima; Dokumentacija o arhitekturi; Programski kod; Dokumenti o testiranju; Dokumenti o prethodnim zahvatima održavanja

**Kategorije održavanja:**-Korektivno:pronalaženje i ispravljanje pogrešaka pronađenih u programu nakon isporuke(20% uk.troškova)-ono što se nije otkrilo tijekom testiranja,Ispad:izlazno stanje definirano specifikacijom nije isto kao i stvarni rezultat izvođenja,Neispravnost:dio programskog koda koji pod određenim uvjetima uzrokuje ispad programa,-Adaptivno:održavanje inicirano promjenama u programskoj okolini(novo sklopovlje, OS, nova verzija OS-a)(25%),-Perfektivno:zadovoljavanje novih i modificiranih potreba korisnika; promjena postojećih ili dodavanje novih funkcionalnosti(55%)-uvijek od strane korisnika; nedostatak iskustva s aplikacijom održavatelja; nestrukturirani razvoj-2.Lehmanov zakon,-Preventivno:sprječavanje eventualnih problema prije nego se pojave(5%)

**Nasljeđeni sustavi(Legacy Systems):-**naslijeđeni sustavi su stari sustavi koji postaju teški za održavanje;dokumentacija i tehnologija zastarjeli; akumulacija promjena narušava modularnost originalne arhitekture

**Aktivnosti održavanja-**Nestrukturirano održavanje:dostupan jedino kod PPa; dokumentacije nema ili je oskudna; analiza koda->konfiguracija, performanse PPa, ograničenje dizajna; teško procijeniti utjecaj promjena i nemoguće regresijsko testiranje(dio testiranja sustava kojim se provjerava je li unošenje promjena prouzročilo neispravnosti unutar programskog proizvoda),Strukturirano održavanje:PP je razvijen primjenom metodologija programskog inženjerstva; poznata konfiguracija PPa, dostupna dokumentacija; analiza dokumentacije o dizajnu->određivanje značajki PPa; planiranje promjena i primjena regresijskog testiranja

**Lehmannovi zakoni:** 1. Lehmannov zakon - KONTINUIRANA PROMJENAProgramski sustav koji se koristi u realnoj okolini mora se nužno mijenjati, inače postaje progresivno sve manje koristan u toj okolini. 2. Lehmannov zakon - POVEĆANJE SLOŽENOSTI Uvođenjem promjena u neki programski sustav, njegova struktura ima tendenciju degradacije.čim se nešto mijenja povećava se entropija i složenost 3. Lehmannov zakon - EVOLUCIJA VELIKIH PROGRAMSKIH SUSTAVA Veliki sustavi imaju vlastitu dinamiku evolucije koja se uspostavlja u ranim fazama razvoja (održavanjem se ne može postići svaka željena promjena) nije moguće implementirati svaku promjenu

**Re-inženjerstva i reverznog inženjerstva** Reverzno inženjerstvo (Reverse Engineering ) Proces analize nekog programskog sustava kako bi se identificirale komponente sustava i njihovi međuodnosi s ciljem kreiranja prikaza sustava na višoj razini apstrakcije. Ova tehnika ne mijenja sustav i ne kreira novi sustav. Primjer je obnavljanje dokumentacije i dizajna. Iz izvršnog koda se izvlači logika funkcioniranja sustava Programsko reinženjerstvo (Re-engineering ) Kombinacija reverznog inženjerstva i restrukturiranja programskih podataka, arhitekture i logike razvije se dio sustava koji radi isto što i stari dio, ali ima poboljšane performanse Cilj: razvoj programskog proizvoda iste funkcionalnosti i veće kvalitete Ponovna implementacija cijelog ili dijela naslijeđenog sustava kako bi postao pogodan za održavanje; Pogodno kad neki dijelovi sustava zahtijevaju česte zahvate održavanja;Uz sustav treba se ažurirati i dokumentacija

**Održavanje PP od strane nekog vanjskog odjela/firme:**Prednosti: veća pažnja posvećuje se stvaranju bolje dokumentacije; uspostavlja se formalna procedura za prebacivanje programskog proizvoda iz faze razvoja u fazu održavanja; osoblje koje vrši održavanje ima mogućnost prepoznavanja prednosti i mana programskog proizvoda Nedostaci: prebacivanje sustava iz razvoja u održavanja može biti sporo; pojavljuju se problemi refundiranja troškova održavanja; potrebno je određeno vrijeme da organizacija/odjel za održavanje stekne znanja o novom programskom proizvodu koji je preuzet

**Održavanje PP od strane odjela za razvoj PP-a:**Prednosti:osobe koje su se bavile razvojem nekog programskog proizvoda imaju najbolje znanje o njemu; nema potrebe za detaljnom dokumentacijom; nema potrebe za uspostavom formalnog komunikacijskog sustava između održavanja i razvojaNedostaci:preveliko dotjerivanje(ono što se nije stiglo pri razvoju)

**Pogodnost za održavanje i koji faktori utječu na nju-**Mjera za definiranje jednostavnosti (lakoće) s kojom se neki programski proizvod može razumjeti, ispraviti, prilagoditi ili poboljšati.Pogodnost za održavanje: 1. Upravljački faktori(metodologija razvoja-dizajn,kodiranje, razvojna okolina-osoblje, progr.jezici), 2. Kvantitativna mjerenja (odnose se na vrijeme)

**Intelektualno vlasništvo:**-Industrijsko vlasništvo: patent, žigovi, industrijski dizajn,topografija poluvodičkih proizvoda,-Autorsko pravo i srodna prava:odnosi se na intelektualne tvorevine poput: književno, glazbeno, fotografska,koreografsko djelo, prijevodi

**Razlike intelektualnog vlasništva:** Patent:Rok(20g,10g),Gdje(nacion,međunarod),Protiv(upotrebe,proizvodnje,prodaje),Što(invencija), Znak:Rok(15g,obnovljivo),Gdje(nac,međun),Protiv(kopiranje,upotrebe),Što(identitet),Autorstvo:Rok(Život+,50g),Gdje(autom,međun),Protiv(kopiranje,reprodukcija),Što(Orig.rad)

**Patent:**-isključivo pravo priznato za izum koji nudi novo rješenje nekog tehničkog problema(može se prenositi i ima ga samo vlasnik)-svojstva:novost, primjenjivost, inventivnost(90% su unapređenja postojećih)

**Patentni sustav**-razmjena u kojoj izumitelj pristaje otkriti društvu pojedinosti svoga izuma u zamjenu za pravo isključivog korištenja tijekom određenog razdoblja

**Patentna prijava**-izričita naznaka da se zahtijeva priznanje patenta,opis izuma postaje javno dostupan svimzainteresiranima,Sadržaj:-naziv izuma(bez žigova i komerc naziva), podaci o podnositelju prijave(tvrka), podaci o izumitelju(fiz.osob), opis izuma(teh.detalji na nivou koji omogućuje izradu)

**Opis izuma**-područje tehnike:A do H naznaka oznake,-tehnički problem:ne cilj već samo opis problema,-stanje tehnike:slična rješenja(opis 3-5),što je riješeno i na koji način,kritika,-izlaganje biti izuma:novost rješenja u odnosu na ona opisana kod stanja tehnike,-način ostvarivanja izuma:detaljan opis bar jednog načina ,-primjena izuma:način industrijske ili druge primjene,-patentni zahtjevi:moraju biti sastavljeni tako da definiraju izum isključivo tehničkim karakteristikama

**Tehnologija**-koristiti tehnologiju s kojom se ima najviše iskustva-dva tipa rješenja:1.open source:besplatan, naplaćuje se podrška, velika zajednica razvijatelja2.vlasnička:skupo na početku(godišnje licence), dobra podrška, obično stabilnija opcijaa)razvoj vlastitog rješenja:dugotrajno, zna se točno kako sve funkcionira,dobro za manje projekte)korištenje radnih okvira:drastično ubrzava razvoj, problem ako dođe do greške