Programsko inženjerstvo i informacijski sustavi

Josip Nađ

Čakovec, 2020.

dr. sc. Josip Nađ
Programsko inženjerstvo i informacijski sustavi

Recenzija:

prof. dr. sc. Domagoj Hruška
Sveučilište u Zagrebu, Ekonomski fakultet

dr. sc. Bruno Trstenjak, viši predavač
Međimursko veleučilište u Čakovcu

Lektura:

Isidora Stupar, prof.

Nakladnik:

Međimursko veleučilište u Čakovcu

Za nakladnika:

doc. dr. sc. Igor Klopotan

ISBN: 978-953-8095-18-4

Sadržaj

1 Uvod

Izazovi poslovanja rastu nesmiljenom brzinom i stavljaju pred menadžere odgovornost izbora i dizajniranja odgovarajućeg informacijskog sustava za svoju organizaciju. Informacije se moraju sakupljati, obrađivati, spremati i distribuirati, a da bi to sve radilo mora postojati dobro kreiran, provjeren, pouzdan i funkcionalan softver. Da bi takav softver dobio odgovarajuću formu, potrebno je, prije svega, detaljno proučiti i analizirati poslovne zahtjeve.

Ovaj pisani materijal prikazuje osnovna pravila igre prilikom uspostave poslovnih informacijskih sustava, od same ideje i kategorizacije poslovnih zahtjeva, sve do same implementacije i početka korištenja jednog velikog informacijskog sustava. Ovdje nije riječ o programiranju i kodiranju, nego isključivo o pripremi posla za programere i kasnijem testiranju i korištenju programa, koji su programeri kreirali na osnovi dobre pripreme posla. Za te je aktivnosti usvojen termin „Programsko inženjerstvo“, a primjena će biti fokusirana prvenstveno na informacijske sustave.

S obzirom na značaj koji za opću dobrobit društva ima proizvodnja, razmatrat će se isključivo informacijski sustavi u proizvodnim tvrtkama. Da bi razmatranje bilo standardizirano, na svim primjerima u sklopu nastave koristit će se tzv. „Osnovni proizvodni model“, koji je prikazan u poglavlju 2. Na vrlo jednostavan način objasnit će se aktivnosti primarnih poslovnih funkcionalnosti: prodaja, logistika, proizvodnja i nabava, i djelomično kontroling (interno računovodstvo), u područjima izračuna proizvodne cijene i prikupljanja proizvodnih troškova.

Poglavlje 3 posvećeno je inženjerstvu zahtjeva, s obrađenim relevantnim slučajevima koji se odnose na sve poslovne funkcije prikazane u osnovnom proizvodnom modelu. Za inženjerstvo zahtjeva ne može se reći da je najvažniji korak u programskom inženjerstvu, ali je toliko bitan da se može usporediti s autokartom ili navigacijom kad vozimo po nepoznatom terenu.

Poglavlja 4–7 posvećena su preostalim procesima i aktivnostima programskog inženjerstva (dizajniranje, testiranje, isporuka i održavanje), dok su poglavlja 8–15 usmjerena na tematiku informacijskih sustava. Ukazuje se na bitnu povezanost poslovanja i informacijskih sustava, te se dodatno naglašava povećana potreba za sudjelovanjem visokog menadžmenta u izboru strategije vezane za digitalizaciju poslovanja. Digitalna transformacija više nije izbor – postala je neminovnost, odnosno neophodan korak u procesu preživljavanja u našoj novoj realnosti.

U nastavku ovog uvodnog poglavlja bit će predstavljene glavne definicije programskog inženjerstva i informacijskih sustava, kao polazište za daljnja razmatranja.

1.1 Programsko inženjerstvo

Programsko inženjerstvo je disciplina bazirana na računarstvu, menadžmentu, ekonomiji, vještini komunikacije i nadasve inženjerskom pristupu rješavanju problema. Dakle, programsko inženjerstvo nije računarska znanost, a također nije ni programiranje. To je disciplina koja prati razvoj softvera, počevši od definiranja zahtjeva pa sve do problematike održavanja softvera nakon što se on počne koristiti.

Definicija prema [4]:

- Programsko inženjerstvo je primjena sustavnog, discipliniranog i kvantificiranog pristupa razvoju, radu i održavanju softvera.

Programsko inženjerstvo uključuje znanje, alate i metode za prikupljanje zahtjeva, dizajn, izradu, testiranje i održavanje softvera.

1.1.1 Principi oblikovanja softvera

U literaturi se uvijek spominje sljedećih pet principa:

- Uvođenje inženjerski propisanih postupaka (procedura) u proces oblikovanja softvera (precizno definirati faze procesa oblikovanja, tko radi, što radi i kada radi). Disciplina programskog inženjerstva koja se bavi ovim principom zove se Inženjerstvo zahtjeva.
- Svaku fazu procesa oblikovanja softvera treba dokumentirati, po mogućnosti na standardan i formalan način. Važan zaključak: rezultat svake faze procesa je dokument!
- U oblikovanje softvera treba uvesti analizu i izbor stila arhitekture softvera, kao i pripadne modele pogodne za formalnu analizu. Najrašireniji stil arhitekture je „Objektno usmjeren pristup“.
- Softverski produkt treba oblikovati u manjim cjelinama / komponentama i treba nastojati višestruko iskoristiti komponente.
- Uz tradicijsko ispitivanje (testiranje) treba uvesti formalne metode provjere (verifikacije) modela softvera. Formalna verifikacija je postupak provjere zadovoljava li sustav specifikaciju na ispravan način (odsustvo kvarova).

1.1.2 Glavne aktivnosti u proizvodnji softvera

Generalno je usvojen pristup s četiri glavne aktivnosti:

- izrada specifikacije (na temelju analize zahtjeva treba specificirati što sustav treba činiti i koja su očekivana ograničenja u razvoju)
- razvoj softvera (dizajn i programiranje)
- validacija i verifikacija (potvrđivanje ispravnog rada softvera)
- evolucija softvera (rukovanje promjenama sukladno korigiranim / novim zahtjevima).

1.1.3 Svojstva dobrog softvera

Svojstva dobrog softvera su da je isporučen u skladu s definiranim zahtjevima, da razvija tražene performanse, te da je održiv, pouzdan i upotrebljiv.

Prema [3], u programskom inženjerstvu su *kvaliteta, cijena i vrijeme* tri međusobno ovisna faktora. Na žalost, moguće je kontrolirati bilo koja dva od navedena tri faktora, ali nikako sva tri.

Slika 1: Omjer kvalitete, cijene i vremena

Na slici 1 je vidljivo da nije moguće u kratkom vremenu dobiti neku kvalitetnu i jeftinu aplikaciju:

- Ako se traži brzina izrade, mora se žrtvovati ili cijena ili kvaliteta.
- Ako se traži visoka kvaliteta, mora se žrtvovati ili cijena ili brzina izrade.
- Ako se traži niska cijena, mora se žrtvovati ili brzina ili kvaliteta.

Svojstva softvera vezana za faktor kvalitete:

- robusnost
- točnost
- pouzdanost
- djelotvornost
- proširivost
- održivost
- cjelovitost.

Svojstva softvera vezana za faktor vremena:

- razvoj
- implementacija
- trajanje.

Svojstva softvera vezana za faktor cijene:

- razvoj
- implementacija
- korištenje
- održavanje.

1.1.4 Temeljne postavke programskog inženjerstva

Osnovni principi:

- Sustavi se trebaju razvijati pomoću upravljanog i razumljivog procesa razvoja (za različite se vrste softvera koriste različiti procesi).
- Pouzdanost i performanse važni su za sve vrste sustava.
- Razumijevanje i upravljanje specifikacijama softvera i zahtjevima su važni.
- Ako je prikladno, trebalo bi moći ponovo koristiti softver koji je već razvijen, a ne pisati novi softver.

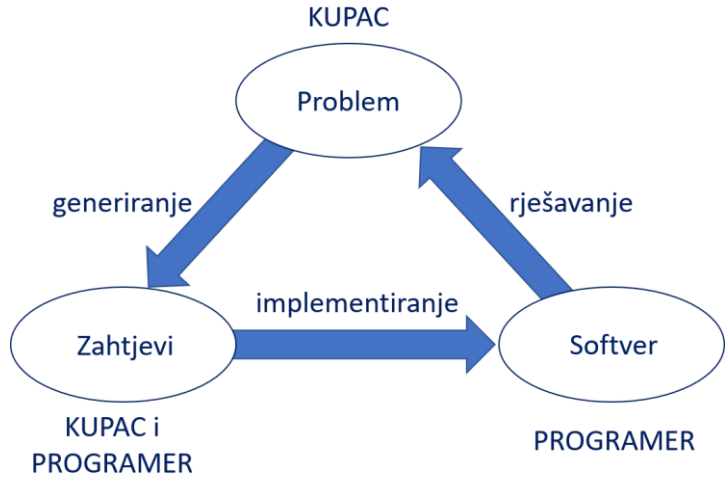
Etika programskog inženjerstva:

- Programsko inženjerstvo uključuje šire odgovornosti od obične primjene tehničkih vještina.
- Programski inženjeri moraju se ponašati pošteno i etično da bi ih se moglo uvažavati kao profesionalce.
- Etičko ponašanje je više od jednostavnog pridržavanja zakona i uključuje slijeđenje niza načela koja su moralno ispravna.

Primarni cilj programskog inženjerstva je unapređenje kvalitete softvera i povećanje produktivnosti samog procesa kreiranja softvera. U postizanju tog cilja, razlikuju se dva prilično različita pristupa: projektni (engl. *project-based* SE) i proizvodni (engl. *product-based* SE).

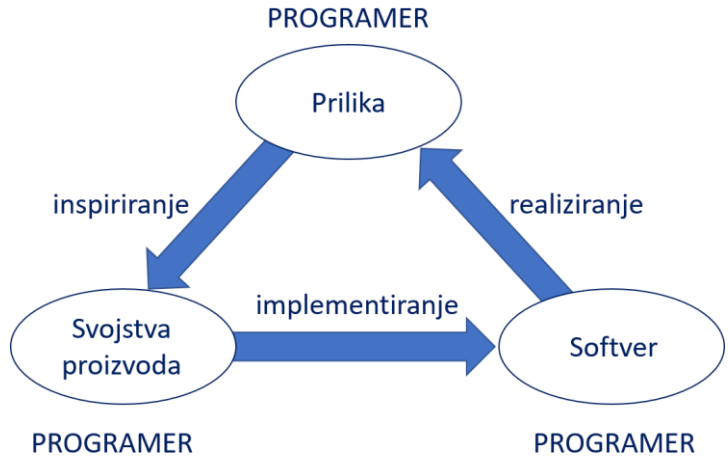
Kod projektno baziranog programskog inženjerstva (slika 2), polazna točka razvoja softvera je skup „softverskih zahtjeva“ u vlasništvu klijenta, koji određuju ono što softverski sustav treba učiniti kako bi podržao svoje poslovne procese:

- Softver razvija softverska tvrtka (izvođač) koja dizajnira i implementira sustav koji pruža funkcionalnost za ispunjavanje zahtjeva.
- Kupac može promijeniti zahtjeve u bilo kojem trenutku kao odgovor na poslovne promjene (to se često događa).
- Izvođač mora promijeniti softver da bi odražavao promjene ovih zahtjeva.
- Prilagođeni softver obično ima dug vijek trajanja (10 godina ili više) i mora ga se podržavati tijekom tog životnog vijeka.



Slika 2: Projektno bazirano programsko inženjerstvo; prema [7]

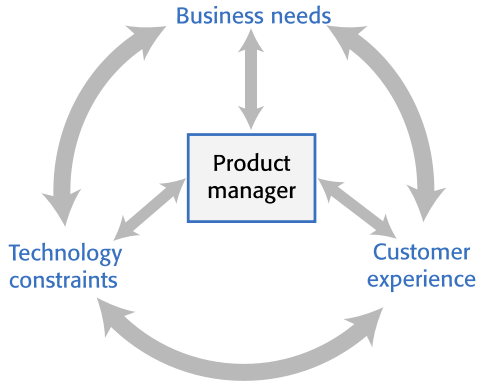
Kod proizvodno baziranog programskog inženjerstva (slika 3) polazna točka razvoja proizvoda je poslovna prilika koju prepoznaju pojedinci ili tvrtka. Oni razvijaju softverski proizvod da iskoriste ovu priliku i prodaju ga kupcima. Tvrtka koja je identificirala priliku, dizajnira i implementira niz softverskih značajki koje trebaju realizirati tu priliku i koje će biti korisne kupcima.



Slika 3: Proizvodno bazirano programsko inženjerstvo; prema [7]

Tvrtka za razvoj softvera odgovorna je za odlučivanje o vremenskom rasponu razvoja, koje značajke treba uključiti i kada se proizvod treba promijeniti. Brza isporuka softverskih proizvoda neophodna je za privlačenje tržišta za tu vrstu proizvoda. Ključne odgovornosti voditelja proizvoda su vlasništvo nad vizijom proizvoda, razvoj mape

proizvoda, stvaranje korisničkih priča, korisničko testiranje i dizajn korisničkog sučelja. Menadžeri proizvoda također rade na „sučelju“ između tvrtke, tima za razvoj softvera i kupaca proizvoda, tj. olakšavaju komunikaciju između ovih skupina.



Slika 4: Voditelj proizvoda (*Product Manager*); preuzeto iz [7]

1.2 Informacijski sustavi (IS)

Informacijski sustavi trebaju osigurati relevantne informacije korisnicima radi donošenja odluka [12]. To je neosporno najvažnija uloga IS-a, odnosno, može se reći da se IS gradi prvenstveno zbog potreba poslovnog sustava.

Dvije dodatne definicije IS-a:

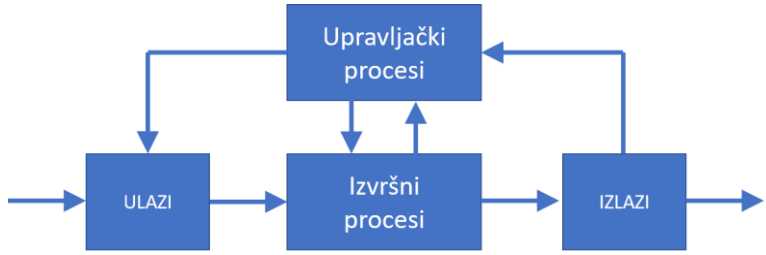
- Formalni, socijalno-tehnički, organizacijski sustavi dizajnirani za skupljanje, obradu, čuvanje i distribuciju informacija [1].
- Skup povezanih dijelova kojima je cilj pribaviti i prenijeti informacije i podatke za funkcioniranje, planiranje, odlučivanje i upravljanje poslovnom organizacijom [12].

Dakle, IS treba opskrbljivati informacijama poslovne procese i operacije, što u igru uvodi i najviši menadžment. Da bi poslovni sustav djelovao optimalno, menadžment mora uspostaviti sustav primanja informacija o funkcioniranju organizacije. Na osnovi tih će informacija onda menadžment upravljati funkcioniranjem cijelog poslovnog sustava.

Bitne značajke informacija:

- točnost, pouzdanost, potpunost
- fleksibilnost, provjerljivost
- relevantnost, jednostavnost, pravodobnost
- dostupnost, sigurnost, količina
- razumljivost (uklapanje u sustav znanja primatelja informacije).

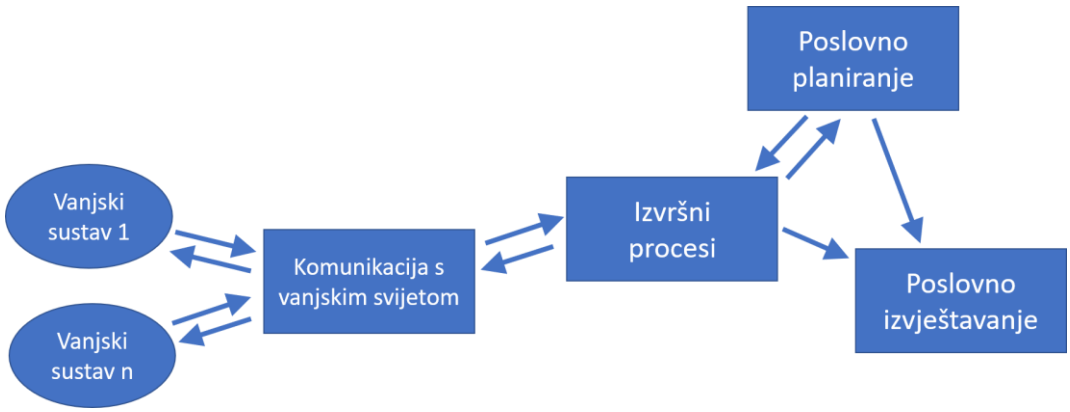
Ako se na ovu situaciju pogleda s procesne strane, vidjet će se da je IS jednako važan i za izvršne i za upravljačke procese. Primjer upravljačkog procesa je planiranje proizvodnje, a primjer izvršnog procesa sama proizvodnja.



Slika 5: Upravljački i izvršni procesi

Komponente poslovnog IS-a:

- obrada transakcija
- poslovno planiranje
- poslovno izvještavanje
- komunikacija s vanjskim sustavima (npr. direktno primanje narudžbi).



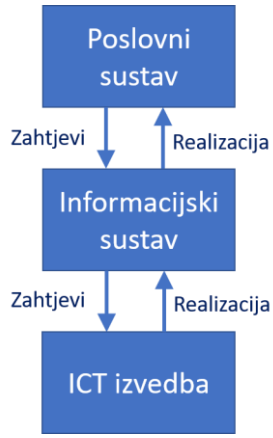
Slika 6: Komponente poslovnog IS-a

IS, dakle, mora pratiti poslovne događaje, mora omogućiti planiranje, mora davati odgovarajuće izvještaje i mora komunicirati s vanjskim svijetom. Da bi to sve uspješno odradio, IS traži određene uvjete:

- ispravni i relevantni matični podaci
- formalna organizacija koja odgovara stvarnom stanju
- korektno posloženi poslovni procesi
- ažurnost poslovnih dokumenata (stari dokumenti su zatvoreni i arhivirani).

Analizom navedenih uvjeta dolazi se do zaključka da je IS pozicioniran točno između poslovnog sustava i ICT strukture (slika 7). Sada se vide osnovni razlozi uvođenja IS-a:

- smanjivanje troškova poslovanja i uočavanje kritičnih točaka
- poboljšanje efikasnosti ubrzavanjem manipulativnih procesa i boljim korištenjem opreme i radne snage
- ostvarivanje potpune proizvodne i materijalne sljedivosti, kako u smjeru od dobavljača do kupca, tako i obratno, od kupca do dobavljača
- optimiziranje zaliha, pomoću povećane točnosti planiranja
- poboljšanje procesa donošenja odluka, zahvaljujući potpunoj transparentnosti procesa i podataka.



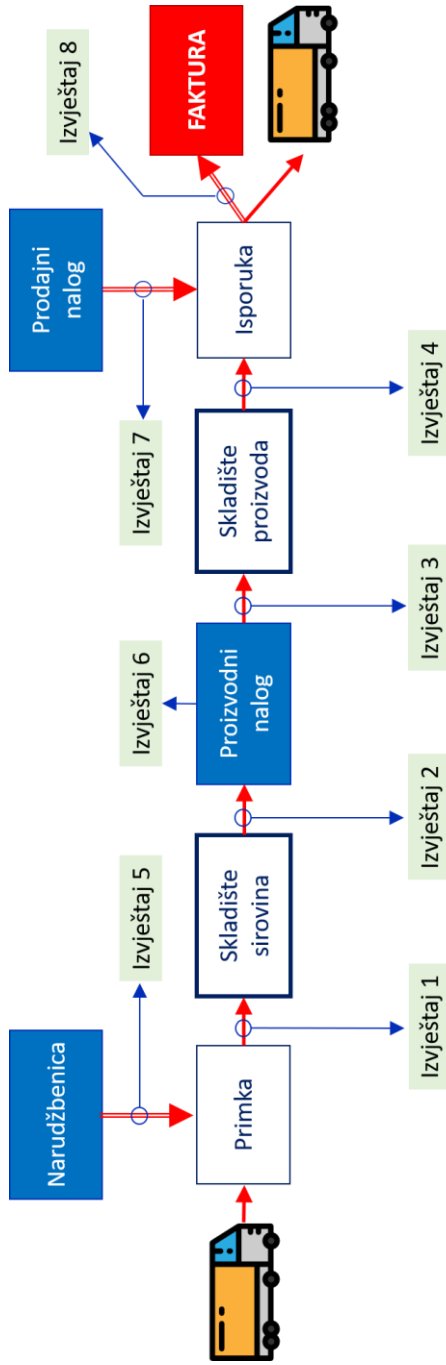
Slika 7: Komponente poslovnog IS-a

Sve ovo nabrojano vodi zaključku da IT odjel nikako ne smije biti startna točka uvođenja IS-a u tvrtku. To postaje pravo i obveza najvišeg menadžmenta koji se mora uključiti u proces budžetiranja i dizajna IS-a, jer se uočava da uspjeh cijele tvrtke sve više ovisi o izboru korektnog softvera [1]. To se odnosi na sve razine poslovnog menadžmenta. Primjer korištenja IS-a za dobivanje osnovnih informacija (izvještaja) o trenutnom stanju i funkcioniranju logističkih funkcija (kretanje materijala duž lanca opskrbe) dan je na slici 8. Jasno se uočava osam osnovnih izvještaja koje bi odgovarajući IS morao davati menadžmentu:

1. Izvještaj o zaprimanju materijala od dobavljača
2. Izvještaj o utrošku materijala u proizvodnji
3. Izvještaj o proizvedenim količinama
4. Izvještaj o isporučenim količinama kupcima
5. Izvještaj o trenutnim statusima svih narudžbenica
6. Izvještaj o trenutnim statusima svih proizvodnih naloga
7. Izvještaj o trenutnim statusima svih prodajnih naloga
8. Izvještaj o izdanim računima i stanju naplate.

Navedeni izvještaji se lako mogu svrstati u tri kategorije:

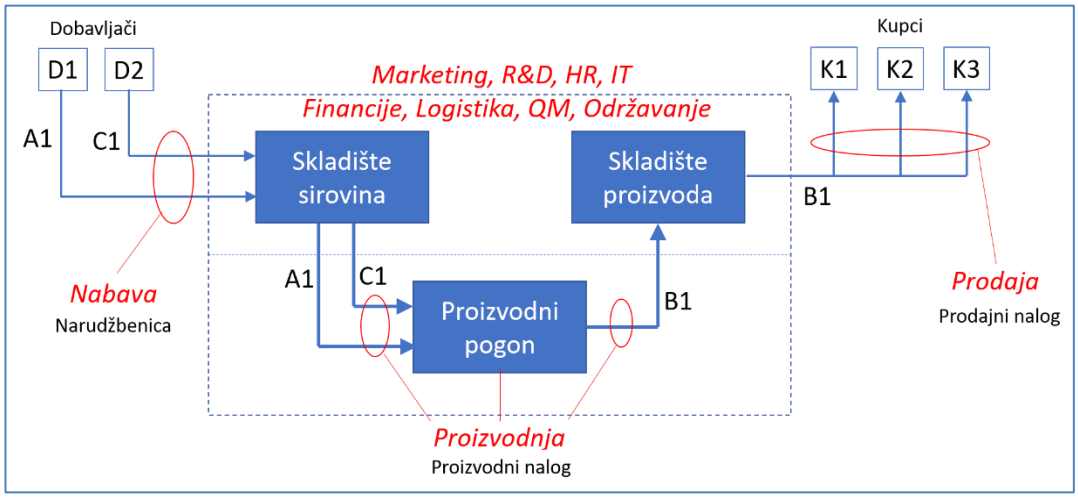
- Izvještaji o materijalnim kretanjima (1, 2, 3, 4)
- Izvještaji o trenutnom statusu osnovnih dokumenata (5, 6, 7)
- Izvještaj o financijskoj situaciji prodaje (8)



Slika 8: Dobivanje izvještaja s raznih točaka u poslovnim procesima

2 Osnovni proizvodni model

Osnovni proizvodni model poslužit će u svrhu dobivanja ideje o tome kako poslovanje funkcionira i kakvi se poslovni zahtjevi redovito postavljaju pred programere.



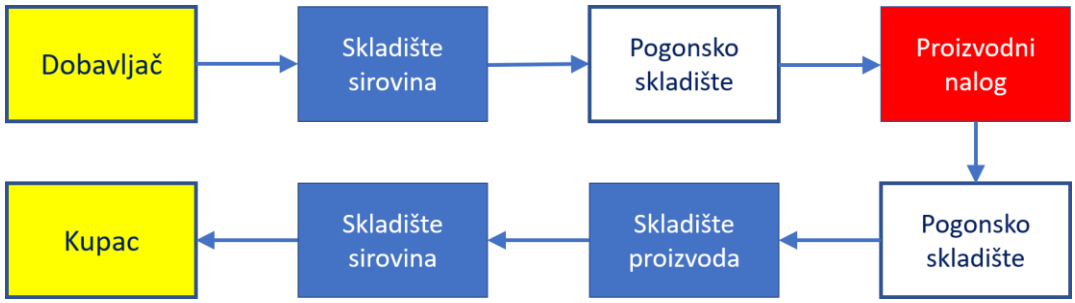
Slika 9: Osnovni proizvodni model

U osnovnom proizvodnom modelu ne uzimaju se u obzir mogući poluproizvodi, nego se iz sirovine i ambalaže „odmah“ dobije gotov proizvod. Iz detaljnog su razmatranja izostavljene još neke proizvodne funkcije. Protok materijala u ovom jednostavnijem slučaju ide prilično glatko i jednostavno, kao što je prikazano na slici 10.

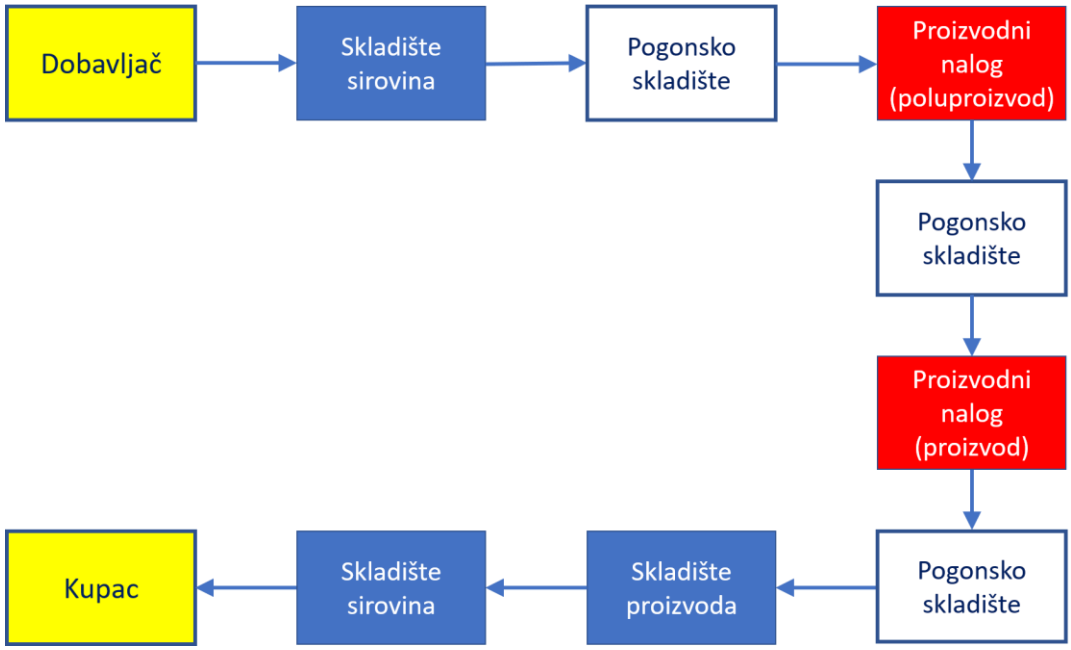
Slika 10: Protok materijala u osnovnom proizvodnom modelu A

U ovaj se jednostavni prikaz može dodati malo kompleksnosti na način da se u proces uključe dva tzv. pogonska skladišta (jedno za privremeni smještaj materijala, koji ulaze u proizvodnju i jedno za privremeni smještaj proizvoda, koji izlaze iz proizvodnje), kao i jedno distribucijsko skladište. Taj se slučaj može nazvati model B i prikazan je na slici 11.

U slučaju dodatnog povećavanja složenosti, proces proizvodnje proširuje se s jednom ili više razina poluproizvoda. Taj se slučaj može nazvati modelom C. Na slici 12 prikazana je proizvodnja s jednom dodatnom razinom.



Slika 11: Protok materijala u osnovnom proizvodnom modelu B



Slika 12: Protok materijala u osnovnom proizvodnom modelu C

Situacija s modelom C puno je češća u tvornicama (jako se rijetko može vidjeti proizvodnju bez poluproizvoda ili sklopova i podsklopova).

Primjer iz proizvodnje tableta, gdje se nakon početne obrade ulaznih sirovina dolazi do četiri razine poluproizvoda:

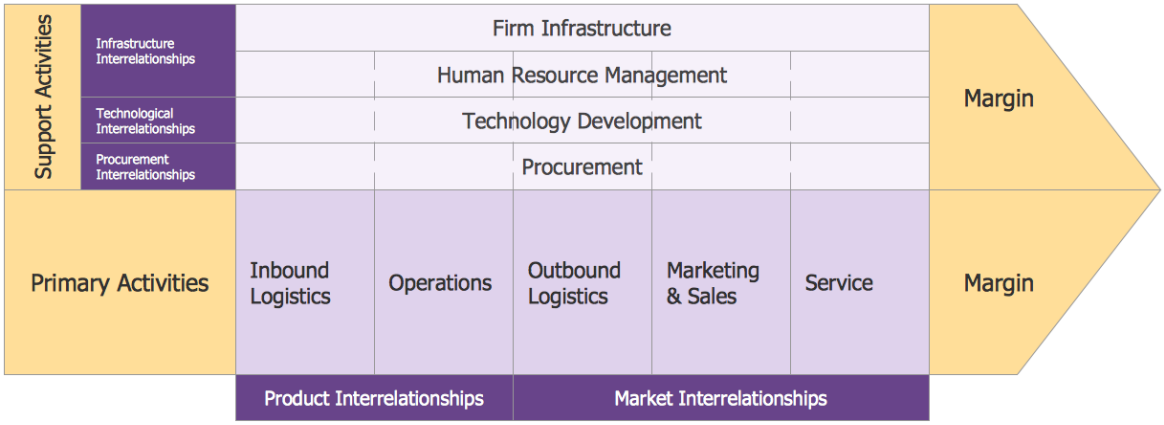
granulat → tablete → obložene tablete → blistirane tablete.

Blistirane se tablete pakiraju i tek onda se dobije gotov proizvod – zapakirane tablete.

2.1 Poslovne funkcije

Primarnim poslovnim funkcijama smatraju se funkcije koje stvaraju vrijednost (Michael Porter, 1985):

- prodaja i marketing
- proizvodnja i usluge
- logistika.



Slika 13: Porterov model lanca vrijednosti, 1985.

Podržavajuće ili sekundarne poslovne funkcije su one koje omogućavaju funkcioniranje primarnih procesa:

- financije, kontroling
- nabava
- održavanje
- razvoj i istraživanje
- ljudski resursi.

Poslovni partneri koji će se razmatrati u osnovnom proizvodnom modelu su kupci i dobavljači.

Glavni poslovni dokumenti koji se koriste u osnovnom proizvodnom modelu:

- ponuda i prodajni nalog (osnovni dokumenti za komunikaciju s kupcima)
- zahtjevnica (dokument za interno naručivanje)
- narudžbenica (osnovni dokument za komunikaciju s dobavljačima)
- proizvodni nalog (osnovni dokument za praćenje proizvodnih troškova)
- radni nalog održavanja (osnovni dokument za praćenje troškova održavanja)
- kalkulacija (dokument za izračun proizvodne cijene proizvoda).

2.2 Preduvjeti za funkcioniranje modela

Da bi bilo koja aplikacija vezana za rad poslovnih funkcija mogla pravilno funkcionirati, u sustavu trebaju biti postavljeni temelji, odnosno matični podaci i organizacijska struktura.

2.2.1 Matični podaci

Za ispravno funkcioniranje osnovnog proizvodnog modela, potrebno je postojanje osam matičnih podataka:

- materijali (proizvodi, sirovine, ambalaža, rezervni dijelovi)
- kupci
- dobavljači
- sastavnice (popis materijala / BOM)
- radni centri
- postupnici (receptura / plan operacija / routing)
- mjerne jedinice
- zaposlenici
- radne aktivnosti.

Matični slog materijala sadrži sve podatke o materijalima koji se vode u skladišnom poslovanju, tj. materijale koji se nabavljaju, proizvode, skladište i prodaju. Potpuno definirani matični slog materijala sadrži parametre za planiranje, skladišnu i financijsku evidenciju, izvještavanje, nabavu, prodaju itd.

Kupac je jedan od dva ključna partnera u poslovanju. U matičnim podacima kupaca definirani su svi podaci potrebni za prodaju i naplatu prodane robe i izvršenih usluga.

Dobavljač (drugi ključni poslovni partner) poduzeću isporučuje robu ili usluge. To je proizvođač ili neki drugi poduzetnik od kojeg se nabavlja materijal za našu proizvodnju. Matični slog dobavljača održava se u svrhu upravljanja materijalima i u svrhu financijskih transakcija (fakture, predujmovi, plaćanja i sl.).

Sastavnice sadrže normativni popis materijala potrebnih za proizvodnju proizvoda. Sastoje se od zaglavlja i popisa stavaka koji čine potrebne količine određenih vrsta materijala nužnih za proizvodnju. Sastavnica je osnova za nabavu materijala od dobavljača, za planiranje proizvodnje i za izračun cijene proizvoda.

Radni centri se koriste za izvršenje operacija prilikom proizvodnje. Svaki radni centar ima svoj kapacitet strojeva, radne snage, energije i ostalih potrebnih proizvodnih parametara. Za njih se mogu vezati formule za proračun vremena obrade, troškova i kapaciteta. Svaki radni centar obavezno je dodijeljen odgovornom mjestu troška, preko kojega se određuju nazivne cijene rada strojeva i ljudi, proizvodnih energenata i ostalih parametara koji se trebaju uključiti u izračun proizvodne cijene.

Postupnici sadrže tehnološke podatke, koji definiraju proces izrade nekog materijala, počevši od popisa potrebnih operacija i radnih centara, koji se koriste u pojedinoj

operaciji, pa do detalja svake operacije (osnovna proizvodna količina, kao i vrijeme pripreme, izrade i raspremanja stroja za svaku operaciju).

Mjerne jedinice su po definiciji odabrane, dogovorene i objavljene poznate vrijednosti mjernih (fizikalnih) veličina, s kojima se pri mjerenju uspoređuju sve druge istovrsne veličine. U našem osnovnom proizvodnom modelu koristit će se standardne jedinice po SI sustavu.

Zaposlenici su, u slučaju našeg modela, radnici u proizvodnji.

Radne aktivnosti su strukturirane proizvodne aktivnosti, raspodijeljene po vrijednosti. Svaka radna aktivnost ima definiranu financijsku vrijednost u odnosu na vrijeme, tzv. satnicu (npr. 80 kn za 1 h).

2.2.2 Organizacijska struktura

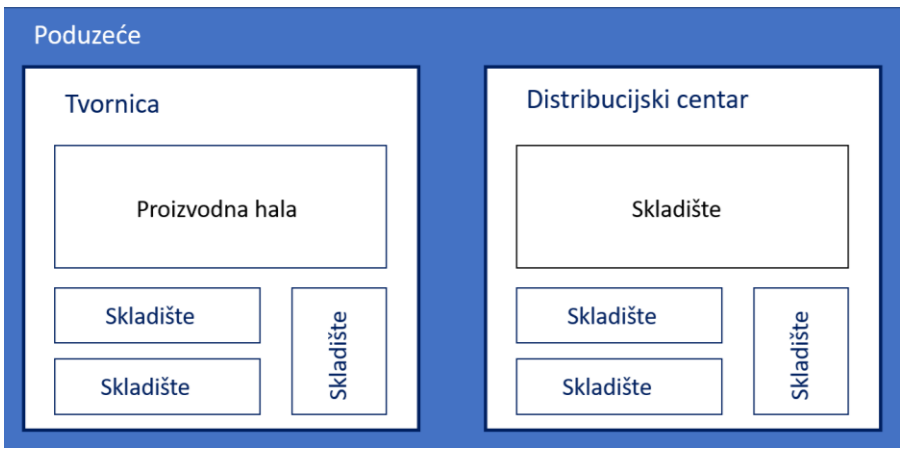
Za ispravno funkcioniranje osnovnog proizvodnog modela, potrebno je postojanje tri organizacijska podatka:

- poduzeće
- pogon (tvornica)
- skladište.

Poduzeće je pravno nezavisna poslovna jedinica. To je centralna organizacijska jedinica u financijskom računovodstvu.

Pogon je osnovna jedinica logističkog planiranja unutar koje se jedinstveno planira materijal i proizvodnja, a u isto vrijeme predstavlja područje vrednovanja materijala (cijena materijala određuje se na razini pogona). Pogon može biti tvornica ili (udaljeni) distribucijski centar (DC).

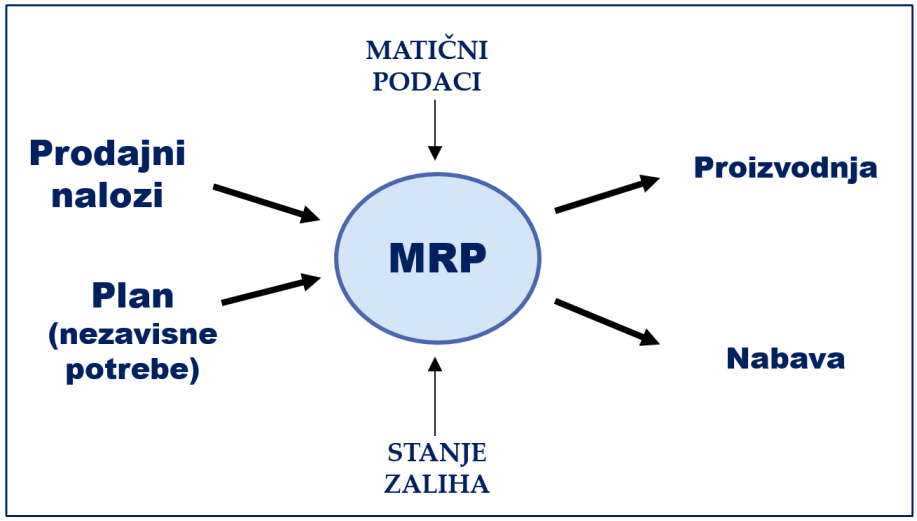
Skladište predstavlja mjesto fizičkog skladištenja materijala. Pogonu može biti pridruženo jedno ili više skladišta. Skladišta mogu biti realna i fiktivna (tzv. pogonska skladišta).



Slika 14: Primjer fzičke organizacije poduzeća s jednom tvornicom i jednim DC-om

2.3 Planiranje

Služba planiranja obrađuje plan prodaje i prodajne naloge te, na osnovi toga, kreira planske podatke za službu nabave i za službu proizvodnje.



Slika 15: Osnovna ideja planiranja

Kao što je vidljivo na slici 15, ulazni podaci za proces planiranja su:

- plan prodaje (proizvodnja za nepoznatog kupca; nezavisne potrebe)
- prodajni nalozi (proizvodnja za poznatog kupca).

Izlazni podaci iz procesa planiranja su:

- potreba za proizvodnjom (planski nalozi za proizvodnju)
- potrebe za nabavom (zahtjevnice za nabavu).

U slučaju proizvodnje za nepoznatog kupca (tj. za tržište), proizvodnja se mora isplanirati unaprijed – ne bi se smjelo dogoditi da kupac zatraži određenu količinu proizvoda, a da tog proizvoda nema na skladištu.

Podaci koji se unose u plan prodaje:

- proizvod koji se želi prodati
- količina proizvoda koja se želi prodati
- period na koji se odnosi plan, tj. kada se navedena količina planira prodati.

Da bi se od plana prodaje i prodajnih naloga došlo do potrebnih rezultata u vidu plana nabave i plana proizvodnje, potrebno je koristiti određene matične podatke i podatke sustava:

- trenutno stanje zalihe proizvoda i potrebnih materijala
- sastavnice proizvoda

- aktivni proizvodni nalozi
- aktivne narudžbenice.

Podaci koji trebaju biti dio plana proizvodnje:

- proizvod koji se treba proizvesti
- količina proizvoda koju je potrebno proizvesti
- period na koji se odnosi količina za proizvodnju.

Podaci koji trebaju biti dio plana nabave:

- potrebni materijal koji se mora kupiti
- količina materijala koja se mora kupiti
- period na koji se odnosi potreba.

2.4 Prodaja

Osnovni je zadatak prodaje dogovoriti prodaju i realizirati naplatu isporučene robe kupcima.

2.4.1 Kreiranje ponude

Prvi korak u procesu prodaje je kreiranje neobvezujuće ponude kupcima. S kupcima se dogovaraju osnovni elementi buduće prodaje – što nudimo kupcu, u kojoj količini, do kada i po kojoj cijeni:

- tko (kojem kupcu nudimo naše proizvode; poveznica na matični slog kupca)
- što (koji proizvod; poveznica na matični slog materijala)
- koliko (koja količina proizvoda se nudi kupcu; poveznica na mjerne jedinice)
- kada (za kada proizvod treba biti raspoloživ za kupca; poveznica na kalendar)
- cijena (jedinična cijena svakog proizvoda iz ponude)
- iznos (ukupni iznos za svaki proizvod i ukupni iznos cijele ponude; jednostavna matematika).

2.4.2 Kreiranje prodajnog naloga

Nakon što je kupac odobrio ponudu, u sustavu se kreira prodajni nalog kao obvezujući dokument za obje strane. Prodajni nalog može se kreirati s referencom na ponudu, ali može i bez ponude:

- referentni dokument / ponuda (neobvezni podatak)
- tko (kojem kupcu nudimo naše proizvode; poveznica na matični slog kupca)
- što (koji proizvod; poveznica na matični slog materijala)
- koliko (koja količina proizvoda se nudi kupcu; poveznica na mjerne jedinice)
- kada (za kada proizvod treba biti raspoloživ za kupca; poveznica na kalendar)

- cijena (jedinična cijena svakog proizvoda iz ponude)
- iznos (ukupni iznos za svaki proizvod i ukupni iznos cijele ponude; jednostavna matematika).

2.4.3 Kreiranje fakture kupcima za isporučene proizvode

Nakon što je proizvodnja proizvela proizvod i nakon što je logistika isporučila proizvod kupcu, potrebno je kreirati račun za isporučenu količinu proizvoda.

Osnovni elementi računa za kupca su:

- referentni dokument (prodajni nalog)
- tko (kojem kupcu izdajemo račun za isporučene proizvode; poveznica na matični slog kupca)
- što (koji proizvod se naplaćuje; poveznica na matični slog materijala)
- koliko (koja količina proizvoda se naplaćuje kupcu; poveznica na mjerne jedinice)
- kada (kada je proizvod isporučen kupcu; poveznica na kalendar).
- cijena (jedinična cijena svakog proizvoda)
- iznos (ukupni iznos za svaki proizvod i ukupni iznos cijele ponude; jednostavna matematika)
- korisnik (ime osobe iz službe prodaje koja kreira račun; poveznica na popis zaposlenika).

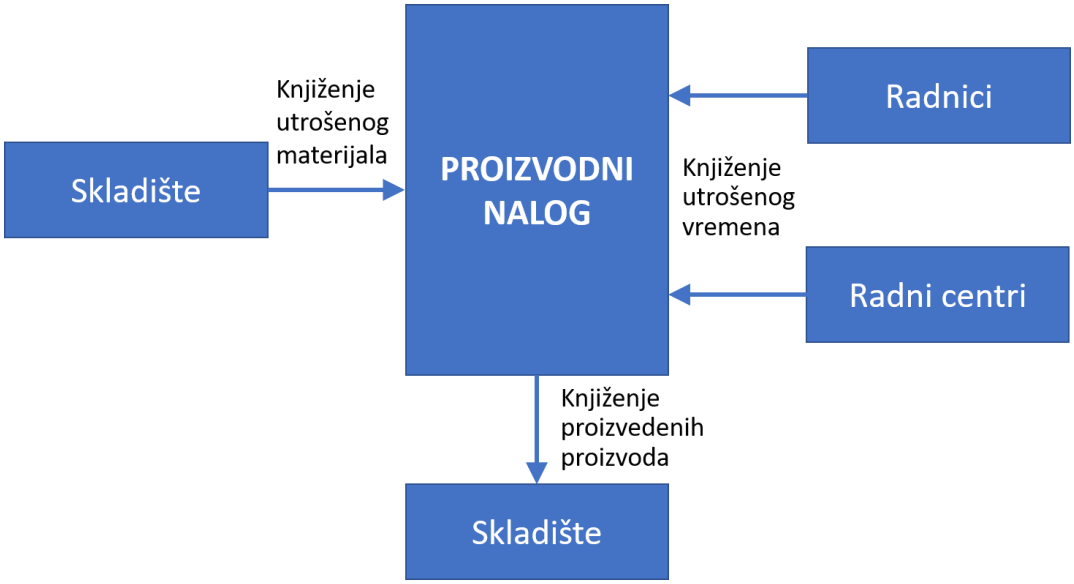
2.5 Proizvodnja

Osnovni zadatak službe proizvodnje je proizvesti svu količinu proizvoda koja je zahtijevana od strane službe prodaje, u skladu s traženim terminima.

Proces proizvodnje može se podijeliti na pripremu proizvodnje i na praćenje izvršenja proizvodnje. U praćenje izvršenja proizvodnje spadaju knjiženje utrošenog materijala, knjiženje utrošenog vremena i proizvodne režije, kao i prijava proizvedene količine traženog proizvoda.

Kao osnovni dokument koristi se proizvodni nalog (slika 16). On prikuplja sve količinske i vrijednosne informacije o proizvodnji:

- koliko je materijala utrošeno u proizvodnji i po kojoj cijeni
- koliko je vremena utrošeno za proizvodnju, na kojim radnim centrima i s kojim zaposlenicima
- koliko je proizvoda proizvedeno po predmetnom proizvodnom nalogu.



Slika 16: Materijalna i vremenska knjiženja na proizvodni nalog

2.5.1 Priprema proizvodnje

U sklopu pripreme proizvodnje, odgovorne osobe kreiraju proizvodne naloge koji trebaju sadržavati sve informacije potrebne za samu proizvodnju:

- što (koji se proizvod treba proizvesti; poveznica na matični slog materijala)
- koliko (koja se količina proizvoda treba proizvesti; poveznica na mjerne jedinice)
- kada (u kojem roku tražena količina treba biti raspoloživa za isporuku kupcu; poveznica na kalendar)
- tko (za kojeg kupca se veže tražena količina; poveznica na matični slog kupca)
- koji su materijali potrebni za proizvodnju (poveznica na matičnu sastavnicu; proporcionalni izračun potrebnih količina svih materijala)
- na kojim strojevima se proizvodnja treba odraditi (poveznica na matični postupnik; proporcionalni izračun potrebnog vremena rada strojeva)
- koja radna snaga je potrebna za proizvodnju (poveznica na matični postupnik; proporcionalni izračun potrebnog vremena rada radnika).

Nakon kreiranja proizvodnih naloga, slijede dodatne aktivnosti u sklopu pripreme proizvodnje: provjeravanje raspoloživosti materijala i provjeravanje raspoloživosti proizvodnih kapaciteta (strojevi i ljudi).

Ako je sve u redu, proizvodni nalog može se otpustiti, tj. dati znak svima u lancu da proizvodnja po tom dokumentu može početi. Ako radnici nemaju prikaz svih potrebnih

detalja i uputa na svojim ekranima pokraj stroja, ispisuju se potrebni dijelovi proizvodnog naloga i odnose radnicima.

Slika 17: Priprema proizvodnje

2.5.2 Knjiženje potrošnje sirovine i ambalaže na proizvodni nalog

Da bi se proizvodnja traženog proizvoda mogla obaviti, potrebno je utrošiti određene sirovine (materijal). Evidencija o tome vodi se na način da se utrošena količina „knjiži“ iz skladišta na proizvodni nalog. Trenutkom knjiženja materijal se smatra potrošenim i više nije „vidljiv“ na stanju u skladištu.

Slika 18: Tok materijala: iz skladišta na proizvodni nalog

Samo knjiženje može se obaviti na tri osnovna načina:

- količinsko knjiženje (podaci se u sustav upisuju „ručno“ i sva odgovornost je na osobi koja administrira knjiženje, nema nikakve sljedivosti materijala)
- skeniranje univerzalne etikete na pakiranju (podaci se u sustav unose automatski, putem skeniranja; smanjena mogućnost greške, ali i dalje nema nikakve sljedivosti)

- skeniranje jedinstvene etikete na pakiranju utrošenog materijala (podaci se u sustav unose automatski, putem skeniranja; mogućnost greške je svedena na minimum, moguće je ostvariti sljedivost).

Podaci koji se moraju prijaviti u slučaju količinskog knjiženja:

- broj proizvodnog naloga
- datum knjiženja
- šifra utrošenog materijala
- količina utrošenog materijala
- skladište iz kojeg je utrošen materijal.

Podaci koji se moraju prijaviti u slučaju skeniranja jedinstvene etikete:

- broj proizvodnog naloga
- datum knjiženja (sustav ga određuje automatski)
- jedinstvena oznaka etikete (bar-kod) s dodatnim podacima: šifra utrošenog materijala, količina utrošenog materijala, skladište iz kojeg je utrošen materijal.

2.5.3 Knjiženje utrošenog vremena na proizvodni nalog

Da bi se proizvodnja traženog proizvoda mogla obaviti, potrebno je utrošiti određeno vrijeme. Evidencija o tome vodi se na način da se utrošeno vrijeme „knjiži“ na proizvodni nalog.

Samo knjiženje može se obaviti na dva osnovna načina:

- količinsko knjiženje (podaci se u sustav upisuju „ručno“ i sva odgovornost je na osobi koja administrira knjiženje)
- skeniranje osobne kartice ili davanje otiska prsta (sustav utvrđuje vrijeme početka i završetka aktivnosti te automatski izračunava cjelokupno trajanje).

Podaci koji se moraju prijaviti u slučaju količinskog knjiženja:

- broj proizvodnog naloga
- radni centar na kojem je izvršena proizvodnja (poveznica na matični slog radnih centara)
- vrijeme koje je stroj utrošio za predmetni proizvodni nalog
- šifra radnika koji je sudjelovao u proizvodnji (poveznica na matični slog zaposlenika)
- vrijeme koje je radnik utrošio za predmetni proizvodni nalog
- radna aktivnost (poveznica na matični slog radnih aktivnosti).

Podaci koji se moraju prijaviti u slučaju knjiženja skeniranjem:

- broj proizvodnog naloga
- radni centar na kojem je izvršena proizvodnja
- šifra radnika koji je sudjelovao u proizvodnji (poveznica na matični slog zaposlenika)

- početak rada na predmetnom proizvodnom nalogu
- završetak rada na predmetnom proizvodnom nalogu.

2.5.4 Prijava proizvodnje

Nakon što se proizvede određena količina proizvoda, u sustav je potrebno upisati što se sve proizvelo po određenom proizvodnom nalogu, u određenom vremenu.

Slika 19: Tok materijala: iz proizvodnog naloga u skladište

Samo knjiženje može se obaviti na dva osnovna načina:

- količinsko knjiženje (podaci se u sustav upisuju „ručno“ i sva odgovornost je na osobi koja administrira knjiženje, nema nikakve sljedivosti proizvoda)
- označavanje proizvoda jedinstvenom etiketom i skeniranje jedinstvene etikete na pakiranju gotovog proizvoda (podaci se u sustav unose automatski putem skeniranja etikete; mogućnost greške svedena je na minimum, moguće je ostvariti sljedivost proizvoda).

Podaci koji se moraju prijaviti u slučaju količinskog knjiženja:

- broj proizvodnog naloga
- datum knjiženja
- šifra proizvoda
- proizvedena količina
- skladište u koje se predaje proizvod.

Podaci koji se moraju prijaviti u slučaju skeniranja jedinstvene etikete:

- broj proizvodnog naloga
- datum knjiženja (sustav ga određuje automatski)
- jedinstvena oznaka etikete (bar-kod) s dodatnim podacima: šifra proizvoda, količina pakiranja, skladište u koje se predaje proizvod.

2.6 Nabava

Osnovni zadatak nabave je naručivanje potrebne količine svih materijala i usluga potrebnih za neometanu proizvodnju i rad svih ostalih službi.

2.6.1 Interno naručivanje

Prvi korak u procesu naručivanja je tzv. interno naručivanje, tj. kreiranje zahtjevnica od strane svih ostalih službi prema službi nabave. Poslovni korisnici putem zahtjevnica javljaju nabavi što im treba, u kojim količinama i u kojem vremenskom roku:

- što (koji materijal ili usluga; poveznica na matični slog materijala)
- koliko (koja količina materijala ili usluge je potrebna; poveznica na mjerne jedinice)
- kada (za kada je potreban materijal ili usluga; poveznica na kalendar).

2.6.2 Eksterno naručivanje

Drugi korak u procesu naručivanja je tzv. eksterno naručivanje, tj. kreiranje narudžbenica od strane službe nabave prema dobavljačima. Služba nabave obrađuje prethodno kreirane zahtjevnice i kreira narudžbenicu, kao osnovni dokument za kupovinu i plaćanje:

- tko (od kojeg dobavljača se naručuju materijali ili usluge; poveznica na matični slog dobavljača)
- što (koji materijal ili usluga; poveznica na matični slog materijala)
- koliko (koja količina materijala ili usluge je potrebna; poveznica na mjerne jedinice)
- cijena (jedinična cijena za svaku stavku narudžbenice, dogovorena s dobavljačem)
- iznos (ukupni iznos svake stavke i ukupni iznos za cijelu narudžbenicu; jednostavna matematika)
- kada (za kada je potreban materijal ili usluga; poveznica na kalendar)
- gdje (gdje se naručena roba treba dostaviti; poveznica na skladište kao organizacijski podatak)
- korisnik (ime osobe iz službe nabave koja kreira narudžbenicu; poveznica na popis zaposlenika).

2.7 Logistika

Služba logistike bavi se materijalnim kretanjima:

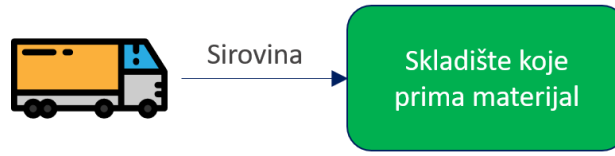
- zaprimanje materijala od dobavljača
- interno preskladištenje materijala
- izdavanje materijala u proizvodnju
- zaprimanje proizvoda nakon proizvodnje
- interno preskladištenje proizvoda
- isporuka proizvoda kupcu.

Ovih šest vrsta materijalnih kretanja grupira se u tri logistička odjela:

- ulazna logistika
- interna logistika
- izlazna logistika.

2.7.1 Ulazna logistika

Odjel ulazne logistike obavlja zaprimanje svih materijala od dobavljača. Cilj je zaprimiti materijal u formi u kojoj je zapakiran.



Slika 20a: Ulazna logistika: zaprimanje materijala od dobavljača

Načelno, zaprimanje sirovine i ambalaže od dobavljača moguće je obaviti na tri načina:

- količinsko (ručno) zaprimanje
- zaprimanje skeniranjem univerzalne etikete
- zaprimanje skeniranjem jedinstvene etikete (sljedivost).

Ako tehnički uvjeti to omogućuju, zaprimanje se treba obaviti jednostavnim skeniranjem etikete na paleti / kontejneru / bačvi / kutiji i sl. Ako skeniranje nije omogućeno, zaprimanje se obavlja „ručno“, što je redovno puno složeniji proces i traži neusporedivo više vremena i vještine od osobe koja radi zaprimanje.

U realnom poslovanju, osim tehničkih preduvjeta postoji i jedan poslovni, a to je da sa svim dobavljačima treba dogovoriti proces dostavljanja materijala uz prikladnu etiketu, koja sadrži sve potrebne informacije u pogodnom obliku za skeniranje. Tu se uočava drugi mogući problem: Što ako dobavljači nisu u stanju dostaviti robu s odgovarajućim etiketama? U tom je slučaju potrebno omogućiti ispis željene etikete na mjestu zaprimanja robe. Nakon ispisa etikete i njezinog priljepljivanja na pakovinu, može se obaviti

skeniranje u svrhu zaprimanja materijala, čime je ujedno omogućena detaljna evidencija takve pakovine u daljnjim koracima proizvodnog procesa.

2.7.2 Interna logistika

Odjel interne logistike radi sva preskladištenja materijala i proizvoda, tj. selidbu (iz bilo kojeg razloga) materijala ili proizvoda iz jednog skladišta u neko drugo:

- iz jednog u drugo logističko skladište
- iz logističkog u pogonsko skladište (sirovina i ambalaža)
- iz pogonskog u logističko skladište (gotovi proizvod).

Slika 20b: Interna logistika: međuskladištenje materijala

Preskladištenje materijala također se može raditi ručno (zapisivanje količine) ili pomoću skeniranja. U slučaju internog preskladištenja iz vanjskih u proizvodna skladišta, priprema materijala za premještaj obavlja se na osnovi definiranih potreba iz proizvodnje. Te se potrebe obično formiraju u obliku rezervacije za preskladištenjem materijala.

2.7.3 Izlazna logistika

Odjel izlazne logistike priprema pakirane proizvode za isporuku kupcima.

Slika 20c: Izlazna logistika: isporuka materijala (proizvoda) kupcu

Izdavanje materijala iz skladišta gotovih proizvoda za potrebe procesa distribucije također se najlakše obavlja pomoću skeniranja. Sve pakovine koje se šalju kupcima se pojedinačno skeniraju.

U slučaju višerazinskih pakiranja (kutija → karton → paleta), ako su sve pakirne jedinice označene jedinstvenom etiketom tijekom proizvodnje, sada se jednim skeniranjem paletne etikete omogućava praćenje kretanja svih kutija koje su zapakirane u skeniranu paletu.

2.8 Kontroling

Kontroling se bavi internim praćenjem proizvodnih troškova, od izračunavanja proizvodne cijene do analize stvarnih proizvodnih troškova i izračuna odstupanja od normativa.

2.8.1 Izračun proizvodne cijene

Proizvodna cijena izračunava se na osnovi matičnih podataka. Prvo se „pozivaju“ sastavnica s popisom potrebnih materijala i postupnik s popisom potrebnih aktivnosti, a onda se za svaki materijal iz sastavnice i za svaku radnu aktivnost iz postupnika uzima jedinična cijena. Nakon toga se iz sastavnice uzima podatak o potrebnim količinama materijala, a iz postupnika podatak o potrebnim količinama radnih aktivnosti (jednostavna matematika). Na sve to se još dodaje planski trošak proizvodne režije, obično u postocima, u odnosu na planski trošak materijala ili planski trošak aktivnosti. Rezultat izračuna jedinične proizvodne cijene je tzv. kalkulacija.

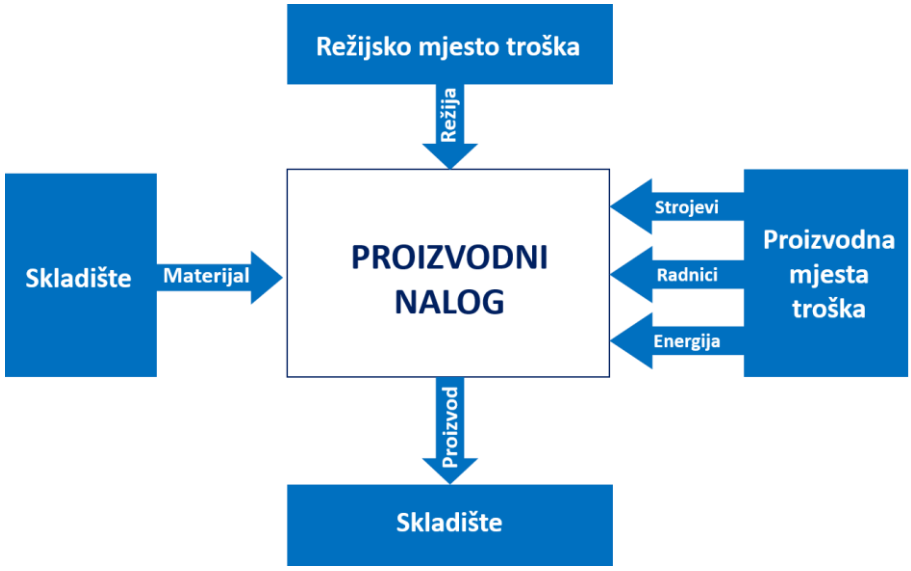
Slika 21: Princip kreiranja proizvodne cijene

2.8.2 Kontrola stvarnih proizvodnih troškova

Stvarni proizvodni troškovi izračunavaju se na osnovi provedenih knjiženja utroška materijala i utroška vremena.

Iznos troškova materijala dobiva se jednostavnim množenjem količine utrošenog materijala s jediničnom cijenom materijala, a iznos troškova proizvodnih aktivnosti dobiva se jednostavnim množenjem količine utrošenog vremena s jediničnom cijenom konkretne radne aktivnosti.

Na sve to se još dodaje stvarni trošak proizvodne režije, na isti način kao kod izračuna proizvodne cijene – u postocima u odnosu na stvarni trošak materijala ili stvarni trošak aktivnosti. Evidencija stvarnih troškova radi se s referencom na proizvodni nalog.



Slika 22: Princip praćenja proizvodnih troškova

2.8.3 Izračun proizvodnih odstupanja

Proizvodna su odstupanja bitan indikator kvalitete same proizvodnje. Uspoređuju se planski troškovi (troškovi prema kalkulaciji) sa stvarnim troškovima (knjižena vrijednost materijala i radnih aktivnosti, zapisano s referencom na proizvodni nalog). Najčešće se javljaju tri vrste odstupanja:

- odstupanje ulazne količine (knjižila se različita količina od predviđene)
- odstupanje ulazne cijene (knjiženi materijal je u trenutku knjiženja imao drukčiju cijenu od one koja je korištena u izračunu proizvodne cijene)
- odstupanje korištenih materijala (umjesto materijala koji je predviđen sastavnicom, knjižen je neki drugi materijal).

Slika 23: Princip izračunavanja proizvodnih odstupanja

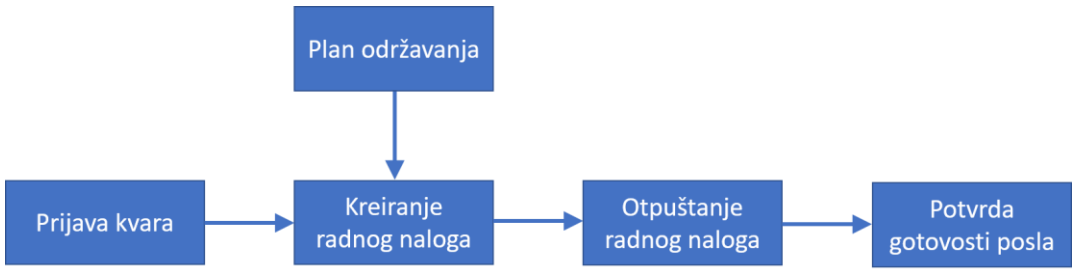
2.9 Održavanje pogona

Služba održavanja brine se da svi strojevi (radni centri) budu ispravni i raspoloživi za proizvodnju.

2.9.1 Preventivno održavanje

Ključni dio procesa održavanja je Preventivno održavanje. Planeri održavanja definiraju periodične preglede i remontne aktivnosti. Osnovni dokument koji se koristi je radni nalog preventivnog održavanja, koji sadrži sljedeće elemente:

- što (koji se uređaj treba pregledati; poveznica na matični slog radnih centara)
- koja aktivnost se treba načiniti (poveznica na matični slog radnih aktivnosti)
- kada (u kojem roku planirana aktivnost treba biti izvedena; poveznica na kalendar)
- tko (koji radnici trebaju izvršiti planiranu aktivnost; poveznica na popis zaposlenika)
- koji su materijali / rezervni dijelovi potrebni za planiranu aktivnost (poveznica na matični slog materijala – rezervnih dijelova).



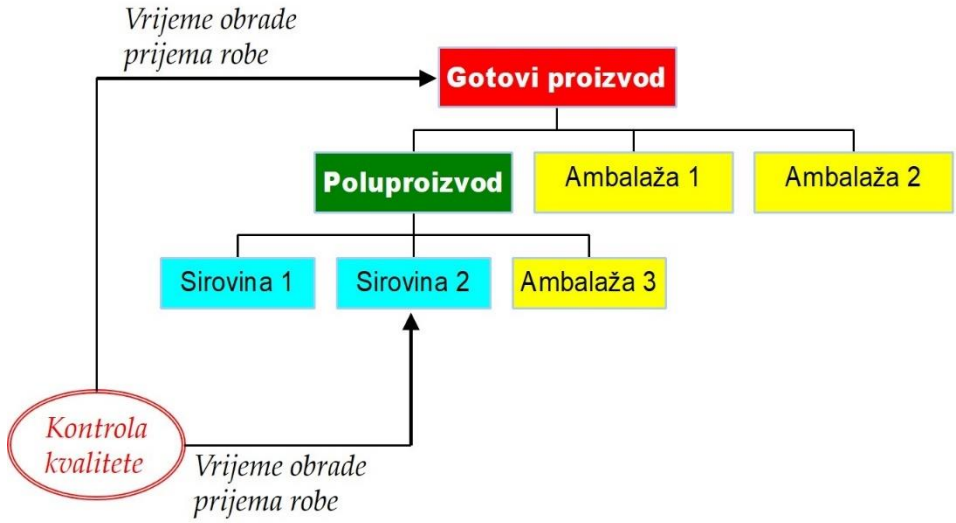
Slika 24: Princip održavanja

2.9.2 Korektivno održavanje

Korektivno održavanje odnosi se na popravak kvarova. Popravak se radi najčešće odmah nakon prijave problema, a sva evidencija vodi se pomoću radnog naloga korektivnog održavanja, koji sadrži iste elemente kao i radni nalog za preventivno održavanje.

2.10 Osiguranje kvalitete

Služba osiguranja kvalitete u osnovnom proizvodnom modelu ima zadatak provjeravanja kvalitete i ispravnosti ulaznih sirovina, kao i kvalitete i ispravnosti proizvedenih proizvoda. Ne smije se dogoditi da loša sirovina uđe u proces obrade, a također ni da loš proizvod bude poslan kupcu.



Slika 25: Princip praćenja kvalitete

Slika 25 prikazuje da se kontrola kvalitete ne mora raditi za sve materijale koji sudjeluju u proizvodnom procesu. Često se za detaljne provjere izabere nekoliko ključnih materijala i onda se vrijeme potrebno za njihovu provjeru treba uključiti u planiranje proizvodnje.

2.10.1 Kontrola kvalitete ulaznih materijala

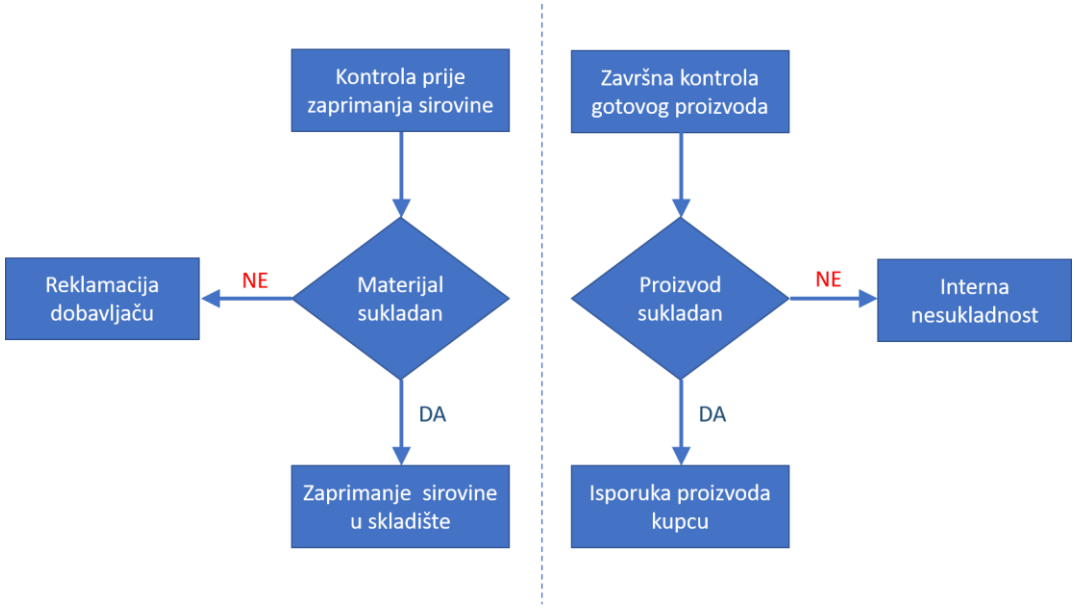
Kontrola kvalitete ulaznih sirovina radi se provjerom definiranih parametara i njihovom usporedbom s traženim vrijednostima. Sva mjerenja zapisuju se u kontrolni evidencijski list. Evidencija sadržava sljedeće komponente:

- što (koji materijal se provjerava; poveznica na matični slog materijala)
- parametri provjere
- tražene vrijednosti svih parametara
- izmjerene (ustanovljene) vrijednosti
- zaključak / odluka o upotrebi.

2.10.2 Kontrola proizvoda prije isporuke

Kontrola kvalitete gotovih proizvoda radi se provjerom definiranih parametara i njihovom usporedbom s traženim vrijednostima:

- što (koji proizvod se provjerava; poveznica na matični slog materijala)
- parametri provjere
- tražene vrijednosti svih parametara
- izmjerene (ustanovljene) vrijednosti
- zaključak / odluka o upotrebi.

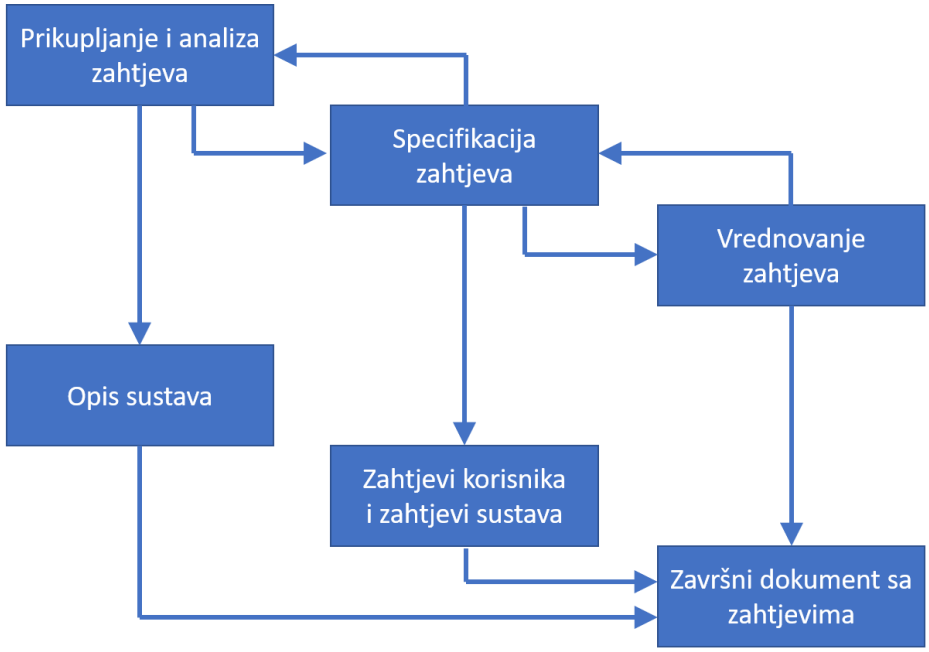


Slika 26: Praćenje kvalitete na ulazu i izlazu iz proizvodnog pocesa

3 Inženjerstvo zahtjeva

Prikupljanje zahtjeva jedna je od ključnih faza projekta izrade aplikacije. Ako su zahtjevi pogrešno definirani, aplikacija će vjerojatno neki zadatak dobro odraditi, ali to neće biti zadatak koji ta aplikacija u stvari treba odraditi.

Inženjerstvo zahtjeva je proces izrade specifikacije programskog produkta. To je prva generička aktivnost tijekom svakog procesa programskog inženjerstva. Zahtjevi su svojstva koja aplikacija mora / treba imati. Količina i detaljnost zahtjeva ovisi o razini formalnosti koju kupac traži. Tipično, što je veći projekt, zahtjevi su brojniji i detaljniji.



Slika 27: Inženjerstvo zahtjeva [6]

S obzirom na razinu detalja, zahtjeve dijelimo na tri vrste: korisnički zahtjevi, zahtjevi sustava i specifikacija softvera.

Korisnički zahtjevi su specifikacija visoke razine apstrakcije (obično u okviru ponude za izradu softverskog proizvoda). Pišu se u prirodnom jeziku i grafičkim dijagramima. Moraju biti razumljivi netehničkom osoblju.

Zahtjevi sustava vrlo su detaljna specifikacija (uobičajeno nakon prihvaćanja ponude, a prije sklapanja ugovora). Pišu se strukturiranim prirodnim jezikom, posebnim jezicima za oblikovanje sustava, dijagramima i matematičkom notacijom.

Specifikacija softvera najdetaljniji je opis i objedinjuje korisničke zahtjeve i zahtjeve sustava.

3.1 Uvodni pojmovi

Karakteristike dobro definiranih zahtjeva su:

- jasnoća
- konciznost
- laka razumljivost
- razumnost
- provjerljivost
- međusobna konzistentnost.

3.1.1 Prioritizacija zahtjeva

Nakon što se zahtjevi prikupe, moraju se prioritizirati. Prioritizacija se uglavnom radi u četiri stupnja:

- MORA (svojstva koja obavezno moraju biti uključena u aplikaciju)
- TREBA (važna svojstva koja treba uključiti ako je ikako moguće)
- MOŽE (poželjna svojstva koja se mogu izbjeći u slučaju da se ne uklapaju u budžet i vremenski raspored)
- NE TREBA (opcionalna svojstva za koja se kupac složio da neće biti ugrađena u trenutnoj verziji).

Nakon što se definira potpuna lista zahtjeva, potrebno je pažljivo pregledati svaki zahtjev i zapitati se: „Može li se ovaj zahtjev izostaviti?“ Odgovor „ne“ svrstava te zahtjeve u stupanj „mora“.

Da bi se utvrdilo koji zahtjevi spadaju u stupanj „treba“, potrebno je pregledati preostale zahtjeve i zapitati se: „Unapređuje li ovo konkretno svojstvo značajno aplikaciju?“

Za identifikaciju stupnja „može“, treba pregledati sve preostale zahtjeve i za svaki se ponaosob zapitati: „Bi li ovaj zahtjev bio koristan korisnicima?“

Sve što preostane na listi, treba još jednom pojedinačno analizirati i postaviti pitanje: „Je li ovaj zahtjev nepotreban, zbunjujuć ili glup?“ Ako se ne može odgovoriti s „da“, treba se zapitati: „Hoće li se ovaj zahtjev koristiti samo u rijetkim slučajevima?“ Ako se ni sada ne može odgovoriti s „da“, onda treba razmisliti o tome da se konkretni zahtjev prebaci u neku od prethodnih stupnjeva.

3.1.2 Kategorizacija zahtjeva

Zahtjevi se mogu kategorizirati u pet skupina:

- **poslovni** – pogled s visoke razine; objašnjavaju što se kupac nada ostvariti predmetnim projektom
- **korisnički** – opisuju kako će aplikacija biti korištena od strane krajnjih korisnika; često su vrlo detaljni
- **funkcionalni** – detaljni opis željenih svojstava, uz dodatak stvari koje korisnici neće direktno vidjeti – sučelja i tijek aktivnosti / *workflow*; izjave o uslugama koje sustav mora pružati, kako će sustav reagirati na određeni ulazni poticaj, i kako bi se sustav trebao ponašati u određenim situacijama
- **nefunkcionalni** – performanse sustava, pouzdanost, sigurnost, broj korisnika; ograničenja u uslugama i funkcijama, kao što su vremenska ograničenja, (ne)usvojeni standardi, zahtjevi kvalitete, platformski zahtjevi, ograničenja u procesu razvoja i oblikovanja itd.
- **implementacijski** – privremena svojstva potrebna radi uspješnog prelaska sa starog na novi način rada.

3.1.3 Prikupljanje zahtjeva

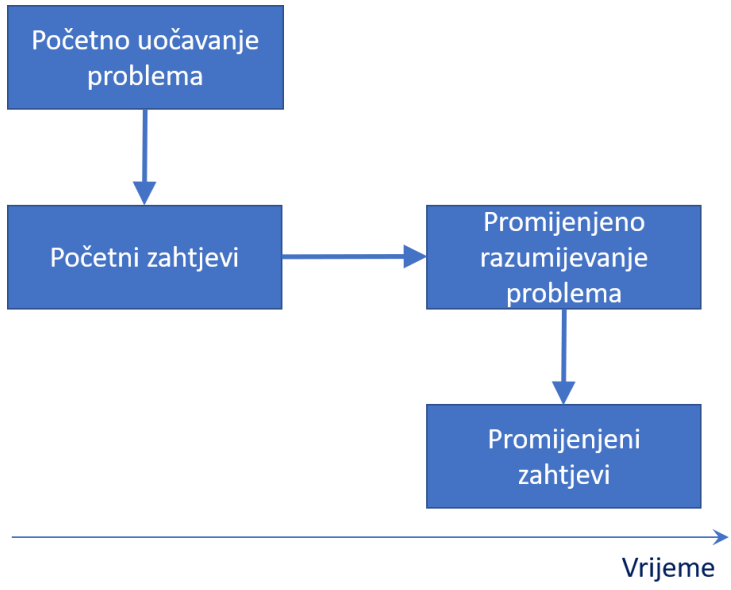
Prikaz postupka prikupljanja zahtjeva dan je na slici 28. Nakon „otkrivanja“ zahtjeva kreće se dalje u njihovo detaljno definiranje. Zahtjevi se klasificiraju i prioritiziraju te se dolazi do završne specifikacije. Nakon toga se cijeli ciklus može ponoviti kako bi se dobio osjećaj sigurnosti da je „sve pokriveno“.

Slika 28: Prikaz postupka prikupljanja zahtjeva [6]

Vremenom se razumijevanje zahtjeva mijenja, te se kaže da „zahtjevi evoluiraju“ (sl. 29).

Tehnike koje se mogu koristiti za prikupljanje i dokumentiranje zahtjeva:

- intervju i slušanje kupca / korisnika:
  - Korisnici znaju više o svom poslovanju nego mi.
  - Fokusirati se na problem, ne na sugerirana rješenja.
  - Ako sugovornik inzistira na „čudnim“ stvarima, treba otkriti zašto mu je to tako važno.
- upitnici:
  - Mnogo jeftiniji od intervjua.
  - Nije moguće na ovaj način dobiti sve potrebne informacije.
- zajedničke radionice:
  - Zajednički rad menadžera, korisnika, analitičara i programera.
  - Radionice moraju biti sponzorirane od nekog visoko pozicioniranog menadžera.
  - Skup, ali vrlo efikasan način definiranja zahtjeva.
- metoda Five Ws (and one H):
  - WHO: Tko će koristiti aplikaciju.
  - WHAT: Što korisnici žele da aplikacija radi.
  - WHEN: Kada je aplikacija potrebna.
  - WHERE: Gdje će se aplikacija koristiti, na kakvim uređajima, u kakvim uvjetima i okolnostima.
  - WHY: Zašto korisnici trebaju aplikaciju.
  - HOW: Kako korisnici misle da bi se problem mogao riješiti.



Slika 29: Evolucija zahtjeva

3.1.4 Dokumentiranje zahtjeva

Nakon prikupljanja zahtjeva, treba ih dokumentirati na način da budu dostupni za analizu i raspravu:

- **običan tekst** – opisivanje zahtjeva jednostavnim jezikom; u slučaju pažljivog formuliranja rečenica, ovaj način može biti vrlo efikasan
- **UML dijagrami** – prikazivanje zahtjeva standardiziranim dijagramima; problem ovog načina je što kupci / korisnici imaju puno boljih načina za trošenje svog vremena od učenja kompliciranih UML principa
- **korisničke priče** – kratki zapis o tome kako bi aplikacija trebala dopustiti korisniku da načini određene akcije
- **slučajevi upotrebe** – opis interakcije između sudionika (sudionici mogu biti korisnici ili sama aplikacija)
- **prototip** – pružanje prilike korisniku da dobije osjećaj kako će aplikacija raditi; može biti funkcionalni i nefunkcionalni; popularno kod razvoja igara
- **specifikacija zahtjeva** – zapis detalja zahtjeva na posebnom predlošku.

Nakon dokumentiranja slijedi verifikacija zahtjeva te dogovaranje procedure za kasnije promjene zahtjeva.

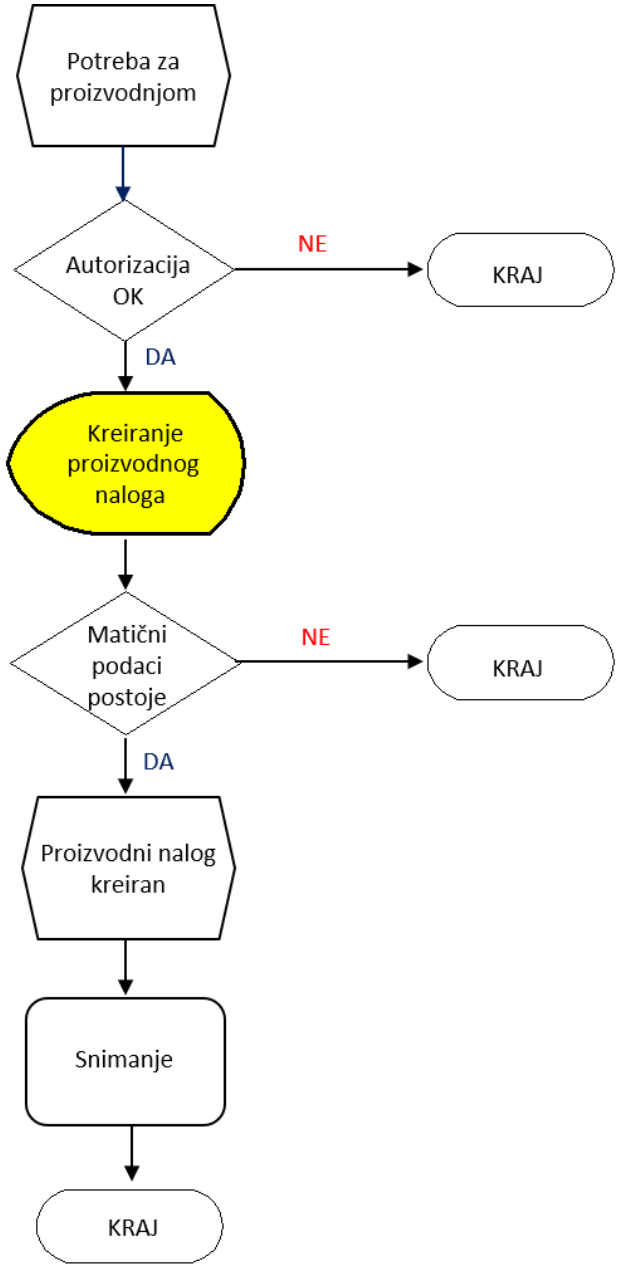
Primjer predloška za specifikaciju zahtjeva, prema [4]:

Sekcija	Tema
1	Kratki opis proizvoda / aplikacije
2	Razvojna i radna okolina
3	Sučelja i tok podataka
4	Funkcionalni zahtjevi
5	Zahtjevi za performansama
6	Rukovanje iznimkama
7	Prioriteti za implementaciju
8	Predvidljive modifikacije i unapređenja
9	Kriteriji prihvaćanja
10	Savjeti i smjernice za dizajn
11	Indeksi
12	Rječnik

3.1.5 EPC dijagrami

Za prikaz zahtjeva, odnosno poslovnih procesa koji se trebaju obuhvatiti novim softverom, često se koriste EPC dijagrami (*Event-driven Process Chain diagram*).

Primjer jednog jednostavnog EPC dijagrama za proces kreiranja proizvodnog naloga, prikazan je na slici 30.



Slika 30: EPC dijagram za proces kreiranja proizvodnog naloga

Čitanje je dijagrama na slici 30 jednostavno. Nakon što je uočena potreba za proizvodnjom neke količine proizvoda, korisnik u sustavu treba pokrenuti akciju kreiranja proizvodnog naloga.

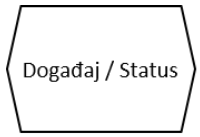
Prije nego što sustav dopusti pokretanje kreiranja, mora se automatski provjeriti ima li korisnik autorizaciju za predmetnu transakciju. Ako autorizacija nije odgovarajuća, postupak se prekida. Ako je autorizacija odgovarajuća, aktivnost može početi.

Nakon što su uneseni podaci o potrebnim količinama i datumima proizvodnje, sustav mora automatski provjeriti postoje li svi potrebni matični podaci:

- matični slog materijala (proizvoda)
- aktivna sastavnica proizvoda
- aktivni postupnik za izradu proizvoda.

Ako bilo koji matični podatak nedostaje, postupak se prekida. Ako svi matični podaci postoje u sustavu, proizvodni nalog može se kreirati u potpunosti. Nakon što je nalog kreiran, potrebno ga je snimiti. S korakom snimanja, postupak se završava.

Značenje najčešće korištenih elemenata EPC dijagrama:



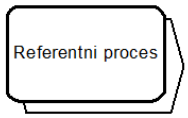
Poslovni događaj ili stanje. Pokretač ili rezultat poslovnog procesa.

Funkcija (aktivnost) transformira ulazne parametre.
Aktivira ju događaj. Funkcija rezultira događajem.

Operator „I" (AND): Svi ulazi ili svi izlazi su aktivni.
Mogući slučajevi:
- I jedan i drugi događaj uzrokuju aktivnost.
- I jedan i drugi događaj posljedica su aktivnosti (funkcije).
- I jedna i druga aktivnost posljedica su događaja.
- I jedna i druga aktivnost uzrokuju događaj.

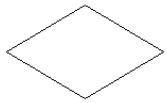
Operator „ILI" (OR):Jedan ili više ulaza ili izlaza su aktivni.
Mogući slučajevi:
- Ili jedan ili drugi događaj, ili oba, uzrokuju aktivnost.
- Ili jedan ili drugi događaj, ili oba, posljedica su aktivnosti (funkcije).
- Ili jedna ili druga aktivnost, ili obje, uzrokuju događaj.

Operator „Ekskluzivni ILI" (XOR): Samo jedan ulaz ili izlaz je aktivan.
Isključuje druge ulaze ili izlaze.
Mogući slučajevi:
- Ili jedan ili drugi događaj uzrokuju aktivnost.
- Ili jedan ili drugi događaj posljedica su aktivnosti (funkcije).
- Ili jedna ili druga aktivnost uzrokuju događaj.



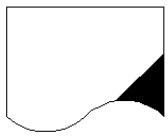
Proces opisan u drugim dijelovima (modulima) Konceptualnog dizajna, a na koji se referencira u aktualnom dijagramu.

Aktivnost u sustavu: transakcija, unos, program itd.
Time se eksplicitno upotpunjuje dijagram procesa naglašavanjem uloge aktivnosti u novom softveru, u određenoj fazi procesa.

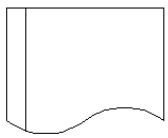


Odluka; izbor između dviju ili više grana procesa, ovisno o događaju.
Odluka može biti tipa da-ne, ili višestruko grananje.

Ovim elementom ističe se početak ili kraj procesa kada je to u dijagramu potrebno naglasiti.



Dokument. Označeni dokument (donji desni kut koji je zacrnjen) naglašava specifičnost dokumenta. Neoznačeni dokument predstavlja tipični dokument procesa. (Oznaka se može maknuti koristeći funkciju u meniju koji se poziva desnim gumbom miša).



Izvještaj.

3.2 Zahtjevi vezani za matične podatke

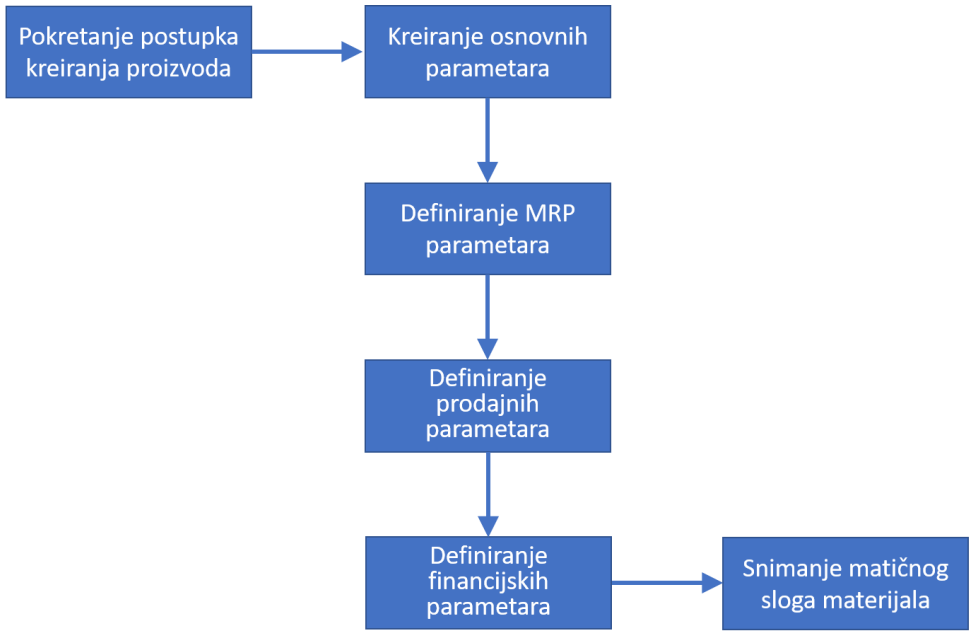
Pomoćna pitanja koja se u procesu prikupljanja zahtjeva mogu uputiti korisnicima u svrhu dobivanja potrebnih informacija vezanih za tematiku matičnih podataka:

- Koje vrste (tipove) materijala treba razlikovati?
- Koji su parametri bitni za materijale?
- Koji su podaci bitni za dobavljače?
- Koji su podaci bitni za kupce?
- Koji su parametri bitni za sastavnice?
- Koji su parametri bitni za postupnike?
- Koji su parametri bitni za radne centre?
- Koji su parametri bitni za zaposlenike?
- Koji su parametri bitni za radne aktivnosti?
- Koji su statusi bitni za pojedine vrste matičnih podataka?

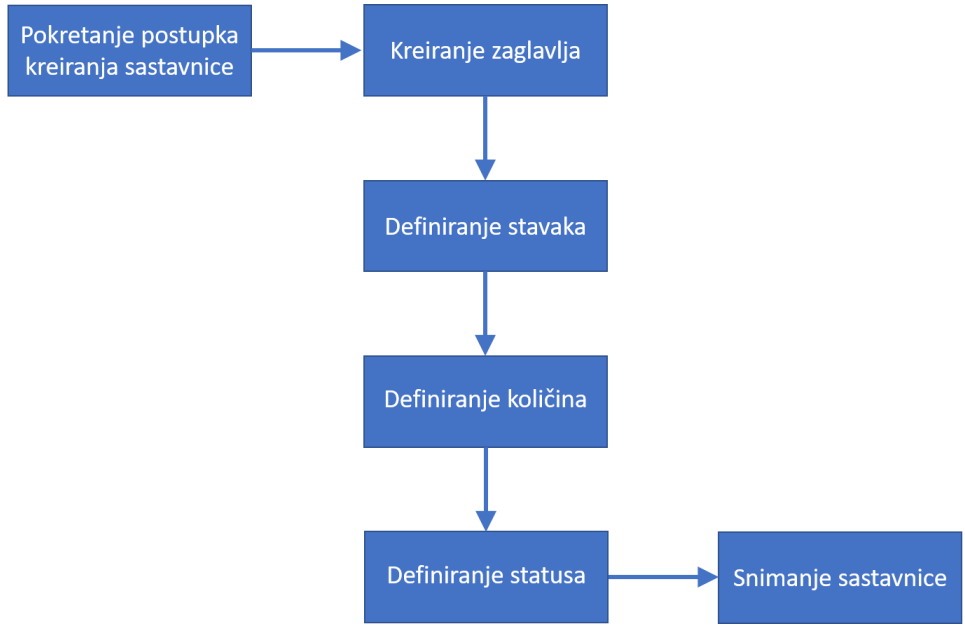
Sljedeći ciklus odnosi se na procesne postupke vezane za sve vrste matičnih podataka:

- postupak kreiranja
- postupak promjene parametara
- postupak promjene statusa
- postupak arhiviranja.

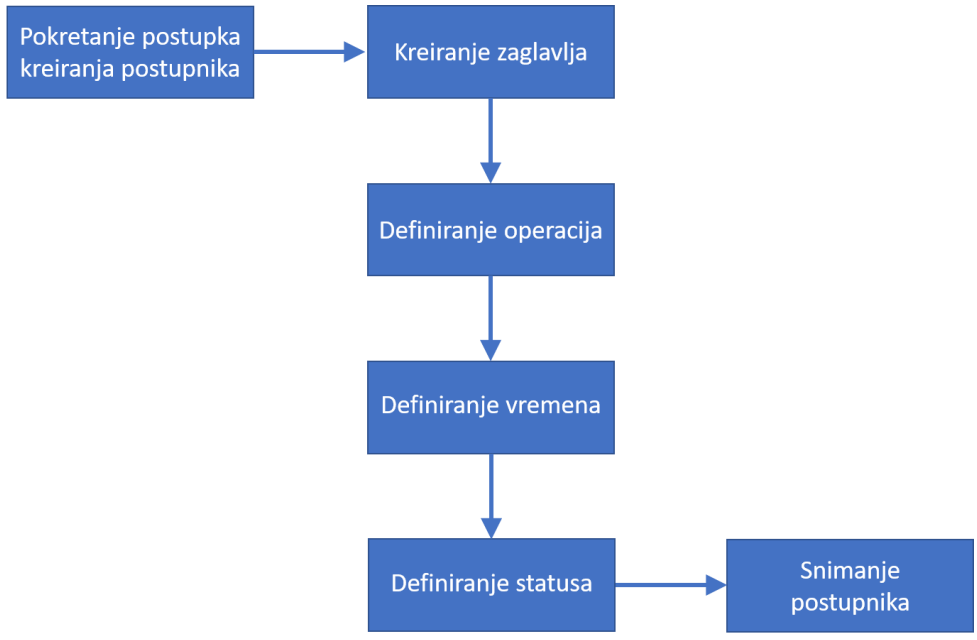
Na sljedećim trima slikama prikazan je poslovni proces kreiranja triju osnovnih matičnih podatka potrebnih za proces proizvodnje: materijal (proizvod), sastavnica i postupnik.



Slika 31: Grubi prikaz postupka kreiranja matičnog sloga materijala



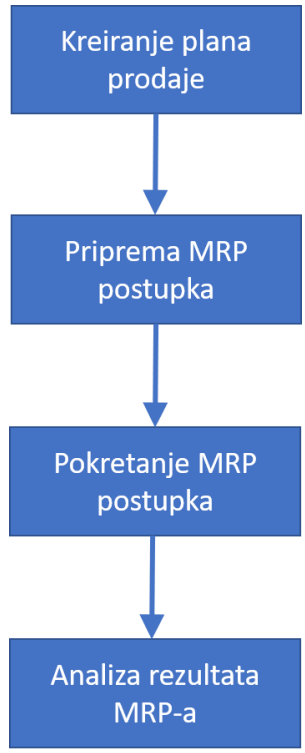
Slika 32: Grubi prikaz postupka kreiranja sastavnice proizvoda



Slika 33: Grubi prikaz postupka kreiranja postupnika za izradu proizvoda

3.3 Zahtjevi vezani za planiranje

Proces planiranja kreće u najvećem broju slučajeva od plana prodaje. Zaposlenici službe prodaje unose svoje tjedne, mjesečne i godišnje planove u sustav. Zajednički plan prodaje obrađuje se putem MRP procedure (objašnjeno u poglavlju 2.3 i prikazano na slici 15).



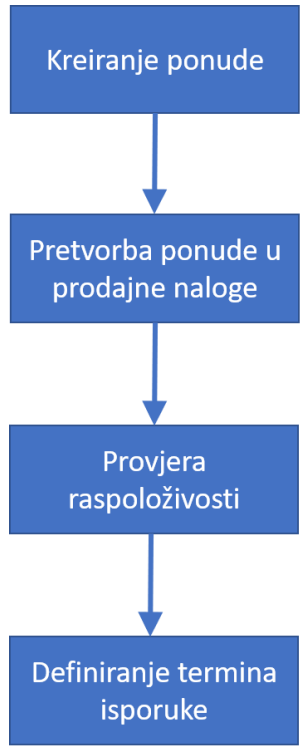
Slika 34: Grubi prikaz postupka planiranja

Pomoćna pitanja koja se u procesu prikupljanja zahtjeva mogu uputiti korisnicima, u svrhu dobivanja potrebnih informacija vezanih za tematiku planiranja:

- Tko pokreće planiranje?
- Koliko često se radi planiranje?
- Koliko vrsta planiranja se koristi?
- Koji je postupak s rezultatima planiranja?
- Kakvi se izvještaji žele nakon završetka planiranja?

3.4 Zahtjevi vezani za prodaju

Proces prodaje uglavnom kreće kreiranjem ponude, koja se, zatim, nakon potvrde od strane kupca, pretvara u prodajni nalog. Na osnovi prodajnog naloga radi se provjera raspoloživosti (postoji li dovoljna količina proizvoda na zalihi) i, ako je sve u redu, definira se datum isporuke.



Slika 35: Grubi prikaz prodajnog postupka

Pomoćna pitanja koja se, u procesu prikupljanja zahtjeva, mogu uputiti korisnicima u svrhu dobivanja potrebnih informacija vezanih za tematiku prodaje i prodajnih ponuda:

- Tko kreira ponude?
- Koji su elementi sastavni dio ponude?
- Kakav je postupak kreiranja ponude?
- Što se zbiva nakon kreiranja ponude?
- Kako se mijenja ponuda?
- Konvertira li se ponuda u prodajni nalog?
- Brišu li se / Arhiviraju li se ponude?

Pomoćna pitanja koja se, u procesu prikupljanja zahtjeva, mogu uputiti korisnicima u svrhu dobivanja potrebnih informacija vezanih za tematiku prodaje i prodajnih naloga:

- Tko kreira prodajne naloge?
- Koji su elementi sastavni dio prodajnih naloga?
- Koliko vrsta prodajnih naloga ima?
- Kako su klasificirani (grupirani) proizvodi?
- Kakav je postupak kreiranja prodajnih naloga?
- Kako se izračunava prodajna cijena proizvoda?
- Koja se metoda koristi za provjeru raspoloživosti?
- Što se zbiva nakon kreiranja prodajnog naloga?
- Kako se mijenja prodajni nalog?
- Koji su tipovi prodajnih naloga relevantni za planiranje?
- Postoje li aktivnosti s povratnom ambalažom?
- Brišu li se / Arhiviraju li se prodajni nalozi?
- Koje vrste izvještaja o prodaji treba prodajna operativa?
- Koje vrste izvještaja o prodaji trebaju izvršni menadžeri?
- Koje vrste izvještaja o prodaji treba uprava?

Dodatne napomene:

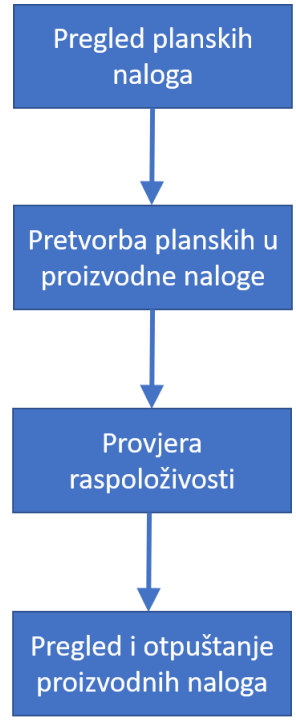
- Na ovaj način rade tvrtke koje proizvode robu za nepoznatog kupca, tj. proizvode prema planu prodaje, po principu *Make to Stock* i proizvedenu robu stavljaju na zalihu.
- Tvrtke koje proizvode za poznatog kupca rade po principu *Make to Order*. Ne proizvodi se unaprijed, nego se čeka konkretni zahtjev kupca. Kad stigne zahtjev, kreira se prodajni nalog i tek se onda pokreće proizvodnja željenog artikla u željenoj količini.
- Neke vrste industrije koriste ambalažu za višekratnu upotrebu, koja kruži između subjekata kao povratna ambalaža. U nekim slučajevima, povratna ambalaža zna biti jako skupa i zahtijeva detaljno praćenje u sustavu, a o tome obično vodi računa služba prodaje ili logistike (bez te ambalaže nema isporuke kupcu).

3.5 Zahtjevi vezani za proizvodnju

Proizvodnja se uvijek radi uz pažljivu pripremu. Zato je redovno, u proizvodnim tvrtkama, služba pripreme proizvodnje odvojena i često potpuno nezavisna od službe izvršenja proizvodnje.

Priprema proizvodnje počinje s pregledom rezultata MRP-a, odnosno pregledom planskih naloga za proizvodnju i završava s korakom otpuštanja proizvodnih naloga.

Otpušteni proizvodni nalozi su ulaz u proces izvršenja proizvodnje. Služba proizvodnje ima zadatak proizvesti traženu količinu proizvoda u traženom roku i na definiranim radnim centrima (sve to piše u proizvodnim nalozima).



Slika 36: Grubi prikaz postupka pripreme proizvodnje

Tijekom izvršenja proizvodnje, događaju se tri vrste materijalnih i vremenskih knjiženja:

- knjiženje utroška materijala (povlačenje sirovine iz skladišta u proizvodnju)
- knjiženje utroška vremena
- prijava proizvedenih količina (slanje proizvoda iz proizvodnje na skladište).

3.5.1 Priprema proizvodnje

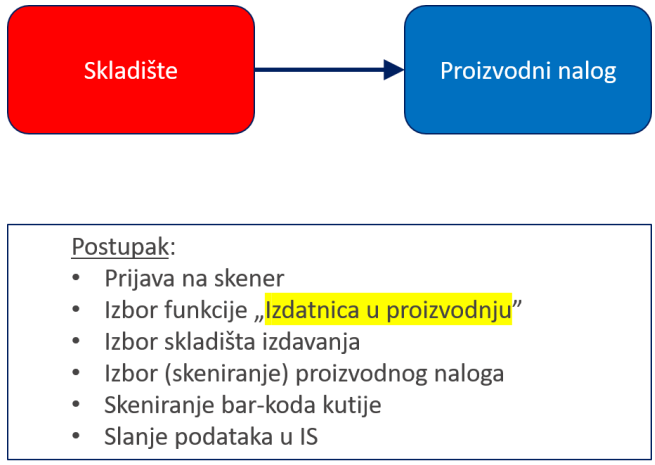
Pomoćna pitanja za područje pripreme proizvodnje:

- Tko kreira proizvodne naloge?
- Koji su elementi sastavni dio proizvodnih naloga?
- Kakav je postupak kreiranja proizvodnih naloga?
- Kakvi su mogući statusi proizvodnih naloga?
- Što se zbiva nakon kreiranja proizvodnih naloga?
- Koje se provjere rade prije pokretanja proizvodnje?
- Kako se mijenja proizvodni nalog?
- Treba li se proizvodni nalog dodatno odobriti prije proizvodnje?
- Brišu li se / Arhiviraju li se proizvodni nalozi?
- Kakvi su izvještaji potrebni o proizvodnim nalozima?

3.5.2 Knjiženje utroška materijala

Pomoćna pitanja vezana za područje knjiženja utroška materijala:

- Tko knjiži utrošak materijala?
- Na koji se način knjiži utrošak materijala?
- Iz kojeg skladišta se knjiži utrošak na proizvodni nalog?
- Koji podaci su sastavni dio knjiženja?
- Kako se rade ispravci načinjenih knjiženja?
- Kakvi su izvještaji potrebni o utrošku materijala?



Slika 37: Grubi prikaz postupka knjiženja utroška materijala skeniranjem

3.5.3 Knjiženje utroška vremena

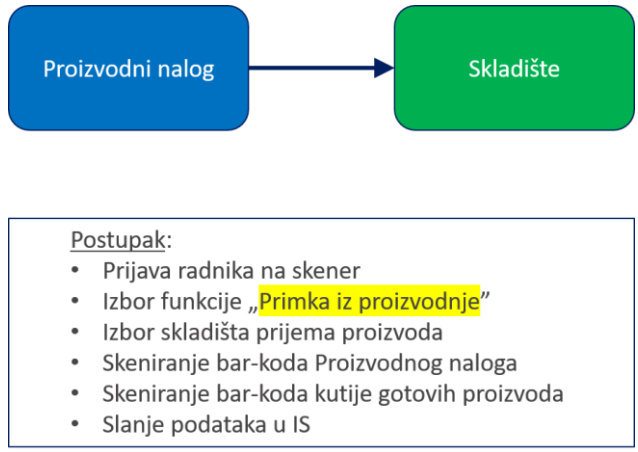
Pomoćna pitanja vezana za knjiženje utroška vremena:

- Tko knjiži utrošak vremena?
- Na koji se način knjiži utrošak vremena?
- Koji podaci su sastavni dio knjiženja?
- Kako se rade ispravci načinjenih knjiženja?
- Kakvi su izvještaji potrebni o utrošku vremena?

3.5.4 Prijava proizvodnje

Pomoćna pitanja vezana za prijavu proizvodnje:

- Tko knjiži proizvedene količine?
- Na koji se način knjiži proizvedena količina?
- U koje skladište se knjiži prijem iz proizvodnje?
- Koji podaci su sastavni dio knjiženja?
- Kako se rade ispravci načinjenih knjiženja?
- Kakvi su izvještaji potrebni o proizvedenim količinama?



Slika 38: Grubi prikaz postupka knjiženja proizvedene robe na skladište skeniranjem

3.6 Zahtjevi vezani za nabavu

Proces nabave počinje pregledom postojećih zahtjevnica u sustavu (zahtjevnice vezane za proizvodnju kreirane su automatski pomoću MRP-a). Radi se izbor dobavljača te se zahtjevnice, vezane za jednog dobavljača, zajednički pretvaraju u narudžbenicu prema izabranom dobavljaču. Kreirane narudžbenice provjeravaju se i po potrebi odobravaju (svaka tvrtka ima svoju logiku odobravanja), te se šalju dobavljačima.

Slika 39: Grubi prikaz postupka naručivanja materijala od dobavljača

Pomoćna pitanja vezana za zahtjevnice i narudžbenice:

- Koji je postupak kreiranja zahtjevnica?
- Koji su elementi potrebni na zahtjevnicama?
- Koji je postupak s kreiranim zahtjevnicama?
- Moraju li se zahtjevnice dodatno odobravati i po kojim kriterijima?
- Na koji se način kreiraju narudžbenice?
- Koji su kriteriji za izbor dobavljača?
- Treba li izbor dobavljača biti automatski ili će ga raditi korisnik?
- Koji su elementi potrebni na narudžbenicama?
- Koji je postupak s kreiranim narudžbenicama?
- Moraju li se narudžbenice dodatno odobravati i po kojim kriterijima?
- Mogu li se kreirane / odobrene narudžbenice mijenjati?

3.7 Zahtjevi vezani za logistiku

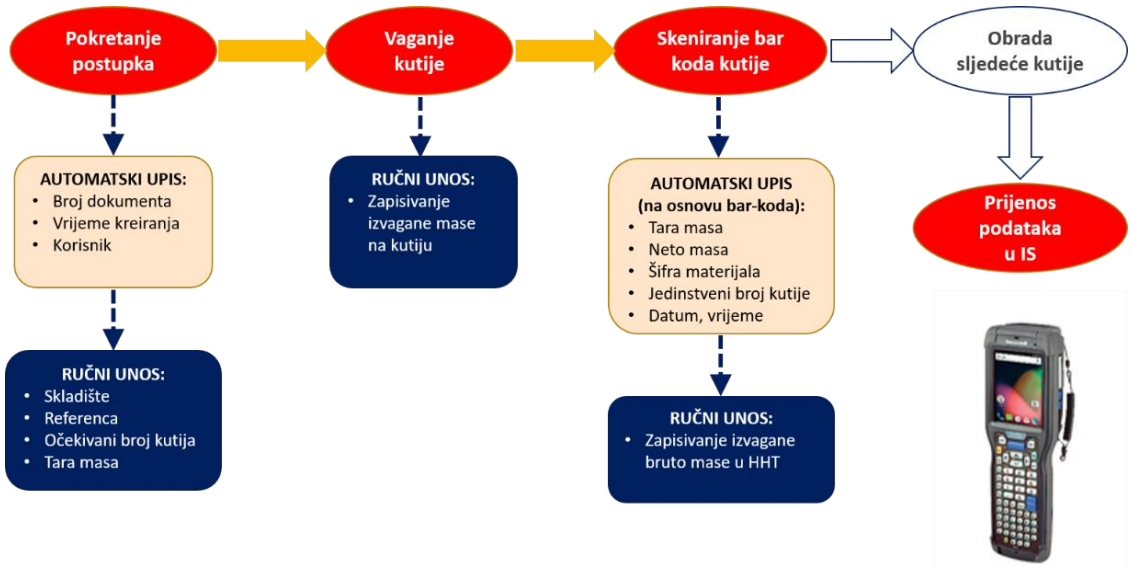
Logistika se bavi organizacijom, planiranjem, kontrolom i realizacijom tokova svih vrsta materijala. Uglavnom se dijeli na ulaznu (zaprimanje materijala od dobavljača), internu (premještanje materijala između dvaju skladišta) i izlaznu (isporuke materijala kupcima).

3.7.1 Ulazna logistika – prijem materijala

Zaprimanje materijala od dobavljača bitan je proces koji se odvija prije proizvodnje. Ako se ne odradi dobro, proizvodnja će izgubiti mogućnost praćenja sljedivosti, odnosno neće se moći saznati ni dokazati koji je dobavljač poslao lošu sirovinu. Samim tim, neće se moći saznati ni u kojim je sve pakiranjima proizvoda ugrađena loša sirovina.

Na slici 40 prikazan je primjer korištenja skenera kod zaprimanja sirovine zapakirane u kutije nejednakih masa:

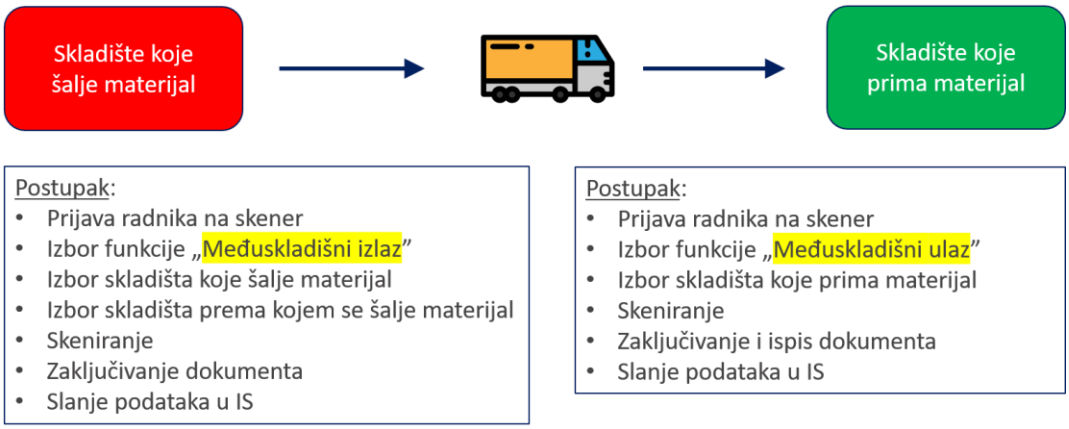
- Svako zaprimanje veže se na referentni dokument.
- Količine u kutijama zapisane su u sklopu bar-koda na etiketi.
- Količine su međusobno nejednake i potrebno ih je provjeriti vaganjem.
- Svaka kutija ima etiketu s važnim podacima za ispravno funkcioniranje poslovnog sustava.
- Nakon što se etiketa odskenira, podatke za svaku kutiju potrebno je putem sučelja poslati u informacijski sustav (IS).



Slika 40: Grubi prikaz postupka zaprimanja materijala od dobavljača skeniranjem

3.7.2 Interna logistika

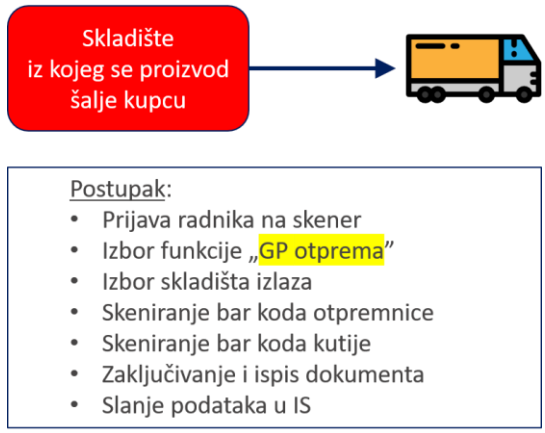
Na slici 41 prikazan je primjer preskladištenja zapakiranog materijala (sirovina ili proizvod, svejedno – bitno je da je zapakirano i da ima etiketu s jedinstvenim brojem).



Slika 41: Grubi prikaz postupka preskladištenja materijala skeniranjem

3.7.3 Izlazna logistika – isporuka proizvoda

Na slici 42 prikazan je primjer pripreme isporuke proizvoda za kupca. Proizvod je zapakiran i svaka je pakovina označena etiketom s jedinstvenim brojem.



Slika 42: Grubi prikaz postupka preskladištenja materijala skeniranjem

4 Osnovni principi dizajniranja

Nakon prikupljenih projektnih zahtjeva, može se početi s postupkom pretvaranja specifikacija u izvršni sustav. Počinje se s dizajnom više razine (*High Level Design*), nakon čega slijedi detaljni dizajn (*Low Level Design*).

4.1 Dizajn više razine

Dizajn više razine grubo oslikava strukturu aplikacije:

- identificira opće okruženje sustava (hardver, operativni sustav, mreža itd.)
- identificira arhitekturu (monolitna, klijent / server, servisno orijentirana)
- identificira glavne komponente sustava (moduli izvještavanja, baze podataka i klase najviše razine)
- opisuje na koji će način djelovati sustav.

Dizajn više razine omogućuje prikaz sustava na apstraktnom nivou. Pokazuje kako će se glavni dijelovi gotove aplikacije uklopiti i međusobno komunicirati. Dizajn više razine trebao bi, također, specificirati pretpostavke o okruženju u kojem će se gotova aplikacija pokretati. Na primjer, trebao bi opisati hardver i softver, koji će se koristiti za razvoj aplikacije, i hardver, koji će na kraju pokrenuti program. Dizajn na višoj razini, ne usredotočuje se na detalje o načinu funkcioniranja dijelova aplikacije. Ti će se detalji moći naknadno razraditi tijekom detaljnog dizajna.

Razvoj softvera može se promatrati kao proces koji usitnjava sustav na sve manje i manje dijelove, dok ne budu dovoljno mali za implementaciju. Koristeći to gledište, dizajn više razine prvi je korak u procesu usitnjavanja. Cilj je podijeliti sustav na dijelove koji su dovoljno samostalni da bi ih se moglo dati zasebnim timovima na realizaciju.

4.1.1 Sigurnost

Dizajn više razine trebao bi skicirati sve sigurnosne potrebe aplikacije. Te potrebe mogu uključivati sljedeće:

- sigurnost operativnog sustava – to uključuje vrstu postupaka za prijavu, pravila isteka lozinke i standarde zaporki
- sigurnost aplikacija – neke se aplikacije oslanjaju na sigurnost operativnog sustava i ne pružaju vlastitu, dok druge traže zasebno korisničko ime i zaporku. Zatim, sigurnost aplikacije znači osiguravanje odgovarajuće razine pristupa različitim korisnicima (nekim korisnicima nije dopušten pristup svim dijelovima sustava).
- sigurnost podataka – pružanje sigurnosti da osjetljive informacije ne padnu u ruke neovlaštenih osoba ili hakera
- mrežna sigurnost – sprečavanje *cyber* razbojnika da kradu podatke s mreže
- fizička sigurnost – ovu komponentu sigurnosti mnogi softverski inženjeri zanemaruju; ako netko ukrade prijenosno računalo iz otključanog ureda, aplikacija se neće moći koristiti.

4.1.2 Hardver

U današnje vrijeme moguće je raditi aplikacije koje će se izvoditi na serverima, stolnim računalima, prijenosnim računalima, tabletima, telefonima i sl. Sve se više pojavljuju nosivi uređaji, kao što su narukvice, satovi, naočale i slušalice.

4.1.3 Korisnička sučelja

Tijekom dizajna više razine, može se skicirati korisničko sučelje. Na primjer, mogu se navesti glavne metode za navigaciju kroz aplikaciju.

Starije aplikacije, za radnu površinu koriste obrasce s izbornicima koji prikazuju druge oblike. Korisnik često može prikazati više obrazaca istovremeno i prelaziti između njih klikom miša (ili dodirom ako hardver ima zaslon osjetljiv na dodir).

Suprotno tome, novije aplikacije, u stilu tableta, imaju tendenciju korištenja jednog prozora (koji obično pokriva cijeli tablet) i tipki ili strelice za navigaciju. Kada se klikne na gumb, pojavljuje se novi prozor koji ispunjava uređaj.

Koji god se navigacijski model odabere, treba odrediti obrasce ili prozore koje će aplikacija uključivati. Tada se može provjeriti dopuštaju li korisniku izvršavanje zadataka definiranih u zahtjevima (treba se proći kroz korisničke priče).

Uz osnovni stil navigacije aplikacije, dizajn korisničkog sučelja na višoj razini može opisati posebne značajke, kao što su ikonice ili tabovi na koje se može kliknuti, važne tablice ili metode za određivanje postavki sustava (poput klizača ili tekstnih okvira).

Ovaj dio dizajna, također, može riješiti opća pitanja poput izgleda u boji, rasporeda logotipa tvrtke i prekrivanja obrasca.

4.1.4 Interna sučelja

Kad se program podijeli na dijelove, trebalo bi se odrediti na koji način će dijelovi međusobno djelovati. Tada timovi, dodijeljeni pojedinim dijelovima, mogu raditi odvojeno bez potrebe za stalnom koordinacijom.

Važno je da dizajn više razine precizno i nedvosmisleno precizira te unutarnje interakcije, kako bi timovi mogli raditi što neovisnije moguće. Ako se dva tima, koja trebaju komunicirati, ne dogovore kako bi ta interakcija trebala nastupiti, mogu izgubiti puno vremena. Mogli bi gubiti vrijeme u svađi oko toga koji je pristup bolji. Vrijeme će se, također, izgubiti ako jedan tim treba promijeniti sučelje, a to prisiljava i drugi tim da promijeni svoje sučelje. Problem se drastično povećava ako više interakcija mora funkcionirati preko istog sučelja.

Vrijedno je potrošiti malo više vremena za pažljivo definiranje takvih internih sučelja prije nego što programeri počnu pisati kod. Nažalost, možda neće biti moguće definirati sučelja prije nego što se napišete bar dio programa. U tom slučaju, možda će trebati izolirati dva projektna tima definiranjem privremenog sučelja. Nakon što su timovi napisali dovoljno koda da znaju koje informacije trebaju razmjenjivati, mogu definirati konačno sučelje.

4.1.5 Vanjska sučelja

Mnoge aplikacije moraju komunicirati s vanjskim sustavima.

Na neki su način vanjska sučelja često lakša za odrediti od unutarnjih, jer se obično ne traži kontrola na oba kraja sučelja. Ako nova aplikacija treba komunicirati s postojećim sustavom, tada taj sustav već ima zahtjeve za sučeljem koji se moraju ispuniti.

Suprotno tome, ako se želi da se budući vanjski sustavi povezuju s postojećim, obično se definira da kasnije razvijeni sustavi trebaju udovoljiti zahtjevima postojećeg sustava.

4.1.6 Arhitektura

Arhitektura aplikacije opisuje kako se njezini dijelovi slažu na visokoj razini. Programeri koriste puno „standardnih“ vrsta arhitektura. Mnogi od njih bave se određenim karakteristikama problema koji se rješava.

Na primjer, sustavi bazirani na pravilima, često se koriste za rješavanje složenih situacija, u kojima se rješavanje određenog problema svodi na slijeđenje skupa pravila. Korisnik pozove definirani broj telefonski, jer se njegovo računalo ne može povezati s internetom, a sugovornik mu, s druge strane linije, postavlja niz pitanja kako bi se pokušao dijagnosticirati problem. Sugovornik čita pitanje s računalnog zaslona, korisnik odgovara i sugovornik klikne na odgovarajući gumb, kako bi se došlo do sljedećeg pitanja. Pravila unutar dijagnostičkog sustava odlučuju koje će se pitanje postaviti korisniku kao sljedeće.

Druge arhitekture pokušavaju pojednostaviti razvoj, smanjivanjem interakcija među dijelovima sustava. Na primjer, arhitektura bazirana na komponentama, pokušava svaki dio sustava učiniti što izoliranijim kako bi različiti timovi programera mogli zasebno raditi na njima.

Najčešće arhitekture koje se koriste su:

- monolitna arhitektura (*Monolithic*)
- arhitektura klijent / poslužitelj (*Client / Server*)
- uslužna arhitektura (*Service-Oriented*)
- komponentna (*Component-Based*).

Monolitna arhitektura karakterizirana je činjenicom da jedan program čini sve. Isti program prikazuje korisničko sučelje, pristup podacima, obrađuje narudžbe kupaca, ispisuje račune i čini sve što aplikacija treba učiniti.

Ova arhitektura ima značajne nedostatke. Konkretno, dijelovi sustava su usko povezani, tako da nema velike fleksibilnosti. Na primjer, ako aplikacija pohranjuje podatke o adresi kupca, a kasnije će se morati promijeniti oblik adrese, tada će se, također, morati promijeniti svaki komad koda koji koristi adresu. To možda nije pretjerano teško, ali znači da programeri, koji rade na povezanim dijelovima koda, moraju stati s onim što rade i nositi se s promjenom prije nego što se vrate na trenutne zadatke.

Monolitna arhitektura također zahtijeva razumijevanje kako se svi dijelovi sustava spajaju od početka projekta. Ako pogrešno shvatite neke detalje, tijesno spajanje dijelova sustava čini kasnije njihovo teško fiksiranje.

Prednost monolitne arhitekture je da je sve izgrađeno u jednom programu i da nema potrebe za kompliciranom komunikacijom putem mreža. To znači da se ne treba pisati i uklanjati ispravke komunikacijskih rutina i ne treba se brinuti o padu mreže.

Monolitne arhitekture također su korisne za male aplikacije u kojima jedan programer ili jedan tim radi na kodu.

Arhitektura klijent-poslužitelj temeljna je arhitektura današnjeg interneta i temeljni stil raspodijeljenih sustava [2]. Kod ovog arhitekturnog stila, neka računala služe kao centri za razmjenu informacija i pružanje usluga, te se zovu poslužitelji (engl. *server*). Druga računala služe za pristup informacijama i dobivanje usluga, te se ona nazivaju klijenti (engl. *client*). Uobičajeni je slučaj da više klijenata pristupa jednom poslužitelju, ali moguće je i da više klijenata pristupa većem broju poslužitelja istovremeno, ovisno o tome kako je točno ostvarena određena usluga na strani poslužitelja.

Model arhitekture klijent-poslužitelj uključuje razmatranje funkcioniranja klijenta i poslužitelja prije, tijekom i po završetku zajedničke komunikacije. Razlikuju se dva pogleda na arhitekturu klijent-poslužitelj:

- Pod pojmom klijenta i poslužitelja podrazumijevaju se fizička računala koja komuniciraju.
- I klijent i poslužitelj su programi na fizičkim računalima koja se povezuju radi zajedničke komunikacije.

Uslužna arhitektura (engl. *Service-Oriented Architecture,* SOA) organizira programsku potporu kao kolekciju usluga, koje međusobno komuniciraju uporabom dobro definiranih sučelja putem mrežnih protokola. Uslužna arhitektura slična je posredničkoj arhitekturi u mnogim svojim značajkama. Glavna razlika je u konceptu usluge. Svaka usluga ima dobro definiranu funkciju, samodostatna je i njezino unutarnje funkcioniranje ne ovisi o drugim uslugama ili stanju njezinog okruženja. Iako se kod posredničke arhitekture može govoriti o usluzi određene odredišne komponente, fokus tog arhitekturnog stila nije na samoj usluzi, već na raspodijeljenoj komunikaciji.

Uslužna arhitektura temelji se na trima entitetima: zahtjevatelju ili klijentu usluge, pružatelju usluge i posredniku (registru, repozitoriju) usluge.

Klijent, različitim operacijama, pronalazi podatke o usluzi u registru usluga, i zatim se povezuje s pružateljem usluge kako bi pozvao neku od njegovih usluga. Klijent može koristiti više usluga ako ih pružatelj nudi.

Pružatelj usluge stvara uslugu i oglašava sve potrebne informacije o njoj u registru usluga.

Posrednik (registar) služi za otkrivanje informacije o postojanju usluge potencijalnom klijentu. Javni registri omogućuju pristup usluzi bilo kojem klijentu, dok privatni registri omogućuju pristup usluzi samo određenim klijentima.

Arhitektura zasnovana na događajima (engl. *Event-Based Architecture*) je arhitektura kod koje je programska potpora organizirana na način da postoje komponente, koje stvaraju događaje i komponente, koje reagiraju na određene događaje.

Temeljne značajke ove arhitekture su:

- Komponente se međusobno ne pozivaju eksplicitno.
- Neke komponente generiraju signale = događaje.
- Neke komponente su zainteresirane za pojedine događaje te se prijavljuju na strukturu za povezivanje komponenata.
- Model izvođenja je takav da se događaj javno objavljuje te se pozivaju prijavljene procedure za obradu tog događaja.
- Komponente koje objavljuju događaj nemaju informaciju o tome koje će sve komponente i kako reagirati na događaj.
- Rukovanje događajima je asinkrono, nema jamstva da će se neki događaj obraditi prije nekog drugog događaja (nedeterminizam).

Važna je značajka implicitno pozivanje procedure / metode, što znači da jedna metoda ne zove izravno drugu, već se druga metoda poziva (od strane sustava / strukture za povezivanje komponenti) ako je prijavljena za određeni događaj.

Primjeri sustava koji koriste arhitekturu zasnovanu na događajima su oni koji obuhvaćaju komunikaciju s korisnikom putem sučelja (npr. klik mišem je događaj) i web-aplikacije.

4.1.7 Izvještaji

Gotovo svaki ozbiljni softverski projekt može koristiti neke vrste izvještaja. Poslovne aplikacije mogu uključivati izvještaje koji se bave kupcima (tko kupuje, tko ima neplaćene račune, gdje kupci žive), proizvodima (inventar, cijene, što se dobro prodaje) i korisnicima (koji zaposlenici prodaju, raspored poslova zaposlenika).

Čak i relativno jednostavne aplikacije ponekad mogu imati koristi od izvještaja. U slučaju jednostavne *shareware* igre, koju će korisnici preuzeti s interneta i instalirati na svoje mobilne telefone, korisnici neće trebati izvještaje (osim možda popis njihovih visokih rezultata), ali bi ih svakako bilo dobro imati. Korisne su informacije o tijeku igre (gdje su korisnici, kada koriste igru, koliko često se igraju, koji dijelovi igre traju dugo itd.). Ti se podaci poslije mogu upotrijebiti za poboljšanje igre i marketinga.

4.1.8 Baza podataka

Dizajn baze podataka važan je dio većine aplikacija. Prvi dio dizajna baze podataka je odluka o tome kakvu će bazu podataka program trebati. Trebate odrediti hoće li aplikacija pohranjivati podatke u tekstualne datoteke, XML datoteke, cjelovitu relacijsku bazu podataka ili nešto egzotičnije, poput vremenske baze podataka ili spremišta objekata (čak i program koji ne koristi nijednu bazu podataka i dalje treba pohraniti podatke).

Ako se odluči koristiti vanjsku bazu podataka (tj. više od podataka ugrađenih u kod), trebalo bi odrediti proizvod baze podataka koji će se koristiti. Mnoge aplikacije

pohranjuju svoje podatke u relacijske baze podataka, kao što su: *Access*, *SQL Server*, *Oracle* ili *MySQL*.

Ako se koristi relacijska baza podataka, tijekom dizajniranja više razine mogu se skicirati tablice koje sadrže i njihove odnose. Kasnije se može navesti više pojedinosti, kao što su specifična polja u svakoj tablici i polja koja čine ključeve koji povezuju tablice.

4.1.9 Protok podataka i statusi dokumenata

Mnoge aplikacije koriste podatke koji se odvijaju u različitim procesima. Na primjer, prodajni nalog kreira se prilikom dogovaranja s kupcem, zatim se prebacuje na završni dio proizvodnje (montaža i skupljanje stavaka za pripremu otpreme), i, na kraju, prelazi na stvarnu isporuku. Na kraju podaci kolaju od isporuke do konačnog procesa naplate.

Također, svi dokumenti mogu se kretati kroz niz statusa, koji često odgovaraju procesima u povezanom protoku podataka. Tako se promatraju statusi prodajnih naloga, proizvodnih naloga i statusi narudžbenica.

Statusi prodajnog naloga:

- kreirano, proizvedeno, otpremljeno, naplaćeno.

Statusi proizvodnog naloga:

- kreirano, otpušteno, proizvedeno, obračunato.

Statusi narudžbenice:

- kreirano, odobreno, poslano, zaprimljeno, plaćeno.

4.2 Detaljni dizajn

Nakon što se dizajnom više razine razdvoji projekt na više dijelova, može ih se dodijeliti skupinama unutar projekta kako bi se moglo raditi na detaljnom dizajnu.

Detaljni dizajn treba osigurati nedostajuće detalje koji su potrebni, prije nego što programeri mogu započeti s pisanjem koda. Daju se konkretnije smjernice o tome kako će dijelovi sustava raditi pojedinačno i kako će raditi zajedno. Radi se na detaljima baze podataka, glavnih klasa i internih i vanjskih sučelja. Dizajn više razine fokusira se na „što“, a detaljni dizajn počinje se usredotočavati na „kako“.

Kao analogija može poslužiti primjer s gradnjom autocesta. Dizajn više razine određuje koji bi gradovi trebali biti povezani autocestama. Detaljni dizajn treba točno naznačiti gdje će se postavljati autoceste, gdje će biti naplatne kućice i gdje će biti odmorišta.

Granica između ove dvije vrste dizajna često ostaje nejasna. Obično, nakon što je dio sustava dodan dizajnu više razine, članovi tima nastavljaju raditi na tom dijelu kako bi razvili njegov detaljni dizajn. Na velikim projektima čest je slučaj da dio ekipe još uvijek radi na dizajniranju više razine, dok su drugi već počeli raditi na detaljima. Programeri mogu čak započeti raditi na dijelovima sustava koji su adekvatno definirani.

Detaljiziranje dizajna više razine nije nužno jednostavno. Možda se općenito zna što je potrebno u bazi podataka (podaci o klijentima i slično), ali ako se to znanje ne pretvori u dobar detaljni dizajn baze podataka, kasnije se može naići na razne vrste problema. Podaci mogu postati nedosljedni, mogu se izgubiti kritične informacije, a financijski izvještaji mogu biti ubitačno spori. Različiti dizajni baza podataka mogu napraviti razliku između pronalaženja podataka, koji su vam potrebni u sekundi, u satima, ili ih se uopće ne može dobiti.

Dizajn više razine identificirao je glavne vrste klasa koje će aplikacija koristiti. Sada je vrijeme za pročišćavanje dizajna, kako bi se identificirale posebne klase koje će trebati programu. Nove klase trebaju sadržavati definicije svojstava, metoda i događaja koje će pružiti aplikaciji za upotrebu.

4.2.1 Identificiranje klasa (razreda)

Jasno je da bi se trebalo identificirati glavne klase koje će aplikacija koristiti. Preostaje vidjeti kako to učiniti. Jedan od načina odabira klasa je traženje imenica u opisu značajke aplikacije.

Na primjer, pretpostavka je da se radi na kreiranju aplikacije pod nazivom *FreeWheeler Automatic Driver* (FAD) koja automatski upravlja automobilima [3]. Početna rečenica: „Program upravlja automobilom do odabranog odredišta.“ sadrži tri imenice, a to su program, automobil i odredište.

Program ne treba izravno manipulirati samim sobom, pa neće biti potrebna klasa „Program“. Međutim, trebat će raditi s automobilima i odredištima, tako da će vjerojatno biti potrebne klase „Automobil“ i „Odredište“.

Kad se proučavaju moguće klase, treba razmisliti o tome koje su vrste klasa potrebne (svojstva), što sve treba učiniti (metode) i koje su promjenjive okolnosti (događaji) nastupili. Na primjer, klasa „Automobil“ trebat će:

- svojstva: Trenutna_Brzina, Trenutni_Pravac i Razina_Goriva
- metode: Ubrzanje, Usporavanje, Aktiviranje_Pok_Smjera, Aktiviranje_Trube
- događaji: Start_Motora, Niska_Razina_Goriva, Sudar_Neizbjezan.

Klasa „Odredište“ vjerojatno je puno jednostavnija, jer u osnovi samo predstavlja određeno mjesto. Zapravo, moguće je da aplikaciji treba samo jedan primjerak ove klase da bi zabilježio trenutno odredište. Izrada samo jedne instance klase, znak je upozorenja da možda klasa nije potrebna. Činjenica da klasa „Odredište“ ne radi ništa ili se ne mijenja sama (tako da ne pruža metode ili događaje), još je jedan pokazatelj da ta klasa možda nije potrebna.

Definicija klase jako ovisi o načinu korištenja objekata. Primjerice, može se definirati klasu „Putnik“ koja će predstavljati ljude koji voze u automobilu. Putnik ima sve vrste zanimljivih podataka kao što su: Ime, Adresa, Dob i Kreditni_Rejting. Međutim, program *FreeWheeler* ne treba znati nijednu od tih informacija. Možda čak ne treba znati ni jesu li putnici u automobilu.

4.2.2 Građenje hijerarhija s nasljeđivanjem

Nakon što se definiraju glavne klase aplikacije, treba dodati više detalja da bi se prikazale varijacije tih klasa. Primjerice, *FreeWheeler* će trebati klasu „Automobil“ da predstavlja vozilo koje vozi, ali različita vozila imaju različite karakteristike. Toyota Yaris sa 65 kW vozi se drugačije nego Audi A6 od 200 kW. Bilo bi loše kada bi program pretpostavio da bi Yaris mogao dostići brzinu od 0 do 100 km/h u 3,7 sekundi.

Razlike između povezanih klasa mogu se uhvatiti izvođenjem (engl. *deriving*) podređene klase (engl. *child class*) iz nadređene (engl. *parent class*). U prethodnom slučaju mogu se izvesti klase Yaris i A6 iz nadređene klase „Automobil“.

Podređene klase automatski nasljeđuju svojstva, metode i događaje definirane nadređenom klasom. Na primjer, klasa „Automobil“ može definirati metode kao što su: Uklj_Park_Kocnica, Skreni_Lijevo i Naglo_Kocenje. Budući da A6 nasljeđuje iz klase „Automobil“, objekt A6 automatski zna kako treba obaviti tražene zadatke (metode).

Ovo je važan način kako objektni programski jezici ostvaruju ponovnu upotrebu koda. Kod se piše jednom u nadređenoj klasi, a bilo koje podređene klase koriste isti kod. Činjenica da A6 nasljeđuje od klase „Automobil“, također znači da je A6 vrsta automobila. Intuitivno, to ima smisla. U stvarnom životu A6 je automobil, pa bi trebao raditi sve što bilo koji drugi automobil može učiniti.

Budući da instanca podređene klase također pripada nadređenoj klasi, program bi trebao moći tretirati objekt kao da je iz nadređene klase ako bi to bilo od pomoći. U ovom primjeru, to znači da bi program trebao biti u mogućnosti tretirati A6 kao A6 ili kao generički automobil. Program, primjerice, može stvoriti niz objekata klase „Automobil“ i uvrstiti unutra primjerke A6, Yarisa, Accorda ili Golfa. Program bi trebao biti u mogućnosti sve te predmete tretirati kao da su „Automobil“, a da i ne znaju njihove prave klase. Sposobnost tretiranja predmeta kao da su iz druge klase naziva se polimorfizam.

Iz jedne nadređene klase može se izvesti više klasa. Nasuprot tome, većina objektno orijentiranih programskih jezika ne dopušta višestruko nasljeđivanje, pa klasa može imati samo jednu direktno nadređenu klasu. Budući da klasa može imati najviše jednu nadređenu klasu i bilo koji broj podređenih, odnosi između klasa tvore tzv. drvoliku nasljednu hijerarhiju.

Postoji puno načina na koje se mogu izmijeniti osnovni nasljedni odnosi. Podređena klasa, na primjer, može imati dodatna svojstva, metode i događaje (tzv. članovi) koji nisu dostupni u nadređenoj klasi. Podređena klasa, također, može zamijeniti člana nadređene klase s novom verzijom.

Dva glavna načina za izgradnju hijerarhije nasljeđivanja su pročišćavanje i generalizacija.

4.2.3 *Refinement* (pročišćavanje)

Pročišćavanje je postupak razbijanja nadređene klase u više potklasa kako bi se zabilježile neke razlike između objekata u klasi. Kad se iz klase „Automobil“ izvedu klase „A6“, „Accord“ i „Golf“, to je pročišćavanje. Jedna od opasnosti ovog postupka je preveliko pročišćavanje, što se događa kada se nepotrebno pročišćava hijerarhija čineći previše klasa, koje čine stvari kompliciranima i dodatno zbunjuju programere.

U primjeru s automobilima, moguće je unaprijediti većinu modela s različitim opcijama, poput različitih veličina motora, radija, zvučnika, alu felgi, spojlera i grijača sjedala. Moguće je dodati još neke potklase koje bi predstavljale različite boje. Na cestama je nekoliko stotina modela automobila. Rezultirajuća hijerarhija sadržavala bi tisuće (možda milijune) klasa. Očigledno hijerarhija ne bi bila korisna. Ne možete napisati dovoljno koda da biste zapravo mogli koristiti svaku od klasa, a ako ne želite koristiti klasu, zašto biste je kreirali?

Ovdje postoje dva glavna problema. Prvo, klase bilježe podatke koji nisu relevantni za aplikaciju. Aplikacija *FreeWheeler* ne brine koje je boje automobil ili ima li CD mjenjač. Zanimaju je samo vozne karakteristike automobila: prijeđena kilometraža, maksimalno ubrzanje, radijus okretanja i slično. Drugi problem ove hijerarhije je taj što razlike između automobila lako mogu biti predstavljene svojstvima, a ne različitim klasama. Dosad identificirane razlike zapravo su samo različite vrijednosti za ista svojstva. Na primjer, Audi, Toyota i Honda različite su vrijednosti za svojstvo „Proizvođač“. Može se ukloniti cijela jedna razina hijerarhije jednostavnim dodavanjem svojstva „Proizvođač“ u klasu „Automobil“.

Slično tome, model automobila (A6, Yaris i Accord) samo je naziv za određenu vrstu automobila. Ljudi uvijek imaju određena očekivanja na temelju imena (očekuje se da je A6 brži od Yarisa), ali za program *FreeWheeler* to su samo sporedne oznake. Ovakve hijerarhijske probleme moguće je izbjeći ako se usredotoči na razlike u ponašanju između različitih vrsta objekata, umjesto da se gledaju razlike u svojstvima.

Na primjer, kakve su razlike u ponašanju između A6 i Yarisa? A6 ubrzava brže, ali oba automobila mogu ubrzati, samo ne jednako brzo. I dalje se isto ponašaju pri ubrzanju, tako da se ta razlika može predstaviti kao svojstvo ubrzanja u klasi automobila.

Primjer s razlikom u ponašanju je vrsta prijenosa. Da biste ubrzali automobil s automatskim mjenjačem, jednostavno se nagazi na papučicu gasa dok automobil ne dostigne željenu brzinu. Povećanje brzine automobila s ručnim mjenjačem mnogo je složenije, pa je potrebno koristiti papučicu gasa, spojku i ručnu promjenu stupnja prijenosa. Obje vrste vozila ubrzavaju, no detalji o tome kako to rade su različiti.

4.2.4 Generalizacija

Pročišćavanje započinje s jednom klasom i stvara podređene klase koje predstavljaju razlike između objekata. Generalizacija čini suprotno: započinje s nekoliko klasa i kreira nadređenu klasu koja treba predstavljati zajedničke značajke.

Isto kao što se može prekomjerno usavršavati, kako bi se izradila hijerarhija nasljeđivanja koja sadrži tisuće klasa automobila, tako se može iscrpljivati i generalizacijom.

Primjerice, izrada aplikacije za praćenje stanja robe u trgovinama za kućne ljubimce. Definira se klasa „Korisnik“ i klasa „Zaposlenik“. Oni dijele neka svojstva kao što su: Ime, Adresa i Horoskopski_Znak, pa ih se može generalizirati, čineći klasu „Osoba“ koja će sadržavati zajednička svojstva. Dalje, moguće je definirati različite klase ljubimaca, kao što su: Pas, Mačka i Zamorac. Generalizira ih se da bi se dobila klasa „Ljubimci“.

Moguće je ići još dalje, i kreirati klasu „Sisavci“ koja će biti nadređena klasa klasama „Osoba“ i „Ljubimci“. Te dvije klase mogu dijeliti neka svojstva, primjerice, Ime. To ima smisla. Ljudi i kućni ljubimci zaista jesu sisavci. Međutim, malo je vjerojatno da će program ikada iskoristiti tu činjenicu. Teško je zamisliti da program izgradi niz ili popis koji sadrži i zaposlenike i ptice, a zatim ih obrađuje na jedinstveni način. Prema svemu sudeći, program će na različite načine tretirati ljude i kućne ljubimce, tako da ih nije potrebno spajati u jednu hijerarhiju nasljeđivanja.

4.2.5 Kompozicija objekata

Nasljeđivanje je jedan od načina na koji možete ponovno koristiti kod. Podređena klasa nasljeđuje sav kod definiran u njegovoj nadređenoj klasi, tako da ga ne treba ponovo pisati. Drugi način ponovne uporabe koda je kompozicija objekata, tehnika koja koristi postojeće klase za izgradnju složenijih klasa.

Na primjer, definira se klasa „Osoba“ sa svojstvima:

- Ime
- Prezime
- Adresa
- Telefon.

Sada se želi napraviti klasa „Tvrtka“ koja bi trebala sadržavati podatke o osobi za kontakt. Moglo bi se podesiti da klasa „Tvrtka“ nasljeđuje od klase „Osoba“ tako da nasljeđuje svojstva: Ime, Prezime, Adresa i Telefon. To bi dalo mjesta za pohranjivanje podataka o osobi za kontakt, ali to nema smisla. Tvrtka nije vrsta osobe, tako da klasa „Tvrtka“ ne bi trebala nasljeđivati svojstva od klase „Osoba“.

Bolji pristup je davanje klasi „Tvrtka“ novo svojstvo pod nazivom KontaktOsoba. Sada klasa „Tvrtka“ dobiva koristi od koda definiranog Osobom, bez nelogičnosti i moguće zbrke nasljeđivanja od Osobe. Ovaj pristup, također, omogućuje postavljanje više objekata jedne Osobe unutar klase „Tvrtka“. Na primjer, ako se odluči da klasa „Tvrtka“ treba pohranjivati podatke o kontaktima za naplatu i kontaktima za otpremu, može se dodati više objekata Osobe. To se ne može učiniti nasljeđivanjem.

4.2.6 Dizajn baze podataka

Postoji mnogo različitih baza podataka koje se mogu koristiti za izradu aplikacije. Na primjer, specijalizirane vrste baza podataka pohranjuju hijerarhijske podatke, dokumente, grafikone i mreže, parove ključeva / vrijednosti i objekte. Daleko najpopularnija vrsta baza podataka su relacijske baze podataka.

Relacijske baze podataka prirodno su jednostavne. Jednostavne su za korištenje i pružaju dobar skup alata za pretraživanje, kombiniranje podataka iz različitih tablica, sortiranje rezultata i preuređivanje podataka na drugi način.

Relacijska baza podataka pohranjuje srodne podatke u tablice. Svaka tablica sadrži zapise koji, pak, sadrže dijelove podataka koji su povezani. Dijelovi podataka u svakom zapisu nazivaju se polja. Svako polje ima ime i vrstu podataka. Sve vrijednosti u različitim zapisima za određeno polje imaju tu vrstu podataka.

Osnovne karakteristike tablice:

- ne postoje dva jednaka retka
- ne postoje dva jednaka stupca
- redoslijed redaka nije bitan
- redoslijed stupaca nije bitan.

Polje ili više polja (atributa), kojima se može jednoznačno definirati redak (slog) tablice, naziva se primarni ključ. Primarni ključ upotrebljava se za povezivanje tablica i ima dvostruku ulogu: jednoznačno definira retke tablice, a preko njega se ostvaruje i veza s drugim tablicama.

Primarni ključ mora zadovoljavati sljedeće:

- Vrijednost primarnog ključa mora biti jednoznačna.
- Primarni ključ ne može imati vrijednost NULL (ne može biti prazno polje).
- Primarni ključ mora postojati kod kreiranja i spremanja sloga.

Pravila dobrog dizajna tablice su:

- jedinstvenost polja – svako polje u tablici mora predstavljati jedinstveni tip informacije
- primarni ključevi – svaka tablica mora imati primarni ključ koji se sastoji od jednog ili više polja tablice
- funkcionalna ovisnost – vrijednosti stupca s podacima, pridružene svakoj od jedinstvenih vrijednosti primarnog ključa, moraju se odnositi na subjekt tablice i u potpunosti ga opisivati
- nezavisnost polja – mora postojati mogućnost mijenjanja podataka u bilo kojem polju (osim primarnog ključa), a da se pritom ne utječe na podatke u ostalim poljima.

Dizajn baze podataka određuje koje tablice sadrži baza podataka i kako su povezane. Objektno orijentirani dizajn i dizajn baze podataka nije sve što se treba učiniti kako bi se osigurao uspjeh, ali neosporna je činjenica da loši dizajni gotovo uvijek vode k neuspjehu.

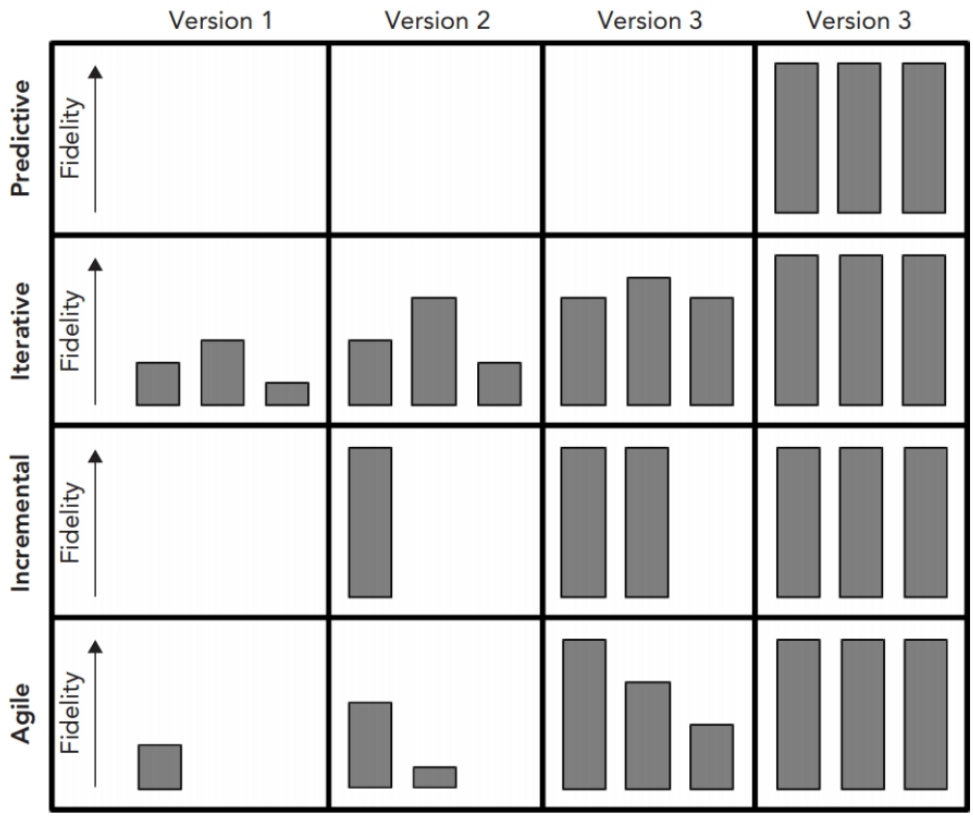
5 Modeli programskog inženjerstva

Procesi programskog inženjerstva obuhvaćaju razvoj programske potpore od specifikacije korisničkih zahtjeva sve do krajnje implementacije, isporuke i održavanja. Zadatak koji pred inženjera postavlja disciplina programskog inženjerstva je taj da on mora prihvatiti sustavni i organizirani pristup procesu izrade softvera. Cjelokupni proces izrade softvera dijeli se na faze i većinom se koristi pristup s njih pet:

- definiranje zahtjeva
- dizajn
- implementacija
- završni skup testiranja
- isporuka i održavanje.

Te faze mogu biti povezane na različite načine, pa onda metode razvoja razlikujemo, načelno, po načinu povezivanja faza.

Na slici 43 dan je usporedni prikaz agilnih, inkrementalnih, iterativnih i prediktivnih modela s obzirom na kompletnost rješenja tijekom vremena (preuzeto iz [3], stranica 287):



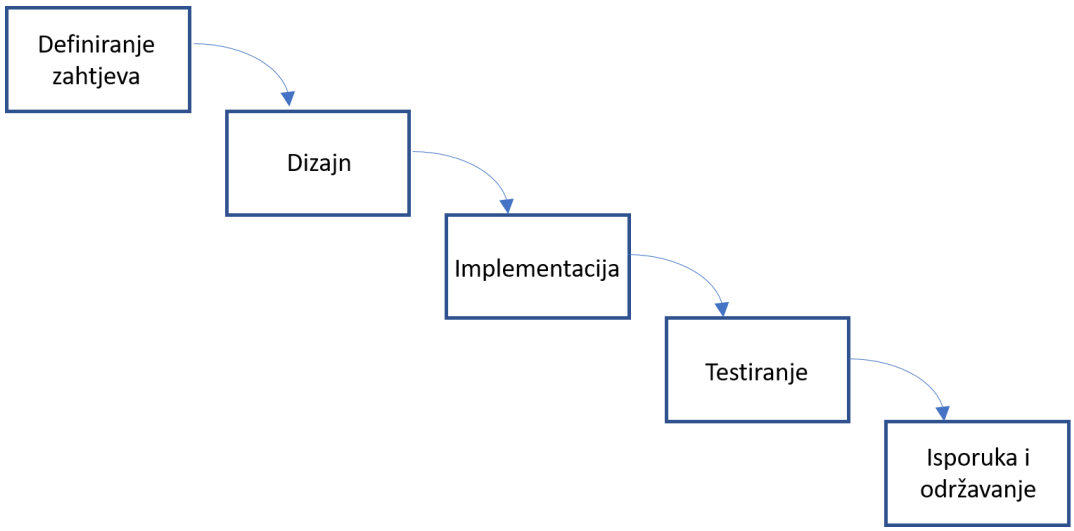
Slika 43: Usporedni prikaz agilnih, inkrementalnih, iterativnih i prediktivnih modela [3]

Kod metoda vođenih planom, glavno svojstvo je da svaka faza razvoja ima svoje preddefinirane rezultate, a iteracije se događaju unutar faza.

S druge strane, glavna je karakteristika skupine metoda, koje su nazvane agilne, da se ishodi procesa razvoja dogovaraju za vrijeme procesa razvoja softvera.

5.1 Vodopadni model

Model s potpuno odvojenim fazama, koji podrazumijeva da se svaka faza u potpunosti završi prije nego što se prijeđe na sljedeću fazu.



Slika 44: Vodopadni model (*Waterfall*)

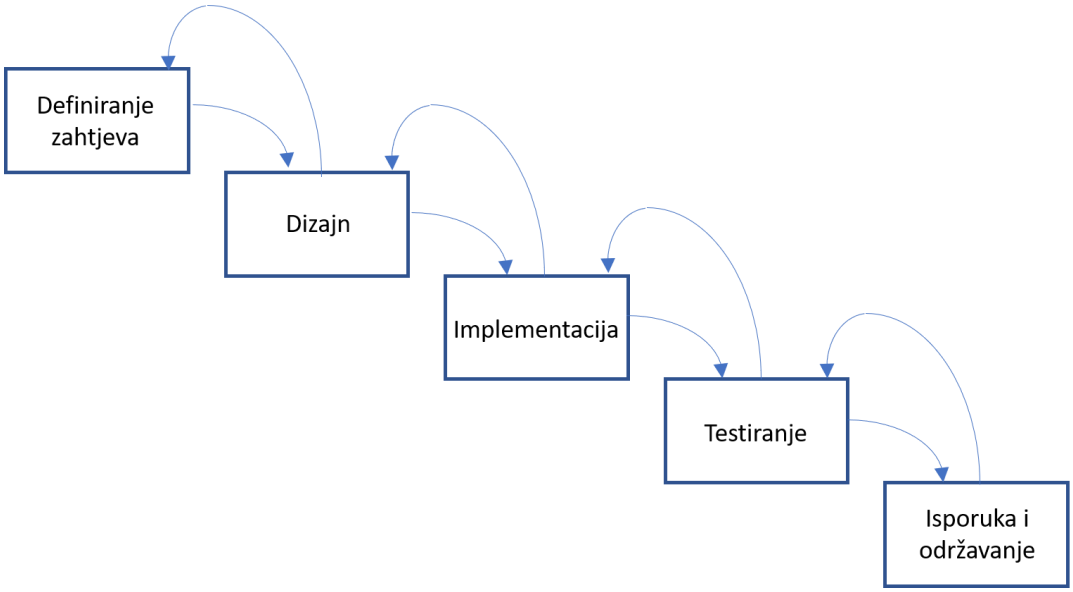
Svih pet prethodno navedenih faza promatraju se kao potpuno odvojene.

Glavni nedostatak modela su teškoće u prihvaćanju promjena tijekom procesa. Faza mora, u načelu, biti gotova prije nego što se prijeđe na sljedeću fazu, te će ovaj model biti prikladan samo ako su zahtjevi dobro razumljivi i promjene ograničene tijekom procesa dizajniranja.

Model vodopada uglavnom se koristi za velike inženjerske projekte sustava, gdje se sustav može razvijati na više fizičkih lokacija. U takvim okolnostima vodopadni model pomaže koordinaciji poslova.

5.2 Vodopadni model s povratnom vezom

Model vodopada s povratnom vezom omogućava pomak unatrag, odnosno povratak na prethodnu fazu. Naime, klasični vodopadni model dobar je samo ako se svaki korak može izvesti savršeno (a da se tijekom razvoja ne mijenjaju zahtjevi). To je vrlo teško izvesti, a budući da model ne dopušta povratak na ranije korake, ako se ne uspije u bilo kojem koraku, svi kasniji koraci bit će pogrešni.



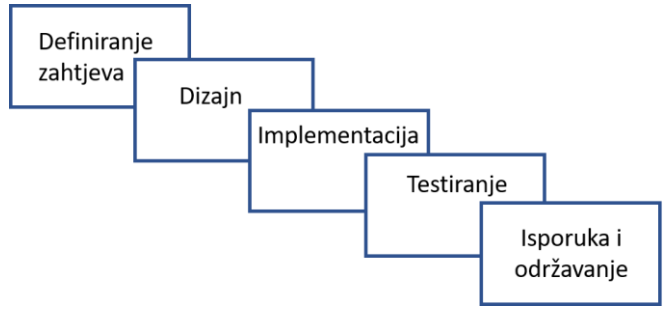
Slika 45: Vodopadni model s povratnom vezom

Kod ovog modela, ako se, primjerice, radi na dizajnu i otkrije se da je došlo do problema u zahtjevima, može se nakratko vratiti zahtjevima i popraviti ih. Međutim, što bi se više trebalo vraćati unatrag, to bi postajalo sve teže. Na primjer, ako se radi na implementaciji i otkrije se problem s pogrešno definiranim zahtjevima, povratak unatrag postaje problem. Teško je preskočiti sigurnosne kopije (dva koraka) da bi se riješio problem. Također, ako se naiđe na problem tijekom održavanja, vjerojatno bi se taj problem trebao tretirati kao zadatak održavanja, umjesto da se vraća natrag u fazu implementacije.

Budući da je kretanje kaskadom teško, mora se i dalje sve dobro predviđati. Moguće je popraviti neke pogreške, ali cilj je, još uvijek, dovršiti svaki korak u potpunosti i učinkovito, prije nego što se krene dalje.

5.3 Vodopadni model s preklapanjem faza

Ovaj je model sličan klasičnom vodopadnom modelu, samo što se dopušta preklapanje.



Slika 46: Vodopadni model s preklapanjem faza

U prvoj fazi projekta (definiranje zahtjeva) neki će zahtjevi biti definirati u potpunosti, a na drugima će se još raditi. U tom trenutku, neki od članova tima mogu započeti s dizajniranjem definiranih značajki, dok drugi nastavljaju raditi na preostalim zahtjevima.

Nešto kasnije, dizajn nekih dijelova aplikacije bit će manje-više gotov, ali dizajn za druge dijelove sustava neće biti. U tom trenutku, neki programeri mogu započeti pisati kod za dizajnirane dijelove sustava, dok drugi nastavljaju raditi na ostalim dizajnerskim zadacima, a možda čak i na preostalim zahtjevima.

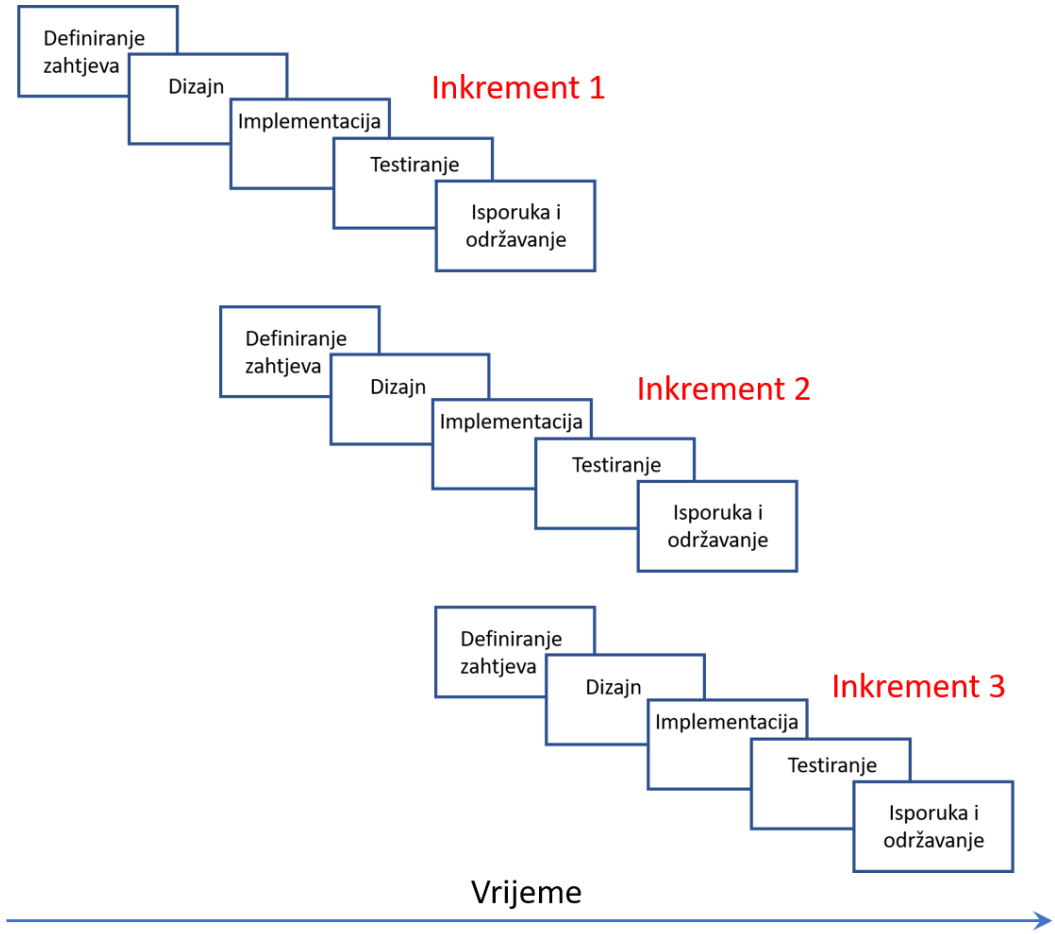
Slično tome, drugi dijelovi razvojnog procesa mogu se preklapati iako vjerojatno postoje ograničenja. Možda se želi odgoditi implementacija dok aplikacija ne bude testirana i potvrđena. Može se čak dopustiti veće preklapanje između faza projekta. Moguće je, na primjer, imati ljude koji rade na zahtjevima, dizajnu, implementaciji i testiranju istovremeno.

Model s preklapanjem faza ima nekoliko prednosti:

- Ljudi s različitim vještinama mogu se usredotočiti na svoje specijalnosti, a da ne čekaju druge. Na primjer, dizajner baze podataka može početi postavljati tablice i indekse, čak i ako zahtjevi korisničkog sučelja nisu gotovi. Ova vrsta preklapanja čini članove tima produktivnijima.
- Omogućuje se da duboko zaronite u određenu temu kako bi se saznalo više o njoj. Primjerice, za vrijeme rada na zahtjevima, moguće je dopustiti brzo dizajniranje i implementaciju prototipa. Tada se istovremeno radi na zahtjevima, dizajnu i implementaciji. Na temelju rezultata prototipa, moguće je bolje definirati zahtjeve.
- U kasnijim fazama je moguće modificirati ranije faze. Ako se tijekom dizajna otkrije da su zahtjevi nemogući ili da su potrebne izmjene, moguće je izvršiti potrebne promjene.

5.4 Inkrementalni vodopadni model

Inkrementalni vodopadni model (ili model s višestrukim vodopadima) koristi niz zasebnih kaskada. Svaka kaskada završava isporukom upotrebljive aplikacije koja se naziva inkrement. Svaki inkrement sadrži više značajki od prethodnog, tako da se postupno gradi finalna aplikacija.



Slika 46: Inkrementalni vodopadni model s preklapanjem

Tijekom svakog inkrementa, dolazi se do boljeg razumijevanja izgleda finalne aplikacije. Nauči se što nije bilo dobro u prethodnom inkrementu. Korisnici će vjerojatno dati i dugi popis novih značajki koje žele dodati. Sve je to od pomoći u pripremi za sljedeći inkrement.

Na slici 46 vidljivo je da se inkrementi preklapaju u dimenziji vremena. Ako se razumije što treba učiniti u sljedećoj iteraciji, ne mora se čekati da se trenutna iteracija u potpunosti dovrši prije nego što se počne s pisanjem novih dokumenata sa zahtjevima. S druge strane,

ako se započne sljedeći inkrement, prije nego što su korisnici imali priliku raditi s trenutnim, neće se moći dobiti toliko povratnih korisničkih informacija.

Moguće je koristiti bilo koju varijaciju vodopada za svaki inkrement.

Ovaj model je u stvari prilagodljiv, jer omogućuje da se ponovno ocijeni smjer na početku svakog novog inkrementa. Međutim, nije sve tako prilagodljivo:

- Može se promijeniti smjer kada se pokrene novi inkrement, ali unutar svakog inkrementa model radi prediktivno.
- Moguće je izvršiti samo male izmjene, dopuštene bilo kojim modelom vodopada koji se koristi za inkrement.
- Činjenica da se radi na prethodnom inkrementu, daje projektu određenu inertnost, koja ograničava količinu promjena koju je moguće dodati u sljedećoj verziji. Na primjer, moguće je dodavati, uklanjati i prilagođavati značajke u novom inkrementu, ali vjerojatno ne bi trebalo radikalno mijenjati korisničko sučelje (to bi više bilo kao potpuno novi projekt, a ne kao inkrementalna promjena trenutne verzije).
- Inkrementi imaju tendenciju postojanja u duljem razdoblju. Primjerice, neki dugotrajni projekt može pokrenuti novi inkrement svake godine ili svake dvije godine. To korisnicima daje lijep, stabilan i predvidljiv raspored novih verzija (za usporedbu, neki adaptivni modeli proizvode nove verzije aplikacija svaki mjesec ili čak svaki tjedan). Dakle, model inkrementalnih vodopada donekle je prilagodljiv, ali uglavnom tijekom dugog razdoblja.

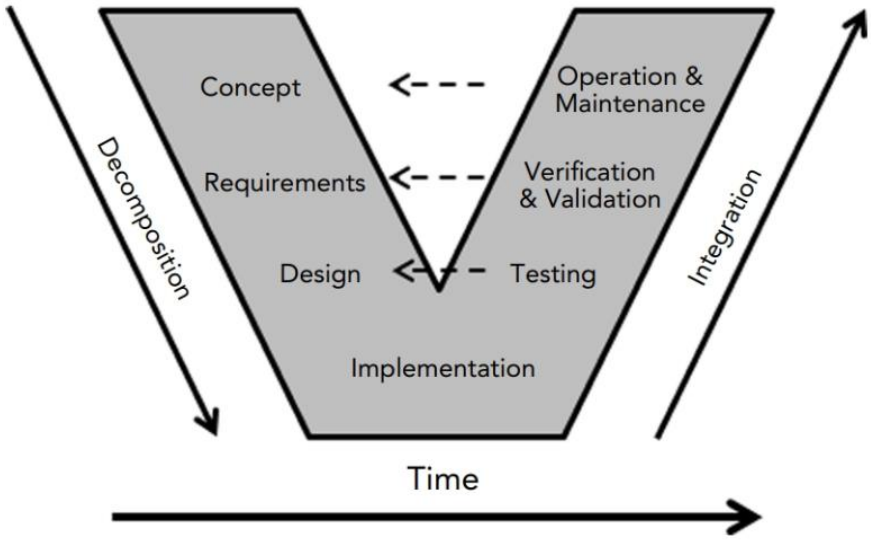
5.5 V-model

V-model je u osnovi vodopadni model savijen u obliku slova V [3].

Zadaci s lijeve strane V-a, raščlanjuju aplikaciju s njezine najviše konceptualne razine na sve detaljnije i detaljnije zadatke. Ovaj postupak razbijanja aplikacije na dijelove, koji mogu biti implementirati, naziva se dekompozicija.

Zadaci s desne strane V-a, promatraju gotovu aplikaciju na sve višim razinama apstrakcije. Na najnižoj razini, testiranje potvrđuje da kod radi. Na sljedećoj razini, verifikacija potvrđuje da aplikacija zadovoljava zahtjeve, a validacija potvrđuje da aplikacija zadovoljava potrebe kupaca. Taj se proces vraćanja do idejnog vrha aplikacije naziva integracija.

Svaki od zadataka s lijeve strane, odgovara zadatku s desne strane, sa sličnom razinom apstrakcije. Na najvišoj razini, početni koncept odgovara načinu rada i održavanju. Na sljedećoj razini, zahtjevi sasvim izravno odgovaraju provjeri i validaciji. Testiranje potvrđuje da je dizajn bio dobar.



Slika 47: V-model; preuzeto iz [3], stranica 276

5.6 Prototipni model

Prototip je pojednostavljeni model koji odrađuje neko ponašanje koje se želi dodatno proučiti. To je program koji oponaša dio aplikacije koja se želi kreirati.

Dvije važne činjenice o prototipovima:

- Ne moraju raditi na isti način kao što će raditi konačna aplikacija.
- Ne trebaju implementirati sve značajke konačne aplikacije.

Primjer, zahtjev za radom sustava za naručivanje proizvoda:

- Unesu se neki parametri, kao što su datumski raspon ili korisnički ID broj.
- Klikne se na gumb „Popis“, kako bi sustav prikazao popis odgovarajućih naloga.
- Dva puta se klikne da bi se vidjeli detalji narudžbe.
- Iz detalja se mogu kliknuti veze da bi se prešlo na detaljne informacije o stavkama u narudžbi.

Tijekom faze prikupljanja zahtjeva, može se kreirati prototip da kupci mogu vidjeti kako će izgledati gotova aplikacija. Kada se klikne na gumb „Popis“, prototip prikazuje unaprijed definirani popis naloga. Kada se dvaput klikne na narudžbu, prototip neće prikazati podatke za tu narudžbu, nego će, umjesto toga, prikazati informacije o odabranoj narudžbi. Na kraju, ako se klikne na poveznicu na narudžbi, mogu se vidjeti informacije o odabranom kupcu. Ovaj prototip ne funkcionira na isti način na koji će raditi gotova aplikacija. Da funkcionira isto, bila bi to stvarna aplikacija, a ne samo prototip. Međutim, omogućuje kupcima da vide kako će aplikacija izgledati. Omogućuje im da kliknu na neke

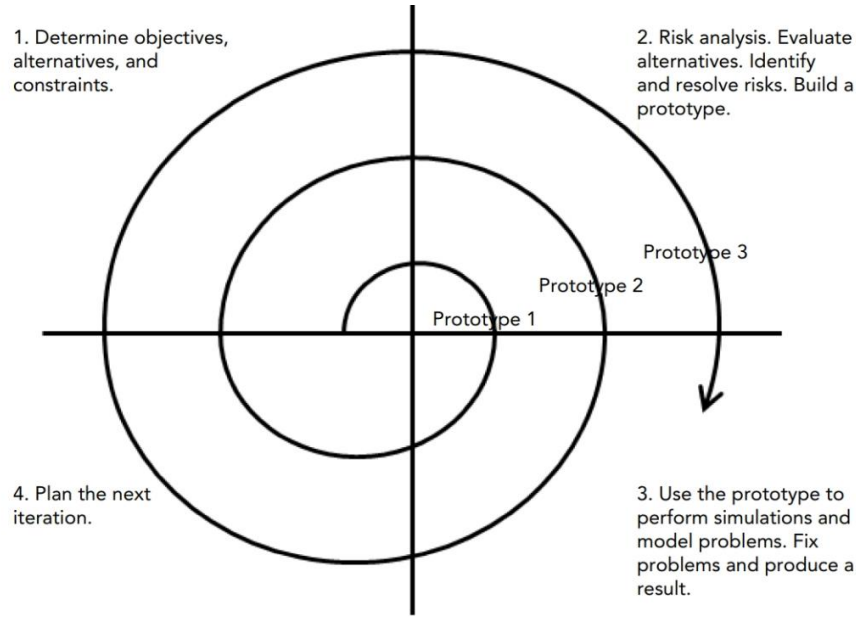
gumbe i poveznice, kako bi mogli isprobati programsku metodu za interakciju s korisnicima.

Nakon što kupci eksperimentiraju s prototipom, mogu dati povratne informacije, kako bi se još mogla poboljšati definicija zahtjeva. Primjerice, kad vide popis narudžbi, mogli bi razmišljati o drugim poljima koja žele prikazati. Nakon što pokušaju dvostrukim klikom otvoriti detaljne podatke o narudžbi, mogu odlučiti da će radije provjeriti potvrdne okvire pored jedne ili više narudžbi, a zatim kliknuti na gumb „Detalji“, za otvaranje stranica s detaljima za sve odabrane narudžbe. Kad pregledaju podatke o kupcu, mogu odlučiti da žele na brz način vidjeti povijest narudžbe tog kupca.

Prototipovi ponekad nemaju ni korisničko sučelje. Na primjer, kreira se aplikacija za naplatu koja će obraditi troškove kupca i generirati račun. Moguće je napisati prototip koji skuplja podatke određenog kupca, izračunava neplaćene troškove i ispisuje račun. To bi programerima omogućilo da prouče kako funkcionira kod za obradu podataka, prije nego što pokušaju učiniti to isto, ali za cjelokupnu bazu podataka o kupcima (koja može biti ogromna).

5.7 Spiralni model

Spiralni model koristi pristup temeljen na riziku, da bi se projektnim timovima pomoglo odlučiti koji razvojni pristup trebaju poduzeti za različite dijelove projekta. Ako se, primjerice, ne razumiju svi zahtjevi, upotrijebi se iterativni pristup za njihovo razvijanje. Opći spiralni pristup, prikazan na slici 48, sastoji se od četiri osnovne faze [3].



Slika 48: Spiralni model; preuzeto iz [3], stranica 291

U prvoj fazi (faza planiranja), određuju se ciljevi tekućeg ciklusa. Definiraju se sve alternative i ograničenja u ciljevima. U drugoj fazi (faza analize rizika), izvodi se analiza rizika, kako bi se utvrdilo koji su najveći faktori rizika koji bi mogli spriječiti postizanje ciljeva ovog ciklusa. Rješavaju se rizici i gradi se prototip kako bi se postigli ciljevi.

U trećoj fazi (faza inženjerstva), koristi se prototip, koji je upravo izgrađen, za procjenu rješenja. Obavljaju se simulacije i modeliraju određeni problemi. Ono što se nauči, postiže se za postizanje izvornih ciljeva. Sada bi se već trebalo moći pokazati nešto konkretno.

U četvrtoj fazi (faza evaluacije), procjenjuje se dosadašnji napredak i uvjerava da su glavni dionici projekta složni da je rješenje ispravno i da bi se projekt trebao nastaviti. Ako dionici odluče da se u nečemu pogriješilo, ide se još jedan krug oko spirale kako bi se riješili svi preostali problemi.

5.8 Agilni model

Agilni je razvoj više skup smjernica, nego što je stvarni razvojni model [3]. Uključuje skup principa za koje se vjeruje da mogu pomoći u bilo kojem razvoju. Budući da je to skup smjernica, postoji mnogo načina na koje je moguće protumačiti pravila.

Jedna od karakteristika agilne metodologije je tzv. samoorganizirajući tim. To je tim koji ima fleksibilnost i ovlasti u pronalaženju vlastitih metoda za postizanje svojih ciljeva. Timovi s različitim razinama autorizacija također mogu postavljati vlastite ciljeve, pratiti njihov napredak, pa čak i birati članove tima.

Članovi samoorganizirajućeg tima su motivirani da rade na zadatku, bez da čekaju da im se zadaci dodijele. Oni preuzimaju odgovornost za svoj rad i prate vlastiti napredak. Ako nastanu problemi, ne boje se zatražiti pomoć, bilo unutar tima ili izvan njega. Povećani osjećaj vlasništva, koji članovi osjećaju, često ih čini entuzijastičnijima i učinkovitijima u pronalaženju dobrih rješenja za probleme. U idealnom slučaju oni će nemilosrdno „napadati“ svaki problem koji im stoji na putu.

Članovi komuniciraju kao tim, kako bi osigurali da grupa radi na svojim ciljevima (koji se mogu postaviti izvan tima, barem na visokoj razini). Da bi učinkovito komunicirali, članovi tima moraju se osjećati sigurno. Moraju vjerovati jedni drugima da bi konstruktivno koristili povratne informacije.

Međutim, samoorganizirajući timovi ne nastaju uvijek spontano. Čak i ako se okupi grupa kompetentnih, motiviranih ljudi i kaže im se da se organiziraju, vjerojatno će ih se morati voditi. Umjesto da ih se vodi, kao što bi to učinio tradicionalni menadžer, može ih se potaknuti da preuzmu inicijativu i onda im ponuditi podršku kad im zatreba.

Agilni projekti koriste česte (ponekad gotovo neprekidne) komunikacije s klijentima kako bi se projekt održao na ispravnoj putanji. Klijenti pregledavaju najnoviju iteraciju, a zatim nude ispravke, prijedloge i zahtjeve za promjenama. Programeri u skladu s tim rade prilagodbe sljedeće iteracije.

Međutim, ako bi svaki kupac stalno razgovarao sa svakim programerom, nitko ne bi imao vremena ništa učiniti. Da bi se poboljšala učinkovitost, većina metoda imenuje primarnog predstavnika klijenta, koji postaje glavna točka kontakta između klijenta i programera.

Ponekad i razvojni tim ima primarni kontakt. Često je to voditelj projekta ili tehnički voditelj. Ako primarni kontakti, s obje strane, imaju ovlasti donositi odluke za svoje grupe, oni mogu iznijeti većinu komunikacija i ostaviti ostale članove tima slobodnima u izvršavanju svojih drugih dužnosti.

Mnoge metode koriste jednu ili više velikih ploča za prikaz trenutnog statusa projekta. Ploče su postavljene tamo gdje ih svi mogu vidjeti, tako da svi točno znaju kako projekt stoji u svakom trenutku. Neki razvojni modeli koriste česte sastanke kako bi programerima olakšali komunikaciju. Primjerice, u Scrumu se tim svakodnevno susreće na dnevnima stojećim sastancima, gdje svi pregledavaju što su radili od posljednjeg sastanka, što planiraju raditi prije sljedećeg sastanka i što im može stati na put. Budući da postoji toliko sastanaka, oni moraju započeti na vrijeme i biti kratki, kako ne bi trošili puno dragocjenog vremena za programere.

Agilni projekti su iterativni i postupni. Iteracije su relativno kratke, s trajanjem od tjedan dana do par mjeseci. Svaka iteracija uključuje svaki razvojni korak, uključujući analizu zahtjeva, dizajn, programiranje, testiranje i provjeru. Svaka iteracija je u osnovi mini projekt sam po sebi.

Budući da je svaka iteracija temeljito istestirana, greške se pronalaze u najkraćem mogućem vremenu. To ih čini lakšima za popravak, jer su najvjerojatnije u kodu trenutne iteracije. To, također, znači da je kod visoke kvalitete, tako da se ne mora gubiti vrijeme popravljajući stare *bugove* u kasnijim iteracijama.

Cilj je svake iteracije imati u potpunosti testiranu aplikaciju, koja ima dovoljno visoku kvalitetu da bi ju se moglo objaviti korisnicima. No, možda se to zapravo i ne želi, ako iteracija nije dodala dovoljno novih značajki projektu. Često je bolje spremiti promjene i napraviti manje većih izdanja svakih nekoliko mjeseci, nego bombardirati korisnike novom verzijom svaki čas.

Nakon svake iteracije, klijent pregledava što se napravilo, kako bi se utvrdilo ide li projekt i dalje u pravom smjeru. Tada se mogu pružiti povratne informacije. Najkritičnija vrsta povratnih informacija je odluka *Go / No Go*. Ako je projekt beznadežno skrenuo s puta, možda je najbolje da ga se otkaže i krene dalje, možda prema novom projektu koji će imati drukčiji pristup.

U agilnim projektima, sav razvoj mora biti kvalitetan. Zbog brzog ciklusa ponavljanja, programeri nemaju vremena tražiti *bugove* koji su ušli u kod prije nekoliko mjeseci. Mnogo je bolje odvojiti malo više vremena za pisanje dobrog i solidnog koda, nego žuriti s programiranjem i onda naknadnim popravljanjem, pa poslije popravljanjem popravaka ...

Ključni problemi s agilnim metodama:

- manjak dokumentacije
- držanje klijenta uključenim u cijeli razvojni proces
- održavanje kontinuiteta razvojnog tima.

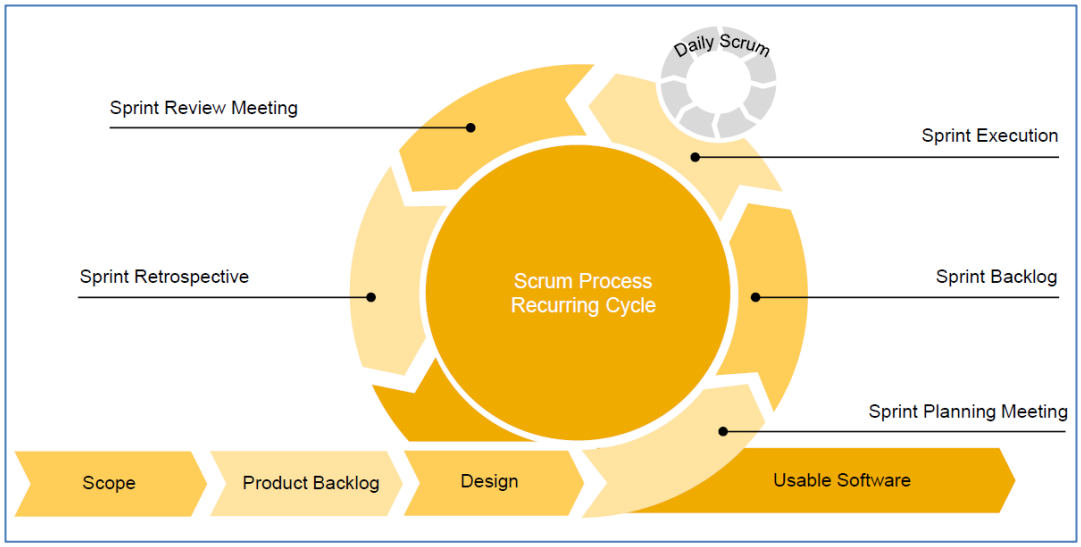
Agilne metode razvoja oslanjaju se na razvojni tim koji zna i razumije što treba učiniti. Za sustave dugog vijeka (engl. *Long-lifetime Systems*), ovo često postaje veliki problem, jer originalni programeri neće zauvijek raditi na sustavu.

5.9 Scrum

Scrum je agilna metoda koja se usredotočuje na upravljanje iterativnim razvojem, a ne na specifične agilne prakse.

U *Scrum* metodi postoje tri faze:

- Početna faza je faza planiranja u kojoj se postavljaju opći ciljevi projekta i dizajnira se softverska arhitektura.
- Nakon toga slijedi niz ciklusa sprinta, gdje svaki ciklus razvija novi inkrement sustava.
- U fazi zatvaranja projekta završava se projekt, dovršava se potrebna dokumentacija i ocjenjuju se naučene lekcije iz projekta.



Slika 49: *Scrum* ciklusi (izvor: SAP SE)

Trajanje sprinta fiksirano je na dva do četiri tjedna. Polazna točka planiranja je tzv. *backlog*, odnosno popis posla koji treba obaviti na projektu. Faza odabira uključuje sav projektni tim koji surađuje s klijentom, na odabiru značajki i funkcionalnosti iz *backloga*, koje se trebaju razviti tijekom sprinta.

Nakon što se postigne dogovor, tim se organizira kako bi se razvio softver. Tijekom ove faze, tim je potpuno izoliran, a sva komunikacija usmjerena je kroz tzv. *Scrum mastere*.

Uloga *Scrum mastera* je zaštititi razvojni tim od vanjskih ometanja. Na kraju sprinta, učinjeni se rad pregledava i prezentira dionicima. Tada započinje sljedeći ciklus sprinta.

Scrum dnevni sastanak održava se na početku svakog dana. Tijekom sastanka, svi članovi tima dijele informacije, opisuju svoj napredak od prethodnog dana, probleme koji su nastali i planove za sljedeći dan. To znači da svi u timu znaju što se događa i, ako se pojave problemi, mogu ponovno planirati kratkoročni rad kako bi se mogli nositi s njima.

Scrum sastanci trebaju biti kratki i usredotočeni. Da bi odvratili članove tima od sudjelovanja u dugim raspravama, sastanci su stojeći. Pregledava se popis stavaka sprinta i dovršeni predmeti uklanjaju se s popisa. Novi elementi mogu se dodati u popis ako se pojave nove informacije. Tim tada odlučuje tko bi trebao raditi na sprinterskim stavkama tog dana.

Idealni *Scrum* tim ima od 5 do 8 članova.

Pet najvećih koristi od *Scrum* metode [6]:

- Cijeli proizvod (aplikacija) rastavljen je na skup dijelova razumljivih svim dionicima.
- Nestabilni zahtjevi ne sprečavaju napredak.
- Klijent ima pravovremeni uvid u inkremente, imajući tako jasnu informaciju o tome kako cijeli projekt napreduje.
- Timska komunikacija je unaprijeđena, jer svi članovi tima imaju uvid u sve.
- Uspostavljeno je povjerenje između klijenta i razvojnog tima.

5.10 Problematika velikih sustava

Veliki sustavi su obično zbirka zasebnih, komunikacijskih sustava, gdje zasebni timovi razvijaju svoje dijelove velikog sustava. Ti timovi često rade na različitim mjestima, ponekad u različitim vremenskim zonama.

Veliki sustavi uključuju i komunikaciju s većim brojem već postojećih sustava. Uvijek se radi o mnoštvu zahtjeva za komunikacijom i zato oni nisu fleksibilni niti pogodni za inkrementalni razvoj.

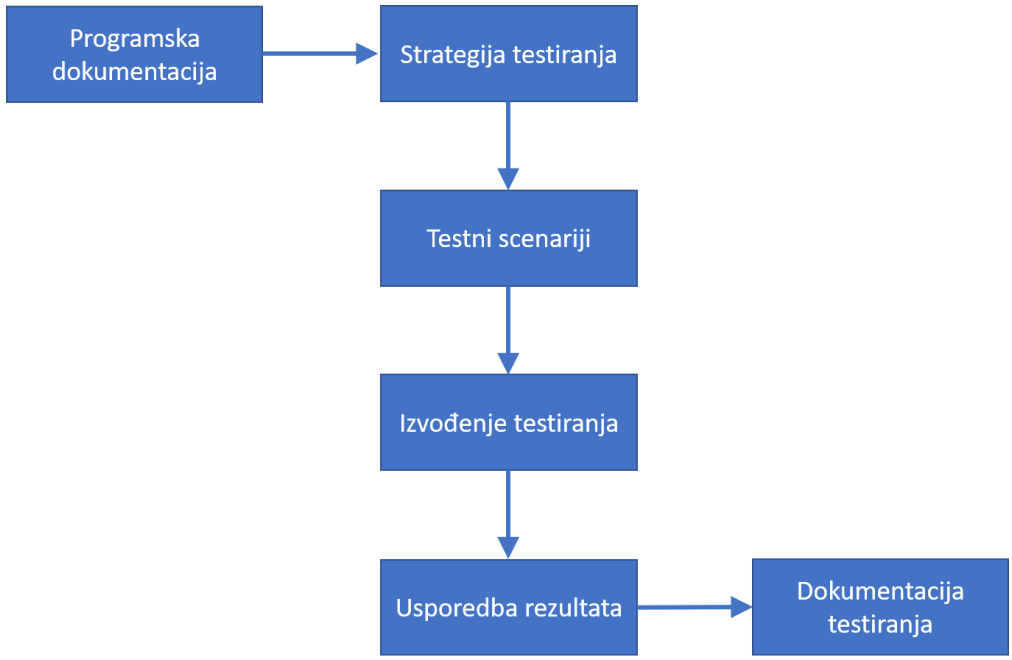
Karakteristike slučaja kad je nekoliko sustava integrirano u veliki sustav:

- Značajan dio razvoja odnosi se na konfiguraciju sustava, a ne na razvoj originalnog koda.
- Veliki sustavi i njihovi razvojni procesi često su ograničeni vanjskim pravilima i propisima, koji ograničavaju način na koji se mogu razvijati.
- Veliki sustavi imaju dugo vrijeme nabave i razvoja; teško je održavati skladne timove koji znaju detalje sustava u cijelom tom razdoblju, jer ljudi neminovno prelaze na druge poslove i projekte.
- Veliki sustavi obično imaju raznolik skup dionika i praktički ih je nemoguće sve uključiti u razvojni proces.

6 Testiranje softvera

Aksiom programskog inženjerstva je da svi ozbiljniji programi sadrže *bugove* [3]. S tim se treba pomiriti i vidjeti što se može učiniti da se otkrije što više velikih *bugova*, onih koji bi ometali ili sprečavali normalan rad aplikacije, odnosno samog poslovanja. Da bi se otkrilo što više *bugova* u softveru, provodi se testiranje.

Proces testiranja ide na način da se, na osnovi definirane strategije testiranja, provodi usporedba očekivanih i testnih rezultata, te se, pomoću definiranih kriterija prihvaćanja, donose zaključci o tome radi li određena funkcionalnost kako treba ili su potrebne dodatne akcije (dorade, korekcije).



Slika 50: Princip testiranja

Testiranje služi za promatranje ponašanja sustava u određenim okolnostima i s određenim ulaznim podacima, da se vidi radi li program ono što bi trebao raditi.

U sklopu testiranja kreiraju se scenariji, gdje se jasno navodi što se treba testirati. Svaka vrsta testiranja (jedinično, funkcionalno, integralno, korisničko i sl.) ima svoja pravila i svoj popis scenarija. Testiranje vodi osoba imenovana za voditelja testiranja.

Zaključivanje o ispravnosti rada aplikacije je jednostavno:

- Test je prošao ako je ponašanje sustava u skladu s očekivanim. Test nije prošao ako ponašanje sustava nije u skladu s očekivanim.
- Ako je program načinio upravo ono što se od njega očekuje, to samo znači da program radi dobro za one ulazne parametre koji su korišteni. Ako su ulazni

parametri dovoljno reprezentativni, može se pretpostaviti da će program raditi dobro u svim slučajevima.

- Ako test nije prošao, u softveru postoje *bugovi* koji se moraju riješiti.

Testiranje se radi prema standardima. Jedan od korištenih, ali i osporavanih, zbog navodne prestrukturiranosti testnog modela, je ISO/IEC/IEEE 29119 s pet dijelova:

- *Part 1: Concepts and definitions*
- *Part 2: Test processes*
- *Part 3: Test documentation*
- *Part 4: Test techniques*
- *Part 5: Keyword-driven testing*.

Navedena norma pruža smjernice za provedbu testiranja u čitavom životnom ciklusu razvoja. Definirani su koncepti i pojmovi, procesi testiranja, testna dokumentacija i tehnike testiranja.

Tim za testiranje obično se sastoji od sljedećih uloga:

- voditelj testiranja – upravlja procesom testiranja i potrebnim resursima, koordinira pripremu scenarija i testnih slučajeva
- analitičar testiranja – radi analizu poslovnih zahtjeva i funkcijskih specifikacija, priprema plan testiranja
- voditelj kvalitete testiranja – definira standarde testiranja koji se moraju provoditi, prati usklađenost provedbe testiranja i predviđenih scenarija, kontrolira kvalitetu ispitne dokumentacije
- testeri – pripremaju podatke i provode ispitivanja.

Strategija testiranja ima za cilj demonstraciju ispravnosti rada softvera:

- Za posebno rađeni softver, to znači da mora postojati bar jedan test za svaki dokumentirani zahtjev kupca.
- Za standardni (generički) softver, to znači da se testiranje mora provesti za svaku funkcionalnost sustava i, dodatno, za kombinacije pojedinih funkcionalnosti, u skladu s poslovnim slučajem.

Strategija testiranja sastoji se od testnih scenarija, posebno pripremljenih za svaku od spomenutih vrsta testiranja. Scenariji bi trebali u dovoljno detaljnom opsegu opisivati poslovne slučajeve. Svaki testni scenarij može zahtijevati više iteracija, tako da se postigne nekoliko identičnih provjera za različite ulazne podatke. U sklopu scenarija je definirana ideja poslovnog slučaja, i navedeni su pojedinačni koraci, koje korisnik (tester) mora načiniti da bi se poslovni proces ostvario.

Ako se testiranje promatra kao proces, onda proces započinje kreiranjem popisa željenih scenarija. Nastavlja se izradom (opisivanjem) samih scenarija i pripremom potrebnih matičnih podataka. Paralelno se priprema plan testiranja, s definiranim vremenima i izvođačima (testerima). U slučajevima velikih projekata, moguće je imati nekoliko stotina scenarija, više desetaka izvođača, a sve skupa može trajati i nekoliko mjeseci. Takva velika testiranja dodatno se usporavaju izradom testne dokumentacije, ali ona se pokazala neophodnom prilikom kasnijih revizija rada sustava.

6.1 Vrste testiranja

Testiranje softvera općenito se može podijeliti na nekoliko vrsta:

- jedinično testiranje (komponente se testiraju pojedinačno, svaka za sebe)
- testiranje sučelja
- integralno testiranje (testiranje cijelog sustava, uključivši i sučelja)
- korisničko testiranje (korisnici sami testiraju sustav da se uvjere da su dobili točno ono što su tražili)
- volumno testiranje
- regresijsko testiranje.

6.1.1 Jedinično testiranje

U jediničnom testiranju provjerava se ispravnost pojedinih dijelova (samostalnih jedinica) sustava. Samostalna jedinica je dio programskog koda, koji se poziva izvan same jedinice i koji može pozvati druge samostalne programske jedinice. Ovo se testiranje provodi uvijek od strane implementatora, prvenstveno samih programera.

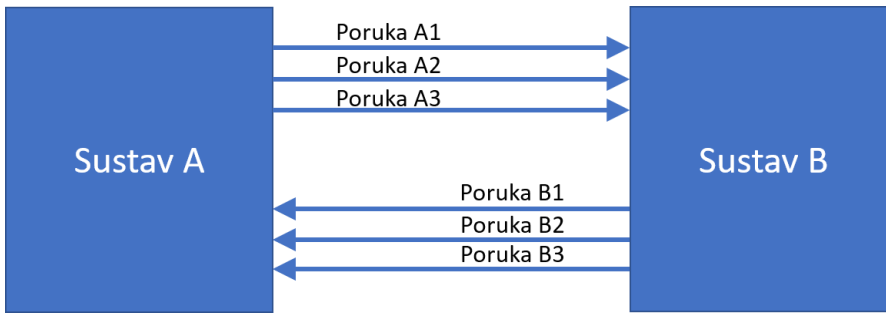
Jedinično testiranje zasniva se na jednostavnom principu [6]:

- Ako se testirana jedinična funkcionalnost ponaša prema očekivanju za skup ulaza koji dijele neke zajedničke karakteristike, onda će se ta jedinična funkcionalnost ponašati na isti način i za neki veći skup, čiji članovi dijele iste karakteristike.

Kod ovog načina testiranja, sve pronađene greške odnose se samo na testiranu jedinicu, te ih je, zbog toga, jednostavnije ispraviti. Pronalazak grešaka u ranoj fazi implementacije je, ujedno, i glavna prednost jediničnog testiranja.

6.1.2 Testiranje sučelja

Termin „sučelje“ odnosi se na komuniciranje dvaju odvojenih sustava, pri čemu svaki sustav može slati i primati poruke. Prilikom testiranja sučelja, mora se provjeriti ispravnost odlaznih i dolaznih poruka.



Slika 51: Sučelje (*Interface*) s odlaznim (*Outgoing*) i dolaznim (*Ingoing*) porukama

Sučelje između sustava A i B na slici 51 sastoji se od šest poruka:

- Poruke A1, A2 i A3 su odlazne poruke za sustav A, a dolazne za sustav B.
- Poruke B1, B2 i B3 su odlazne poruke za sustav B, a dolazne za sustav A.

Kao primjer se može obraditi slučaj proizvodne tvrtke, koja za cjelokupno poslovanje koristi globalni IS, a za detaljno praćenje izvršenja proizvodnje, ima razvijen mali lokalni softver koji komunicira s radnim centrima. Za taj mali softver obično se koristi ime MES (*Manufacturing Execution System*), pa će se termin MES koristiti u ovom primjeru.

Dakle, sustav A u ovom primjeru je globalni IS. Sustav B je lokalni MES, koji dobiva upute od globalnog IS-a, na osnovi kojih se treba nešto proizvesti. Nakon izvršene proizvodnje, MES treba poslati poruku u IS o načinjenim knjiženjima.

Poruke koje sustav A (IS) putem sučelja šalje u sustav B (MES):

- poruka A1 – kreirani matični podaci materijala (sirovine, ambalaža, proizvod)
- poruka A2 – kreirane sastavnice za proizvode
- poruka A3 – kreirani proizvodni nalozi.

Poruke koje sustav B (MES) putem sučelja šalje u sustav A (IS):

- poruka B1 – proizvod koji je proizveden po određenom proizvodnom nalogu
- poruka B2 – materijal utrošen na proizvodni nalog
- poruka B3 – vrijeme utrošeno na proizvodni nalog.

Za svaku od ovih šest poruka, treba pažljivo odabrati parametre (polja) koje će se koristiti, tako da se u porukama ne izostave neki bitni podaci, ali i da se ne prenose suvišne informacije.

U slučaju redovne proizvodnje, globalni IS svakodnevno šalje poruke A3 prema MES-u, da bi proizvodnja znala što se treba proizvoditi. Parametri (polja) koji se trebaju nalaziti unutar poruke A3:

- broj proizvodnog naloga
- šifra proizvoda koji se treba proizvesti
- datum proizvodnje
- šifra radnog centra na kojem se treba načiniti proizvodnja
- šifre svih materijala koji se trebaju utrošiti u proizvodnji
- količine svih materijala koji se trebaju utrošiti u proizvodnji
- mjerne jedinice za sve materijale koji se trebaju utrošiti u proizvodnji
- skladišna lokacija iz koje se treba uzeti materijal za proizvodnju.

Nakon što MES primi poruku A2, proizvodni radnici će, na određenom radnom centru, odmah vidjeti što se, kada, u kojoj količini i po kojem referentnom dokumentu treba proizvesti, i koje materijale (sirovine) trebaju imati na raspolaganju za proizvodnju.

Nakon što je dio proizvodnje gotov, MES šalje tri poruke prema globalnom IS-u (B1, B2 i B3) sa svim potrebnim detaljima, da bi svi zainteresirani znali što se zbiva u proizvodnji.

Prodaja će se naročito zanimati za rezultate poruke B1, jer moraju u svakom trenutku znati što se sve proizvelo, odnosno što je raspoloživo na skladištu za isporuke kupcima.

Parametri (polja) koji se trebaju nalaziti unutar poruke B1:

- broj proizvodnog naloga
- datum knjiženja (datum proizvodnje)
- šifra proizvoda
- proizvedena količina
- mjerna jedinica
- skladišna lokacija u koju je zaprimljen proizvod
- broj serije / šarže proizvoda.

Primjer je jednostavan, ali nikako nije bezazlen. Svaki detalj treba dobro pripremiti i pažljivo istestirati, provjeravajući statuse (i djelovanje) poruka u obama sustavima.

Smjernice za testiranje sučelja [6]:

- Testove dizajnirati tako da parametri za pozvanu akciju budu na krajnjim granicama njihovih raspona.
- Dizajnirati testove koji će namjerno uzrokovati greške u narednim komponentama dolaznog sustava.
- Obavezno treba koristiti stres-test.
- Promijeniti redoslijed aktiviranja komponenata.

Tipične greške kod sučelja:

- pogrešno korištenje – pozivajuća komponenta poziva pogrešne parametre druge komponente
- pogrešno razumijevanje – pozivajuća komponenta ima pogrešnu pretpostavku o ponašanju pozvane komponente
- vremenska greška – pozvana i pozivajuća komponenta rade različitim brzinama što uzrokuje pristup zastarjelim podacima.

6.1.3 Integralno testiranje

Integralno testiranje je testiranje funkcionalnosti cjelokupnog sustava, odnosno radi se o testiranju sustava u cjelini, a ne više pojedinačnih dijelova (svi pojednačni dijelovi su već istestirani).

Ciljevi su otkrivanje pogrešaka u implementaciji sustava i pružanje uvjerljivih dokaza da je sustav prikladan za predviđenu svrhu. Mora se provjeriti da nema neočekivanih interakcija između pojedinih funkcionalnosti sustava. Testiranje može uključiti i provjeru odzivnosti, pouzdanosti i sigurnosti sustava.

Testiranje se fokusira na četiri stvari:

- Ispitivanje o tome postoje li neočekivane i neželjene interakcije između značajki u nekom sustavu.

- Ispitivanje kako bi se otkrilo funkcioniraju li značajke sustava učinkovito kako bi podržali ono što korisnici doista žele učiniti sa sustavom.
- Ispitivanje sustava da bi se bilo sigurnim da djeluje na očekivani način u različitim okruženjima u kojima će se koristiti.
- Ispitivanje odzivnosti, propusnosti, sigurnosti i ostalih atributa kvalitete sustava.

Najbolji način za sustavno testiranje sustava je započeti s nizom scenarija koji opisuju moguće upotrebe sustava, a zatim proći kroz te scenarije svaki put kada se stvori nova verzija sustava.

Koristeći scenarij, treba identificirati skup mogućih puteva *end-to-end*, koji korisnici mogu slijediti prilikom korištenja sustava. Put od kraja do kraja slijed je postupaka od početka korištenja sustava za zadatak, do završetka zadatka.

6.1.4 Korisničko testiranje

Cilj korisničkog testiranja je utvrditi ispunjava li gotova aplikacija kupčeve zahtjeve. Korisnik (ili drugi predstavnik kupca) prolazi kroz sve korisničke slučajeve koji su prepoznati u fazi prikupljanja zahtjeva, kako bi se osiguralo da sve funkcionira kako je traženo. Ovo se testiranje nekad zove i test prihvaćanja (engl. *acceptance testing*). Treba imati na umu da su se zahtjevi mogli promijeniti nakon početne faze. U tom se slučaju potvrđuje da aplikacija zadovoljava revidirane zahtjeve.

Ispitivanje prihvatljivosti može dugo potrajati. Prilično jednostavna aplikacija možda će trebati samo nekoliko načina korištenja, ali velika, složena aplikacija s detaljnim potrebama, može ih imati na desetine ili na stotine. U tom bi slučaju mogli proći dani ili čak tjedni.

Jedna pogreška koju programeri ponekad čine je čekanje da aplikacija bude gotova, prije nego što započnu test prihvaćanja. Naime, ako se test prihvaćanja radi kad korisnik prvi put vidi aplikaciju, to obično znači da će se pojaviti problemi. Može se ustanoviti da se kupčeva i programerska interpretacija prilično razlikuju. Ili, pak, korisnici mogu zaključiti (nakon što su vidjeli aplikaciju) da je ono što im je potrebno, drukčije od onoga što su mislili da im je potrebno tijekom prikupljanja zahtjeva.

6.1.5 Regresijsko testiranje

Regresijsko testiranje koristi se za provjeru sustava, tj. kako bi se provjerilo da promjene u aplikaciji nisu „pokvarile“ prethodno dobar radni kod. Svi se izabrani scenariji ponavljaju svaki put kad se promijeni program. To znači da je, u slučaju ručnog testiranja, regresijsko testiranje skupo, ali uz automatizaciju testiranja, ono postaje jednostavnije i lakše. Uvjet je da se scenariji podese dovoljno dobro i jednostavno, te da se koriste podaci koji, sami za sebe, neće uzrokovati probleme (kod ručnog testiranja to se prepozna i korigira, ali kod automatskog testiranja to uzrokuje loš rezultat).

Testovi se moraju pokazati „uspješnima“ prije nego što se izmjena pusti u rad.

6.1.6 Testiranje performansi sustava

Ovim testiranjem radi se provjera, kako bi se potvrdilo da softver radi brzo i može podnijeti očekivano opterećenje koje korisnici očekuju:

- Treba pokazati da je odziv sustava prihvatljiv za krajnje korisnike.
- Treba pokazati da novi sustav može podnijeti povećana opterećenja.

Testiranje performansi obično uključuje niz iteracija, gdje se opterećenje neprestano povećava sve dok performanse sustava ne postanu neprihvatljive. Jedan oblik ovog testiranja je tzv. stres-test, gdje se sustav namjerno preopterećuje, kako bi se ustanovilo njegovo ponašanje u takvim slučajevima.

6.2 Plan testiranja osnovnog proizvodnog modela

Za sve važnije poslovne procese, treba osmisliti poslovni scenarij, definirati matične podatke, imenovati testere, isplanirati vrijeme testiranja i predvidjeti rezultate.

Primjer poslovnih procesa koji bi se trebali obavezno testirati, u sklopu osnovnog proizvodnog modela:

- kreiranje matičnih podataka
  - kreiranje kupaca
  - kreiranje dobavljača
  - kreiranje materijala / proizvoda
  - kreiranje sastavnice
  - kreiranje materijala / sirovine
- kreiranje narudžbenice
- zaprimanje sirovine od dobavljača
- plaćanje dobavljaču
- preskladištenje materijala
- kreiranje proizvodne cijene
- kreiranje proizvodnog naloga
- knjiženje utroška sirovina na proizvodni nalog
- knjiženje utroška vremena na proizvodni nalog
- knjiženje prijema proizvoda iz proizvodnog naloga na skladište
- kreiranje prodajnog naloga
- kreiranje isporuke
- fakturiranje kupcu
- izvještaji o materijalnim kretanjima
- izvještaji o trenutnom statusu osnovnih dokumenata
- izvještaj o financijskoj situaciji prodaje.

Naravno, da bi sve išlo kako treba, na vrijeme se priprema testna dokumentacija, odnosno formulari, u koje će se zapisivati svi relevantni detalji testiranja.

7 Isporuka i održavanje

Isporuka je proces stavljanja gotove aplikacije ili velikog sustava u korisničke ruke. Isporuka može biti užitak ili noćna mora, ovisno o tome kako je aplikacija napravljena, kako su korisnici pripremljeni i kako je cijeli proces isporuke isplaniran. Kod planiranja je naročito potrebno obratiti pažnju na stvari koje mogu „poći naopako“ i pripremiti alternativne korake.

Općenito, što je veći projekt, veća je mogućnost grešaka i problema kod isporuke. Svi autori koji analiziraju ovu temu kažu: *„Something always goes wrong*“. Hoće li mreža raditi? Hoće li ključni korisnici biti na radnom mjestu? Jesu li ključne točke pokrivene fizičkim prisustvom? Jesu li videolinkovi isprobani? Jesu li korisnička imena i šifre raspodijeljena na vrijeme? Jesu li autorizacije korisnika dobro podešene? Takvih pitanja ima mnogo i sve ih treba analizirati. To znači dobro se pripremiti. Što je veći projekt, potrebna je veća priprema.

Nakon uspješne isporuke slijedi faza održavanja. Vrlo je korisno ostaviti bar nekoliko članova projektnog tima u timu za održavanje. U slučaju da to nije moguće (a obično nije), onda je neophodno načiniti kvalitetnu primopredaju sustava novom timu.

Naslijeđeni sustavi (*Legacy*) koji ostaju raditi zajedno s novim, posebna su priča. To su stariji sustavi koji se oslanjaju na jezike i tehnologiju koja se možda više ne koristi. Naslijeđeni softver može ovisiti o starom hardveru i može imati neke pridružene naslijeđene procese i postupke.

Legacy sustave skupo je mijenjati iz više razloga:

- nekonzistentan stil programiranja
- korištenje starih programskih jezika
- neadekvatna dokumentacija
- degradacija strukture sustava
- razne optimizacije su načinile program nerazumljivim
- greške, dupliciranje i nekonzistentnost podataka.

7.1 Cutover

Cutover (nema pogodne hrvatske riječi) je proces prelaska korisnika na novu aplikaciju. Stara aplikacija se napušta i počinje se koristiti nova.

Kod manjih aplikacija, moguće je samo staviti novu aplikaciju na internet i dati korisnicima da je koriste. Kod većih su ipak potrebne određene pripreme:

- prebacivanje podataka o svim materijalima, proizvodima, uslugama itd.
- prebacivanje podataka o poslovnim partnerima (dobavljači i kupci)
- prebacivanje trenutnih financijskih stanja po kontima glavne knjige
- prebacivanja trenutnih dugovanja i potraživanja

- prebacivanja aktivnih narudžbenica i prodajnih naloga
- prebacivanje podataka o proizvodnji u tijeku
- prebacivanja trenutnog stanja zaliha itd.

Pripreme za *Cutover* znaju kod velikih projekata trajati mjesecima. Nekoliko mjeseci prije datuma isporuke, imenuje se tzv. voditelj *Cutover* procesa, koji onda planira sve potrebne pripreme i kontrolira provođenje istih (jesu li svi treninzi odrađeni, jesu li sva testiranja uspješno završena itd.). Najvažniji njegov zadatak je izrada detaljnog plana po aktivnostima i imenima za tih nekoliko dana koliko traje sam *Cutover*.

Lista aktivnosti može imati nekoliko stotina ili čak nekoliko tisuća stavaka. Za svaku stavku definira se odgovorna osoba, kao i početak i kraj same aktivnosti.

Tri osnovna koraka za uspješan *Cutover* su:

- priprema plana
- prepoznavanje grešaka i problema
- raditi prema planu, a probleme rješavati usput, kako se pojavljuju.

Najčešći uzroci problema, prema [3]:

- pretpostavka da će sve raditi
- nepostojanje plana B
- planiranje nedovoljnog vremena
- ne znati u kojem trenutku treba odustati ako se dogodi veliki problem
- preskakanje probe
- instaliranje previše stvari odjednom
- korištenje nestabilne okoline
- prerano odustajanje.

7.2 Održavanje sustava

Održavanje je važna aktivnost jer ima veliki udio u ukupnoj cijeni projekta. Kod vrlo velikih projekata, održavanje može doseći i do 75 % ukupne cijene projekta [3].

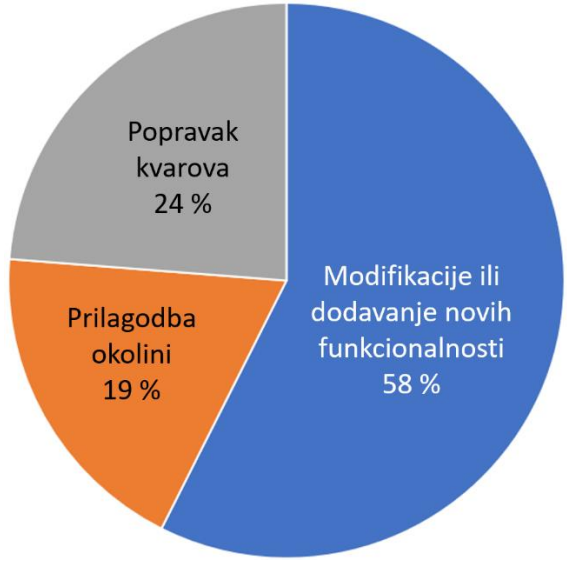
Dva su razloga zašto je održavanje tako skupo:

- aplikacije često ostaju „na životu“ mnogu dulje nego što je to bilo zamišljeno (BASIC i COBOL programeri su još uvijek aktivni)
- često je lakše (u istom programu) napisati novi kod, nego popravljati stari (a onda slijede testiranja samog novog koda, ali i ostatka aplikacije, da bi se bilo sigurnim da novi kod nije poremetio rad ostalih funkcionalnosti).

Prema [6], poslove na održavanju moguće je podijeliti na tri skupine:

- popravak kvarova
- prilagodba okolini
- modifikacije.

Njihov međusobni udio u ukupnim troškovima održavanja prikazan je na slici 52.



Slika 52: Međusobni udio poslova održavanja u ukupnim troškovima; prema [6]

Dodatni je razlog velikih troškova održavanja to što je skuplje dodavati nove funkcionalnosti u fazi održavanja, nego što bi to bilo tijekom razvoja (prije isporuke):

- Novi tim mora dobro razumjeti program koji se održava, da bi se mogle dodavati nove funkcionalnosti.
- Odvajanje razvojnog tima od tima za održavanje, znači da razvojni tim neće biti motiviran da stvara softver koji bi bio lako održiv.
- Poslovi na održavanju aplikacija nisu popularni, a timovi su često neiskusni i s limitiranim znanjem konkretnog područja.
- Kako program stari, tako stari i njegova struktura i postaje teža za promjene.

7.3 Vođenje promjena

Najbolji način za vođenje promjena je njihovo predviđanje i planiranje. To, međutim, znači istinsko razumijevanje rada cijele aplikacije i poveznice s okolnim sustavima. U novije vrijeme, sve je više sustava međusobno čvrsto povezano, i onda promjene u jednom sustavu, neupitno traže promjene u skoro svim drugim sustavima.

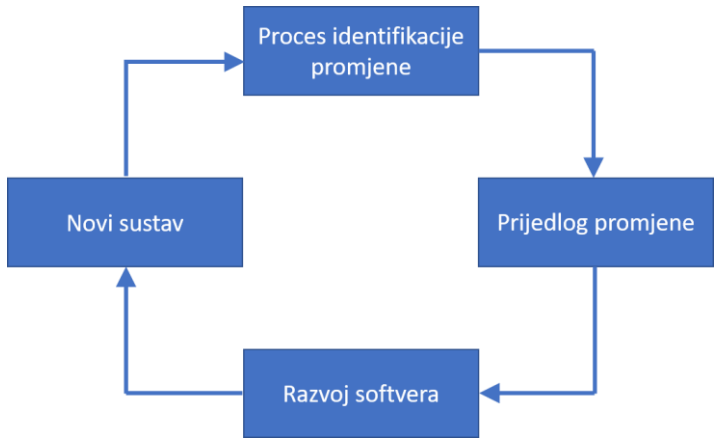
Faktori koji povećavaju složenost promjene:

- broj i složenost sučelja
- broj labilnih (nepostojanih) poslovnih zahtjeva
- broj poslovnih procesa u kojima se koristi predmetni sustav.

Promjene u softveru neizbježne su iz više razloga:

- kad se softver koristi, stalno se pojavljuju novi zahtjevi
- poslovna se okolina stalno mijenja
- greške se moraju otkloniti
- dodavanje nove opreme u sustav
- potreba za poboljšanjem performansi i pouzdanosti sustava.

Može se reći da je, za sve organizacije, provođenje i upravljanje promjenama postojećih softverskih sustava, jedan od ključnih problema [6]. Proces identifikacije i praćenja potrebnih promjena, kao kružni proces prikazan je na slici 53.



Slika 53: Ciklus identifikacije potrebnih promjena [6]

Uvijek će postojati hitne (urgentne) promjene koje će se možda morati implementirati po posebnim procedurama, bez prolaženja po svim prethodno navedenim koracima:

- Kritična greška u sustavu bez čijeg popravka poslovanje ne može ići.
- Poslovna promjena koja zahtijeva vrlo brzu akciju.

8 Kategorizacija informacijskih sustava

Informacijski sustavi sve dublje prožimaju moderne organizacije te je razumijevanje njihove klasifikacije i organizacije, preduvjet za upravljanje infrastrukturom.

Prema [1], informacijski su sustavi ultimativno odgovorni za uspjeh svojih poduzeća ili pokrivenih funkcionalnih dijelova, te su uključivanje novih informacijskih tehnologija (IT) i implementacija modernih informacijskih sustava, (IS) sve češći pokretači velikih poslovnih promjena u organizacijama. Događaju se i slučajevi da IT ne daje nikakav impuls za promjenu, ali da su organizacijske promjene, same po sebi, toliko velike da zahtijevaju značajnu adaptaciju postojećeg informacijskog sustava.

Iz navedenih razloga, poznavanje osnovne terminologije i problematike vezane za informacijske sustave, neophodno je za razumijevanje koje to klase i vrste softverskih proizvoda podupiru moderno poslovanje.

Kategorizacija se može raditi s hijerarhijske, funkcionalne i procesne perspektive.

8.1 Hijerarhijska perspektiva

Hijerarhijska perspektiva prepoznaje da se donošenje odluka i aktivnosti u organizacijama događa na različitim razinama. Na svakoj razini hijerarhije, uključeni pojedinci imaju različite odgovornosti, donose različite vrste odluka i provode različite vrste aktivnosti, a vrsta informacijskog sustava, koji se uvodi za podršku na svakoj razini, mora uzeti u obzir te razlike.

Hijerarhijska perspektiva definira tri razine:

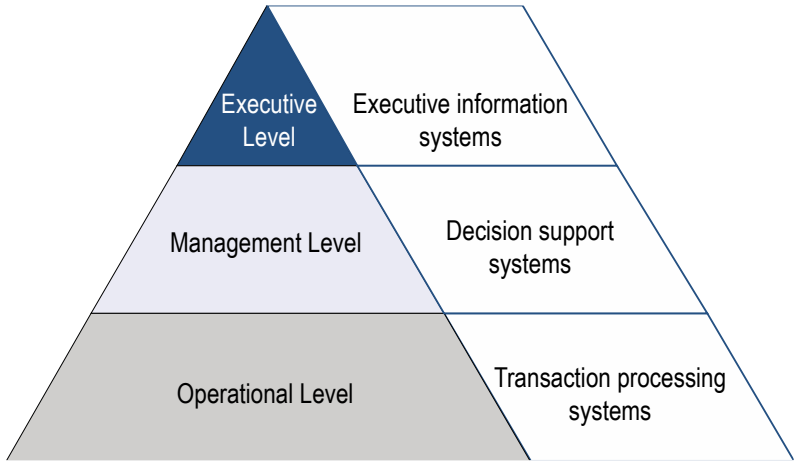
- izvršna, strateška
- upravljačka, taktička
- operativna.

Operativna razina organizacije uglavnom se bavi kratkoročnim aktivnostima, obično onima koje se događaju „upravo sada“. Operativno osoblje usredotočeno je na učinkovitu obradu poslovnih transakcija. Informacijski sustavi, koji podržavaju ovu organizacijsku razinu, nazivaju se sustavima za obradu transakcija (TPS).

Informacijska tehnologija na kojoj se zasniva TPS, obično se koristi za automatizaciju ponavljajućih aktivnosti i za strukturiranje svakodnevnih operacija, osiguravajući da se one izvode s potrebnom brzinom i točnošću.

Menadžerska razina organizacije, uglavnom se bavi srednjoročnim donošenjem odluka i funkcionalnim fokusom. Aktivnosti koje se obavljaju su polustrukturirane, imaju dobro poznate komponente i određen stupanj neizvjesnosti. Cilj aktivnosti ove razine je poboljšati učinkovitost organizacije ili jedne od njezinih funkcija, u okviru širokih strateških smjernica koje postavlja nadređeni izvršni tim.

Informacijski sustavi koji podržavaju ovu organizacijsku razinu, obično se nazivaju sustavi za podršku odlučivanju (DSS). DSS sustavi pružaju informacije potrebne funkcionalnim menadžerima za sudjelovanje u taktičkom odlučivanju.



Slika 54: Tri razine menadžmenta i informacijskih sustava; preuzeto iz [1], stranica 49

Izvršni dio organizacije bavi se dugoročnim odlukama na visokoj razini. Menadžeri ove najviše razine, fokusirani su na strateško odlučivanje i na tumačenje načina reagiranja tvrtke na trendove na tržištu i u konkurentnom okruženju. Njihov je cilj predvidjeti buduća kretanja evaluacijom trendova, koristeći visoko agregirane podatke i analize različitih scenarija.

Informacijski sustavi koji podržavaju ovu organizacijsku razinu, nazivaju se izvršni informacijski sustavi (EIS). U zadnje vrijeme je postalo popularno korištenje izvršnih nadzornih ploča (engl. *dashboards*), koje omogućuju brzu procjenu visoko agregiranih podataka o organizaciji i trendovima, ali i uvid u određene (prethodno definirane) detalje.

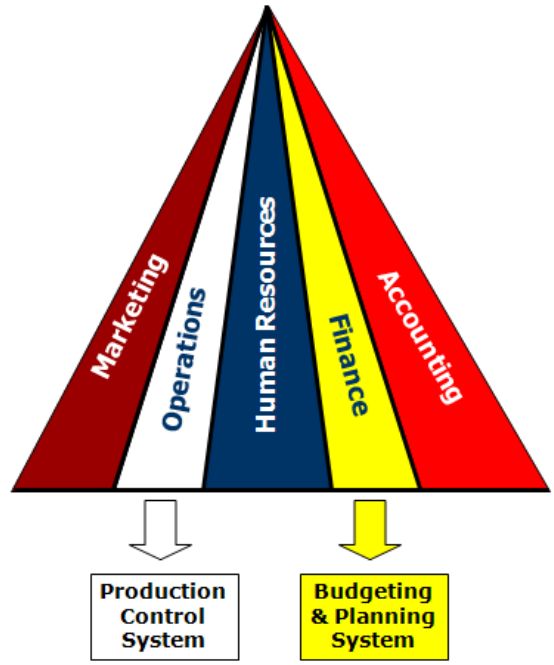
Primarno je ograničenje hijerarhijske perspektive nepostojanje integracije među zasebnim sustavima i uvođenje znatne redundancije.

8.2 Funkcionalna perspektiva

Funkcionalna organizacija unutar poslovnih jedinica obično je predstavljena u obliku organizacijske sheme. Ova decentralizirana upravljačka struktura rješava probleme koordinacije, koji se događaju kada tvrtke postanu dovoljno velike.

Svaka poslovna funkcija samostalno upravlja svojim proračunom i ima jedinstvene potrebe za obradom informacija. Funkcionalni sustavi su izričito dizajnirani da podupiru specifične potrebe pojedinaca u jednom funkcionalnom području:

- Temelje se na načelu lokalne optimizacije, što sugerira da su potrebe za obradom informacija jedinstvene i homogene unutar funkcionalnog područja.
- Prilagođeni su specifičnim potrebama i koriste se terminima poznatim profesionalcima s tog područja.



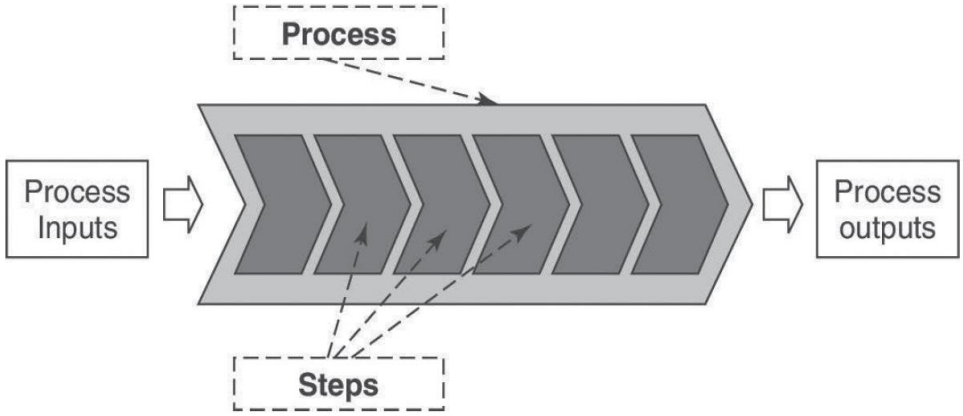
Slika 55: Funkcionalna perspektiva; preuzeto iz [1], stranica 51

Kao i kod hijerarhijske perspektive, i ovdje je primarno ograničenje nepostojanje integracije među zasebnim sustavima i postojanje znatne redundancije. Naime, iz tehnološkog pogleda, funkcionalni pristup dovodi do razvoja višestrukih aplikacija. Te aplikacije vrlo dobro služe vertikalnim (funkcionalnim) potrebama, ali otežavaju (horizontalnu) komunikaciju između različitih funkcionalnih područja.

8.3 Procesna perspektiva

Početkom devedesetih godina prošlog stoljeća, uočena je potreba za razbijanjem organizacijskih (funkcionalnih) silosa.

Novi pristup doveo je do niza alata za reinženjering poslovnih procesa (engl. *Business Process Reengineering*, BPR). To je upravljački pristup, koji koristi procesni pogled na organizacijske aktivnosti i traži dramatična poboljšanja performansi racionalizacijom aktivnosti i uklanjanjem dupliranja napora u odvojenim funkcijama i jedinicama.



Slika 56: Funkcionalna perspektiva; preuzeto iz [1], stranica 52

Fokusiranje na procese omogućuje tvrtki da eliminira suvišnost i neučinkovitost, povezane s višestrukim predajama zadataka iz jednog funkcionalnog područja u drugo. U skladu s procesnim pristupom, tvrtke bi trebale reorganizirati svoj rad u nizu procesa dizajniranih oko željenih ishoda. Svaki bi proces imao svog „vlasnika“, koji nadgleda korake procesa od početka do kraja.

Osnovni problem BPR metodologije je njezina vrlo radikalna narav, jer zahtijeva potpuno nepoštivanje postojećih procesa kako bi se omogućilo redizajniranje.

Osim toga, BPR inicijative su vrlo skupe, jer često zahtijevaju od tvrtke da povuče svoje stare sustave i da razvije posve novu (skupu) integriranu tehnološku infrastrukturu.

9 Integrirani informacijski sustavi

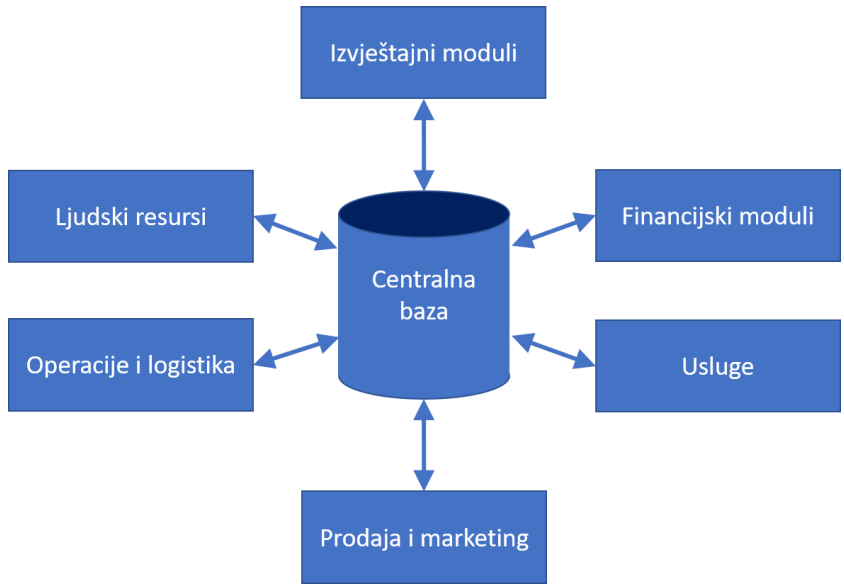
Opći je cilj integracije organiziranje, pojednostavljivanje i pojednostavljenje procesa ili aplikacije:

- Primarni fokus je na tehnološkoj komponenti informacijskih sustava, koja mora podržavati strategije poslovne integracije.
- Ishod integracije je prikupljanje kompatibilnih sustava koji redovito razmjenjuju informacije, ili razvoj integriranih aplikacija koje zamjenjuju bivše samostalne.

Vrste integracije:

- integracija aplikacija → omogućavanje komunikacije između zasebnih softverskih programa
- integracija podataka → omogućavanje spajanja spremišta podataka i baze podataka
- unutarnja integracija → objedinjavanje ili povezivanje internih organizacijskih sustava
- vanjska integracija → objedinjavanje ili povezivanje međuorganizacijskih sustava.

Poslovni sustavi integrirani su softverski programi koji obuhvaćaju (sve) organizacijske funkcije i u osnovi se oslanjaju na jednu bazu podataka. Poslovni sustavi tradicionalno su usmjereni na unutarnje organizacijske procese.



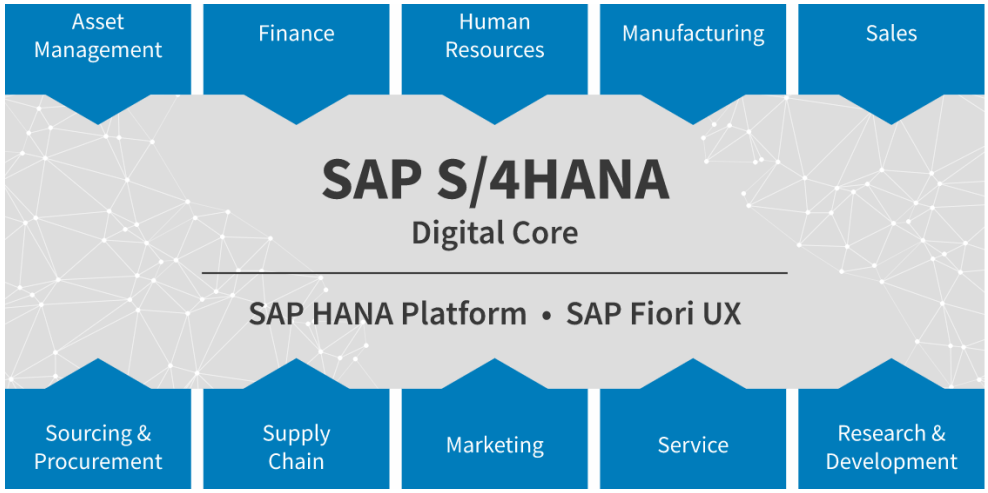
Slika 57: Modularnost integriranih poslovnih sustava

9.1 ERP (*Enterprise Resource Systems*)

ERP sustavi su informacijski sustavi koji, pored mogućnosti pristupa i obrade svih relevantnih podataka u poduzeću, omogućuju planiranje proizvodnje, usluga i resursa, kao i primjenu modela za upravljanje procesima poslovanja, pripreme i proizvodnje [15].

Glavne karakteristike ERP-a su:

- modularnost – ERP omogućuje organizaciji koja ga kupuje da odluči koje će funkcionalnosti biti aktivirane, a koje se neće koristiti
- integriranje aplikacija i podataka – ERP omogućava da događaj koji se dogodi u jednom od modula aplikacije, automatski pokrene događaj u jednom ili više drugih zasebnih modula
- konfigurabilnost – ERP nudi konfiguracijske tablice koje omogućavaju odabir definiranog skupa opcija tijekom implementacije aplikacije.



Slika 58: Vodeći svjetski ERP sustav SAP S / 4HANA (izvor: SAP SE)

Prednosti ERP sustava:

- poboljšanje učinkovitosti uštedom izravnih i neizravnih troškova
- brzina odziva sustava
- širenje znanja među zaposlenicima
- prilagodljivost sustava.

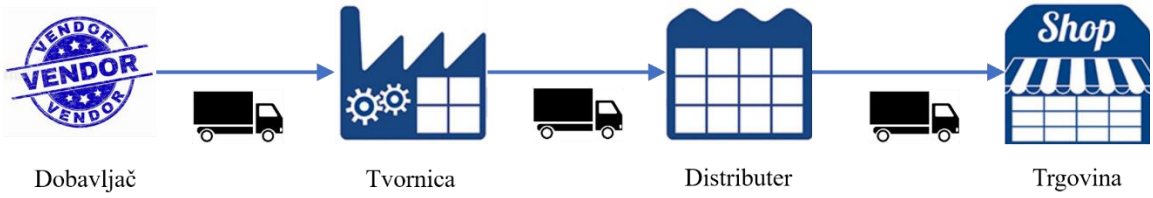
Ograničenja ERP sustava:

- sukob između želje za standardizacijom i potrebe za fleksibilnošću
- ograničenost softvera koji predstavlja „najbolju praksu“
- skupa i rizična implementacija.

9.2 SCM (*Supply Chain Management*)

Lanac opskrbe (engl. *Supply Chain*) skup je koordiniranih funkcija koji doprinose premještanju proizvoda iz njegove proizvodnje u njegovu potrošnju.

Upravljanje lancem opskrbe predstavlja koordinaciju materijalnog protoka robe između dobavljača, proizvođača, skladišta, distributivnih centara i trgovina.



Slika 59: Princip lanca opskrbe

Funkcionalnost upravljanja lancem opskrbe (SCM) skup je logističkih i financijskih procesa povezanih s planiranjem, izvršavanjem i nadgledanjem poslovanja lanca opskrbe.

Slika 60: Funkcionalni pogled na SCM

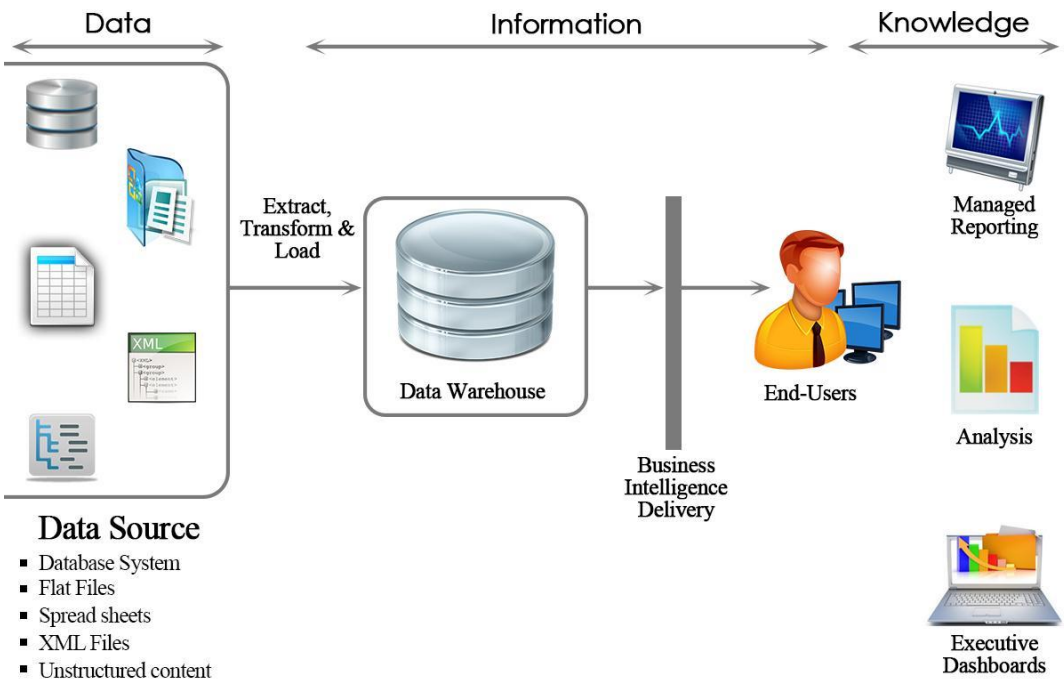
Termin upravljanja lancem opskrbe koristi se za objašnjavanje logističkih aktivnosti, za planiranje i kontrolu materijala i informacija koji su se kretali unutar ili izvan poduzeća, za objašnjenje strateških i interorganizacijskih pitanja, za ispitivanje i proučavanje alternativnih organizacijskih formi, te za objašnjavanje poveznica koje su poduzeća razvijala sa svojim dobavljačima i kupcima.

9.3 *Business Intelligence*

Poslovna inteligencija obuhvaća skup tehnika, procesa i tehnologija, dizajniranih tako da se menadžerima omogući „pogled od gore“ na samo poslovanje i posljedično donošenje boljih odluka. Kontinuirani efekti informacijskog napretka i promjenjiva priroda poslovnih podataka stvaraju savršenu priliku za kreativno izvlačenje ključnih i vrijednih informacija iz podataka. Alati poslovne inteligencije imaju sposobnost prikupljanja i davanja smisla informacijama o poslovanju.

Komponente BI sustava:

- *Data Warehouse* – skladište podataka koje sakuplja i konsolidira podatke dobivene iz višestrukih izvora podataka, sa svrhom omogućavanja kasnijih analiza
- *Data Mart* – manja verzija skladišta podataka koja se usredotočuje na potrebe specifičnog dijela poslovanja
- *Online Analytical Processing* (OLAP) – softver koji omogućuje korisnicima laku i selektivnu ekstrakciju podataka iz baze
- *Data Mining* – proces automatskog otkrivanja ne-tako-jasno-uočljivih veza između podataka u velikim bazama.



Slika 61: BI principi (izvor: http://www.sapspot.com/introduction-sap-bi/)

9.4 *Big Data*

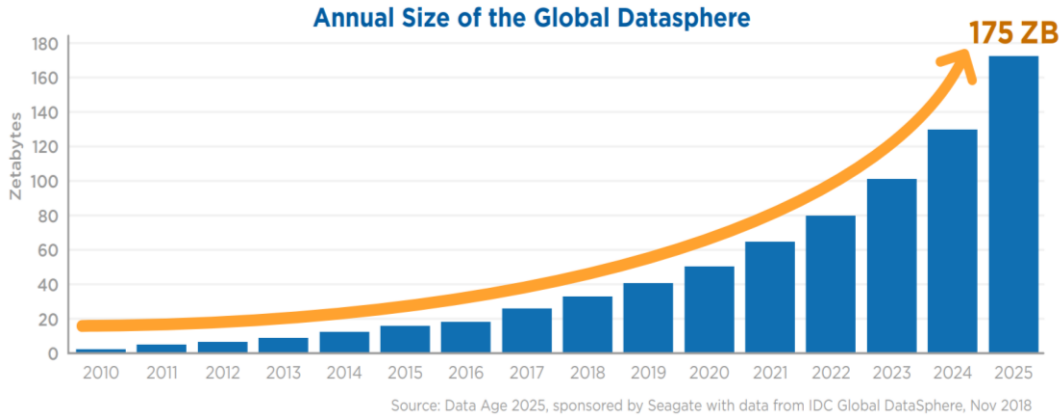
Termin *Big Data* krovni je izraz pod kojim konvergiraju važni skorašnji trendovi. Učinci na odlučivanje i stjecanje znanja predstavljaju promjenu paradigme u načinu na koji menadžment tradicionalno donosi odluke.

Prodornost digitalnih tehnologija i sveprisutnost interneta omogućuju organizacijama prikupljanje i pohranjivanje sve veće količine strukturiranih i nestrukturiranih podataka, vjerojatno više podataka nego što mogu koristiti:

- volumen / količina digitalnih podataka koju organizacije moraju spremati i njima upravljati
- brzina kreiranja i korištenja novih podataka; ova se dimenzija odnosi na prikupljanje, obradu i razdiobu informacija u tzv. stvarnom vremenu
- raznolikost podataka koju treba spremati i njome upravljati.

NoSQL baze podataka predstavljaju novu vrstu tehnologija, razvijenih za prevladavanje ograničenja postojećih pristupa upravljanju podacima pri radu s ogromnim količinama podataka.

Porast količine, brzine i raznovrsnosti podataka (tzv. eksplozija podataka), u zadnjih desetak godina i s prognozom do 2025. godine, prikazan je na slici 62.



Slika 62: Porast količine podataka; izvor: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf/

9.5 *Cloud Computing*

Računarstvo u oblaku revolucija je na tržištu softvera. Omogućava ulazak na tržište IT usluga novim i malim tvrtkama. Tvrtke s ograničenom investicijskom sposobnošću, imaju pristup dinamički formiranim cijenama informatičkih izvora, prebacujući IT troškove s kapitalnih ulaganja (engl. *CapEx*) na operativne troškove (engl. *OpEx*).

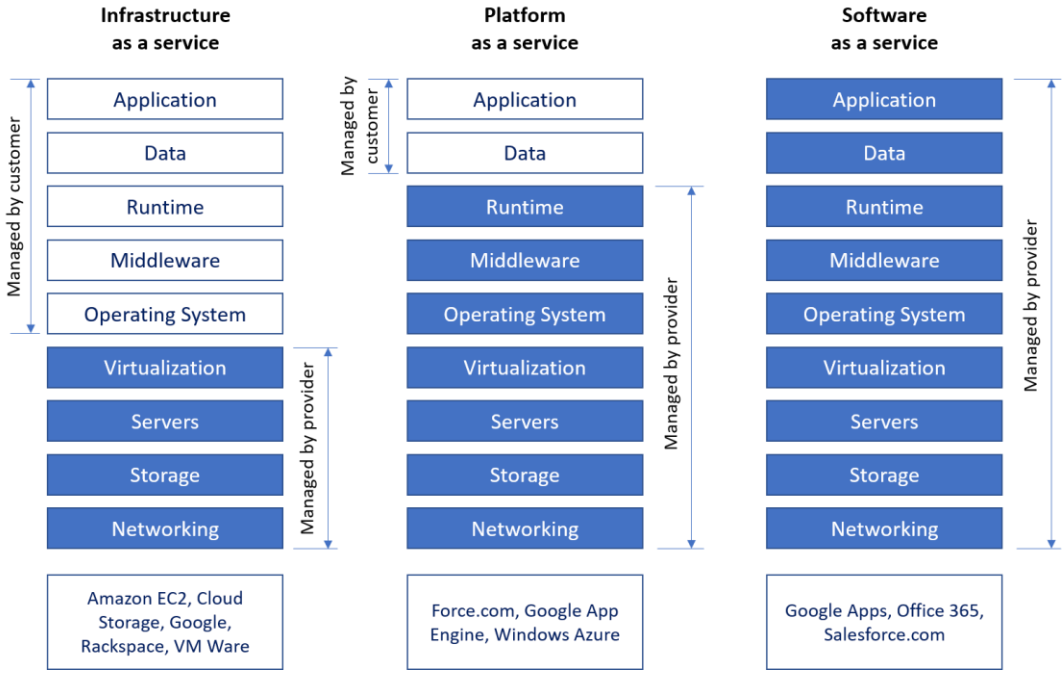
Rad u oblaku predstavlja sasvim nov pristup računarstvu kroz tri osnovne komponente:

- virtualizacija
- servisno orijentirana arhitektura
- plaćanje usluga isključivo po principu korištenja.

Ostale prednosti rada „u oblaku“:

- ubrzavanje procesa inovacija trenutnim pristupom IT resursima, bez prethodnog investiranja
- veća skalabilnost na način da novi IT resursi mogu odmah biti dodijeljeni mjestu potrebe, u ovisnosti o trenutnim potrebama.

Oblak se temelji na arhitekturi klijent-poslužitelj s osobnim računalima, tabletima i pametnim telefonima, koji pristupaju moćnim udaljenim poslužiteljima. Modeli upotrebe prikazani su na slici 63.



Slika 63: Načini rada u oblaku; preuzeto iz [1], stranica 84

Software as a service (*SaaS*) je kada aplikacija radi u oblaku. Primjeri: *Salesforce, Gmail, Dropbox, Office 365*. Fokus je na pružanju i naplaćivanju usluge, tj. na funkcionalnosti, a ne na prodaji softvera.

Platform as a service (*PaaS*) je kada se ne iznajmljuje puna aplikacija, nego samo platforma na kojoj onda korisnik gradi svoje aplikacije.

Infrastructure as a service (*IaaS*) je kada klijent plaća samo za korištenje hardvera (snaga računanja, kapacitet usklađivanja podataka, mreža). Sav softver, uključujući operativni sustav *back-up* i r*ecovery*, odgovornost je korisnika. *IaaS* dobavljač brine se o ispravnom radu infrastrukture.

10 IS u uvjetima digitalnog poslovanja

Internet se široko definira kao skup umreženih računala koja mogu međusobno „razgovarati“. Internet je infrastruktura na kojoj se isporučuju usluge, poput web-stranica, e-pošte, razmjene trenutnih poruka (engl. *instant messaging*) i mnogih drugih. Internet može povezati bilo koji uređaj koji se temelji na arhitekturi digitalnih računala (npr. prijenosna računala, pametni telefoni, digitalni fotoaparat za prepoznavanje lica).

Najbitnija stvar kod interneta je da je on dostupan svima i da nitko nema vlasništvo nad njim. Internet je digitalna mreža koja se sastoji od milijuna manjih digitalnih mreža. Svaka od tih manjih digitalnih mreža obuhvaća zbirku digitalnih uređaja, nazvanih čvorovi.

Internet se oslanja na otvorene tehnološke standarde i protokole (dogovoreni skup pravila ili konvencija koji upravljaju komunikacijom među elementima mreže). Da bi komunicirali, mrežni čvorovi moraju slijediti dogovoreni skup pravila. Na internetu je takav skup pravila TCP / IP protokol. Njega također nitko ne posjeduje – to je otvoreni, tj. slobodno dostupni standard.

Pojam *Web 2.0* popularizirian je 2004. godine i označava drugi val inovacija i evolucije, koji se događa na internetu nakon potresa uzrokovanih prvotnim internetskim inovacijama (1993. – 2001.). *Web 2.0* karakteriziraju:

- dvosmjerni razgovori (npr. blogovi)
- korisnički sadržaj (npr. mrežne recenzije, *YouTube* videozapisi, *Wikipedia*)
- dinamičke strukture (npr. automatska veličina slova, *Flickr*).

U novije vrijeme tendencija je pristup uslugama na internetu mobilnim uređajima – pametnim telefonima, tabletima ili *IoT* uređajima. Taj trend potaknut je rastom dvaju konkurentnih ekosustava: *Apple iOS* i *Google Android*.

Karakteristike mobilnih uređaja:

- sveprisutnost – ideja da korisnici uređaja mogu pristupiti potrebnim resursima (teoretski) s bilo kojeg mjesta – oni su prenosivi i povezani
- prepoznatljivost – ideja da mobilni uređaji jedinstveno identificiraju svog korisnika pomoću modula za identifikaciju pretplatnika (SIM kartica)
- svijest o kontekstu – omogućena činjenicom da se mobilni uređaji mogu geolocirati, pa softverske aplikacije mogu iskoristiti lokaciju osobe koja nosi uređaj.

Poslovni model je apstrakcija koja bilježi koncept i vrijednosti tvrtke, a istovremeno prenosi informacije o tržišnoj prilici koju tvrtka provodi, koji proizvod / uslugu nudi i koju će strategiju slijediti kako bi se našla u vodećoj poziciji. Poslovni model govori tko je kupac tvrtke, što tvrtka radi za svoje kupce, kako to radi i kako će biti nagrađena za ono što radi.

Elektronička trgovina je postupak distribucije, kupnje, prodaje, marketinga i servisa proizvoda i usluga, preko računalnih mreža kao što je internet. Elektroničko poslovanje se definira kao upotreba internetskih, drugih naprednih IT tehnologija, kako bi se omogućili poslovni procesi i operacije.

Elektronička trgovina i elektroničko poslovanje oslanjaju se na isti skup mogućnosti, te se, kao općeniti (zajednički) termin, koristi izraz digitalno poslovanje.

Kategorizacija digitalnog poslovanja po vrsti transakcija:

- *Business-to-Consumer* (B2C) – transakcije koje uključuju profitnu organizaciju, s jedne strane, i krajnjeg potrošača, s druge (*Amazon.com*)
- *Business-to-Business* (B2B) – transakcije u kojima sudjeluju dva poslovna subjekta ili više njih (*Alibaba.com*); transakcije mogu varirati, od onih koje se pojavljuju samo jednom, sve do strogo strukturiranih veza između dvaju poduzeća (autoindustrija)
- *Consumer-to-Consumer* (C2C) – transakcije koje omogućuju pojedinačnim potrošačima neposrednu interakciju i ostvarivanje transakcija (*eBay*)
- *Consumer-to-Business* (C2B) – transakcije se događaju kada pojedinci sklapaju transakcije s poslovnim organizacijama, ne kao kupci robe i usluga, već kao dobavljači (*Upwork.com*)
- *eGovernment* – transakcije koje uključuju zakonodavne i administrativne institucije (npr. elektroničko prijavljivanje poreza na dohodak).

Dominantni poslovni modeli:

- internetska trgovina na malo (engl. *online retail*) – organizacija koja uzima kontrolu nad zalihama i onda ih preprodaje s profitom
- posrednici (engl. *infomediaries*) – organizacije koje koriste internet za pružanje specijaliziranih informacija u ime davatelja proizvoda ili usluga
- pružatelji sadržaja (engl. *content providers*) – organizacije koje razvijaju i objavljuju sadržaj, pri čemu mogu, ali i ne moraju, biti vlasnici sadržaja koji objavljuju
- društveno umrežavanje (engl. *social networking*) – poslovanje bazirano na internetskim zajednicama
- davatelji infrastrukture – tvrtke koje kreiraju vrijednost razvijanjem i upravljanjem infrastrukture za digitalno poslovanje.

11 Strateško planiranje IS-a

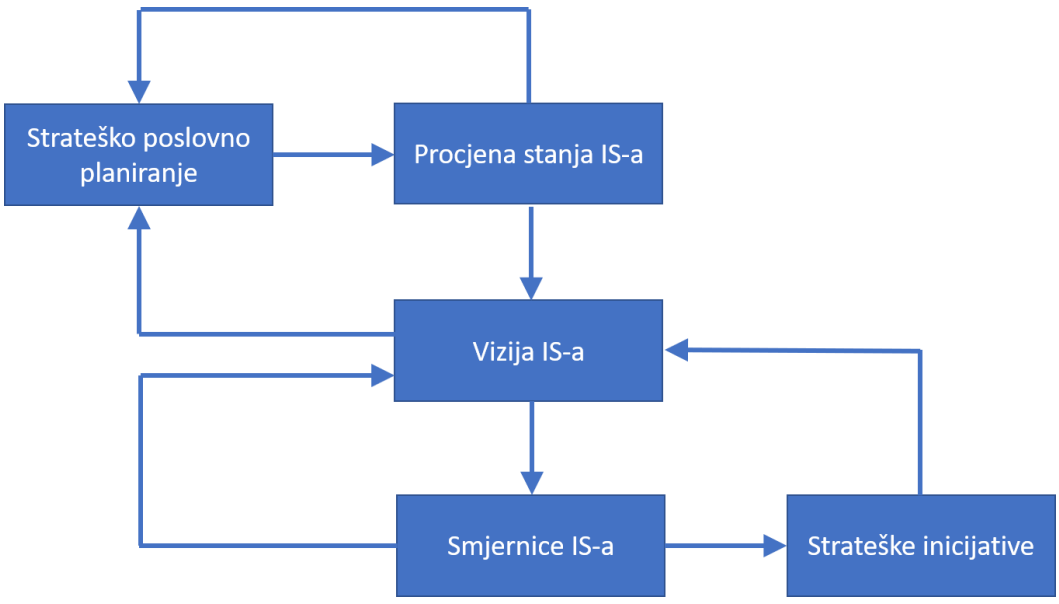
Informacijski sustavi sve više postaju činitelji koji omogućuju provođenje poslovne strategije i samih operacija. Zato je strateško planiranje informacijskog sustava, proces kojim tvrtka, timom za planiranje, razvija zajedničko razumijevanje uloge IS-a u organizaciji.

Menadžeri iz poslovnog dijela (izvan IT-a) igraju presudnu ulogu u timu za planiranje. Njihova je uloga pomoći identificirati strategiju tvrtke, i onda, u skladu s poslovnom strategijom, pomoći u odlučivanju kako bi se resursi informacijskog sustava trebali koristiti za njezino postizanje.

Spomenuti menadžeri trebali bi preuzeti vodeću ulogu u definiranju količine novca koje treba potrošiti na informatičke tehnologije, zatim, treba definirati, u koje poslovne procese ta sredstva trebaju biti usmjerena, koje bi IT mogućnosti trebale prožimati organizaciju, i kakve se razine IT usluga očekuju.

Strateško planiranje informacijskog sustava uključuje prepoznavanje dugoročnog smjera upotrebe i upravljanja informacijskim sustavima u organizaciji.

Uz uspostavljene osnovne mehanizme planiranja, tvrtka kreće u akciju i identificira strateške inicijative, koje treba provoditi u svrhu postizanja navedene vizije informacijskog sustava. Te strateške inicijative često proizlaze iz onoga što organizacija vjeruje da su dostupne mogućnosti, kao i slabosti kojima se mora upravljati.



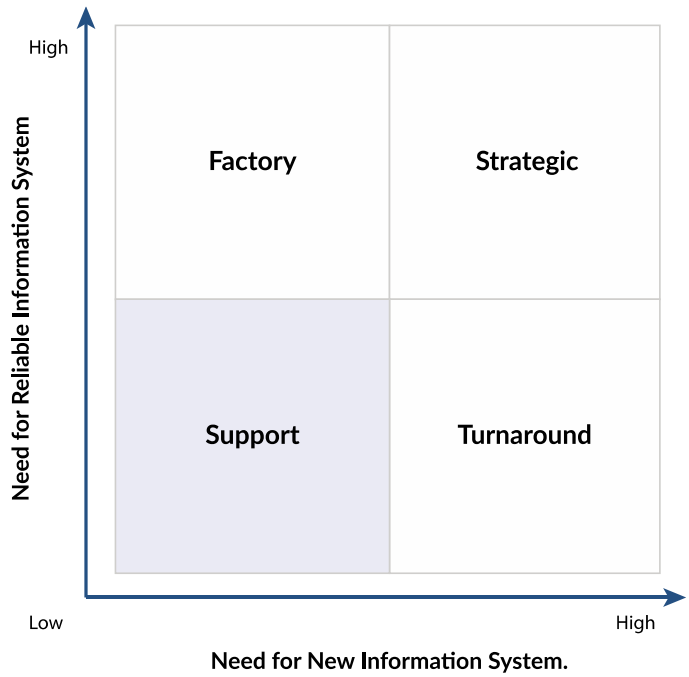
Slika 64: Proces strateškog planiranja IS-a

Kratki opis osnovnih dijelova procesa prikazanih na slici 64:

- Strateško poslovno planiranje sastoji se od misije i budućeg usmjerenja organizacije, ciljeva i strategije djelovanja.
- Procjena stanja IS-a sastoji se od uvida u trenutne tvrtkine resurse u IS-u i procjene koliko dobro IS ispunjava potrebe organizacije.
- Vizija IS-a sastoji se od precizne i jasne izjave o tome što tim za planiranje vjeruje da bi trebala biti uloga resursa IS-a u tvrtki; mora biti usklađena s poslovnom strategijom tvrtke i odražavati poslovnu strategiju tvrtke.
- Smjernice za informacijske sustave predstavljaju skup izjava koje navode kako tvrtka treba koristiti svoje tehničke i organizacijske IS resurse.
- Strateške inicijative su dugoročni (od tri do pet godina) prijedlozi koji identificiraju nove sustave i nove projekte ili nove upute za organizaciju IS-a.

Vrlo vrijedan način analize uloge IS-a u poslovnoj organizaciji je tzv. strateška mreža utjecaja. Glavna prednost koju nudi strateška mreža utjecaja je mogućnost omogućavanja istodobne procjene trenutnih i budućih potreba poduzeća za informacijskim sustavima. To se postiže promatranjem poduzeća, pomoću sljedeće dvije dimenzije:

- trenutna potreba za pouzdanim IS-om – usredotočena je na svakodnevno poslovanje i funkcionalnost postojećih sustava
- buduće potrebe za novim funkcijama IS-a – dimenzija usmjerena prema naprijed, koja se tiče strateške uloge koju nove IT mogućnosti igraju za organizaciju.



Slika 65: Strateška mreža utjecaja; preuzeto iz [1], stranica 187

Definicija i opis pojedinih kvadranata:

- kvadrant „Podrška“:
  - IS nije kritični faktor za trenutni rad poslovnog subjekta
  - novi sustavi ne obećavaju značajnu stratešku prednost ili razlikovnost
  - poslovni subjekt ima pogled na IS kao na obični alat koji treba podržati rad tvrtke, te je vrlo konzervativan s obzirom na moguće investicije u informacijski sustav.
- kvadrant „Tvornica“:
  - čak i mali poremećaji IS infrastrukture mogu ugroziti dobrobit i buduću održivost tvrtke
  - ograničeni potencijal da novi sustavi i funkcionalnosti daju znatan doprinos
  - poslovni subjekt pažljivo nadgleda trenutni sustav, trebao bi biti voljan financirati njihovo održavanje i nadogradnju, ali često zauzima konzervativni stav prema budućim ulaganjima.
- kvadrant „Preokret“:
  - IS nije kritičan za trenutne operacije
  - novi IS ili nove funkcionalnosti postojećeg IS-a, bit će presudne za buduću održivost i uspjeh poslovanja
  - poslovni subjekt je spreman promijeniti svoj IS; bilo bi potrebno sudjelovanje u nekoj reorganizaciji.
- strateški kvadrant:
  - IS je kritičan za trenutno poslovanje tvrtke
  - novi IS ili nove funkcionalnosti postojećih sustava od presudne su važnosti za buduću održivost i prosperitet poslovanja
  - poslovni subjekt trebao bi biti ektremno proaktivan u vezi s informacijskim sustavom i IT ulaganjima.

12 Kreiranje vrijednosti pomoću IS-a

Analiza dodane vrijednosti formalni je mehanizam koji menadžeri i analitičari koriste kako bi procijenili koliko se vrijednosti, koju je tvrtka kreirala, može zadržati u obliku profita.

Ova je analiza ključni korak u odlučivanju treba li se započeti s nekom inicijativom ili ne. Ako analiza pokaže da predložena inicijativa neće stvori opipljivu vrijednost, trebalo bi je odbiti. Ova vrsta analize je korisna, ne samo kod inoviranja, već i kada se procjenjuje kako odgovoriti suparniku, koji je preuzeo lidersku poziciju.

Ekonomska vrijednost nastaje kada se kreiraju vrijedne stvari koje prije nisu postojale. Poduzetnici se fokusiraju na tržišne mogućnosti i razvoj rješenja, kako bi se iskoristile te mogućnosti. Dakle, poduzetnici prvenstveno traže nove načine stvaranja vrijednosti, a ne nove tehnologije ili nove proizvode (to bi bio posao izumitelja).

Ekonomska vrijednost stvara se procesom transformacije. Vrijednost se stvara kada se resursi, koji bi se u svojoj sljedećoj najboljoj upotrebi pretvorili u nešto za što je kupac spreman platiti više od vrijednosti samih transformiranih resursa [1].

Dva su načina za kreiranje nove vrijednosti:

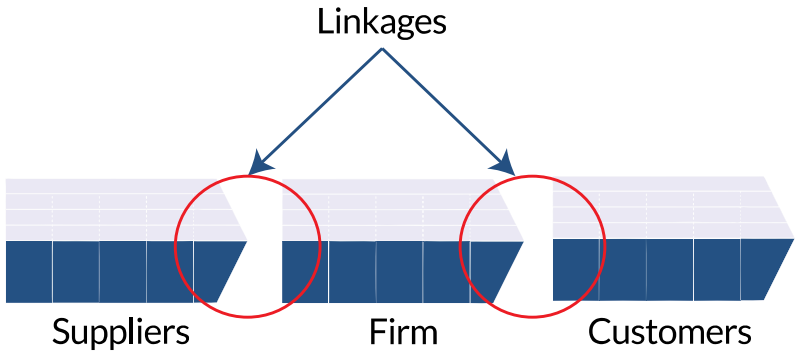
- povećanje kupčeve volje za plaćanjem – napraviti nešto od naročite vrijednosti za kupca
- smanjenje dobavljačevih potraživanja – kreiranje posebnih poticaja za dobavljače, kako bi oni pristali dostavljati potrebne resurse za manje novaca.

Pitanja za dublju analizu za potrebu investiranja u informacijske sustave:

- Može li korištenje IT-a povećati ulaznu industrijsku barijeru? Investiranje u informacijski sustav može biti takvo da smanjuje prijetnju mogućih novih konkurenata (*Playstation*).
- Može li korištenje IT-a smanjiti dobavljačevu pregovaračku poziciju? Pojava interneta i njegovih tehnologija, kao održivih poslovnih alata, pridonijeli su prebacivanju moći s dobavljača i na kupce (*Alibaba*).
- Može li korištenje IT-a smanjiti kupčevu pregovaračku poziciju? Jednako kao što tvrtke mogu koristiti alternativna rješenja kada nabavljaju sirovinu za proizvodnju, tako to mogu ostvariti i njihovi kupci. Ipak, još uvijek postoje mogućnosti za jačanje odnosa s kupcima, smanjujući njihovu potrebu za kupovinom od drugih strana (*Orbitz*).
- Može li korištenje IT-a promijeniti bazu industrijske konkurencije? Uvođenje novog informacijskog sustava od strane konkurentske tvrtke, bilo da je aktivni igrač ili novi sudionik, ponekad potiče revoluciju, koja prisiljava konkurente da primijete promjene i reagiraju (*Amazon.com*).

Menadžeri nastoje nadmašiti svoje konkurente, koristeći informacijske sustave za stvaranje dodane vrijednosti i konkurentskih prednosti. Menadžeri moraju prepoznati, razumjeti i analizirati aktivnosti koje tvrtka obavlja, tako da se mogu poboljšati ili transformirati pomoću IS resursa.

Vlastiti lanac vrijednosti tvrtke postoji u mreži veće vrijednosti koja sadrži dobavljače i kupce. Točke dodira između ovih zasebnih lanaca vrijednosti zovu se veze i nude značajne mogućnosti za implementaciju strateških inicijativa ovisnih o IT-u.



Slika 66: Mreža vrijednosti (kupci – dobavljači); preuzeto iz [1], stranica 219

Temeljna pretpostavka je da tvrtka može koristiti informacijske sustave za stvaranje vrijednosti, nudeći vrhunsku korisničku uslugu. Dobila je novi zamah s pojavom interneta i omogućavanjem čvrstih povezivanja. Kako sve više i više organizacija uspijeva uspostaviti izravne odnose s kupcima, povećava se potencijal za stvaranje nove vrijednosti pomoću vrhunske usluge za kupce.

Tvrtka koja je u stanju prikupiti i koristiti superiorne informacije, i time smanjiti neizvjesnost, može ograničiti zalihe i pokretati jednostavnije poslovne operacije. Pristupačna, snažna, međusobno povezana računala i jeftina pohrana, stvorili su pozadinu za brojne nove načine stvaranja ekonomske vrijednosti s informacijskim sustavima. Informacije više nisu samo izvori podrške za fizičke aktivnosti (one opisane modelom fizičkog lanca vrijednosti), već se same mogu tretirati kao ulaz produktivnog procesa transformacije.

Informacija je subjekt čija se vrijednost povećava kroz pet povezanih aktivnosti:

- prikupljanje informacija
- organizacija informacija
- izbor relevantnih informcija
- sinteza važnih informacija
- distribucija sažetih informacija.

Kada se napajaju kroz aktivnosti virtualnog lanca vrijednosti, organizacijski se podaci mogu transformirati u vrijedne uvide, nove procese ili nove proizvode ili usluge. Primjer može biti prikupljanje podataka i informacije o kupcima kako bi se učinilo nešto vrijedno za njih, povećavajući, na taj način, njihovu spremnost za plaćanje. Da bi se to učinilo, potrebna je značajna analiza i razumijevanje karakteristika same tvrtke i prijedloga vrijednosti.

13 Financiranje IS-a

Strateško planiranje informacijskih sustava pruža okvir za donošenje odluka i odabir pojedinih projekata. Tvrtka razvija godišnje operativne planove i proračune kako bi se prioritizirala potrošnja vezana za informacijski sustav.

Funkcioniranje procesa proračuna i određivanja prioriteta korisno je znanje za IT, jer je godišnji proračun prilika za formalno ocjenjivanje konkurentskih projekata i donošenje sveobuhvatnih odluka o prioritizaciji, radi usklađivanja razvoja IS-a sa strategijom tvrtke. Naime, tvrtka se mora brinuti o svom IS-u i financirati njegovu imovinu i troškove, jer se informacijski sustavi ne rade sami zbog sebe, nego u svrhu ostvarivanja poslovnih ciljeva.

Tri glavne metode financiranja su:

- naplata korištenja
- alokacija troškova
- princip režijskih troškova.

Naplata korištenja zahtijeva izravno naplaćivanje resursa i usluga informacijskih sustava, organizacijskoj funkciji ili odjelu koji ih koristi. To je utemeljeno na principu plaćanja po upotrebi. Glavna je prednost ovog mehanizma osjećaj realnosti i odgovornosti, koji se stvaraju kod korisnika i kod funkcije IS-a. Prednost je također i stupanj kontrole koji se na ovaj način pruža menadžerima, jer mogu proaktivno kontrolirati troškove informacijskog sustava svoje funkcije.

Ova metoda tretira IS kao mjesto troška, odnosno poslovne funkcije plaćaju korištenje IS-a na temelju stvarnih troškova.

Princip alokacije također zahtijeva izravno naplaćivanje resursa i usluga informacijskih sustava, organizacijskoj funkciji ili odjelu koji ih koristi, ali ne na osnovi mjerenja upotrebe, nego na osnovi indirektnih pokazatelja, poput veličine, prihoda i broja korisnika. Ti se pokazatelji obično postavljaju jednom godišnje, pa su troškovi, koje svaka jedinica može očekivati u narednom periodu, sasvim predvidljivi.

Princip režijske naplate tretira informacijske sustave kao zajednički trošak, koji se izvlači iz cjelokupnog proračuna organizacije, a ne da ga plaća svaka poslovna jedinica posebno. To je najjednostavniji pristup financiranju. Odlučivanje se radi jednom godišnje, u sklopu postupka odobravanja proračuna, i pruža najviše kontrole nad troškovima IS-a.

Glavni nedostatak ovog pristupa je nedostatak odgovornosti za obje strane – i za funkcionalna područja i za IS. Naime, korisnicima se ne naplaćuje izravno, tako da je manje vjerojatno da će proaktivno upravljati njihovom uporabom i filtrirati zahtjeve za uslugom i novim projektima. Osim toga, korisnici su u velikoj mjeri nesvjesni svog utjecaja na ukupni proračun IS-a. S druge strane, funkcija IS-a ima malu odgovornost prema pojedinim funkcionalnim područjima, pa postoji mogućnost nuđenja lošije usluge.

13.1 Proces godišnjeg proračuna

Godišnji postupak proračuna je alat koji organizacije koriste za komuniciranje o planovima i provođenju sustava kontrole:

- Kao alat za planiranje, proračun daje procjenu onoga što tvrtka vjeruje da će biti budući financijski tokovi.
- Kao mehanizam kontrole, proračun pomaže poticanje i provođenje financijski odgovornog ponašanja.

Kontrola IT proračuna obično se dijeli na dva dijela:

- Dio se dodjeljuje funkciji IS-a za infrastrukturne troškove i projekte koji omogućuju da poslovanje radi pouzdano i sigurno.
- Ostatak kontroliraju pojedine poslovne jedinice, za financiranje operacija postojećih sustava i za financiranje novih IS projekata.

Proces izrade proračuna zahtijeva kompromise između različitih interesa i prioriteta projekata s ograničenim resursima. Obično se donose dvije odluke:

- određivanje odgovarajućeg proračuna za tekuće operativne troškove (održavanje)
- procjena velikih kapitalnih izdataka (npr. novi sustavi), podržavanje trenutnih poslovnih funkcija ili razvijanje poslovnih kapaciteta.

Tijekom procesa izrade proračuna, tvrtka ima priliku procijeniti rizik predloženih projekata, pojedinačno i kao cijeli portfelj.

Kao priprema za izradu godišnjeg proračuna, izrađuju se tzv. poslovni slučajevi. Poslovni slučaj je službeni dokument koji priprema i prezentira viši menadžment, obično svatko za svoju područje. Glavni cilj dokumenta je objasniti i pružiti dokaze izvršnom menadžmentu o tome da će se inicijativa isplatiti i da je njezino financiranje opravdano.

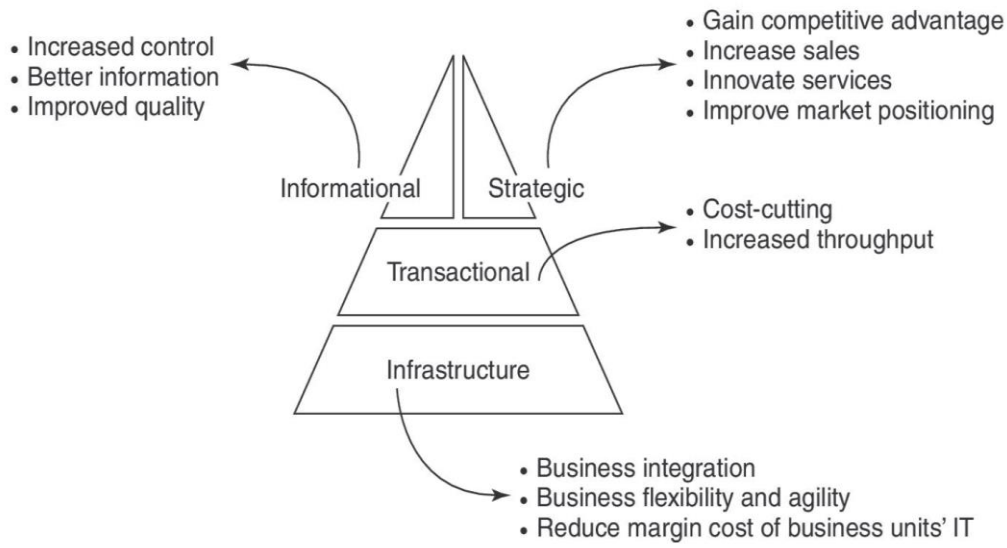
Važan dio argumentacije čini i procjena ukupnih troškova vlasništva (engl. *Total Cost of Ownership*, TCO). To je u stvari financijska procjena troškova IT imovine, tijekom cijelog životnog ciklusa. Troškovi informacijskog sustava i tehnologije obično znatno prelaze troškove nabave i implementacija. Uključuju troškove koji nastaju nakon što je sustav pokrenut, troškove za njegovo održavanje tijekom cijelog životnog vijeka. Kod ovoga je važno reći da identificiranje projektnih troškova zahtijeva značajan udio procjenjivanja (engl. *estimation*). Uključuje pretpostavljanje budućih događaja.

Sljedeći bitan dio procesa izrade proračuna je kategorizacija projekata i određivanje korektnih financijskih odnosa između pojedinih kategorija. Tijekom procesa izrade proračuna, kada tvrtka procjenjuje prikupljanje inicijativa za narednu godinu, trebala bi se utvrditi odgovarajuća razina zbirnih rizika, koje je tvrtka spremna prihvatiti.

Profil rizika mijenjat će se ovisno o spoju strateških, informacijskih, transakcijskih i infrastrukturnih projekata, u koje se tvrtka odluči uključiti. Projekti se tako mogu podijeliti na četiri kategorije [1]:

- strateški
- informacijski

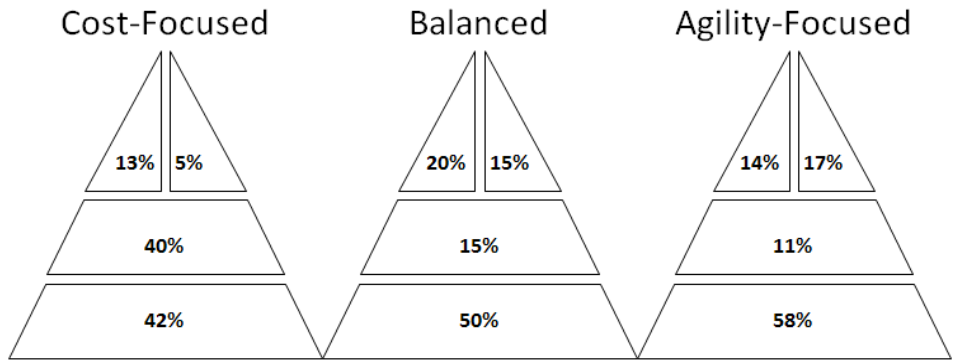
- transakcijski
- infrastrukturni.



Slika 67: Kategorizacija IS projekata; preuzeto iz [1], stranica 282

Bitno je reći da su ovakve procjene menadžerske, a ne informatičke odluke. Portfeljski pristup upravljanju rizicima u informacijskim sustavima, osigurava da financirane inicijative odgovaraju profilu rizika koji tvrtka smatra prikladnim:

- baziran na troškovima
- fokusiran na agilnost
- uravnotežen.



Slika 68: Kategorizacija projekata prema raspodjeli sredstava; preuzeto iz [1], stranica 282

13.2 *Outsourcing* i *Offshoring*

IS *Outsourcing* je sklapanje ugovora s vanjskom tvrtkom za dobivanje usluga informacijskog sustava. Koristi se kao sredstvo financiranja operacija informacijskog sustava, angažiranjem vanjskih pružatelja usluga. Takve se usluge mogu kretati u rasponu, od automatizacije određenih procesa (npr. plaće), upravljanja određenom imovinom (npr. podatkovnim centrom), razvojem novih aplikacija, sve do izravnog upravljanja funkcijom IS-a u cjelini (tj. potpuni *outsourcing*).

Motivi za *outsourcing*:

- smanjivanje troškova (iskorištavanje sposobnosti pružatelja usluga za kreiranje ekonomije razmjera – engl. *economies of scale*)
- pristup vrhunskim ekspertima (pružatelji IT usluga imaju potrebnu IS ekspertizu)
- poboljšanje kontrole (uklanjanje neefikasnog IT odjela)
- poboljšanje strateškog fokusa (uklanjanje IS funkcije i oslobađanje resursa za usredotočavanje na najvažnije poslove)
- financijski utjecaj (jačanje bilance na osnovi otpisa materijalne i nematerijalne imovine).

Rizici *outsourcinga*:

- paradoks *outsourcinga* (sam po sebi, *outsourcing* ne garantira performanse, menadžersko uključivanje iz tvrtke još je uvijek neophodno)
- mračna strana partnerstva (svaki partner ima odgovornost prema svojim vlasnicima da maksimizira svoje performanse – mogući sukob interesa)
- promjena zahtjeva (tijekom duljeg razdoblja, tehnološki i poslovni zahtjevi se mijenjaju i traže se drukčije stvari od IS-a)
- skriveni troškovi koordinacije (postoji mnogo skrivenih i neočekivanih troškova koordinacije poslova između tvrtke i dobavljača usluga)
- varljiva uloga IS-a (mnoge tvrtke podcjenjuju kritičnu ulogu koju IS ima u omogućavanju poslovnog uspjeha).

Offshoring (*Offshore Outsourcing*) postupak je angažiranja stranog dobavljača za opskrbu usluga koje tvrtka više ne namjerava interno provoditi:

- *Offshoring* je dobio puno poticaja od komercijalizacije interneta, što je značajno umanjilo utjecaj geografskih i vremenskih razlika na izvršenje informatičkih i informacijskih usluga.
- Rast *offshoringa* potaknuli su isti pokretači *outsourcinga* (troškovi i kvaliteta), pri čemu se velik dio posla seli u Indiju i Kinu – zemlje sa znatno nižim životnim troškovima od Sjedinjenih Država ili Europe, a nude, naizgled, beskonačno mnoštvo visoko kvalificiranih IT talenata.

13.3 Novi informacijski sustavi

Moderne tvrtke uvode nove informacijske sustave, koristeći jedan od sljedećih pristupa:

- specifični dizajn i razvoj (implementacija softvera koji je kreiran od strane specijalističkih proizvođača IS sustava, na način da je maksimalno prilagođen jedinstvenim potrebama organizacije)
- standardna verzija IS-a (implementacija standardne / univerzalne aplikacije; interno povjerenstvo za izbor odabire jedno od ponuđenih rješenja)
- interni razvoj (implementacija softverske aplikacije koju kreira interni IT tim, a ne profesionalne tvrtke koje razvijaju IS sustave).

Moguća je i kombinacija internog razvoja i kupovine:

- Tvrtka prvo kupi standardni softver, a onda ga modificira prema svojim potrebama.
- Neke tvrtke uopće ne rade promjene softvera nakon kupovine.
- Standardne aplikacije postaju sve veće i kompleksnije te često zahtijevaju dodatne modifikacije radi prilagođenja poslovanju.

Prednosti specifičnog razvoja:

- Potpuno prilagođavanje potrebama poslovanja.
- Prilagodljivost i kontrola (kreiranje „od nule“ dopušta da softver bude oblikovan u željenoj formi i naknadno modificiran).
- Korisničko zadovoljstvo (korisnici aktivno sudjeluju u dizajniranju posve novog rješenja te će, vjerojatno, biti zadovoljni rezultatom).

S druge strane, specifični razvoj nosi svoje rizike. Najveći su rizici kontinuiteta. Tvrtka koja je za korisnika izradila softver po narudžbi, može odustati od te vrste poslovanja ili jednostavno može doći u teške financijske probleme. Također, za samu je softversku tvrtku teško naći pogodnu zamjenu za dizajnere i programere, koji su kreirali aplikaciju, ako napuste tim.

Prednost kupovine standardnog softvera:

- manje ukupno vrijeme implementacije (standardni softver prilično smanjuje vrijeme cjelokupne implementacije, od kupovine sve do završetka implementacije, a znatno je olakšan proces naknadne implementacije u ostale dijelove korporacije, tzv. *roll-out*)
- prijenos znanja (korištenje najboljih praksi donosi obilje novih znanja u tvrtku, od standardnih IT znanja sve do specifičnih poslovnih znanja)
- ekonomska atraktivnost (tvrtka može iskoristiti princip ekonomije razmjera u pregovorima s dobavljačima)
- kvaliteta softvera (kupovni softveri koji su dovoljno dugo na tržištu, već su višestruko istestirani).

14 Projekt implementacije standardnog IS-a

Implementacija velikih informacijskih sustava višegodišnji je projekt, koji traži detaljnu pripremu, planiranje i kontrolu.

Vrste projekata implementacije IS-a:

- probni projekt (*Pilot*)
- poslovni predložak za korporaciju (*Template*)
- implementacija predloška u centralni dio korporacije (*Core*)
- implementacija u ostale dijelove korporacije (*Roll-Out*).

Svaki se od nabrojanih vrsta projekata implementira fazno, na sličan način. U nastavku će biti navedene i kratko objašnjene osnovne projektne uloge, kao i dva osnovna načina implementacije (5-fazni i 4-fazni).

14.1 Projektne uloge

Važnije projektne uloge:

- voditelj projekta — *Project Manager*
- arhitekt poslovnog rješenja — *Solution Architect*
- voditelj promjena — *Change Manager*
- voditelj kvalitete projekta — *Quality Manager*
- voditelji modula — *Team Leads*

Voditelj projekta vodi projekt prema planu i odgovoran je za dostizanje projektnih ciljeva u okvirima vremena, troškova, resursa i kvalitete. Odgovoran je za planiranje projekta, alociranje osoblja, i resursa za ispunjenje projektnih zadataka, te, na kraju, za sam nadzor projekta.

Arhitekt poslovnog rješenja intenzivno surađuje sa svim projektnim timovima, povezujući njihov rad u svrhu kreiranja najboljih ukupnih rješenja. Analizira i definira prijedloge složenih *end-to-end* poslovnih procesa, uključujući pojedinačne procese i rješenja u jedinstveno i sveobuhvatno rješenje.

Voditelj promjena ima ulogu prvenstveno vezanu za komunikaciju prema poslovanju, odnosno treba olakšati proces predavanja razvijenog rješenja, od strane projektnog tima prema poslovanju. Voditelj promjena određuje što će se, kada i kako komunicirati, koji će kanali komunikacije biti korišteni, te kako će ljudi komunicirati prema vodstvu projekta.

Voditelj kvalitete projekta ima nadzor nad ukupnom realizacijom projekta, posebno u smislu primijenjene metodologije rada na projektu, kao i nadzora nad točkama provjera kvalitete. Radi u suradnji s vodstvom projekta i daje preporuke za osiguranje isporuke u skladu s dogovorenim opsegom, trajanjem i budžetom.

Voditelji timova odgovorni su za planiranje, organiziranje i koordinaciju resursa pojedinih timova na projektu. Voditelj tima može biti odgovoran za neke potprojekte, odnosno preuzima dio poslova vođenja potprojekta. Pri izvršavanju ovih zadataka, odgovara voditelju projekta, kojeg redovito izvještava o statusu potprojekata.

Osim projektnih funkcijskih timova, tijekom projekta, povremeno se definiraju i tzv. paralelni timovi. Neki članovi funkcijskih timova imenuju se, primjerice, u tim za matične podatke ili u tim za provedbu testiranja i sl.

Voditelji paralelnih timova:

- voditelj tima za matične podatke — *Data Lead*
- voditelj testiranja — *Test Lead*
- voditelj obuke korisnika — *Training Lead*
- voditelj pripreme za produkciju — *Cut-Over Manager*
- voditelj rješavanja testnih problema — *Defect Manager*
- voditelj postprodukcijske podrške — *Hypercare Manager*
- voditelj rješavanja postprodukcijskih grešaka — *Incident Manager*

14.2 Projektne faze

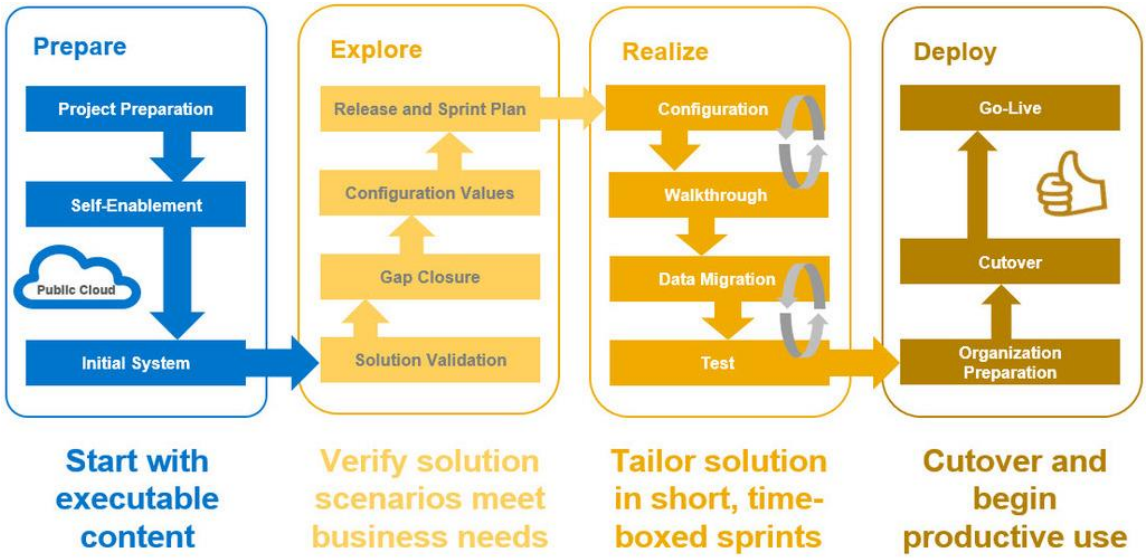
Za implementaciju standardnih informacijskih sustava, ima različitih metodologija, a donedavno je najraširenija bila metoda s pet faza:

- priprema projekta
- konceptualni dizajn (*Business Blueprint*)
- prilagodba sustava (konfiguracija i programiranje)
- testiranje i obuka korisnika
- produkcija (*Go-Live*) i održavanje.

U novije vrijeme, ERP gigant SAP počeo je koristiti novu metodu s četiri faze, nazvanu *Activate* [14]. Ova metodologija temelji se na kombnaciji agilnog i vodopadnog pristupa. Koristi se iterativnim postupkom implementacije s mogućnošću veće fleksibilnosti na promjene tijekom projekta. Opisuje relevantne isporuke i nudi predloške dokumenata za pojedine projektne aktivnosti.

Na temelju poslovnih prioriteta, projektni zadaci (poslovni zahtjevi) realiziraju se kroz pojedinačne cikluse i završavaju potvrdom od strane poslovnih korisnika. Nekoliko pojedinačnih ciklusa završava integracijskim ciklusom, koji objedinjuje korisničku potvrdu poslovnog scenarija. Utjecaj vodopadnog pristupa, očituje se u izvršenju svih projektnih aktivnosti u jednom projektnom ciklusu izvršenja. Utjecaj agilnog pristupa, vidi se u granuliranoj strukturi od više manjih ciklusa i potvrda, od strane korisnika, unutar svake faze.

Početna faza obuhvaća planiranje i pripremu projekta, uključujući organizaciju projekta, kao i raspored, proračun i upravljanje planovima (planovi rizika i kvalitete). Budući da svaki projekt ima svoje jedinstvene ciljeve, opseg i prioritete, navedene isporuke trebaju osigurati inicijaciju i planiranje koraka na učinkovit i djelotvoran način. Okruženje sustava je postavljeno, uključujući i najbolju praksu s procesima spremnim za izvođenje.



Slika 69: SAP *Activate* metodologija (izvor: SAP SE)

U fazi istraživanja, korisnički tim istražuje funkcionalnosti SAP rješenja, a implementator istražuje poslovne procese korisnika. Na zajedničkim *fit-gap* radionicama, timovi surađuju s ciljem identificiranja potrebne konfiguracije i eventualnih modifikacija, koje će najbolje odgovarati zahtjevima korisnika.

U fazi realizacije, korisnički tim i tim implementatora, zajedničkim naporima, na temelju prioritetnih zahtjeva identificiranih u fazi istraživanja, konfiguriraju i proširuju funkcionalnost sustava. Konfiguracija i implementacija izvode se u kratkim, iterativnim ciklusima, uz redovitu provjeru valjanosti procesa i dobivenih povratnih informacija od korisnika. Aktivnosti provedbe strukturiranog testiranja i migracije podataka, trebaju osigurati kvalitetu budućeg sustava.

Faza isporuke ima za cilj finalizirati spremnost rješenja i pratećih alata i postupaka za prijelaz u produkcijski rad. To uključuje ispitivanje sustava, obuku krajnjih korisnika, upravljanje sustavom, kao i *Cutover* aktivnosti (uključujući i migraciju podataka i početnu podršku nakon *Cutovera*).

Nakon puštanja sustava u rad, slijedi podrška. Primarni ciljevi pružanja podrške su dodatna optimizacija i stabilizacija operativnosti rješenja, zaključenje liste otvorenih pitanja i primopredaja projekta.

Elementarna razlika agilnog projektnog pristupa, u odnosu na vodopadni, je način na koji se izvršava faza realizacije. SAP *Activate* koristi iterativni inkrementalni pristup, u kojem se primjenjuju kraći ciklusi isporuke s potvrdama korisnika na kraju svakog ciklusa. To je u suprotnosti s tradicionalnim vodopadnim pristupom, gdje poslovni korisnici intenzivno sudjeluju u fazi konceptualnog dizajna, na temelju kojeg konzultanti realiziraju buduće rješenje, uz minimalno sudjelovanje poslovnih korisnika (poslovni korisnici se ponovno aktivno uključuju tek u trenutku korisničkog testiranja sustava).

U SAP *Activate* metodologiji poslovni su korisnici uključeni u svaku fazu projekta.

15 Vođenje promjena

Vođenje promjena disciplina je koja, strukturiranim pristupom, osigurava uspješan prijelaz ljudi, poslovnih procesa i organizacije, od početnog do željenog, budućeg stanja.

Općenito, vođenje promjena (engl. *change management*) najviše se zasniva na komunikaciji, edukaciji i treningu. Zato voditelj promjene (engl. *change manager*) treba pripremiti planove komunikacije, edukacije i treninga. Naime, nije dovoljno znati samo zašto je promjena potrebna. Ljudi moraju biti sposobni raditi nove stvari koje nikada prije nisu radili, odnosno treba im obuka.

Postoji nekoliko bitnih razlika između voditelja projekta i voditelja promjena. Posao voditelja projekta, prvenstveno je upravljanje serijom zadataka, resursa i aktivnosti, unutar određenog vremena i dogovorenog budžeta, u svrhu kreiranja jedinstvenog proizvoda, usluge i rezultata.

S druge strane, posao voditelja promjena, relativno je nepoznata uloga. Radi se na osiguravanju da će se koristi, koje su predviđene kao rezultat projekta, stvarno dogoditi. Voditelj promjena brine se o tome što bi eventualno, s ljudske točke gledišta, moglo „poći po zlu“. Jedan od njegovih važnih zadataka je osiguravanje da će se potrebne i dogovorene edukacije i treninzi u potpunosti provesti, a da će poslovanje na kraju projekta, stvarno postati vlasnik svih novih procesa (što je gotovo nemoguća misija ako bi se projekti, s tzv. razornim promjenama, radili bez voditelja promjena).

Uloga voditelja promjena blisko je vezana za komunikaciju, te on mora naći odgovore na pitanja što će se komunicirati, kako će se komunicirati, koji će kanali komunikacije imati najbolje efekte, i kako će ljudi komunicirati prema vodstvu projekta. Zato bi se osnovni preduvjeti, za biti dobar voditelj promjena, mogli sažeti u sljedeće četiri stavke:

- relevantno poslovno iskustvo
- slobodna i komotna komunikacija s pripadnicima svih korporativnih razina
- pristanak na rad u pozadini (dopuštanje drugima da budu heroji)
- sposobnost da se bude dobar trener, mentor i savjetnik.

Voditelj promjena mora u svom poslu biti uporan, dosadan, provokativan i optimističan. Mora biti spreman govoriti ljudima nepopularne stvari i ostati uporan u objašnjavanju potrebe za provođenjem projektnih zahtjeva. Mora imati i sposobnost utjecaja na ljude u svrhu uspješnog provođenja promjene, i treba moći pregovarati s visokopozicioniranim menadžerima radi dobivanja potrebne podrške i suradnje.

Ljudima je potrebno, u više navrata, davati ključne informacije: uvodna objašnjenja, informacije o trenutnom statusu, dodatna objašnjenja, kad god se osjeti da je to potrebno. Budući da raditi nešto novo znači griješiti, slobodno se može zaključiti da su greške veće i brojnije, što je promjena razornija. Zato voditelj promjena mora pripremiti poslovanje za takvu situaciju. Ljudi žele znati kako će trebati raditi u budućnosti i kako će se trebati ponašati u novim okolnostima. Zato s njima treba neprestano komunicirati i pružati im svu moguću pomoć, kako bi što prije došli u stanje povjerenja i prihvaćanja novih okolnosti.

Nakon što se definiraju početni detalji oko projekta i nakon što se uspostavi osnovni projektni tim, pokreću se aktivnosti, čiji je cilj ustanoviti koji se sve detalji (stavke)

promjene događaju. Zatim će uslijediti analiza i razrada njihovog utjecaja na poslovanje, pa izrada komunikacijskog plana, i, na kraju, planiranje i provedba edukacije i treninga.

Zanimljivo pitanje je o mogućoj poziciji voditelja promjena, u sklopu projekta implementacije nekog informacijskog sustava. Dvije su moguće pozicije, mogle bi se nazvati modelima A i B:

- model A – voditelj promjena u paraleli s voditeljem projekta; voditelj promjena je odgovoran sponzoru i programskom odboru, nezavisno od voditelja projekta; voditelj promjena sudjeluje na svim sastancima programskog odbora i ima direktni pristup svim korporativnim razinama
- model B – voditelj promjena kao dio projektnog tima; odgovoran je direktno voditelju projekta; ne sudjeluje na sastancima programskog odbora i nema pristup visokim upravljačkim strukturama.

Prednost je modela A sloboda djelovanja, povećani utjecaj i aktivni doprinos vođenju projekta. Mana ovog modela je mogući sukob voditelja projekta i voditelja promjena. Prednost modela B upravo je jasna hijerarhija (koja sprečava prethodno navedene sukobe), a mane modela B su manjak utjecaja, pasivni pristup i ograničeno djelovanje.

15.1 Analiza razlika (*Fit-Gap Analysis*)

Analiza razlika uspoređuje početnu i krajnju točku projekta, definirajući tako sve razlike koje se trebaju riješiti. Ishod je analize razlika razumijevanje različitosti trenutnog i budućeg stanja, stoga je neophodni dio procesa vođenja promjena.

Koraci procesa analize razlika:

- interaktivne vođene radionice
- vođenje rezultata u jednostavnim tablicama za prikaz razlika
- identificiranje potrebnih stavaka promjene
- prioritiziranje stavaka
- definiranje problema i rizika.

Rezultat analize razlika predstavlja osnovu za sljedeći korak procesa, a to je dublje specificiranje samih stavaka promjene i analiza njihovih utjecaja na poslovanje.

Sadržaj *Fit-Gap Excel* dokumenta:

- jedinstvena oznaka stavke promjene
- naziv i opis stavke promjene
- poslovna funkcija – marketing, operacije, prodaja
- kategorije stavaka promjena – *Fit, Fit with Process change, Gap*
- *Gap* vrsta – konfiguracija, podaci, poboljšanje, sučelje, procesna varijanta
- korist od implementacije stavke promjene
- ukupni učinak ako se ne implementira
- vlasnik stavke promjene.

Važan rezultat analize razlika je korektno kategoriziranje svih stavaka promjene:

- uklapa se (engl. *fit*) – svi poslovni zahtjevi su ispunjeni, a poslovni proces se ne mijenja upotrebom novog softvera
- uklapa se uz promjenu (engl. *fit with change*) – svi poslovni zahtjevi su ispunjeni, ali se poslovni proces mora prilagoditi novom softveru
- značajna razlika (engl. *gap*) – poslovni proces nije pokriven novim softverom, potrebna je odluka o tome hoće li se ići u dodatni razvoj softvera ili će poslovni proces biti pokriven na neki drugi način.

15.2 Analiza utjecaja promjene (CIA, *Change Impact Assessment*)

Analiza utjecaja promjene (CIA) postupak je koji analizira utjecaj i posljedice predmetne promjene na sve aspekte poslovanja i koristi se kao podloga za donošenje odluka vezanih za samu promjenu. Jednom kad je promjena propisno definirana, važno je provesti potpunu procjenu utjecaja.

CIA je u srcu procesa vođenja promjena i, kada je korektno napravljena, omogućava dobivanje jasnog pogleda na sve rizike i zahtjeve za kontinuitetom poslovanja. CIA se izrađuje na višednevnim izoliranim radionicama, uz obavezno prisustvo vodećih ljudi iz centralnog i lokanog poslovanja.

Sadržaj dokumenta analize utjecaja promjene:

- kategorije stavaka promjena – *Fit, Fit with Process change, Gap*
- opis stavke promjene – trenutno stanje i buduće stanje
- ukupni utjecaj (djelovanje) – visok / srednji / niski
- korist od implementacije
- rizici za uspješnu implementaciju
- plan akcije za implementaciju promjene.

15.3 Komunikacija i uključivanje ključnih ljudi

Uključivanje ključnih ljudi (engl. *stakeholders*) središnji je faktor uspjeha cijelog procesa promjene. Uključivanjem ključnih ljudi, voditelj promjena uspostavlja zamah cijelog projekta. Voditelj promjena mora ustrajati u održavanju svakodnevne komunikacije s ključnim ljudima.

Svi ključni ljudi iz poslovanja moraju biti prepoznati i uključeni u proces komunikacije. Neuključivanje nekih bitnih ljudi, može biti veliki rizik za uspjeh projekta. Poznato je da će bilo koji tim raditi bolje ako postoji potrebna razina jasnoće oko pojedinačne i zajedničke odgovornosti, vezane za odgovarajuće zadatke.

Taj se zaključak može primijeniti i na komunikacijske aktivnosti u sklopu vođenja promjena, te se, u tu svrhu, definira matrica komunikacijske odgovornosti RACI-X:

- R = odgovoran za izvršenje (*Responsible*)
- A = odgovoran za rezultat (*Accountable*)
- C = konzultiran (*Consulted*)
- I = informiran (*Informed*)
- X = isključivo odgovoran.

Komunikacija je u srcu svih uspješnih inicijativa promjena. Zato razumijevanje različitih komunikacijskih pristupa, i kada je koji prikladan za korištenje, čini kritičnu razliku između uspjeha i neuspjeha komunikacijskih napora, kao i uspjeha provedbe same promjene. Voditelj promjene treba segmentirati primatelje informacija i jasno identificirati njihove potrebe za informacijama.

Primjeri komunikacijskih kanala su: e-pošta, printane lokalne novosti, info portali, velike ploče s obavijestima, TV monitori s porukama i prezentacijama, intranet, radionice, prezentacijske sesije s diskusijom itd. Potrebno je koristiti što više različitih kanala, jer svaki od njih može prenijeti različite poruke na različite načine, odnosno drugačije djeluje na primatelje. Priprema dobrog plana komunikacije, znači pažljivo određivanje ciljane publike, ključne poruke, glavnih aktivnosti, kao i definiranih odgovornosti i vremena izvedbe.

Najteži dio projekata, koji se nalaze u području visoke složenosti i kaotičnosti, balansiranje je između održavanja redovnog poslovanja i projektnih zahtjeva. Česta je pojava sukoba između potrebe da se održava kontinuirana proizvodnja, i potrebe da se na vrijeme izvrše sva potrebna testiranja nove opreme, i školovanje radnika na proizvodnim linijama. Bez kontinuirane komunikacije i naglašavanja značaja projekta za budućnost tvornice, to jednostavno „ne bi išlo“.

Tri su uobičajena načina rada voditelja promjena:

- neutralno vođenje promjena s praćenjem standardne metodologije
- iskustveno vođenje promjena s aktivnim uključivanjem u rad projektnog tima
- intuitivno vođenje promjena s povremenim uključivanjem u vođenje projekta.

Prvi je način zadovoljavajući u slučaju jednostavnih projekata. Međutim, kod takvih projekata, postavlja se pitanje ima li uopće potrebe za pozicijom voditelja promjena.

Drugi način primjenjuje se kod složenih projekata, kada se pozicija voditelja promjena pokazuje neophodnom, a na poziciju se redovito postavlja osoba s relevantnim iskustvom na sličnim projektima. Voditelj promjena prema potrebi se aktivno uključuje u redovne projektne aktivnosti (npr. priprema testiranja ili nadzor nad pojedinim implementacijskim fazama).

Treći način primjenjuje se kod tzv. kaotičnih projekata, kada se, s obzirom na to da ne postoje prethodna iskustva, često intuicija pokazuje jedinim mogućim načinom vođenja. Voditelj promjena tada djelomično preuzima koordinaciju rada projektnog tima.

16 Završni komentar

Temi programskog inženjerstva ovdje je pristupljeno na jedan posve eksperimentalan način. Ne govori se o programiranju i programskim jezicima, niti se daju savjeti programerima. U uvodu je naglašeno da programsko inženjerstvo nije programiranje. Ista tvrdnja stoji i za informacijske sustave.

Knjiga je napisana iz pogleda osobe koja treba od korisnika saznati što se točno zahtijeva s poslovne strane, osobe koja programerima daje upute što trebaju isprogramirati, osobe koja treba zajedno s korisnicima istestirati rezultate programiranja, i osobe koja će, na kraju, zajedno s korisnicima, pustiti u život rezultat rada programera.

Prikazano je, između ostalog, kako skupiti informacije o tome što se točno treba isprogramirati, odnosno koja pitanja se mogu koristiti da bi se od poslovnih korisnika saznalo što im točno treba u njihovu radu. Spomenuto je koji tip funkcionalnosti trebaju operativci, a koji menadžeri.

U radu je (osim literature) korišteno autorovo iskustvo u radu s operativcima i menadžerima iz nekoliko velikih svjetskih korporacija, kao i s programerima iz Hrvatske, Europe i Indije. U radu s Indijcima, naučena je jedna ključna lekcija: sve im se može objasniti i oni sve mogu napraviti ako se dovoljno pažnje obrati (izraženim) kulturološkim različitostima. Ali to je već jedna druga tema.

17 Literatura

1. G. Piccoli, F. Pigni: Information Systems for Managers 4.0, Prospect Press, 2019.
2. Jović. N. Frid, D. Ivošević: Procesi programskog inženjerstva, Sveučilište u Zagrebu, skripta, 2019.
3. R. Stephens: Beginning Software Engineering, Wrox, 2015.
4. A. Kummarath, H. Mohapatra: Fundamentals of Software Engineering, BPB Publications, 2020.
5. A. Jović, M. Horvat, I. Grudenić: UML dijagrami, zbirka primjera; Graphis, 2014.
6. I. Sommerville: Software Engineering, Pearson, 10th edition, 2015.
7. I. Sommerville: Engineering Software Products: An Introduction to Modern Software Engineering; Pearson, 2019.
8. R. S. Pressman: Software Engineering – a Practitioner's Approach McGraw-Hill Education, 2014.
9. G. O'Regan: Concise Guide to Software Engineering, Springer, 2017.
10. K. Klarin: Programsko inženjerstvo Sveučilište u Splitu, skripta, 2012.
11. R. Idlbek, O. Hip: Informacijske tehnologije u poslovanju, Veleučilište u Požegi, skripta, 2017.
12. M. Pavlić: Informacijski sustavi Školska knjiga, 2011.
13. Ž. Panian, K. Ćurko: Poslovni informacijski sustavi Sveučilište u Zagrebu, 2010.
14. S. Denecken, J. Musil, S. Santhanam: SAP Activate Project Management SAP Press, Rheinwerk Publishing, 2020.
15. J. Nađ: Izrada i moguće primjene analitičkog modela električnog stroja u ERP sustavu; Konex, 2020.