Analiza agilnih metodologija za razvoj informacijskih sustava

Povoljnjak, Tomislav

Sveučilište u Zagrebu

Fakultet prometnih znanosti

DIPLOMSKI RAD

ANALIZA AGILNIH METODOLOGIJA ZA RAZVOJ INFORMACIJSKIH SUSTAVA

AGILE METHODOLOGIES ANALYSIS FOR THE INFORMATION SYSTEMS DEVELOPMENT

Mentor: Doc. dr. sc. Marko Matulin

Student: Tomislav Povoljnjak

JMBAG: 0135245496

Zagreb, rujan 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET PROMETNIH ZNANOSTI
POVJERENSTVO ZA DIPLOMSKI ISPIT

Zagreb, 4. svibnja 2022.

Zavod: **Zavod za informacijsko komunikacijski promet**
Predmet: **Analiza i modeliranje prometnih sustava**

DIPLOMSKI ZADATAK br. 6688

Pristupnik: **Tomislav Povoljnjak (0135245496)**
Studij: Promet
Smjer: Informacijsko-komunikacijski promet

Zadatak: **Analiza agilnih metodologija za razvoj informacijskih sustava**

Opis zadatka:

U radu prikazati značajke informacijskih sustava i njihova životnog ciklusa. Identificirajući osnovne procese takvih sustava, prikazati prikladnost korištenja UML jezika za njihovo modeliranje na različitim razinama promatranja. Korištenjem relevantne literature, opisati tradicionalne i agilne modele razvoja informacijskih sustava. Detaljno analizirati mogućnosti primjene RUP metodologije te evaluirati njene prednosti i nedostatke u odnosu na ostale agilne metodologije u kontekstu kvalitete ispunjavanja ciljeva sustava definiranih kroz korisničke zahtjeve.

Mentor:

doc. dr. sc. Marko Matulin

Predsjednik povjerenstva za diplomski ispit:

prof. dr. sc. Štefica Mrvelj

SAŽETAK

Informacijski sustavi se smatraju temeljnim alatima modernog društva pa tako i poslovanja. Razvoj informacijskih sustava je kompleksan i poprilično zahtjevan proces u kojem informacijski sustav koji se nastoji izgraditi prolazi u svojem životnom vijeku kroz određene faze koje se nazivaju životni ciklus razvoja informacijskih sustava (*System Development Life Cycle* – SDLC). SDLC se najčešće sastoji od šest faza, a to su planiranje i analiza, definiranje i dokumentiranje zahtjeva, dizajniranje arhitekture proizvoda, faza izrade, faza testiranja i faza implementacije. Navedeno predstavlja teorijski temelj na kojem se baziraju metodologije razvoja informacijskih sustava. Odabir ispravne metodologije je ključan aspekt prilikom izrade informacijskih sustava. Ovaj rad će prikazati osnovne karakteristike tradicionalnih i agilnih metodologija razvoja informacijskih sustava s posebnim naglaskom na *Rational Unified Process* – RUP. RUP je iterativna metoda razvoja informacijskih sustava koja kombinira najbolje prakse tradicionalnih i agilnih metodologija.

SUMMARY

Information systems are considered fundamental tools of modern society and thus of business society. The development of information systems is a complex and quite demanding process in which the information system that is being built goes through certain stages during its lifetime, which in technical literature has a name *System Development Life Cycle* - SDLC. The SDLC usually consists of six phases: planning and analysis, defining and documenting requirements, designing the product architecture, manufacturing phase, testing phase and implementation phase. SDLC represents the theoretical foundation on which the methodologies for the development of information systems are based. Choosing the right methodology is a key aspect when creating information systems. This paper will present the basic characteristics of traditional and agile methodologies for the development of information systems with a special emphasis on the *Rational Unified Process* - RUP. RUP is an iterative method of developing information systems that combines the best practices of traditional and agile methodologies.

Sadržaj

1. Uvod

Informacijsko komunikacijske tehnologije i njihov intenzivan razvoj u posljednjih nekoliko desetljeća uvelike su promijenile tempo života ljudi i njihovu svakodnevicu. Današnja ljudska svakodnevica bazira se na primjeni informacijskih sustava, aplikacija, platformi i usluga za obavljanje najosnovnijih ljudskih potreba, kao što su npr. potreba za komunikacijom, informiranjem i sl. Nadalje, što se tiče poslovnog aspekta, primjena modernih i inovativnih informacijskih sustava i usluga omogućava digitalizaciju i automatizaciju poslovnih procesa društva, tvrtke ili organizacije te se iz tog razloga s pravom smatraju temeljem poslovnog digitalnog društva. Razvoj kvalitetnog informacijskog sustava zahtjeva kvalitetnu i opsežnu analizu poslovnih procesa i korisničkih zahtjeva što nije nimalo jednostavno te stoga primjena ispravne metodologije može biti ključna u razvoju konačnog proizvoda, odnosno informacijskog sustava. Osim odabira metodologije za kvalitetan konačni proizvod, potrebno je ispoštovati detaljan prolazak kroz sve procesne korake razvoja za što se u stručnoj literaturi uvriježio naziv životni ciklus razvoja sustava (*System Development Life Cycle* - SDLC). U ovom radu bit će napravljena analiza agilnih metodologija razvoja sustava s naglaskom na ujedinjen proces razvoja sustava - RUP i njegove faze u razvoju informacijsko komunikacijskih sustava.

Cilj ovog istraživanja je analizirati primjenjivost odabrane agilne metodologije za razvoj informacijsko komunikacijskog sustava, u ovom slučaju RUP metodologije i prikazati njena osnovna načela, i primjenjivost ovisno o veličini projekta, odnosno kompleksnosti sustava koji se nastoji razviti te utjecaj na konačne atribute njegove kvalitete. Svrha istraživanja je analizirati prednosti i mane pojedine metodologije za razvoj informacijsko komunikacijskih sustava, s naglaskom na *RUP* koristeći dostupne izvore u kojima su opisani pojedini slučajevi uporabe različitih metodologija te također prikazati faze razvoja sustava kao i dokumente koji proizlaze kao rezultat pojedine faze. Rad je podijeljen u sedam cjelina:

1. Uvod
2. Informacijski sustav i njihov životni ciklus
3. Upravljanje projektom razvoja informacijskog sustava
4. Modeli i metodologije za pristup izgradnji IS
5. Modeliranje sustava primjenom UML jezika
6. Analiza koraka razvoja sustava primjenom RUP metodologije
7. Zaključak.

U drugom poglavlju definiran je pojam sustava, što ga čini i koji su njegovi osnovni elementi. Također, pružen je pogled na temeljne elemente informacijskih sustava te je opisan životni ciklus informacijskog sustava te su detaljno prikazane njegove faze i aktivnosti koje projektni tim poduzima u pojedinoj fazi. U trećem poglavlju diskutirana je važnost upravljanja projektom

razvoja informacijskog sustava te su predstavljene najčešće uloge projektnog tima koji sudjeluje u razvoju informacijskih sustava. Naglasak je stavljen na ulogu voditelja projekta koji je u najvećoj mjeri odgovoran za ispravno upravljanje projektnim aktivnostima. Nadalje, prikazana je važnost timskog rada i utjecaj dobre komunikacije između članova projektnog tima na ispunjavanje projektnih aktivnosti.U četvrtom poglavlju predstavljena su dva temeljna pristupa razvoju i izgradnji informacijskih sustava, a to su tradicionalni modeli razvoja informacijskih sustav kao što su vodopadni model, prototipiranje, spiralni model itd. te agilni modeli i metodologije kao što su *Scrum*, *Kanban*, *Lean* i RUP koji se može smatrati kombinacijom tradicionalnih i agilnih metodologija.

U petom poglavlju prikazana je važnost dokumentiranja procesa modeliranja i razvoja i utjecaj UML grafičkog jezika na navedeno. Prikazani su najčešći oblici UML dijagrama koji se koriste za dokumentiranje i vizualizaciju tijekom svih faza razvoja informacijskih sustava.

U šestom poglavlju detaljnije je analizirana RUP metodologija za razvoj informacijskih sustava te su prikazane njene faze i produkti koji nastaju kroz te faze. Također, analizirane su prednosti koje RUP donosi u razvoju informacijskih sustav u odnosu na ostale vrste metodologija. Na kraju je donesen zaključak na obrađenu tematiku.

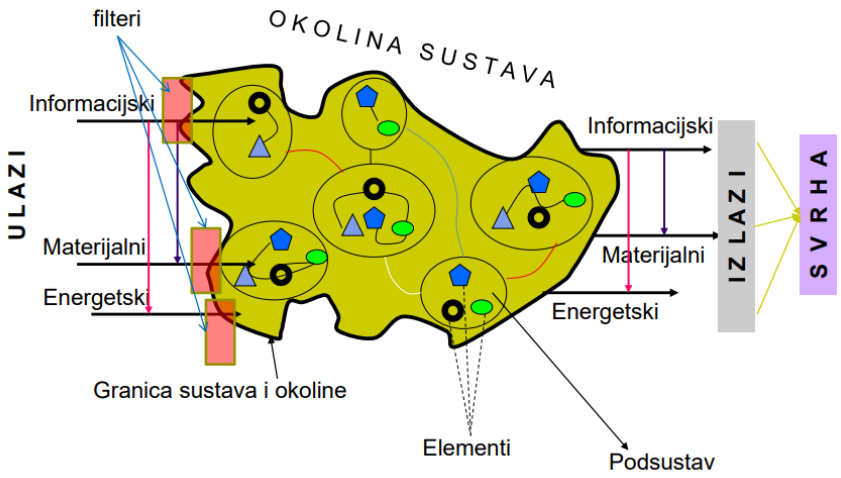
2. Informacijski sustavi i njihov životni ciklus

Intenzivni razvoj novih tehnologija, ekspanzija područja obuhvata i primjene informacijsko komunikacijskih tehnologija u društvu te procesne i tehnološke mogućnosti koje se javljaju u posljednjih nekoliko desetljeća uvelike doprinose promjeni životnog tempa ljudi. Informacijsko komunikacijske tehnologije i njihova intenzivna i sveobuhvatna primjena smatraju se temeljem digitalnog društva te se ljudska svakodnevica u velikoj mjeri oslanja na moderne tehnologije. Trenutno je vrlo malo društvenih područja koje na bar neki način nisu obuhvaćene informacijsko komunikacijskim tehnologijama (*Information and Communication Technologies* – ICT) pa se tako tehnološke mogućnosti iz dana u dan proširuju što rezultira razvojem novih i inovativnih informacijskih sustava u raznim društvenim sferama.

Za razumijevanje tematike i potrebe za razvojem informacijskih sustava potrebno je krenuti od određenih osnova. Pod osnove se podrazumijevaju pojmovi kao što su sustav te pitanja vezana u sustav, a to su npr. što ga čini, što je u njegovu okruženju, koja je njegova uloga te koje su njegove vrste.

2.1. Teorijska podloga informacijskih sustava

Sustav se može definirati kao svaki uređeni skup međusobno povezanih elemenata koji zajedničkom interakcijom ostvaruju funkciju cjeline. Sastoji se od niza elementarnih podsustava koji mogu biti više ili manje povezani, a njihova međusobna djelovanja i veze nazivaju se sučelja. Sustav je okružen svojom okolinom koja zapravo predstavlja dio cjeline koji nije obuhvaćen sustavom. Sama podjela sustava može se definirati s obzirom na više različitih pogleda, a najčešća je podjela na otvorene i zatvorene sustave. S obzirom na povezanost sustava s okruženjem, dijele se na zatvorene i otvorene ovisno o tome razmjenjuju li informacije, materiju i energiju s okruženjem ili ne, odnosno jesu li adaptivni prema promjenama u okruženju [1]. Strukturu sustava čine njegove komponente te njihov položaj u međusobnom odnosu te u odnosu na cjelinu. Funkcija sustava, odnosno svrha postojanja se očituje u ulozi koju sustav obavlja u svojoj okolini te način ostvarivanja te svrhe. Jasno definirana funkcija, odnosno svrha omogućavaju utvrđivanje zahtijevanih komponenti i njihovih međusobnih odnosa, odnosno strukture. Sam cilj sustava se ispunjava kontinuiranim i cikličnim procesom transformacije ulaza u izlaze, dok sam sustav ovisi o konstantnom prilivu materije, energije i informacija kako bi opstao, odnosno kako bi se stanje njegove nesređenosti/kaosa koje se definira kao entropija sveo na minimum [2]. Sam sustav i njegova struktura prikazani su na slici 1.



Slika 1. *Sustav i njegova struktura, [2].*

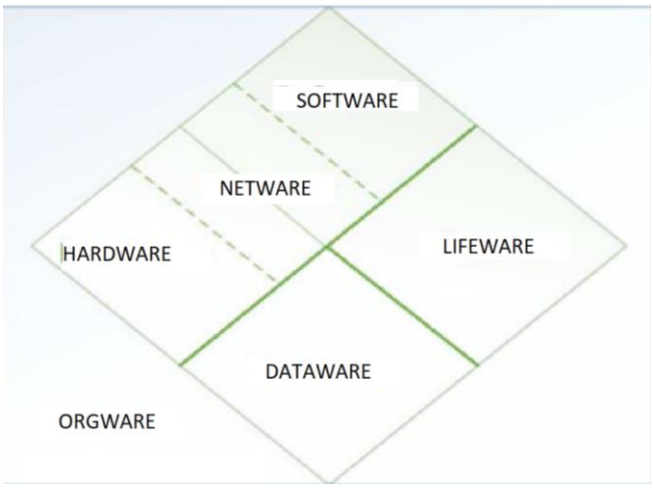
Informacijski sustav se definira kao sustav kojeg sačinjavaju ljudi, programska i računalna oprema koja je napravljena, oblikovana i dovedena u operativno stanje te služi skupljanju, zapisivanju, spremanju i pronalaženju te prikrivanju informacija u odgovarajućem obliku. Informacijski sustavi su vrlo složeni zbog činjenice kako uključuju različite elemente i procese. Iz tog razloga ih je vrlo bitno dobro opisati, a navedeno se ostvaruje kroz pojam projektiranja, odnosno stvaranja koji uključuje prikupljanje kvalitetnih informacija temeljem kojih se omogućava donošenje odluka. Suvremeni informacijski sustavi i njihova primjena jedan su od glavnih aspekata ostvarivanja konkurentske prednosti na tržištu što je iz poslovnog gledišta vrlo bitno [2].

Informacijski sustav kao svoje ulazne i izlazne veličine prepoznaje informacije. Informacije su podaci koji primatelju donose neku relevantnu novost. Ako se informacija opisuje i s tehničkog aspekta ju može se definirati kao skup podataka (elemenata, događaja), koji se nižu po nekom određenom matematičkom zakonu vjerojatnosti. Informacijski sustav je uvijek podsustav nekog tehnološkog ili organizacijskog sustav čija je svrha permanentno opskrbljivanje potrebnim informacijama svih razina njegovog upravljanja i odlučivanja, odnosno omogućuje upravljanje tim organizacijskim ili tehnološkim sustavom ili njegovim podsustavom. Nadalje, informacijski sustav dio je svakog poslovnog sustava čija je funkcija neprekidna opskrba svih razina upravljanja, odlučivanja i svakodnevnog poslovanja potrebnim informacijama [1].

Informacijski sustav kao i svaki sustav sastoji se od međusobno povezanih elemenata u cjelinu čija je međusobna interakcija ključna za ispunjenje zadanog cilja. Prema [1], sastavni elementi svakog informacijskog sustava, kao što je prikazano na slici 2, su:

- *Hardware* – predstavlja fizički, odnosno materijalni dio informacijskog sustava, a u njega spadaju računala, mrežna oprema, itd.,

- *Software* – nematerijalni dio informacijskog sustava u obliku programskih rješenja ili algoritama koji upravljaju računalom ili se izvode na računalu,
- *Orgware* – organizacijski dio informacijskog sustava koji omogućuje logičko povezivanje i upravljanje komponentama informacijskog sustava, a očituje se kroz postupke, procedure i metode te pravila, ograničenja, prijenos znanja i upotrjebljene metodologije,
- *Netware* – mrežni dio informacijskog sustava, odnosno komponenta koja omogućuje komunikacijsko povezivanje elemenata i dijelova sustava u cjelinu,
- *Dataware* – komponenta informacijskog sustava vezana za organizaciju baze podataka i informacijskih resursa,
- *Lifeware* – ljudski faktor informacijskog sustava koji se sačinjava od svih aktera koji sudjeluju u dizajniranju, razvoju, upotrebi, analizi i upravljanju informacijskim sustavom kao što su npr. analitičari, dizajneri, programeri, poslovni korisnici, menadžment, itd.



Slika 2. *Elementi informacijskog sustava, [1].*

2.2. Životni ciklus razvoja informacijskog sustava – SDLC

Prilikom izrade informacijskih sustava potrebno je slijediti određene procesne faze i korake koji su ključni za razvoj kvalitetnog i suvremenog informacijskog sustava koji ispunjava sve zacrtane poslovne ciljeve, a u konačnici i svoju svrhu. Životni ciklus razvoja sustava - SDLC je proces koji se sastoji od niza planiranih aktivnosti za razvoj ili nadogradnju softverskih proizvoda. Koristi se od strane IT industrije, odnosno IT stručnjaka u procesima dizajniranja, razvoja i testiranja kompleksnih programskih rješenja koje zadovoljavaju korisničke poslovne potrebe. SDLC zapravo predstavlja procesni okvir za razvoj programskih rješenja. Sastoji se od detaljnog plana koji opisuje kako razviti, održavati i zamijeniti ili nadograditi određeno programsko rješenje. Životni ciklus definira metodologiju za unaprjeđenje konačnog proizvoda i kompletnog razvojnog procesa. Tradicionalni SDLC sastoji se kao što je prikazano na slici 3. najčešće od šest faza [3].

***Slika 3**. SDLC faze, [3].*

Iako je SDLC iterativni i cirkularni proces, kao početne faze procesa mogu se smatrati faza planiranja i analize zahtjeva (*Planning and Requirement Analysis*). Planiranje i analiza zahtjeva je jedna od ključnih i temeljnih faza u SDLC, a podrazumijeva prikupljanje informacija koji najčešće provodi analitički, odnosno razvojni tim s fokusom na provođenje kvalitetnih radionica s budućim korisnicima odnosno klijentima kako bi se prikupile informacije o poslovnim potrebama radi kojih se i kreće u razvoj ili nadogradnju sustava. Prikupljene informacije se zatim koriste kako bi se odredile temeljne smjernice projekta te se najčešće provodi studija izvodljivosti u ekonomskom, tehničkom i operativnom aspektu [3].

Nakon prve faze i prikupljanja svih potrebnih informacija o poslovnim procesima subjekta koji zahtjeva automatizaciju i razvoj novog programskog rješenja, sljedeća faza je faza definiranja i dokumentiranja zahtjeva (*Defining and Documenting Requirements*). Navedena faza podrazumijeva jasno i nedvosmisleno definiranje i dokumentiranje zahtjeva proizvoda koji se razvija i odobravanje tako dokumentiranih zahtjeva od strane krajnjeg korisnika ili klijenta kroz dokument specifikacije zahtjeva (*Software Requirement Specification* - SRS) kako bi se moglo krenuti u sljedeću fazu projektnog životnog ciklusa [3].

Treća faza je jedna o kompleksnijih i sadrži ponajviše aktivnosti projektnog tima koje se odnose na dizajn arhitekture samog sustava koji se razvoja, a naziva se faza dizajniranja arhitekture

proizvoda (*Designing the Product Arhitecture*). Specifikacija korisničkih zahtjeva koja je produkt prethodne faze smatra se referencom za razvoj arhitekture programskog rješenja koje se razvija. Najčešće se primjenjuje više pristupa prilikom dizajniranja arhitekture koji se dokumentiraju kroz dokument specifikacije koja se zatim daje na pregled svim važnim dionicima. Dokument specifikacije dizajna (*Design Document Specification* - DDS) sadrži procjenu rizika, robusnost projekta, modularnost dizajna te informacije o budžetu i vremenskim ograničenjima. Nakon pregleda dokumenta od strane dionika, donosi se odluka o pristupu koji se odabire za dizajn programskog rješenja. Dizajnerski pristup jasno definira sve arhitektonske module proizvoda zajedno s njegovom komunikacijom i prikazom protoka podataka s vanjskim modulima i modulima trećih strana (ako postoje). Unutarnji dizajn svih modula predložene arhitekture trebao bi biti jasno definiran s najsitnijim detaljima u DDS-u [3].

Faza izrade odnosno razvoja proizvoda kreće se sa stvarnim razvojem sustava. Generiraju se algoritmi i programski kod kao što je specificiran u DDS dokumentu koji je produkt prethodne faze. Razvojni tim slijedi smjernice i principe razvoja te standarde u razvoju koji su definirani ili od strane njihove vlastite organizacije ili na međunarodnoj razini. Razvojni tim se koristi programskim jezicima kao što su C, C++, Pascal, Java i PHP za razvoj programskog koda s obzirom na tip informacijskog sustava koji se razvija, odnosno njegov softverski dio [3].

Nakon faze razvoja slijedi faza testiranja koja se u modernim SDLC modelima najčešće provlači kroz sve faze razvoja informacijskog sustava. Kroz ovu fazu nastoji se otkriti sve greške, dokumentirati ih te popraviti od strane razvojnog tima nakon čega se funkcionalnosti ponovno testiraju. Cilj koji se nastoji ostvariti je podizanje standarda kvalitete kako je definirano u specifikaciji zahtjeva i u potpisanim ugovorima između ugovornih strana [3].

Posljednja faza podrazumijeva implementaciju informacijskog sustava na tržište, odnosno u produkciju te održavanje istog. Ponekad se implementacija programskog rješenja odvija u fazama prema poslovnoj strategiji te organizacije. Proizvod se može najprije izdati u ograničenom segmentu i testirati u stvarnom poslovnom okruženju (*User acceptance testing* – UAT). Zatim se na temelju povratnih informacija proizvod može objaviti takav kakav jest ili s predloženim poboljšanjima u ciljanom tržišnom segmentu. Nakon što je proizvod pušten na tržište, provodi se njegovo održavanje za postojeću bazu kupaca [3].

SDLC se može smatrati sustavnim pristupom rješavanju problema kroz faze koje sadrže više koraka, odnosno može se definirati kao vodič za detaljan opis koraka koje je potrebno odraditi kako bi se ispunili korisnički zahtjevi i ciljevi koji su postavljeni pred informacijski sustav, a sve u svrhu osiguravanja kvalitete i učinkovitosti projekta. SDLC osigurava slijed logičkih faza i njima pripadajućih koraka za razvoj (planiranje, izvršenje, nadzor) projekta pa tako svaka faza rezultira nekim isporučevinama (*deliverables*) koje se isporučuju u slijedeću fazu. Prilikom izvođenja bilo koje faze veoma je važno voditi brigu o kontroli kvalitete i isporuke kako bi se mogle donijeti kvalitetne i pravovremene odluke u daljnjim fazama [3].

Važan aspekt razvoja informacijsko komunikacijskog sustava predstavlja i odabrana metodologija koja određuje kako se proces razvoja treba odvijati, odnosno definiraju se različite faze i njihov opseg u procesu razvoja, način odvijanja aktivnosti i njihovo dokumentiranje. Metodologije proizlaze iz osnovne definicije SDLC procesa koji zapravo predstavlja opći model životnog ciklusa razvoja informacijskih sustava. Osnovna podjela metoda razvoja sustava je na tradicionalne i agilne metode koje će detaljnije biti razrađena kroz naredna poglavlja.

3. Upravljanje projektom razvoja informacijskog sustava

Informacijski sustavi se razvijaju kroz određene procesne korake, odnosno faze razvoja koje se razlikuju po razini kompleksnosti. Za ispunjavanje zadanih poslovnih ciljeva ključna je razmjena pravovremenih i kvalitetnih informacija. U tom pogledu informacijski sustavi i njihov razvoj moraju omogućavati obavljanje osnovnih funkcionalnosti koje zahtijevaju poslovni procesi organizacije. Pravilno rukovanje i upravljanje procesom razvoja veoma je važno za razvoj informacijskog sustava koji je istovremeno kvalitetan i zadovoljava poslovne potrebe korisnika Nadalje korištenje najmodernijih tehnologija omogućava u okvirima zadanog budžeta visoku razinu automatizacije poslovnih procesa. Razvojni proces informacijskog sustava je uvijek definiran kroz pojam projekta.

Projekt kao takav jest privremeni, poduzetni pothvat čija je svrha stvaranje novog, jedinstvenog proizvoda, usluge ili rezultata. Sve ove značajke jesu razlika između projekta i operativnog rada koji se obavlja svakodnevno odnosno procesa obavljanja poslovnih aktivnosti, koje rezultiraju ispunjavanjem svakodnevnih poslovnih planova. Projekt je progresivna razrada odnosno projekt se razvija po koracima gdje nakon svakog dolazi do inkrementa odnosno napredovanja u razumijevanju samog krajnjeg rezultata (proizvod ili usluga). Projekt se može smatrati instrumentom za ispunjavanje zadanih poslovnih ciljeva, odnosno u kontekstu IT industrije projektom se smatra razvoj, implementacija i održavanje informacijskih sustava [4].

Prilikom razvoja tradicionalnih i procesno jednostavnih informacijskih sustava, brigu o razvoju vodio je najčešće IT menadžer, što se u zadnjih nekoliko desetljeća uvelike promijenilo zbog konstantnog razvoja novih tehnologija koje omogućavaju kreiranje i implementaciju kompleksnih sustava. S obzirom na navedeno, IT projekti su puno kompleksniji i zahtijevaju širi opseg stručnog i kompetentnog kadra sa specifičnim znanjima kako bi se osigurao kvalitetan razvoj. Ulogu IT menadžera preuzeo je projektni menadžer. Navedena uloga i opseg potrebnih znanja, a i sama kompleksnost aktivnosti i stručnost u tehničkim područjima uvelike se promijenila kroz godine. Projektni menadžeri koji imaju iskustvo razvoja tradicionalnih programskih rješenja će najvjerojatnije uspjeti u nadzoru i upravljanju procesom razvoja kompleksnih programskih rješenja, zbog usredotočenosti na budžet, raspored aktivnosti, resurse i kompletni projektni plan. Međutim, razvoj modernih informacijskih sustava čija se razina kompleksnosti iz dana u dan povećava, zahtjeva od projektnog menadžera dodatna znanja i upravljačke sposobnosti. Upravljanje projektom razvoja kompleksnih informacijskih sustava zahtjeva višu razinu integracije s internim i eksternim korisnicima. Mora se bazirati na kombinaciji tradicionalnog razvoja s poslovnom kreacijom i također zahtjeva uključenost korisnika od samog početka projekta kroz sve faze razvoja. Odgovornosti upravljanja projektom uvelike ovisi o veličini projekta, ali i donosi razne prepreke koje projektni menadžeri moraju moći savladati kako ne bi došla u pitanje uspješnost projekta [5].

Definiranje projekta je prvi korak u razvoju informacijskog sustava. Definiranjem projekta nastoje se ostvariti tri glavna zadatka, a to su identificiranje ciljeva projekta, identificiranje korisnika informacijskog sustava koji se razvija projektom i definiranje opsega projekta. Ciljevi projekta definiraju se kao rezultati koji se moraju postići tijekom projekta. Ciljevi predstavljaju referentnu točku svake aktivnosti koja se provodi tijekom projekta. Moraju biti dobro definirani, te sadržavati određene karakteristike [5]. Ciljevi trebaju biti specifični (*specific*), odnosno moraju biti vrlo jasni i precizni u izražavanju onoga što se želi postići. Nadalje, ono što se želi postići mora biti mjerljivo (*measurable*) u količinama, trajanju, svotama, postotcima i sl. kako bi se olakšalo mjerenje uspješnosti realizacije cilja. Ciljevi moraju biti ostvarivi (*achievable*) i realistični (*realistic*), odnosno ono što se želi postići mora biti ostvarivo i temeljeno na realističnim očekivanjima. Važno je ne postaviti previsoke ciljeve koje je nemoguće ostvariti, ali istovremeno preniski ciljevi neće predstavljati izazov čime se umanjuje produktivnost. Ciljevi trebaju imati zadani vremenski okvir (*time-bound*) u kojima se moraju postići, u suprotnom ako nema definiranih rokova može se dogoditi da se projekt nikada ne završi. Kombinacijom početnih slova engleskih verzija navedenih pridjeva dolazimo do akronima SMART koji se u stručnoj literaturi projektnog menadžmenta koristi za opis karakteristika koje ciljevi moraju sadržavati [1].

Drugi zadatak koji se nastoji ostvariti je identifikacija korisnika koji će koristiti informacijski sustav ili na bilo koji način biti u doticaju s njim u poslovnom smislu. Najčešća podjela korisnika je na interne korisnike, klijente ili kupce te potrošače. Vrlo je važno za uspješnost projekta napraviti distinkciju između važnih i nevažnih inputa koje korisnici mogu pružiti tijekom razvoja što predstavlja izazov s kojim se voditelji projekata moraju suočiti. Navedeno se postiže kroz radionice s korisnicima na kojima se radi prioritizacija bitnih i manje bitnih korisničkih zahtjeva. Treći zadatak je definiranje opsega projekta koji je usko povezan s budžetom i vremenskim ograničenjima projekta. Na voditelju projekta je odgovornost pregovora oko opsega projekta, odnosno onoga što se može napraviti u definiranom roku i korisničkih želja. Opseg projekta zapravo predstavlja domenu funkcionalnosti i značajki koje će biti uključene u informacijski sustav na temelju vremenskog roka i troškova koje razvoj takvih funkcionalnosti proizvodi [5].

Osim voditelja projekta, ključan u razvoju informacijskog sustava je i projektni tim. Projektni tim osigurava uspješno izvršavanje aktivnosti tijekom svih faza procesa razvoja na način da različite uloge projektnog tima brinu o različitim aspektima razvoja. Najčešće uloge projektnog tima za razvoj informacijskih sustava su [5]:

- Projektni menadžer (voditelj projekta) – odgovoran je za opseg posla, razvoj projektnog plana, izradu rasporeda izvršavanja aktivnosti, dodjelu resursa, budžetiranje, upravljanje projektnim timom, komunikaciju s korisnikom i izvještavanje uprave o napretku projekta. Voditelj projekta se također suočava s potencijalnim poslovnim problemima i ograničenjima koji uključuju pregovore i sklapanja ugovora, licenciranje proizvoda trećih strana, ali i proces odabira i zapošljavanja osoblja.

- *Account* menadžer – najčešće se radi o starijem menadžeru koji je zadužen za više projekata odjednom. Zapravo se radi o ulozi koja obavlja koordinaciju između projekata. Nadalje, *account* menadžer predstavlja korisničke potrebe razvojnom timu te ugovara prodaju novih proizvoda.
- Tehnički menadžer – ova uloga predstavlja osobu koja nadgleda razvojni tim programera, programera baze podataka i sistem integratora i omogućava komunikaciju s ostalim dijelom projektnog tima. Radi se o senior poziciji koju najčešće zauzima najiskusniji programer. Odgovoran je za korištenje i implementaciju ispravnih tehnologija na projektu.
- Programer – osoba u ovoj ulozi je odgovorna za pisanje koda aplikacije koja se razvija kroz projekt. Ove su aplikacije kodirane prema specifikacijama i mogu uključivati mnogo tehnologija uključujući ali ne ograničavajući se na poslužiteljske skripte, aplikacije baze podataka, aplete i ActiveX kontrole. Razvoj i programiranje kompleksnih informacijskih sustava i aplikacija podrazumijeva korištenje različitih programskih jezika, ali najčešće se radi o programskom jeziku Java, JavaScript, Visual Basic, VBScript, SQL i C/C++. Tehnički menadžer je najčešće nadređeni programerima, međutim ovisno o veličini projekta i kompleksnosti, može se dogoditi da postoje i različite razine programera, kao npr. senior programer, junior programer i sl.
- Poslovni analitičar – ovaj pojedinac odgovoran je za prikupljanje korisničkih zahtjeva, dizajniranje logičkih modela i arhitekture sustava, što uključuje procesne modele, podatkovne modele i procesne specifikacije. Poslovni analitičar je uloga u procesu razvoja koja predstavlja sponu između poslovnog dijela i IT dijela. Obavlja dokumentiranje poslovnih procesa koji se zatim od strane razvojnog tima pretvara u funkcionalan kod.
- Dizajner sučelja – ova uloga je odgovorna za kreiranje korisničkih sučelja aplikacije. Koriste se različitim alatima za dizajniranje predloška sučelja.
- Administrator baze podataka – zaduženi su za dizajn i razvoj fizičke baze podataka. Također, odgovorni su za normalizacije baza podataka kako bi se ubrzao rad sustava. Osim navedenog, administratori baze podataka uspostavljaju pohrane, odnosno skladišta podataka, replikaciju podataka i generiranje izvještaja.
- Mrežni/Sistem inženjer – odgovornost ove uloge je dizajniranje mrežnih konfiguracija koji će podupirati rad kompleksnih programskih rješenja. Često, mrežni inženjer, odgovoran je i za sigurnosne aspekte informacijskih sustava i mreža te uspostavljanje domena, e-mail servera, i sl.
- Sigurnosni stručnjak – uloga koja je odgovorna za enkripciju podataka i osiguravanje razmjene podataka između sustava.
- Specijalist za osiguravanje kvalitete – odgovoran je za kreiranje testnih scenarija kako bi se osiguralo da informacijski sustav radi u skladu sa specifikacijama. Svrha

testnih planova, koji se ponekad nazivaju i planovi prihvaćanja, je pružiti minimalni skup testova koji se moraju proći da bi sustav krenuo u produkciju.

- Tester – tester je uloga koja provodi planove testiranja koje je definirao specijalist za osiguravanje kvalitete. Prilikom provođenja testnih scenarija bilježe i prijavljuju sve greške koje se javljaju prilikom rada sustava kako bi se greške mogle ispraviti intervencijom ostalih članova projektnog tima.

U većini slučajeva prilikom razvoja novog informacijskog sustava neće se sve prethodno navedene uloge definirati u sklopu projektnog tima. Većina odgovornosti koje su ovdje podijeljene po specifičnim ulogama, u stvarnim projektima se dijele među različitim ulogama projektnog tima pa se tako projektni tim najčešće sastoji od voditelja projekta, poslovnog analitičara, programera, testera, sistem inženjera i administratora baze podataka.

4. Modeli i metodologije za pristup izgradnji IS

Razvoj informacijskih sustava je kompleksan i specifičan proces i za uspješni razvoj kvalitetnog informacijskog sustav ključna je prilagodba i odabir ispravne metodologije. Prilikom odabira metodologije za pristup izgradnji informacijskih sustava potrebno je definirati ključne kriterije kako bi rezultat projekta bio uspješan. Izbor modela i metodologije ovisi o ponajprije veličini i obujmu projekta informacijskog sustava. Također, sama priroda problema, odnosno ciljevi koji se nastoje ispuniti uvođenjem ili nadogradnjom informacijskog sustava su također vrlo važni aspekti pri odabiru metodologije. Kompleksni problemi vrlo često zahtijevaju kompleksne informacijske sustave za rješavanje problema, zbog njihove mogućnosti obrade velike količine podataka. Nadalje, vrlo važan aspekt predstavlja i vremenski rok stvaranja i isporuke informacijskog sustava kao i raspoloživa saznanja o problemu koji se nastoji riješiti razvojem informacijskog sustava. Naime, produživanje rokova razvoja rezultira najčešće povećanjem troškova razvoja s toga je prilikom odabira metodologije vrlo važno obratiti pažnju i na taj aspekt. Osim navedenog, potrebno je obratiti pažnju i na raspoložive resurse koje tvrtka koja razvija informacijske sustave može ponuditi. Pod resursima se ne smatra samo financijski aspekt, odnosno budžet klijenta već su tu bitni i ljudski resursi, dostupnost informacijsko komunikacijske opreme kao i dostupna infrastruktura [2].

Metodologije i modeli razvoja informacijskih sustava proizlaze iz općeg modela razvoja sustava SDLC-a koji je detaljno opisan u drugom poglavlju. Najčešća podjela modela razvoja informacijskih sustava je na tradicionalne i agilne metode razvoja od kojih svaka metodologija nosi svoje posebnosti i nudi razne razvojne mogućnosti. Tradicionalne metode su temelj razvoja informacijskih pa i drugih tipova sustava te se konstantnim razvojem novih tehnologija, ali i povećanjem korisničkih potreba i smanjenjem rokova isporuke javila potreba za većom prilagodljivosti metodologija razvoja što je rezultiralo definiranjem agilnih metodologija. Ključna riječ u upravljanju projektom razvoja informacijskog sustava je prilagodljivost, gdje se ponajprije misli na prilagodljivost veličini projekta i potrebama razvoja [6].

4.1. Tradicionalni modeli razvoja sustava

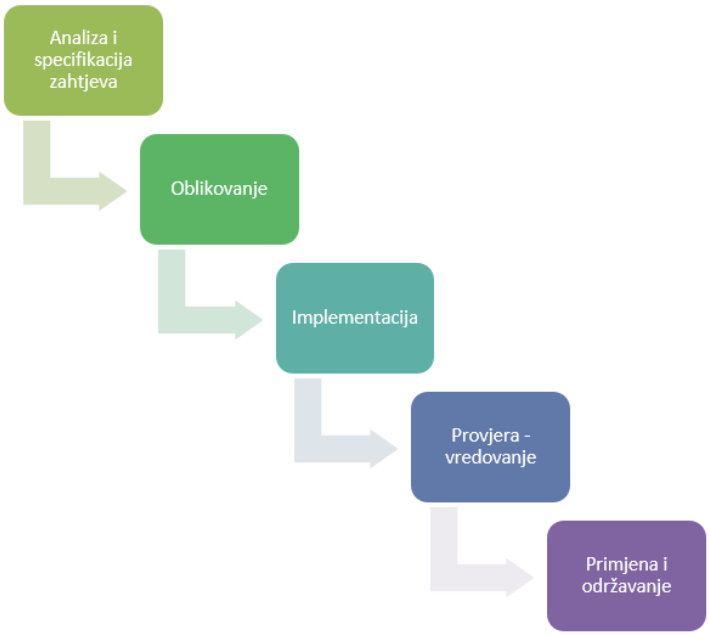
Tradicionalne metode razvoja sustava temelje se na unaprijed definiranom planu razvoja koji se provodi kroz strogo definirane faze. Sustavni razvoj započinje otkrivanjem korisničkih zahtjeva i dokumentiranjem kompletnog skupa zahtjeva, nakon čega slijedi faza razvoja i definicija arhitekture i dizajna sustava. Tradicionalne metodologije se smatraju teškaškim (eng. *heavyweight*) metodama razvoja sustava baš iz tog razloga jer ne dopuštaju previše konceptualnih i funkcionalnih izmjena tijekom različitih faza. Ciljevi koji se nastoje ostvariti primjenom tradicionalnog pristupa očituju se u detaljnom razumijevanju korisničkih potreba, izradi solidnog

vizualnog dizajna koji je potkrijepljen besprijekornim funkcioniranjem komponenti sustava kako bi sustav bio funkcionalan i zadovoljio potrebe korisnika. Stavljen je veliki naglasak na temeljito planiranje i suočavanje s rizicima koji se javljaju u procesu planiranja, razvoja, implementacije i korištenja. Planiranje podrazumijeva i definiranje više alternativnih puteva za postizanje željenih rezultata iz početnog stanja kako bi se kvalitetnom analizom odabrao najispravniji put s zadovoljavajućim rezultatima. Ovakav pristup razvoju sustava podrazumijeva da su procesi koji se nastoje automatizirati implementacijom sustava lako predvidljivi i da se mogu optimizirati i učiniti ponovljivim. Tradicionalni model razvoja sustava u konačnici je usmjeren na sam proces i pronalazak optimalnog načina za iskorištavanje koristi procesa kroz njegovu automatizaciju. Nadalje, sam proces razvoja koji se bazira na slijednom izvršavanju faza podrazumijeva produciranje velike količine tehničke i poslovne dokumentacije koja detaljno opisuje softverske komponente sustava i njihove tehničke i dizajnerske specifikacije [6].

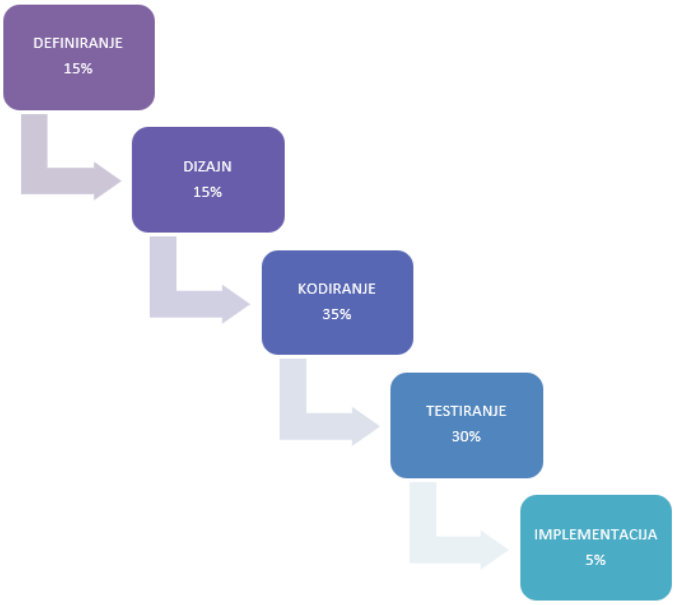
4.1.1. Vodopadni model

Jedna od najpoznatijih i među prvim tradicionalnim metodologijama je vodopadni (*waterfall*) pristup izradi sustava. Vodopadni pristup podrazumijeva slijedno napredovanje iz faze u fazu te ne dozvoljava naknadne promjene rezultata prethodnih faza, odnosno nije dozvoljen iterativni pristup razvoju [2]. Nadalje, svaka faza mora biti dovršena prije nego što sljedeća faza može početi i nema preklapanja u fazama te rezultat jedne faze služi kao input za sljedeću fazu. Također, procesi moraju biti dobro definirani i rezultati implementacije sustava moraju moći biti unaprijed predviđeni bez prevelikih nepredviđenih poteškoća. Prije samog početka projekta obavlja se opsežno planiranje kako bi se mogle izmjeriti i preventivno utjecati na potencijalne neželjene varijacije u radu sustava nakon implementacije [6]. Iako je navedeni koncept pomalo ograničen zbog nemogućnosti utjecaja na rezultate prethodne faze i dalje se u visokoj mjeri primjenjuje, posebno na velikim i visoko investicijskim projektima.

Vodopadni pristup prikladan je za dobro definirano okruženje u kojem postoje razrađene procedure i okruženje koje po svojoj prirodi ne dopušta funkcionalne izmjene. Vodopadni pristup se prema izvornom modelu Winstona W. Roycea sastoji od faza analize i specifikacije zahtjeva, faze oblikovanja, faze implementacije, faze provjere – vrednovanja te na kraju faze primjene i održavanja kao što je prikazano na slici 4 [6]. Iako se u različitoj literaturi navode drugačiji nazivi faza, u vodopadnom modelu prva faza nastoji prikupiti informacije što će sustav raditi. Druga faza određuje kako će sustav biti dizajniran. U trećoj fazi razvojni tim kreće s pisanjem programskog koda, dok se u četvrtoj fazi testiraju funkcionalnosti sustava. U posljednjoj, petoj fazi, fokus je stavljen na implementaciju zadataka kao što su obučavanje korisnika o načinu korištenja sustava, izrada popratne dokumentacije i sl. Prikaz angažiranosti projektnog tima u razvoju sustava gdje se kao model razvoja koristi vodopadni pristup prikazan je na slici 5.



Slika 4. *Vodopadni model, [2]*



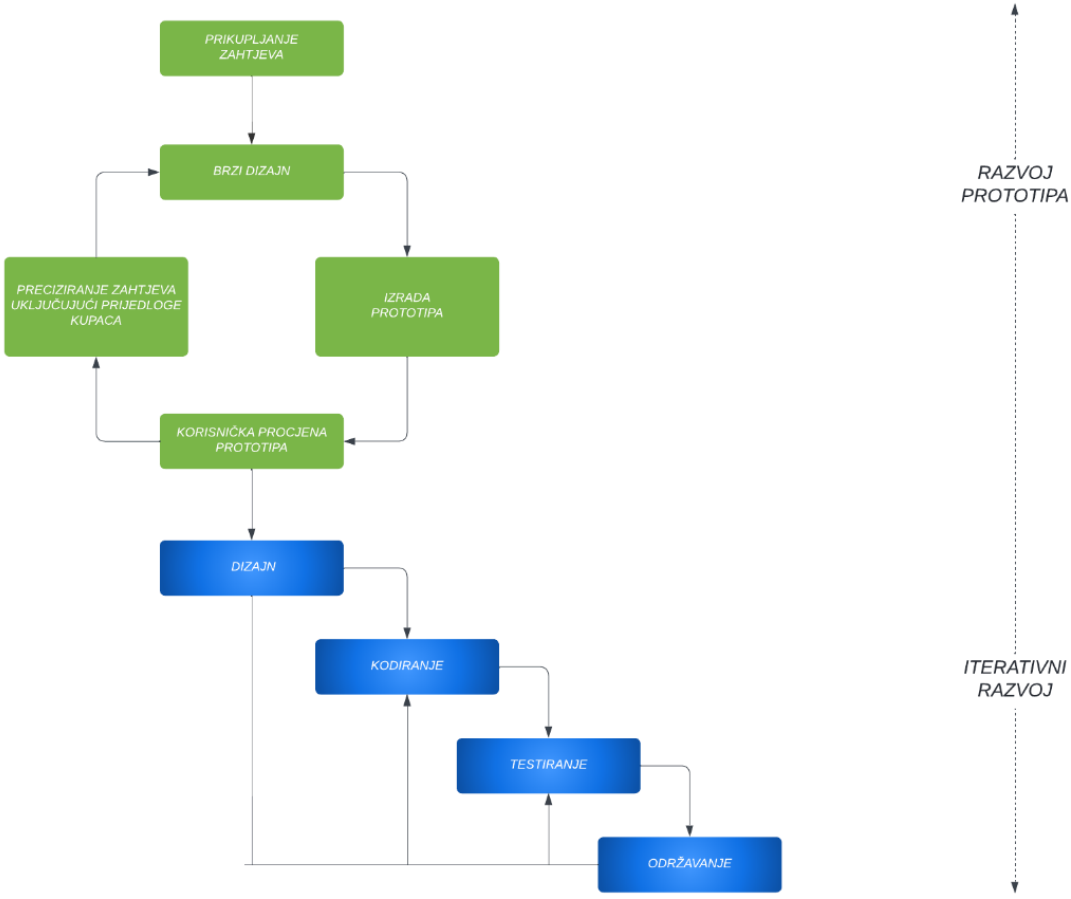
Slika 5. *Postotak aktivnosti primjenom vodopadnog pristupa, [6]*

Vodopadni model zbog svoje strukture ne omogućava vraćanja na prethodne faze razvoja već se izvršava slijedno. Takav pristup ne odgovara razvoju određenih tipova sustava i vođenju i upravljanju određenim projektima, koji zahtijevaju mogućnost vraćanja na prethodne faze razvoja i izmjenu prethodno definiranih specifikacija. Također, proces razvoja prema vodopadnom modelu može potrajati mjesecima pa čak i godinama čime se ne ispunjavaju korisnička očekivanja te su

promjene vrlo spore i skupe. Iz navedenog razloga pojavila se potreba za fleksibilnijim modelima koji će omogućiti iterativnost faza, odnosno ponavljanje određene faze ili dijela faze nakon njenog završetka. Iz vodopadnog modela nastale su određene izvedenice kao što je model prototipiranja, iterativni model, spiralni model, V-model i na posljetku agilne metodologije [7].

4.1.2. Prototipiranje

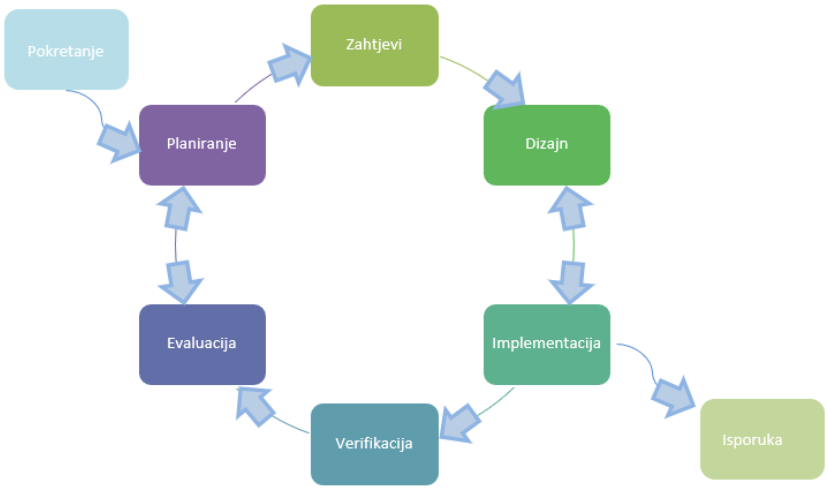
Prototipiranje je model koji je nastao iz vodopadnog i kao alternativa vodopadnom modelu. Koristio se sredinom 70-tih godina prošlog stoljeća. Bazira se na stvaranju prototipa niske vjernosti u svrhu prikupljanja ranih povratnih informacija od potencijalnih korisnika, nakon čega se prototipovi razvijaju u finalna softverska rješenja [7]. Prikaz modela prototipiranja nalazi se na slici 6.



Slika 6. *Prototipiranje, [7].*

4.1.3. Iterativni model

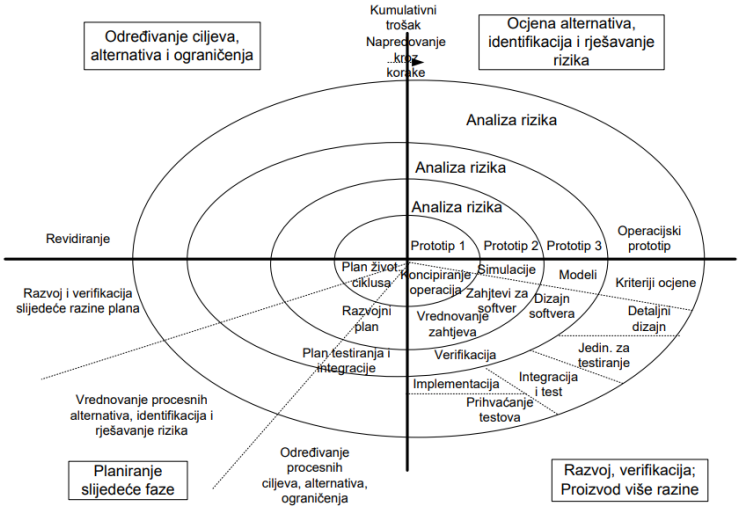
Iterativni model se smatra ranim pretkom agilnih metodologija zbog svoje mogućnosti iteracija faza. Primjena ovog pristupa podrazumijeva poznavanje samo osnovnih korisničkih zahtjeva u početku, na temelju čega projektni tim razvija jednostavno i jeftino prvotno softversko rješenje. Naknadno, nakon što se identificiraju i prikupe i ostali korisnički zahtjevi, provode se dodatne iteracije SDLC iterativnog modela koji se ponavljaju sve dok se ne isporuči kompletno programsko rješenje. Također, ovakav pristup omogućava rad na više projektnih faza istovremeno [7]. Prikaz iterativnog modela nalazi se na slici 7.



Slika 7. *Iterativni model, [7].*

4.1.4. Spiralni model

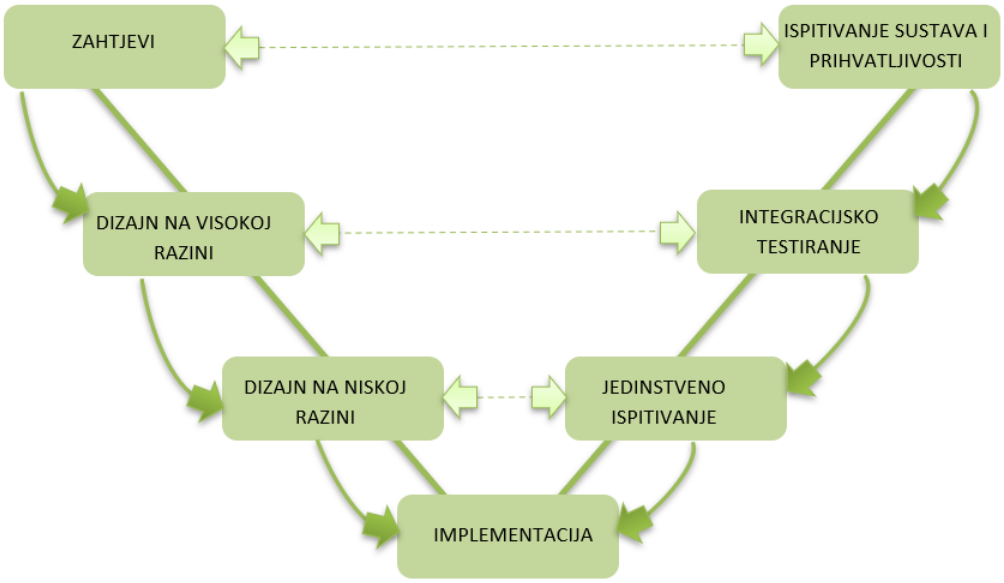
Spiralni model je pristup razvoju informacijskog sustava koji kombinira specifičnosti prototipiranja i vodopadnog modela i njihovih faza. Svodi se na meta-model koji obavlja procjenu rizika projekta prije nego što se preporuči pristup koji spaja aspekte ostalih tradicionalnih metodologija kao što su vodopadni i iterativni pristup. Spiralni model odbacuje jedinstven pristup usvajanju modela procesa koji odgovara svima [7]. Elementi spiralnog modela prikazani su na slici 8.



Slika 8. *Spiralni model, [2].*

4.1.5. V-model

V-model predstavlja nadogradnju vodopadnog modela koji posebnu pažnju pridaje validaciji i verifikaciji. U verifikacijskoj fazi, kreiraju se korisnički zahtjevi i dizajn. Svaka faza validacije povezana je s vlastitom fazom verifikacije koja uključuje funkcionalno testiranje i odobravanje od strane korisnika. Navedene dvije faze su međusobno povezane implementacijskom, odnosno fazom kodiranja [7]. Prikaz V-modela nalazi se na slici 9.



Slika 9. *V-model, [7].*

4.2. Agilni modeli razvoja sustava

Kao odgovor na nedostatke i izazove tradicionalnih metodologija, početkom novog milenija, skupina IT stručnjaka kreirala je manifest agilnih metodologija. Proglas o metodi agilnog razvoja softvera, odnosno manifest agilnih metodologija definiralo je sedamnaest stručnjaka iz područja informacijskih tehnologija i razvoja softvera 2001. godine. Temeljne vrijednosti proglasa su definirati metodologije koje će omogućiti bolju međusobnu suradnju i više se bazirati na međuljudske odnose nego na procese i oruđa razvoja softvera. Nadalje, nastoji se isporučiti uporabljiv softver umjesto iscrpne dokumentacije, podići razinu suradnje s naručiteljem te smanjiti pregovaranja oko ugovora te zaključno reagirati na potencijalne promjene, umjesto ustrajanje na planu. Proglas se sastoji od dvanaest načela koja predstavljaju temelj svih agilnih metodologija [8].

Agilne metodologije nastoje prevladati ograničenja koja projekti razvoja informacijskih sustava donose primjenjujući fleksibilan, brz i pojednostavljen pristup. Agilna metodologija rastavlja projekt u više ciklusa, od kojih svaki prolazi kroz neke ili sve SDLC faze. Fokus je na ljudima i načinu na koji rade zajedno kako bi dovršili projekt. Agilne metodologije pozivaju na kontinuiranu suradnju između članova tima i dionika s redovitim ciklusima povratnih informacija i ponavljanja [7]. Omogućavaju fleksibilan i adaptivan proces razvoja softvera te brzu isporuku proizvoda i usluga. Također, nude mogućnost brzog odgovora na nepredviđene izmjene zahtjeva ili situacije koje se mogu dogoditi tijekom procesa razvoja. Međutim, agilne metodologije zahtijevaju konstantnu komunikaciju i suradnju s korisnicima koristeći njihove inpute i povratne informacije tokom različitih kontrolnih točaka unutar ciklusa. Agilne metodologije su zapravo način razmišljanja i baziraju se na dva znanstvena koncepta, a to su korištenje empirijske kontrole procesa i promatranje razvoja sustava kao složenih adaptivnih sustava i dizajniranje timskih struktura i metoda oko toga. Empirijska kontrola procesa namijenjena je procesima koji nisu dovoljno definirani, a ujedno su i nepredvidljivi i ne sadrže mogućnost ponavljanja. Nastoji implementirati kontrole kroz česti nadzor stanja i prilagodbu što uvelike odgovara razvoju informacijskih sustava što je samo po sebi kompleksan i varijabilan proces te se stoga pokazuje kao metoda koja donosi najbolje rezultate [6].

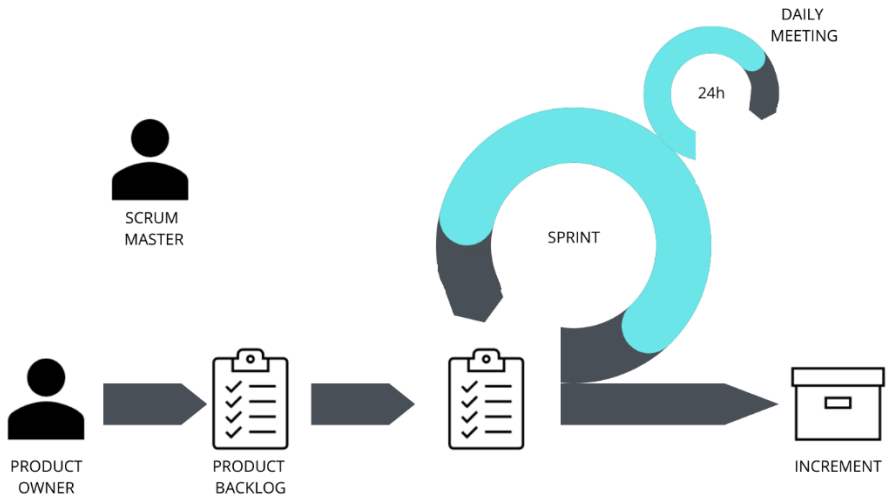
Agilnost kao koncept pod sobom sadrži razne metodologije, odnosno procesne okvire (*framework*) kao što su *Scrum, Kanban, Lean, Extreme Programming (XP), Crystal*, a nerijetko se i *Rational Unified Process* uvrštava u agilne metodologije zbog svoje fleksibilnosti iako sadržava dosta aspekata tradicionalnih metodologija [7].

4.2.1. Scrum

Scrum je veoma popularan procesni okvir kojeg karakterizira konstantna suradnja, česte isporuke i posebni ciklusi razvoja koji se nazivaju „sprintovi“. *Scrum* omogućava cikluse i iterativan je procesni okvir te nije orijentiran na faze razvoja kao ostale metodologije. Osim mogućnosti ponavljanja procesa, *Scrum* je također inkrementalan što znači da se ne nastoji izgraditi cijeli sustav odjednom već se izgrađuje dio po dio sustava koji se zatim spaja u cjelinu. Usredotočuje se na to kako bi članovi tima trebali funkcionirati kako bi proizveli fleksibilnost sustava u okruženje koje se stalno mijenja. Na kraju svake iteracije proizvodi potencijalni skup funkcionalnosti. Ne zahtjeva primjenu određenih praksi razvoja softvera, umjesto toga zahtijeva određene upravljačke prakse i alate u različitim fazama kako bi se izbjegao kaos u razvoju sustava i smanjile nepredvidivosti i kompleksnosti [6]. Prikaz *Scrum* metodologije nalazi se na slici 10.

Specifičnosti *Scrum*-a očituju se u tome što se nakon svakog „sprinta“ može isporučiti stvarni funkcionalni softver ili donijeti odluka da će se nastaviti razvoj kroz još jedan sprint. Vremensko trajanje pojedinog sprinta je najčešće između dva do četiri tjedna unutar kojih se proizvod dizajnira, programira i testira. Ključne upravljačke prakse *Scrum*-a su [6]:

- *Product Backlog* – lista funkcionalnosti i izmjena raspoređenih prema prioritetima koji se još moraju razviti i implementirati, a koje zahtijevaju višestruki dionici. Za održavanje *Product Backlog*-a zadužen je *Product Owner*.
- Sprintovi – postupak prilagodbe promjenjivim varijablama u okruženju tijekom razvoja kao što su zahtjevi, vrijeme, resursi, znanje, tehnologija, itd., te mora rezultirati potencijalno isporučivim inkrementom funkcionalnog softvera,
- Sastanak planiranja sprintova - Sastanku planiranja sprinta prvo prisustvuju kupci, korisnici, menadžment, *Product owner* i *Scrum* tim gdje se odlučuje o skupu ciljeva i funkcionalnosti. Naknadno se *Scrum Master* i projektni tim fokusiraju kako najbolje implementirati proizvod tokom sprinta.
- *Sprint Backlog* – Popis ključnih funkcionalnosti koje su trenutno dodijeljene pojedinom sprintu. Kada se sve funkcionalnosti završe, nova iteracija sustava se isporučuje.
- *Daily Scrum* – dnevni petnaestominutni sastanak na kojem se raspravlja napredak koji je projektni tim ostvario tijekom proteklog vremena sprinta i rasprava o potencijalnim preprekama s kojima se tim susreće.



Slika 10. *Proces Scrum metodologije, [7].*

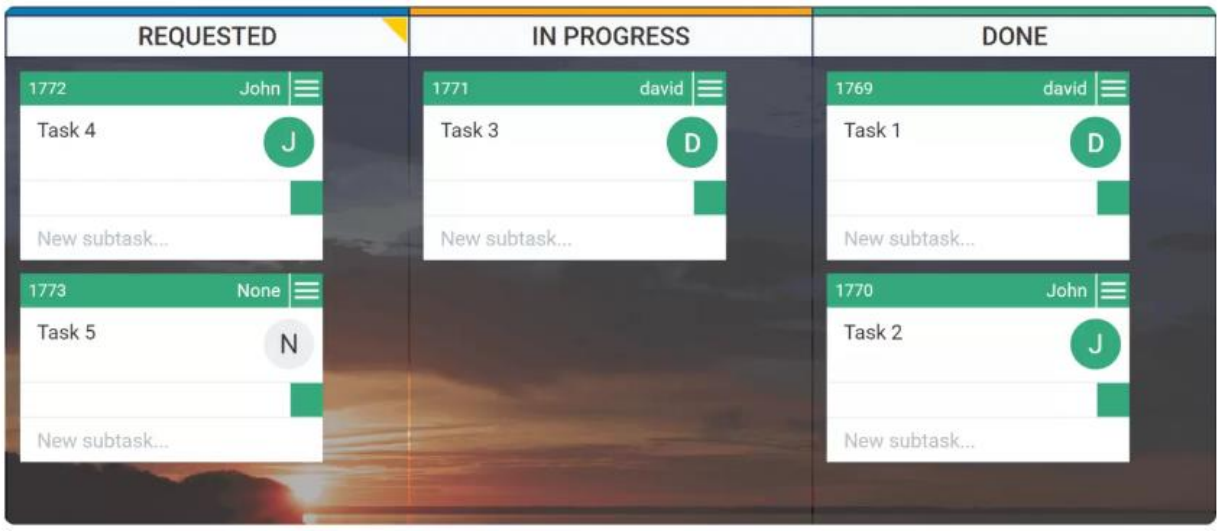
Scrum metodologija se sastoji od određenih uloga koje se dodjeljuju unutar projektnog tima, a to su [7]:

- *Product owner* – uloga koja definira funkcionalnosti proizvoda koji se razvija te odlučuje o datumu isporuke i sadržaju isporuke sustava. Također, odgovoran je za profitabilnost proizvoda, te također izvodi prioritizaciju funkcionalnosti prema trenutnim okolnostima.
- *Scrum Master* – predstavlja upravljačku ulogu u projektnom timu, odnosno voditelja projekta koji je odgovoran za implementaciju temeljnih vrijednosti agilnih metodologija i praksi. Omogućuje funkcionalnost i produktivnost projektnog tima te predstavlja sponu između različitih uloga u projektnom timu.
- Projektni tim – sastoji se od programera, testera, poslovnih analitičara, dizajnera korisničkih sučelja itd.

4.2.2. Kanban

Osim *Scrum*-a, veliku popularnost uživa i *Kanban* procesni okvir koji svoje korijenje ne vuče iz razvoja softverskih rješenja, ali se vrlo dobro uklapa s agilnim metodologijama i uvelike se koristi od strane agilnih projektnih timova. Riječ *Kanban* potiče iz Japanskog jezika te označava riječ „znakovna ploča“. *Kanban* se uvelike koristi od strane projektnih timova prilikom vizualizacije samog procesa razvoja čime se podiže produktivnost, ali i osviještenost na aktivnosti koje su izvršene i koje još trebaju biti izvršene kako bi se kompletirao razvojni proces. Korištenje *Kanban*-a podrazumijeva upotrebu ploče i kartica ili naljepnica kako bi se omogućilo praćenje

procesa kroz vizualni prikaz koraka koje je potrebno odraditi. Zadaci koje je potrebno odraditi se stavljaju na ploču i pomiču kroz različite faze kao što su naprimjer „Za napraviti“, „U tijeku“, „U pregledu“ ili „Dovršeno“. Na taj način projektni tim može efikasno upravljati svojim vremenom na dnevnoj, tjednoj ili mjesečnoj bazi čime se povećava efikasnost projekta [7]. Prikaz *Kanban* ploče nalazi se na slici 11.



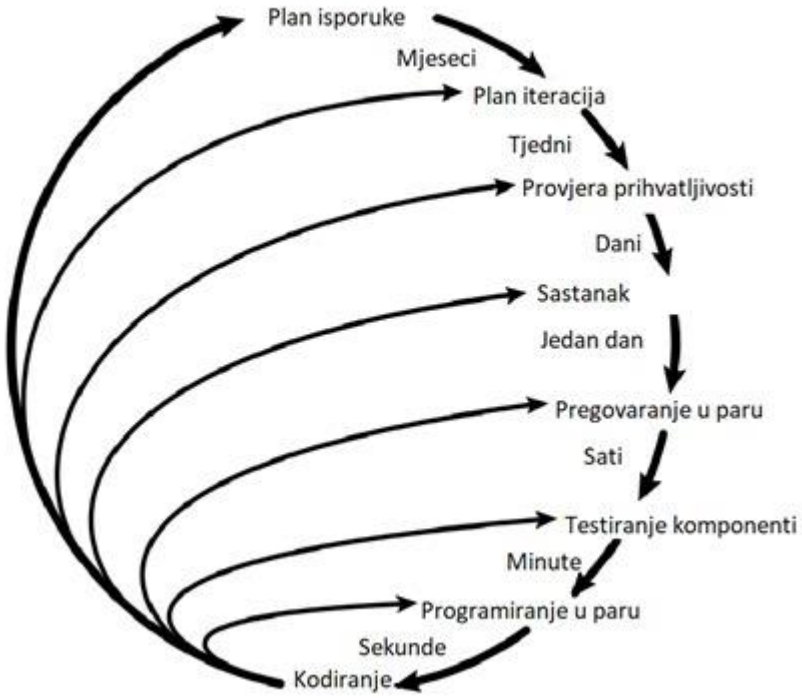
***Slika 11**. Kanban ploča, [7].*

4.2.3. Extreme Programming XP

Osim ranije spomenutih metodologija važno je spomenuti i ekstremno programiranje (*Extreme programming* - XP) koji je agilni procesni okvir koji se fokusira na projektnu fleksibilnost i razvoj softvera kroz visoko kvalitetni i intenzivno testirani programski kod. Prednosti primjene ove metodologije očituju se kroz povećanje kvalitete komunikacije između klijenata i projektnog tima te projektnog tima međusobno. Uvelike je pojednostavljen proces razvoja te se kroz konstantnu komunikaciju i povratne informacije koje klijenti pružaju razvojnom timu smanjuje rizik od neuspješnosti projekta.

XP se fokusira na primjenu najboljih praksi za izradu kvalitetnog i visokoučinkovitog softvera. Primjena najboljih praksi omogućava projektnom timu da definira, izgradi i isporuči kvalitetan softver na vrijeme, ali uz smanjeni stres koji uzrokuju rokovi isporuke i definirani budžet. Navedeno se postiže primjenom tzv. programiranja u paru, što je zapravo tehnika u kojoj dva programera dijele radu stanicu i kreiraju programski kod zajedno. Prednosti ovakvog pristupa su istovremeni pregled koda čime se smanjuje mogućnost pogrešaka u kodu koje se mogu pojaviti te ukoliko se pojave mogu se istovremeno ispraviti. Iako ovakav pristup može zahtijevati više vremena za pisanje programskog koda, rezultat uvelike anulira navedeno vrijeme jer se u konačnici

isporučuje visokokvalitetni programski kod s minimalnim pogreškama. Nadalje, testiranje je automatizirano, čime se smanjuju tehničke poteškoće i osigurava se ponovno korištenje koda. U svemu je ključna konstantna komunikacija između programera i ostalih dionika kako bi se osigurala pravovremena reakcija na korisničke inpute, povratne informacije, i promjene zahtjeva [7]. Vizualni prikaz procesa razvoja primjenom XP metodologije prikaza je na slici 12.



Slika 12. *Extreme Programming proces, [7].*

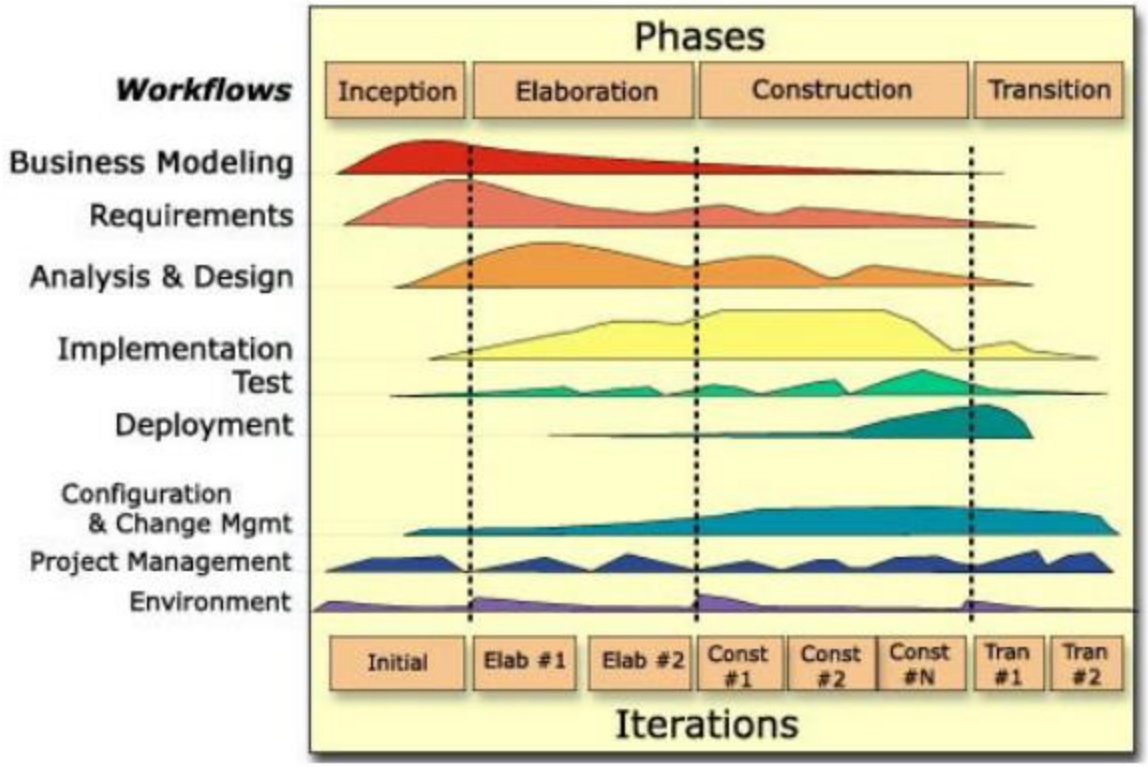
4.2.4. Rational Unified Process

Prethodno spomenute metodologije se smatraju „čistim“ agilnim metodologijama, međutim važno je spomenuti i *Rational Unified Process* koji uzima najbolje prakse iz tradicionalnih metodologija, ali ujedno je dovoljno fleksibilan kako bi se mogao smatrati agilnom metodologijom, iako se i dalje u određenoj stručnoj literaturi prikazuje kao tradicionalna metodologija. RUP predstavlja procesni okvir za razvoj informacijskih sustava koji je definirao, razvio i stavio na tržište *Rational Software*. RUP pruža disciplinirani pristup dodijele i upravljanja zadacima i odgovornostima u organizaciji koja se bavi razvojem softverskih rješenja. Nadalje, radi se o objektno orijentiranom i iterativnom procesu razvoja softvera [9]. Cilj ove metodologije je osigurati smjernice i predloške za izradu visokokvalitetnog softverskog rješenja koje zadovoljava potrebe krajnjih korisnika unutar predviđenih rokova i budžeta. Pruža projektnom timu mogućnost

razvoja softvera kroz primjenu najboljih praksi razvoja softvera kao što su iterativnost, baziranost na arhitekturu, minimiziranje rizika u svakoj fazi razvoja i kontinuirana validacija i verifikacija kvalitete programskog rješenja. Osigurava veću produktivnost projektnog tima na način da se svakom članu tima omogućava pristup bazi znanja sa smjernicama, predlošcima, alatima i mentorstvom za sve aktivnosti razvoja koje su označene kao kritične. RUP se ne fokusira na stvaranje velike količine projektne dokumentacije, već nastoji staviti naglasak na razvoj i održavanje modela koji predstavljaju semantički bogate reprezentacije informacijskog sustava koji se razvija. Navedeno podrazumijeva upotrebu različitih dijagrama, modela i ostalih vizualnih prezentacija kojima su opisane sve komponente sustava te se u tu svrhu koristi unificirani grafički jezik za modeliranje (*Unified Modeling Language* - UML) grafički jezik [10].

Arhitektura *Rational Unified Process-a* se može, kao što je prikazano na slici 13, sagledati kroz dvije dimenzije, odnosno strukture, a to su horizontalna dimenzija koja predstavlja vrijeme i prikazuje životni ciklus aspekata procesa koji se odvija te vertikalna dimenzija koja predstavlja temeljne discipline koje logički grupiraju aktivnosti razvoja softvera prema njihovoj prirodi. Horizontalna dimenzija predstavlja dinamičke aspekte procesa i izražava se kroz cikluse, faze, iteracije i prekretnice dok se vertikalna dimenzija fokusira na statičke aspekte procesa kao što su aktivnosti, artefakti, uloge i radni tijekovi. RUP projektna metodologija se sastoji od četiri faze, a to su [10]:

- Priprema (*Inception*) – cilj ove faze je odrediti izvodljivost i isplativost projekta, odnosno odrediti je li moguće ispuniti korisničke zahtjeve s raspoloživim resursima u definiranom roku,
- Razrada (*Elaboration*) – ova faza objedinjuje procjenu rizika, definira atribute kvalitete te se prepoznaju i evidentiraju slučajevi uporabe koji pokrivaju funkcionalne zahtjeve. Također, procjenjuju se potrebni resursi i stvara se detaljni plan za narednu fazu konstrukcije,
- Provedba (*Construction*) – cilj koji ova faza nastoji ispuniti je dovršetak zahtjeva, analize i oblikovanje modela sustava te dogradnja jezgre arhitekture do konačnog sustava. Kroz ovu fazu se nastoji pokrenuti sustav,
- Tranzicija/puštanje u rad (*Transition*) – faza opisuje pripremanje korisničke okoline sustava i puštanje sustava u rad te se popravljaju potencijalne uočene greške kako bi se ustanovilo da sustav odgovara potrebama korisnika. Također, stvara se korisnička dokumentacija i organizira podrška korisnicima.



Slika 13. *RUP faze i iteracije, [10]*

5. Modeliranje sustava primjenom UML jezika

Sustavni pristup razvoju informacijskih sustava podrazumijeva jasno i nedvosmisleno definiranje komponenti sustava i veza između njih. Nadalje, važan je i način dokumentiranja samog procesa razvoja, prikaz komponenti sustava, njihovu interakciju kako bi se razvio funkcionalan i kvalitetan sustav sa što manje grešaka koje bi se naknadno trebale rješavati. Izrada modela sustava pomaže kod boljeg razumijevanja sustava koji se razvija. Modeliranje pomaže kod vizualizacije stvarnog ili zamišljenog sustava i omogućuje opisivanje strukture i ponašanja sustava. Nadalje, modeli se mogu iskoristiti kao predložak za izgradnju sustava te omogućuju dokumentiranje procesa izgradnje sustava. Modeliranje je dokazana i dobro prihvaćena inženjerska tehnika koja rezultira modelima koji pomažu korisnicima u vizualizaciji konačnog produkta. Modeli kompleksnih sustava se izgrađuju zbog toga što se takav sustav ne može shvatiti u potpunosti, odnosno postoje ograničenja u mogućnosti ljudi da razumiju kompleksnost neke materije. Modeliranje omogućuje približavanje problemu koji se proučava i razrješenju kompleksnosti na način da se obavlja dekompozicija sustava na manje dijelove koji su lakše razumljivi. Fokusiranje na samo jedan aspekt/pogled u vremenu pojednostavljuje razvoj sustava [11].

5.1. Unified Modeling Language – UML

Modeliranje primjenom grafičkih jezika kao što je UML u procesu vizualizacije i dokumentiranja razvoja sustava uvelike pojednostavljuje proces. UML je industrijski standardizirani grafički jezik za modeliranje objektno orijentiranih softvera koji omogućuje projektnom timu jasnu komunikaciju zahtjeva, arhitekture i dizajna [12]. UML je neovisan o ostalim tehnologijama i programskim jezicima te je njegova primjena posebno pogodna kod inkrementalnih i iterativnih procesa razvoja informacijskih sustava. UML predstavlja temelj za razvoj teorija, standarda i specifikacija koje značajno unaprjeđuju strukturiranost dokumentacije analize i dizajna informacijskog sustava.

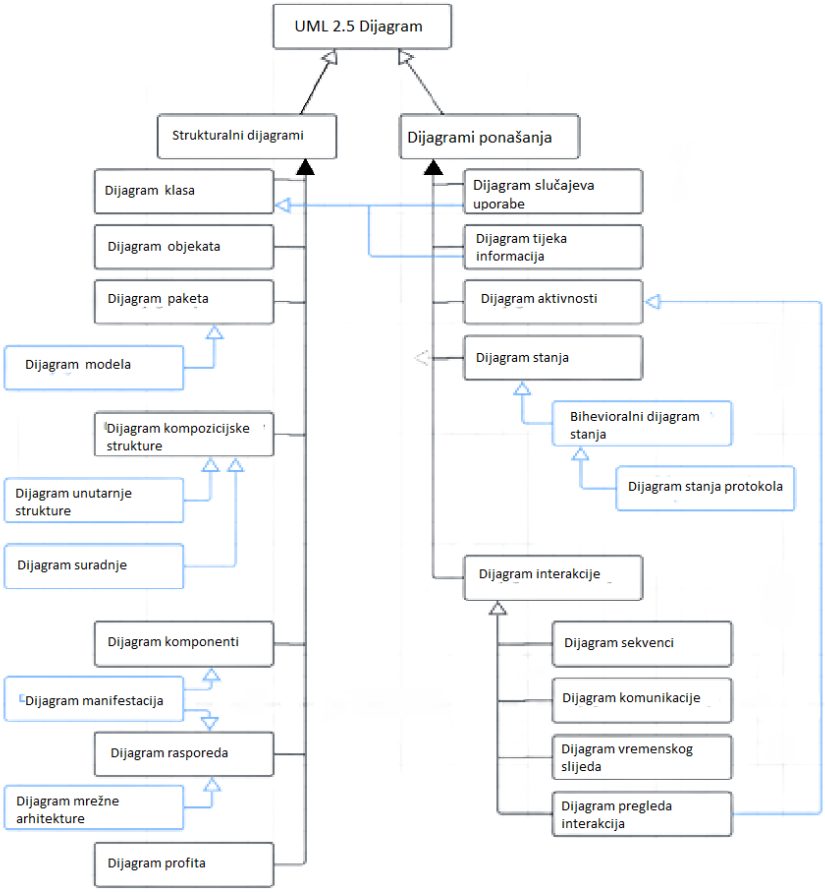
UML je ključan u procesima razvoja informacijskih sustava koje se baziraju na modeliranju. Sastoji se od strukturalnog i funkcionalnog dijela koji rezultiraju različitim tipovima dijagrama. Strukturni dio definira osnovne strukture modela, dok funkcionalni dio omogućuje modeliranje aktivnosti za realizaciju funkcionalnih zahtjeva na informacijski sustav. UML omogućuje modeliranje, odnosno kreiranje modela sustava koji će se na kraju razvojnog procesa pretvoriti u stvarni funkcionalni sustav za korištenje [13]. Pojednostavljenje razvoja kompleksnih sustava je ključna stvar koji se nastoji razriješiti primjenom UML jezika. UML je grafički alat za koji primjenu mogu pronaći svi akteri koji su uključeni u izradu, implementaciju i održavanje softvera. Prije svega koristan je članovima razvojnog tima koji kreiraju modele, ali njegova primjena osjetna

je i od strane ostalih članova tima koji rade zajedno da bi razumjeli izgradnju, testiranje i puštanje u rad sustava [11].

UML je nastao na ujedinjavanju prethodno razvijenih opisa i tehnika analize i konstrukcije programskih sustava zasnovanih na objektu usmjerenoj metodologiji. Radi se o metodama objektno – orijentiranog programiranja (*Object-Oriented Software Engineering* - OOSE) autora Ivara Jacobsona i tehnike objektnog modeliranja (*Object Modeling Technique* - OMT) autora Jima Rumbaugha te objektno orijentirani dizajn i aplikacije (*Object-Oriented Design with Applications* – OODA) autora Gradya Boocha. UML je standardiziran od strane neprofitne *organizacije Object Management Group* OMG 1997. godine, a kroz prethodna desetljeća razvijeno je više verzija UML-a koje se razlikuju po broju i vrsti dijagrama te po izmjenama u oznakama. Posljednja verzija je UML 2.5.1. koja je izdana u prosincu 2017. godine [11].

5.2. UML dijagrami

Kao što je prethodno napomenuto UML kao standard se sastoji od strukturalnog i funkcionalnog dijela koji sadrže prateće dijagrame. Strukturalni dio definiran je od strane strukturalnih dijagrama koji prikazuju statičku strukturu sustava i njegovih komponenti na različitim razinama apstrakcije i njihovu međusobnu povezanost. Prema zadnjoj važećoj specifikaciji 2.5. to su dijagram klasa, dijagram objekata, dijagram paketa, dijagram kompozicijske strukture, dijagram komponenti, dijagram rasporeda, dijagram profila. Funkcionalni dio se smatra dinamičkim dijelom UML-a i prikazan je dijagramima ponašanja koji pokazuju dinamičko ponašanje objekata u sustavu koje se može opisati kao serija promjena u sustavu kroz vrijeme. Prema zadnjoj važećoj verziji radi se o dijagramu slučajeva uporabe, dijagramu aktivnosti, dijagramu stanja te dijagrami interakcije koji se dijele na dijagram sekvenci, dijagram komunikacije, dijagram vremenskog slijeda, interakcijski pregledni dijagram [14]. Na slici 14 prikazana je podjela UML dijagrama prema verziji 2.5.

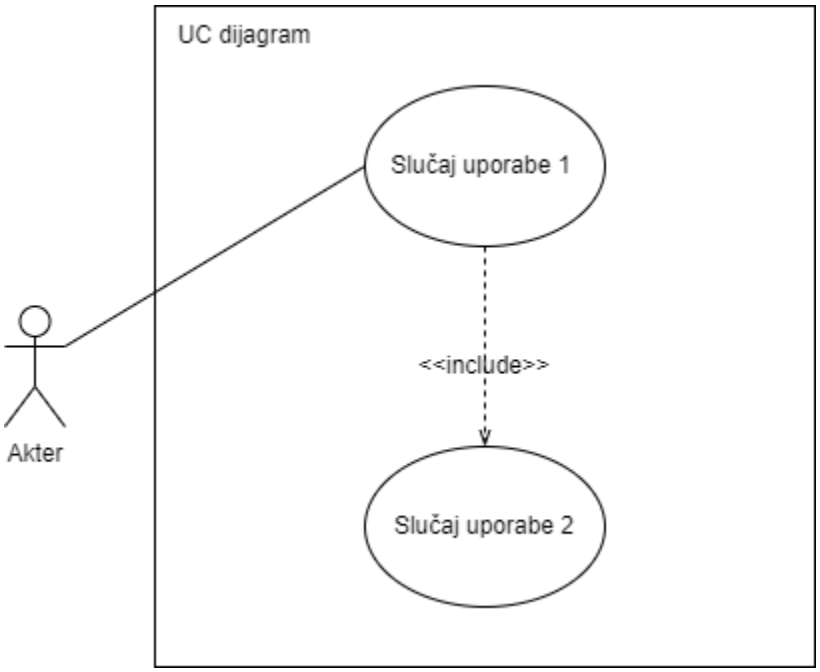


***Slika 14**. UML dijagrami 2.5, [14].*

Na slici 14 vidljivo je kako UML notacija 2.5. sadrži veliki broj dijagrama koji se mogu iskoristiti za modeliranje sustava. U stvarnom svijetu ne koriste se svi tipovi dijagrama, već odabir dijagrama ovisi o korištenoj metodologiji razvoja informacijskih sustava. Najčešće su u uporabi dijagram slučajeva uporabe, dijagram klasa, dijagram objekata, dijagram međudjelovanja/interakcije, dijagram suradnje, dijagram aktivnosti, dijagram stanja i dijagram komponenti. U daljnjem tekstu bit će opisani neki od ključnih dijagrama za modeliranje sustava.

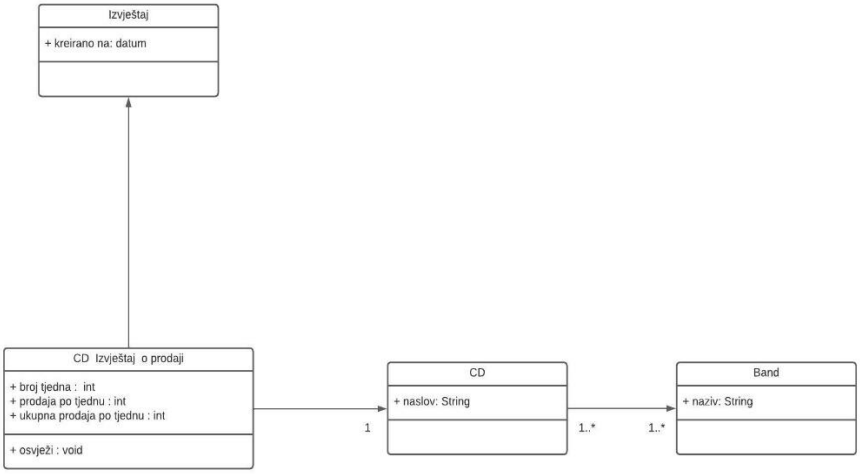
Dijagram slučajeva uporabe opisuje skup radnji (slučajevi upotrebe) koje neki sustav ili sustavi (subjekt) trebaju ili mogu izvesti u suradnji s jednim ili više vanjskih korisnika sustava (aktera) kako bi se akterima ili drugim dionicima pružili neki vidljivi i vrijedni rezultati. Dijagrami slučajeva zapravo spadaju u obje kategorije UML dijagrama. Spadaju i u dijagrame ponašanja jer opisuju ponašanje sustava, a također su i strukturni dijagrami kao poseban slučaj dijagrama klasa gdje su klasifikatori ograničeni da budu ili akteri ili slučajevi upotrebe koji su međusobno povezani asocijacijama. Dijagrami slučajeva uporabe prvi puta su definirani i koriste se unutar RUP metodologije kao način predstavljanja poslovne funkcije u poslovnom modeliranju. Dijagram slučajeva uporabe se sastoji od slučaja uporabe koji opisuju funkcionalnosti koje sustav treba

ispunjavati, sudionika koji ostvaruju interakciju sa sustavom odnosno izravno ili neizravno sudjeluju u izvođenju slučaja uporabe, relacije koje pojašnjavaju međusobnu interakciju između aktera i slučajeva uporabe te granica sustava [14]. Prikaz primjera dijagrama slučajeva uporabe prikazan je na slici 15.



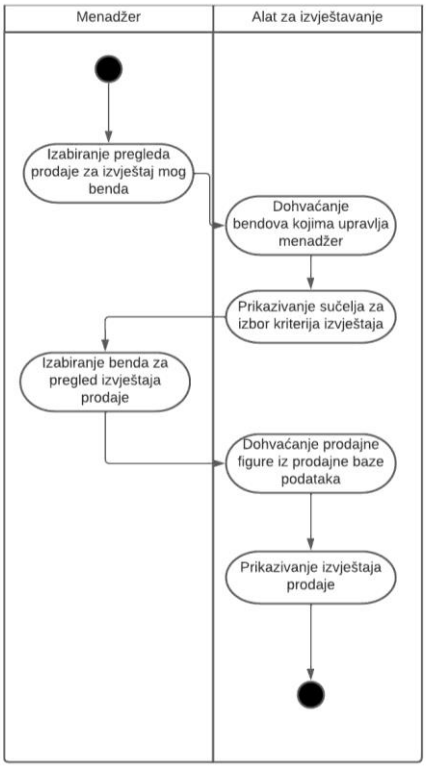
Slika 15. *Dijagram slučajeva uporabe (UC dijagram), [15].*

Dijagram klasa spada u strukturalne UML dijagrame i prikazuje strukturu projektiranog sustava, podsustava ili komponente kao povezanih klasa i sučelja, s njihovim značajkama, ograničenjima i odnosima – asocijacijama, generalizacijama, ovisnostima itd. Dijagrami klasa zapravo prikazuju klase i njihove međusobne veze te vrste objekata unutar nekog sustava i njihove međusobne statične odnose. Klasa se sastoji od naziva, atributa koje klasa poprima i operacija koje klasa izvršava [14]. Prikaz primjera označavanja klasa i primjer dijagrama prikaza je na slici 16.



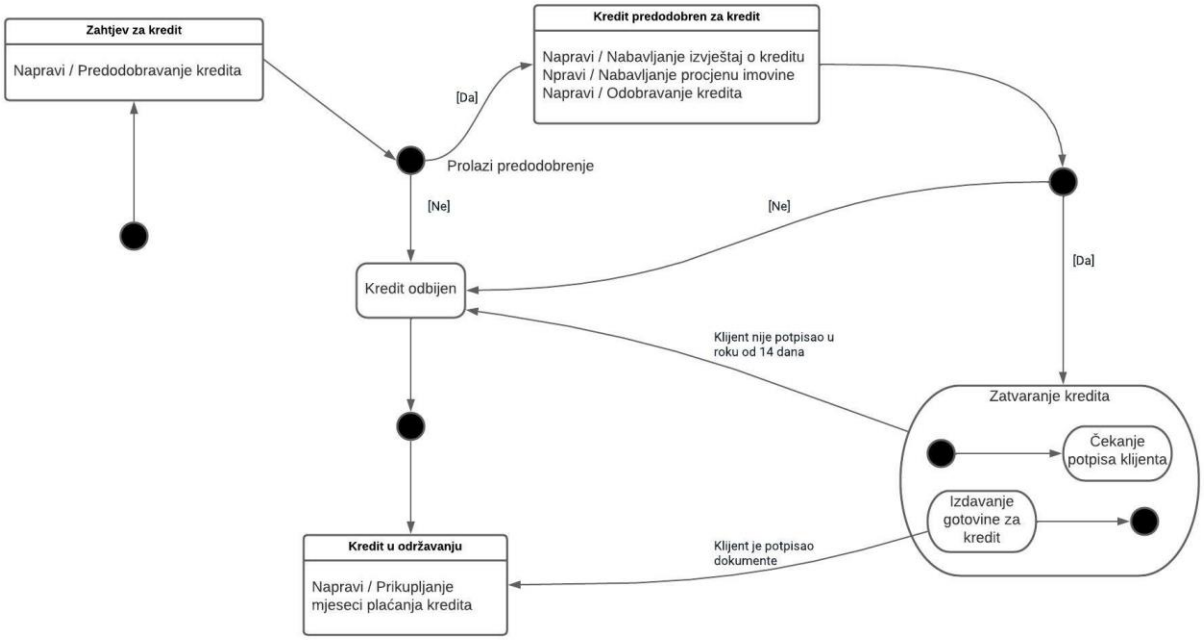
Slika 16. *Dijagram klasa, [15].*

Na slici 17 nalazi se primjer dijagrama aktivnosti. Dijagram aktivnosti omogućuje modeliranje softverskih (i poslovnih) procesa te može pružiti uvid u tok mehanizama i podataka koji čine jedan proces. Ovaj dijagram redoslijed izvođenja aktivnosti pridruženih promatranom objektu i koristi se kod modeliranja radnog tijeka i opisa rada algoritma ili detalja računanja [11].



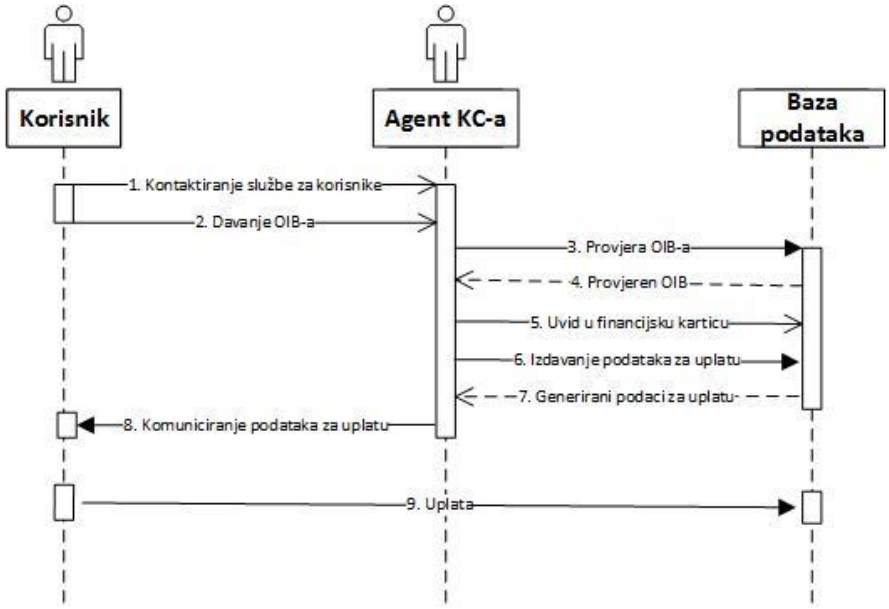
Slika 17. *Dijagram aktivnosti, [15].*

Na slici 18 nalazi se primjer dijagrama stanja promatranog objekta koji predstavlja skup stanja objekta u njegovom životnom vijeku, događaja na koje objekt daje odziv, odziva objekta na događaje i prijelaza između stanja objekta [15].



***Slika 18**. Dijagram stanja objekata, [15].*

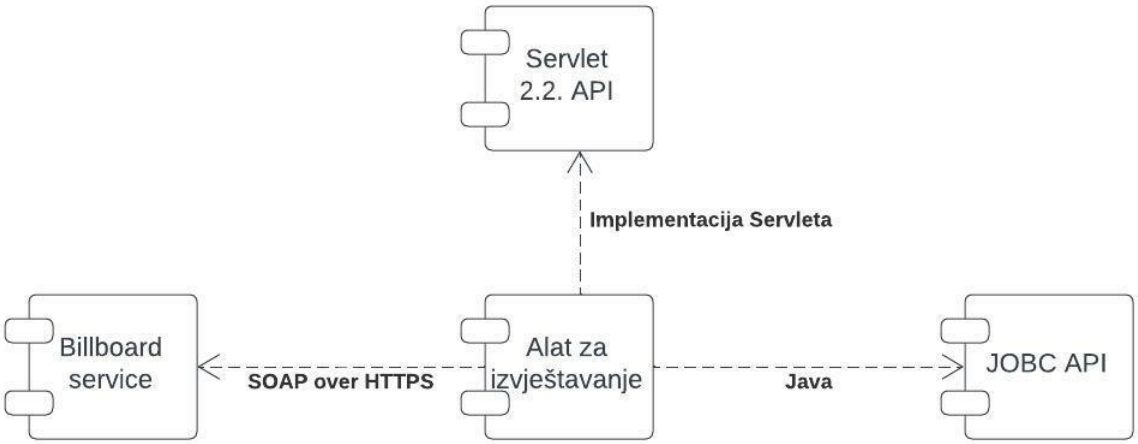
Dijagram sekvenci odnosno međudjelovanja spada u interakcijske dijagrama ponašanja te opisuje interakcije između objekata kako bi se ispunio neki zadatak. Primjer prikaza dijagrama međudjelovanja nalazi se na slici 19. Točnije, dijagram međudjelovanja prikazuje razmjenu poruka između objekata na kronološki način, odnosno usmjeren je na vremenski redoslijed razmjene poruka između objekata. Najčešće je korišten od strane projektnog tima pri dodjeli operacija klasama na temelju dodjele metoda objektima u dijagramu [11].



Slika 19. *Dijagram međudjelovanja, [15].*

Dijagram suradnje je prema opisu vrlo sličan dijagramu međudjelovanja jer također prikazuje interakciju objekata, ali za razliku od dijagrama međudjelovanja ne stavlja naglasak na vremenski ustroj poruka, već prikazuje samo interakciju između objekata i usmjeren je na strukturalnu povezanost objekata [11].

Dijagram komponenti omogućuje prikazivanje komponenti koje čine sustav, a radi se o nezavisnim izvršnim jedinicama koje pružaju ili primaju usluge od drugih komponenti. Primjer dijagrama komponenti nalazi se na slici 20. Mogu se promatrati kao crne kutije, tj. samo kao specifikaciju komponenti, ili kao bijele kutije, odnosno s definiranim načinom implementacije [15].



Slika 20. *Dijagram komponenti, [15].*

6. Analiza koraka razvoja sustava primjenom RUP metodologije

Rational Unified Process metodologija je jedna od najprimjenjivanijih metoda razvoja kompleksnih informacijskih sustava danas. RUP metodologija se često svrstava u kontekst tradicionalnih metodologija razvoja objektno orijentiranih sustava. Međutim, ono što karakterizira agilne metodologije, također karakterizira i RUP, a to je pokret, aktivnost, brzina, te spremnost na promjene. Zapravo, RUP predstavlja metodologiju koja kombinira prakse agilnih i tradicionalnih metodologija i smatra se adaptivnim procesom razvoja koji se može krojiti prema potrebama organizacije, dostupnosti resursa i karakteru tima. RUP je procesni okvir koji je prihvatio agilne koncepte i usvojio je mnoge agilne tehnike [16]. Predstavlja metodologiju koja određuje kako se treba odvijati proces razvoja. Ujedinjeni proces zahtijeva da se softverski proces sastoji od određenih faza koje se realiziraju simultanim izvršavanjem određenih temeljnih aktivnosti. Nadalje, propisuje rezultate koje pojedine faze, odnosno aktivnosti trebaju proizvesti, no ne određuje način kako se ti rezultati trebaju dokumentirati. Obično se koristi za razvoj objektno orijentiranih sustava, makar je primjenjiv i na druge sustave [17].

RUP je pristup izrade programskih rješenja koji je dobro definiran i strukturiran, usmjeren na arhitekturu, te vođen slučajevima uporabe. Omogućava discipliniran pristup vođenju projekta razvoja, dodijeljenim zadacima i odgovornostima unutar organizacije koja razvija informacijski sustav. Njegova primjena je u početku bila namijenjena velikim i opsežnim projektima, s velikim budžetom i propisanim rokovima, što ga je karakteriziralo kao tradicionalnu metodu. Međutim, s vremenom se primjena navedene metodologije proširila i na srednje i male softverske projekte zbog toga što pruža lak uvid u bazu znanja pojedinog projekta koji je zasnovan na različitim uputama. Navedeno doprinosi zajedničkom pogledu projektnog i razvojnog tima na smjer koji treba slijediti prilikom razvoja softverskog rješenja. Glavne prednosti primjene RUP metodologije očituju se kroz visoku razinu dokumentiranosti projekta i mogućnost prilagodbe projektu [16].

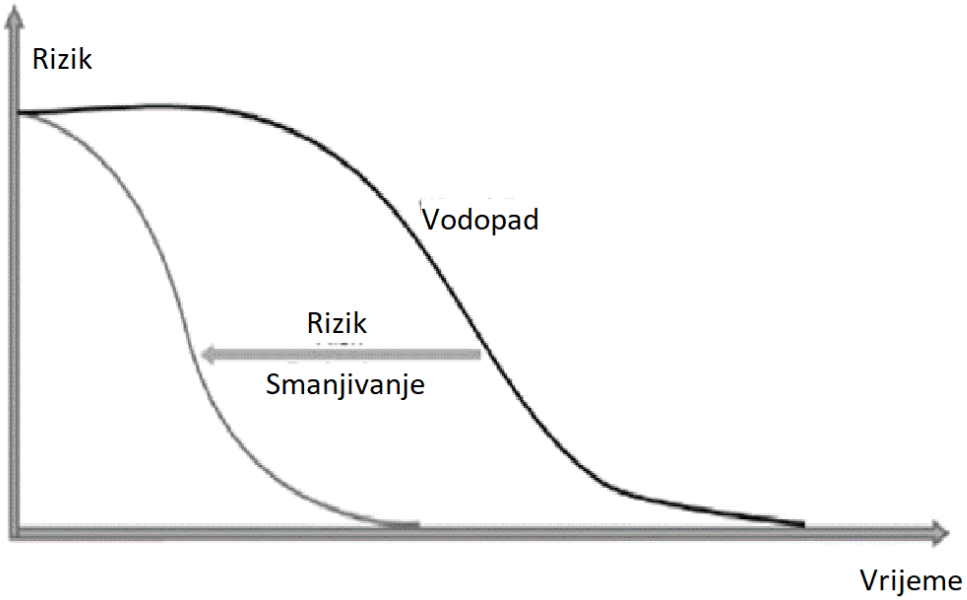
6.1. Osnovna načela RUP-a

Rational Unified Process se temelji na nekoliko temeljnih načela koja podržavaju uspješan iterativni razvoja to su [9]:

1) prepoznati glavne rizike i nastojati ih suzbiti što je prije moguće,
2) osigurati da svaka isporuka korisniku pruža vrijednost,
3) usmjeriti snage prvenstveno na razvoj programskog koda,
4) osigurati održivu i stabilnu arhitekturu sustava,
5) omogućiti da se zahtjevi za promjenom implementiraju u ranoj fazi razvoja,
6) razvijati sustav modularno – po komponentama,

7) projektni tim – radimo zajedno,
8) kvaliteta isporuka kao način poslovanja.

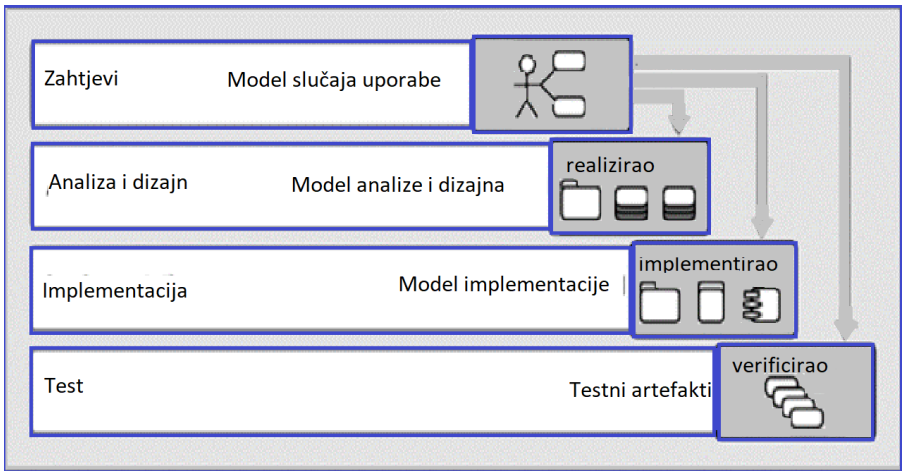
Iterativni razvoj se može izvesti bez primjene ovih načela, međutim, njihovo korištenje povećava vjerojatnost uspjeha projekta. Za optimizaciju iterativnog pristupa, potrebno je pokušati primijeniti što veći broj navedenih načela, uz obazrivost prema izvodljivost pojedinog načela. Naime, pojedina načela neće biti moguće primijeniti, možda zbog veličine projekta, definiranog budžeta ili rokova te su takva načela neprikladan za taj projekt. Nadalje, ova načela prikazuju kako se RUP razlikuje od ostalih iterativnih procesa, posebno iz razloga što stavlja naglasak na suočavanje s rizicima i izradu izvršne arhitekture u ranoj fazi [9]. Na slici 21 prikazan je odnos smanjenja rizika primjenom iterativnih metoda u odnosu na vodopadnu metodu.



Slika 21. *Odnos iterativnih i vodopadne metode u pogledu rizika, [9]*

Identifikacija i suočavanje s poslovnim, tehničkim ili drugim povezanim rizicima u ranoj fazi jedna je od glavnih prednosti iterativnog pristupa. Rizici na koje nije obraćena pozornost mogu značiti da se potencijalno ulaže u pogrešnu arhitekturu ili neoptimalan skup zahtjeva. Nadalje, količina prepoznatih rizika u direktnoj je korelaciji s procjenom završetka projekta, odnosno rizici i načini ophođenja prema njima uvelike mogu povećati ili smanjiti trajanje projekta. Ako se rizici uoče prekasno kad je razvoj u odmakloj fazi, potrebno je znatno više vremena za redizajn i reinženjering. Preporuka za ophođenje s rizicima je stvaranje liste prioriteta rješavanja rizika prije svake iteracije. RUP pruža strukturirani pristup za ophođenje s prioritetnim rizicima u ranoj fazi, što umanjuje sveukupni trošak i omogućuje projektnom timu donošenje realistične i točnije procjene o vremenu potrebnom za završetak projekta [9].

Svaka isporuka treba pružati krajnjem korisniku neku vrijednost. Navedeno se ostvaruje kroz dokumentiranje zahtjeva u formatu koji je lako razumljiv korisnicima, ali ujedno projektni tim mora osigurati ispunjavanje korisničkih zahtjeva kroz kvalitetan dizajn, implementaciju i testne faze. Funkcionalni zahtjevi se dokumentiraju kroz slučajeve uporabe jer oni opisuju interakciju korisnika sa sustavom omogućavajući prikaz funkcionalnih zahtjeva na način razumljiv korisniku i svim drugim članovima tima. Primjena slučajeva uporabe osigurava usklađenost isporuke sa zahtjevima prilikom obavljanja svih projektnih aktivnosti. S obzirom da slučaj uporabe opisuje kako će korisnik komunicirati sa sustavom, primjenjiva je UML notacija i njezini dijagrami za prikaz interakcije, kao npr. dijagram sekvenci ili dijagram suradnje koji prikazuje kako će se interakcija implementirati u završnom dizajnu. U konačnici slučajevi uporabe olakšavaju dokumentiranje funkcionalnih korisničkih zahtjeva i pomažu svim dionicima da razumiju mogućnosti sustava koje se trebaju isporučiti. Također, omogućuju blisku suradnju projektnog tima sa korisničkim zahtjevima prilikom procesa dizajna, implementacije, testiranja čime se osigurava detaljno dokumentiranje i isporuka korisničkih zahtjeva. U kombinaciji s iterativnim razvojem omogućuju isporuku vrijednosti korisnicima. Pod isporukom se smatra upotrebljiva aplikacija koja radi upravo ono što je obuhvaćeno/opisano slučajem uporabe [9]. Na slici 22 prikazana je povezanost slučajeva uporabe s ostalim modelima softverskog inženjeringa.



***Slika 22**. Razrada slučajeva uporabe kroz modele softverskog inženjeringa, [9].*

Treće načelo naglašava važnost usredotočenosti na izvršni programski kod. Izrada dokumentacije je nužna u procesu razvoja informacijskih sustava, ali nije pokazatelj napretka projekta. Evaluacija kvalitete izrađene dokumentacije je subjektivna i njena važnost je sporedna u odnosu na razvijeni programski kod. Funkcionalan softver koji radi je najbolji indikator pravog napretka projekta. Prilikom mjerenja učinka, fokus treba biti na testnim slučajevima/scenarijima te koliko je njih provjereno i potvrđeno, odnosno koliko je isporučeno radećeg programskog koda [9].

Složenost sustav iziskuje vremenski zahtjevan razvoj i ne postoje velike mogućnosti i vrijeme dostupno za implementaciju promjena ili ispravaka nakon što je sustav u cijelosti završen. Trošak promjena se povećava u odmaklim fazama projekta i različiti tipovi promjena imaju različite profile troškova. RUP je postavljen na način da minimizira ukupan trošak promjene, a istovremeno maksimizira mogućnost promjena. Promjena u poslovnim zahtjevima u odmakloj fazi projekta može rezultirati povećanim troškovima implementacije te je stoga važno što ranije postići suglasnost oko svakog funkcionalnog zahtjeva, naročito u fazi pripreme (inception). Nadalje, promjene u arhitekturi sustava moraju biti prihvaćene do kraja faze razrade kako bi se osigurala isplativost projekta. Promjene u dizajnu/implementaciji bi trebale biti definirane do kraja faze provedbe. Važna napomena je da stalne promjene onemogućavaju da se projekt ikada završi [9].

Održiva i stabilna arhitektura jedan je od ključeva uspjeha projekta razvoja informacijskog sustava. Značajni rizici javljaju se upravo vezano uz arhitekturu, osobito prilikom razvoja prve verzije sustava. Najvažniji segment analize i modeliranja je definiranje arhitekture sustava, odnosno određivanja njegovih gradbenih komponenti, prepoznavanje zajedničkih komponenti i definiranje načina povezivanja s postojećim sustavima. Početna arhitektura često ne osigurava ispunjavanje ključnih zahtjeva korisnika, te kako bi se to izbjeglo RUP u početnim iteracijama stavlja težište na definiranje odnosno redefiniranje arhitekture sustava [9].

Modularni razvoj sustava podrazumijeva razvoj sustava po komponentima. Razvoj temeljen na komponentama oslanja se na načelo enkapsulacije i omogućuje izradu aplikacija koje su otpornije na promjene. Komponente osiguravaju visoki stupanj ponovne upotrebe što omogućuje bržu izradu aplikacije više kvalitete. Navedeni pristup radikalno smanjuje troškove održavanja sustava [9].

RUP naglašava važnost dobre komunikacije unutar tima kao jednu od temeljnih pretpostavki za uspješnost projekta. Projektni timovi se organiziraju multidisciplinarno, što znači da se unutar jednog tima mogu pronaći različite uloge kao npr. softver arhitekt, poslovni analitičar, razvojni tim, testni tim, sistemska podrška, IT arhitekt. Iako su uloge usko specijalizirane, aktivnosti obavljaju svi članovi tima te oni nisu striktno dodijeljene određenim ulogama [9].

Osiguravanje visoke kvalitete zahtijeva više od samog sudjelovanja testnog tima. Uključuje sve članove tima i sve dijelove životnog ciklusa. Korištenje iterativnog pristupa omogućava ranije testiranje, a koncept kvalitete po dizajnu omogućuje dizajnerima softverskog rješenja automatizaciju generiranja testnog koda ranije i kvalitetnije testiranje čime se smanjuje broj pogrešaka u testnom kodu. Atributi kvalitete sustava su mogućnost njegovog održavanja, pouzdanost i sigurnost sustava, učinkovitost i upotrebljivost [9].

RUP predstavlja jednu od korištenijih praksi u razvoju informacijskih sustava, a kao ključni koraci koje RUP sadrži su iterativni razvoj softvera, upravljanje zahtjevima, korištenje komponenti baziranih na arhitekturi, vizualno modeliranje softvera, provjera kvalitete softvera i kontroliranje promjena softvera. Temeljne aktivnosti projekata na kojima se primjenjuje RUP metodologija su

prikupljanje korisničkih zahtjeva kako bi se utvrdilo što sustav treba raditi, analiza prikupljenih zahtjeva i njihovo strukturiranje i prioritizacija, oblikovanje odnosno prijedlog građe sustava koja će omogućiti da se korisnički zahtjevi realiziraju, implementacija u kojoj se stvara softver koji realizira predloženu građu i testiranje kako bi se provjerilo radi li kreirani softver zaista onako kako bi trebao [9].

6.2. Faze razvoja RUP metodologije

Kao što je navedeno u četvrtom poglavlju, RUP je fazna metodologija odnosno sastoji se od četiri faze životnog ciklusa koje se izvršavaju slijedno. Četiri faze od kojih se sastoji RUP su faze pripreme, razrade, provedbe i tranzicije.

6.2.1. Faza pripreme - Inception

Faza pripreme je prva faza životnog ciklusa RUP projekta i obuhvaća dobro definirane ciljeve te završava prekretnicom. U fazi pripreme nastoji se razumjeti opseg projekta i ciljeve te prikupiti čim više informacija koje će dati potvrdu kako se treba nastaviti s projektom. Temeljni ciljevi koji se nastoje razriješiti u pripremnoj fazi su razumijevanje što se treba razviti, odnosno odrediti viziju i opseg sustava te njegove granice te identificirati tko su korisnici sustav i koja je vrijednost koju će sustav isporučiti korisnicima. Zatim, identificirati ključne funkcionalnosti sustava i donošenje odluke koji su slučajevi uporabe najkritičniji i koje je potrebno prvo razviti. Definiranje barem jednog rješenja, odnosno identifikacija barem jedne arhitekture sustava. Razumijevanje troška, rasporeda i rizika koji su povezani s projektom te donošenje odluke koje razvojne procese slijediti i koje alate koristiti [9].

Razumijevanje onoga što se treba razviti je jedan od ključnih ciljeva koje je potrebno ispuniti u fazi pripreme. Članovi projektnog tima koji se sastoji od različitih uloga imaju drugačije poglede na način i pristup kojim se treba voditi kako bi se došlo do zajedničkog cilja. Iz tog razloga vrlo je važno postići dogovor između korisnika, menadžmenta, analitičara, programera, testera i ostalih ključnih dionika o tipu sustava koji se nastoji razviti. Nadalje, potrebno je ukratko opisati što se sustavom nastoji postići i što bi taj sustav trebao raditi, bez pružanja pretjeranih detalja koji bi potencijalno zaokupirali pažnju dionika, a u ovoj fazi razvoja nisu potrebni. Nadalje, potrebno je detaljizirati ključne aktere i slučajeve uporabe kako bi ih dionici lako razumjeli i članovi projektnog tima ih mogli koristiti kao direktan input za obavljanje svog dijela posla. Također, vrlo je bitno održati radionice s korisnicima i članovima projektnog tima kako bi se razmišljanja mogla razmijeniti i usuglasiti se oko jednog pravca kojim će se ići s razvojem sustava. Na navedenoj radionici bi se trebali identificirati ključni slučajevi uporabe i akteri koji u njima sudjeluju te se zatim odraditi njihova prioritizacija, Također, cilj je kroz radionice identificirati i ključne funkcionalnosti koje sustav mora ispunjavati. S obzirom da je ključni cilj faze pripreme,

determinirati ima li smisla nastaviti s projektom, potrebno je definirati barem jednu potencijalnu arhitekturu koja će osigurati izradu sustava s minimalnim rizikom i u granicama budžeta. Također, potrebno je obratiti pažnju i na troškove koji će se pojaviti u procesu razvoja, vrijeme koje je dostupno za razvoj sustava i koji se sve rizici mogu pojaviti. Također, potrebno je odraditi i analizu tehnologija i alata koji su najprimjereniji za razvoj sustava takvog opsega. U tom pogledu se posebna pažnja obraća i na projektni tim i iskustvo koje članovi projektnog tima imaju s određenim tehnologijama ili alatima [9].

Na kraju pripremne faze pojavljuje se prva prekretnica koja se naziva „*Lifecycle Objective Milestone*“. Projekt treba prekinuti ili ponovno razmotriti ako ne dosegne ovu prekretnicu. Ako je projekt osuđen na neuspjeh, bolje je to shvatiti rano nego kasno, a iterativni pristup u kombinaciji s ovom prekretnicom može dovesti do ranog shvaćanja što ima svoje pozitivne strane, a to su ne trošenje dragocjenih resursa. *Lifecycle Objective Milestone* uključuje sljedeće kriterije evaluacije [9]:

- suglasnost dionika o definiciji opsega i početnoj procjeni troškova/rasporeda (koja će se doraditi u kasnijim fazama),
- suglasnost da je skupljen pravi skup zahtjeva i da postoji zajedničko razumijevanje istih,
- suglasnost da su procjena troškova/plana, prioriteti, rizici i razvojni proces prikladni,
- suglasnost da su početni rizici identificirani i da za svaki postoji strategija ublažavanja.

Priprema (*Inception*) je prva od četiri faze životnog ciklusa u RUP-u. Bazira se na razumijevanju projekta i dobivanju dovoljno informacija kako bi se stekla potvrda kako treba nastaviti s projektom ili odustati od projekta. Na završetku projekta najčešći produkti pripremne faze su opisani opseg projekta, osigurani resursi i sredstva za njegovu provedbu. Navedeno se manifestira kroz potpisani ugovor od strane klijenta i tvrtke koja razvija softver te dokumenta s opsegom projekta i specifikacija korisničkih zahtjeva [9].

6.2.2. Faza razrade – Elaboration

Faza razrade je faza u kojoj se jasno vide razlike između vodopadnog pristupa i iterativno RUP pristupa. Najveća razlika se iskazuje u vrstama aktivnosti koje se provode. Glavne prednosti RUP se iskazuju kroz prikaz glavnih rizika, izradu rane arhitekture sustava i razradu planova definiranih u fazi pripreme. Cilj faze razrade je definirati i postaviti temelje arhitekture sustava kako bi se osigurala stabilna osnova za većinu aktivnosti dizajniranja i implementacije u narednoj fazi, fazi provedbe. Arhitektura se razvija iz razmatranja najvažnijih zahtjeva (onih koji imaju veliki utjecaj

na arhitekturu sustava) i procjeni rizika. Temeljni ciljevi koji se nastoje razriješiti kroz ovu fazu su detaljnija spoznaja, analiza i razumijevanje zahtjeva. Zatim, dizajniranje, implementacija i validacija temeljne arhitekture. Smanjivanje ključnih rizika i produciranje realističnijih rokova i procjene troškova [9].

Na kraju faze razrade nalazi se prekretnica koja se naziva *„Lifecycle Architecture Milestone"*. Ova prekretnica označava procjenu detaljnih ciljeva i opsega sustava, odabir arhitekture i glavnih rizika na projektu. Ako projekt ne dosegne ovu prekretnicu i dalje se ostavlja mogućnost da se prekine ili da se razmotri ta opcija. *Lifecycle Architecture Milestone* obavlja procjenu je li vizija projekta stabilna i jesu li identificirani zahtjevi stabilni. Također, je li temeljna arhitektura stabilna i jesu li pristupi testiranju i evaluaciji uspješni. Nadalje, jesu li testiranja i evaluacije izvršnih prototipa rezultirali aktivacijom nekih ključnih rizika i na koji način je pristupljeno rizicima. Zatim, jesu li planovi iteracije za narednu fazu dovedeni do zadovoljavajuće razine detalja i vjernosti kako bi se moglo nastaviti s projektom i jesu li izvršene realne procjene. Također, bitan aspekt su i dionici koji u ovoj fazi trebaju podržavati viziju i imati povjerenje u projektni tim kako će se trenutno definirani plan izvršiti uz postojeću temeljnu arhitekturu. Produkti koji proizlaze iz ove faze su jasno raspisane specifikacije slučajeva uporabe kroz koje se nastoje usuglasiti osnovni korisnički zahtjevi i isti prezentirati u dokumentarnom obliku razvoj timu [9].

6.2.3. Faza provedbe – Construction

Faza provedbe je treća faza RUP procesa i uzima najviše vremena jer se ona bazira na izradi visoko kvalitetnog programskog koda. Ciljevi koji se nastoje se uglavnom baziraju na izradi troškovno efikasnog finalnog proizvoda, odnosno operativne verzije sustava koja se može isporučiti u korisničku okolinu. Ciljevi koji se nastoje ostvariti kroz ovu fazu su minimiziranje troškova razvoja i postizanje određene razine paralelnog rada razvojnih timova. Optimizacija resursa i izbjegavanje nepotrebnih popravaka. Nadalje, sljedeći cilj je iterativni razvoj finalnog proizvoda koji će biti spreman za isporuku u korisničku okolinu. Navedeno podrazumijeva razvoj inicijalne verzije operativnog sustava kroz opisivanje preostalih slučajeva uporabe i ostalih zahtjeva, nadopunjavanje detalja dizajniranja i završavanjem implementacije i testiranja sustava [9].

Faza izgradnje završava važnom prekretnicom projekta, *Initial Operational Capability Milestone*, koja se koristi za utvrđivanje je li proizvod spreman za implementaciju u testno okruženje na način da se daje odgovor na sljedeća pitanja [9]:

- Je li proizvod dovoljno stabilan za isporuku u korisničku okolinu?
- Jesu li dionici spremni za isporuku proizvoda u korisničku okolinu?
- Jesu li troškovi utrošenih resursa i dalje prihvatljivi?

Ukoliko je na neko od ovih pitanja odgovor negativan, najčešće se prolazi kroz još jednu iteraciju faze provedbe kako bi se ostvarili ciljevi postavljeni za ovu fazu. Ukoliko je dosegnuta prekretnica ove faze, postignuto je sljedeće [9]:

- izrađuje se troškovno efikasan sustav kroz iskorištavanje temeljne arhitekture definirane u fazi razrade,
- omogućeno je povećanje razine projekta i uključivanje novih članova projektnog tima ukoliko je potrebno,
- izgrađeno je nekoliko internih isporuka kako bi se osigurala iskoristivost sustava i ispunjavanje korisničkih potreba,
- isporučena je prva operativna verzija sustava koji je u potpunosti funkcionalan u testnoj okolini, uključujući instalaciju, potpornu dokumentaciju i korisničke upute.

Produkt koji proizlazi iz ove faze je funkcionalna aplikacija koja je isporučena u testnoj okolini i spremna je za testiranje.

6.2.4. Faza tranzicije – puštanje u rad

Četvrta i posljednja faza RUP procesa je faza tranzicije, odnosno puštanja u rad. Fokus prijelazne faze je osigurati da softver u potpunosti odgovara potrebama svojih korisnika. Faza tranzicije normalno obuhvaća jednu ili dvije iteracije koje uključuju testiranje proizvoda u pripremi za isporuku i izradu manjih prilagodbi na temelju povratnih informacija korisnika. U ovoj fazi životnog ciklusa sva glavna strukturalna pitanja trebala bi biti razrađena, a povratne informacije korisnika bi se trebale odnositi uglavnom na pitanja nekih manjih prilagodbi, konfiguracija i instalacije. Složeniji projekti uglavnom imaju više faza tranzicije, odnosno podijeljeni su u više isporuka, od kojih je svaka novija isporuka zapravo nadogradnja prethodne. Ciljevi koji se nastoje postići u fazi isporuke su popravak uočenih grešaka, pripremiti sustav za isporuku na korisničku okolinu, distribucija sustava na radna mjesta, kreiranje korisničke dokumentacije i organiziranje podrške korisnicima i izrada *post-project review* dokumenta [9].

Faza tranzicije završava s prekretnicom koja se naziva *„Product Release Milestone“*, a očituje se kroz procjenu jesu li svi ciljevi ispunjeni i je li potrebno pokrenuti novu fazu razvoja. Glavna pitanja na koje ova faza treba pružiti odgovor su [9]:

- Jesu li korisnici zadovoljni?
- Jesu li stvarni izdaci resursa u odnosu na planirane prihvatljivi i, ako nisu, koje se radnje mogu poduzeti u budućim projektima za rješavanje ovog problema?

U prekretnici *Product Release Milestone*, sustav odnosno proizvod je u produkciji i započinje proces održavanja, operativnosti i podrške. Navedeno može uključivati novi ciklus razvoja s nekim

većim zahvatima ili nekom dodatnom isporukom održavanja koja služi za popravak pronađenih defekata. Produkt ove faze je u potpunosti funkcionalna aplikacija koja ispunjava korisničke zahtjeve [9].

6.3. Primjer primjene RUP metodologije za razvoj informacijskih sustava

Za bolje razumijevanje procesnih koraka RUP metodologije važno je prikazati procesne faze na nekom primjeru izrade sustava. Informacijski sustav koji je uzet kao primjer u sklopu ovog diplomskog rada nije previše kompleksan, ali je dovoljan za jasno prikazivanje svih faza razrade informacijskog sustava. Kao primjer uzet je sustav za upravljanje Internet trgovinom. Internet trgovina je tip informacijskog sustava podrške čija je svrha pružanje informacija o artiklima korisnicima te također treba omogućiti mogućnost kupovine artikala i korisničku podršku za reklamacije. Funkcionalnosti koje bi takav sustav trebao zadovoljiti razlikuju se od korisnika do korisnika i vrste poslovanja trgovine za koje se takav sustav razvija.

Kao što je opisano u poglavlju 6.2.1., prva faza razvoja informacijskog sustava RUP metodologijom je faza pripreme. U fazi pripreme nastoje se prikupiti osnovne informacije od klijenta o sustavu koji se nastoji izraditi, odnosno prikupiti korisničke zahtjeve. Međutim, prije nego se krene s razvojem informacijskog sustava, potencijalni izvođač radova kreira i dostavlja ponudu za izradu informacijskog sustava klijentu na razmatranje. Nakon što klijent prihvati ponudu i potpiše se ugovor, kreću prve aktivnosti projektnog tima u procesu pripreme. U fazi pripreme kreira se vizija kako bi se projekt trebao razvijati, definiraju se tjedni ili mjesečni planovi, rizici koji se potencijalno mogu javiti tijekom razvoja te na kraju definira se i poslovni slučaj korištenja. Određuje se veza između sudionika u razvoju aplikacije te se utvrđuju načini rada između pojedinih objekata. Fokus ove faze je usmjeren na razumijevanje troškova i rizika, te je poželjno definiranje barem jednog mogućeg rješenja s odgovarajućom arhitekturom [16].

Sustav se u prvom redu razvija kako bi se poslovanje zamišljene Internet trgovine automatiziralo, odnosno kako bi se ubrzao proces odgovaranja na korisničke upite i reklamacije te kako bi se pojednostavnio proces Internet transakcija i kupovine artikala. Radi se o maloj Internet trgovini koja ima potrebu za centraliziranjem svog poslovanja. Time je ugrubo definirana potreba zbog kojeg se kreće s razvojem novog informacijskog sustava. Nakon definiranja potrebe, važno je prikazati viziju te odrediti vrijeme u kojem roku bi sustav trebao biti isporučen. S obzirom da se radi o vrlo jednostavnoj Internet trgovini zamišljena je vrlo jednostavna arhitektura pa se ne predviđaju visoki troškovi za izvođača radova te je vremenski period u kojem bi se sustav trebao razviti nekoliko mjeseci. Rizici koji se mogu javiti u procesu izrade informacijskog sustava najčešće rezultiraju kašnjenjem i neplaniranim povećanjem troškova. Kašnjenje i povećanje troškova mogu rezultirati kao posljedica nekvalitetne poslovne analize, nedovoljno analiziranih korisničkih zahtjeva, nedostatak komunikacije između projektnog tima i klijenta i neispunjavanje

zacrtanih ciljeva. Također, vrlo je važno odraditi prioritizaciju korisničkih zahtjeva kako se ne bi previše vremena trošilo na manje važne funkcionalne zahtjeve. Kao potencijalni rizik Internet trgovine mogao bi se javiti nedostatak prostora za pohranu korisničkih podataka i nestabilna infrastruktura transakcija čiji bi se prijedlozi za rješavanje trebali definirati u narednim fazama razvoja. U fazi pripreme najčešće sudjeluju voditelji projekta čija je uloga uspostaviti projektni tim i inicijalnu komunikaciju s klijentom. Također u fazi pripreme sudjeluju i poslovni analitičari koji prikupljaju inicijalne korisničke zahtjeve koji se zatim detaljnije razrađuju kroz naredne faze procesa razvoja.

Nakon faze pripreme projekta slijedi faza razrade. Definiranje poslovnih slučajeva uporabe i modela poslovnih procesa trebao bi biti fokus ove faze. Projektni tim nastoji analizirati prikupljene funkcionalne i nefunkcionalne zahtjeve i odrediti prioritizaciju pojedinog poslovnog slučaja uporabe. Funkcionalni zahtjevi daju odgovor na pitanje što sustav radi dok nefunkcionalni zahtjevi daju odgovor na pitanje kako sustav radi. Funkcionalni i nefunkcionalni zapisani su od strane projektnog tima, najčešće poslovnog analitičara u specifikaciji zahtjeva iz koje se zatim izrađuje model slučajeva uporabe u kojem su grafički prikazani slučajevi uporabe kroz koje će biti razrađeni funkcionalni i nefunkcionalni zahtjevi. Sustav prije svega treba omogućiti korisnicima prijavu u sustav te pregled dostupnih artikala i obavljanje online transakcija. Navedene funkcionalnosti se mogu smatrati visoko prioritetnim funkcionalnostima kojima treba posvetiti više vremena u procesu razvoja. Također, modeliraju se poslovni procesi i uvrštavaju u dokument modela poslovnih procesa koji se sastoji od dijagrama aktivnosti kako bi se jasno prikazala slijednost koraka u određenoj interakciji sustava i korisnika. Pri izradi modela poslovnih procesa najčešće se koristi grafički prikaz za specificiranje poslovnih procesa, tj. notacija za modeliranje poslovnih procesa (*Business Process Model and Notation* - BPMN). Nakon što se definiraju poslovni procesi i odrede slučajevi uporabe cilj je iste prikazati kroz specifikacije slučajeva uporabe koje razrađuju detaljne korake i poslovna pravila koja je potrebno implementirati kako bi sustav funkcionirao u skladu s predviđenim.

Nakon što se analiziraju poslovni procesi koje informacijski sustav Internet trgovine treba zadovoljiti i jasno se dokumentiraju kroz specifikacije slučajeva uporabe, slijedi faza provedbe. Kroz fazu provedbe razvojni tim nastoji funkcionalne specifikacije slučajeva uporabe koji sadrže detaljne korake i poslovna pravila, pretvoriti u funkcionalna programski kod. Pri tome se za izradu programskog koda koriste razni programski jezici kao što su C#, C++, Python i slično te se također definira struktura i arhitektura baze podataka i koriste se neki od alata za upravljanje bazom podataka kao što su MS SQL Server, DB Visualiser i slično. S obzirom da je RUP metodologija iterativna, ona omogućuje da ukoliko određeni zahtjevi nisu dovoljno detaljno objašnjeni, da se ponovno odradi poslovna analiza koja je dio faze razrade. Nakon što se razvije programski kod, sljedeći korak je isporuka aplikacije u testnu okolinu kako bi se moglo izvršiti testiranje funkcionalnosti i otkloniti potencijalne greške.

Nakon što se sustav razvije i odradi se testiranje, aplikacija se isporučuje u produkcijsku okolinu čime započinje završna faza razvoja sustava Internet trgovine. Produkt završne faze je

funkcionalna aplikacija koju korisnici mogu koristiti za naručivanje artikala putem Internet trgovine. U ovoj fazi je također bitno izraditi korisničke upute koje se implementiraju u sklopu korisničkog sučelja.

7. Zaključak

Današnja svakodnevica u velikom postotku podrazumijeva upotrebu modernih tehnologija. Od privatnog, društvenog pa do poslovnog života, u svim navedenim aspektima neka osoba u nekom trenutku u danu upotrebljava određeni informacijski sustav. Informacijsko komunikacijske tehnologije s posebnim naglaskom na informacijske sustave imaju zadaću automatizacije, pojednostavljenja i ubrzanja svakodnevnih ljudskih procesa. Primjenom informacijskih tehnologija u poslovnom svijetu, nastoje se automatizirati poslovni procesi organizacije kako bi se smanjili troškovi, maksimizirali profiti i smanjila mogućnost ljudske pogreške. Kako bi se navedeno postiglo vrlo je važno imati kvalitetan informacijski sustav koji odrađuje svoju funkciju, a ne troši bespotrebno resurse organizacije. Organizacije čiji se poslovni procesi oslanjaju na interne informacijske sustave i koje se odluče na digitalizaciju svog poslovanja imaju određena funkcionalna očekivanja od tih informacijskih sustava. Navedeno se u stručnoj literaturi predstavlja kao korisnički zahtjevi, a oni opisuju funkcionalnosti koje sustav mora obavljati. Korisnički zahtjevi kao što je prikazano u šestom poglavlju su temeljni element razvoja bilo kojeg informacijskog sustava. Oni moraju biti jasno komunicirani od strane klijenata razvojnom, odnosno projektnom timu. Projektni tim koji mora maksimalno iskoristiti informacije koje dobije od korisnika, kako bi se mogla napraviti kvalitetna specifikacija korisničkih zahtjeva. Korisnički zahtjevi su samo dio razvoja informacijskog sustava. Najvažniji element je odabir ispravne metodologije za razvoj informacijskog sustava.

Prilikom odabira metodologije važno je obratiti pažnju na više faktora, a to su veličina projekta, budžet, opseg projekta, kompleksnost funkcionalnosti koje se razvijaju i slično. Tradicionalne metodologije koje su pojašnjene u četvrtom poglavlju su poprilično zastarjele metode te se vrlo rijetko koriste u današnje vrijeme jer sve više projektnih timova se odlučuje za agilni razvoj koji, kao što je prikazano u poglavlju 4.2. ima karakteristike fleksibilnosti, brzine, jednostavnosti i okrenut je na timski rad i ispripletenost aktivnosti koje članovi projektnog tima izvode. Agilne metodologije su korisne u slučajevima manjih i srednje velikih projekata, a njihova primjena kod velikih i kompleksnih projekata, ne često zna naići na nepremostive prepreke.

RUP čije su procesne faze prikazane u šestom poglavlju je iterativna metodologija razvoja informacijskih sustava koja je orijentirana na arhitekturi i upotrebi slučajeva uporabe koja se sve više koristi u IT industriji. Razlog tome je što se RUP bazira na IT praksama koje su više puta testirane i implementirane u različitim projektima i domenama. RUP metodologija zapravo predstavlja jasno definirani procesni okvir za razvoj sustava koji kombinira najbolje prakse iz tradicionalnih metodologija kao što su iscrpna dokumentacija, slijedno izvršavanje faza razvoja i najbolje prakse agilnih metodologija kao što su iterativnost i inkrementalnost. Navedeno omogućuje projektnom timu koji primjenjuje RUP metodologiju u razvoju visoku razinu dokumentiranosti, jasnu podjelu uloga projektnog tima, ali i brži i jednostavniji pristup cijelom procesu razvoja što u konačnici rezultira visoko kvalitetnim programskim rješenjima. Također kao

što je vidljivo na slici 21. iterativne metodologije od kojih je jedna i RUP uvelike smanjuje rizik i vrijeme utrošeno na razvoj informacijskog sustava. Nadalje, RUP je dobro strukturiran i jasno definira uloge razvojnog tima, od analitičara, programera, testera, voditelja projekata, administratora baze podataka i dr. te njihove odgovornosti i aktivnosti koje poduzimaju u određeno vrijeme životnog ciklusa razvoja softvera. Iako je detaljno strukturiran, RUP je dovoljno fleksibilan procesni razvoj okvir, pa tako omogućuje jednostavnu primjenu na razvojne projekte različitih veličina.

POPIS LITERATURE

1. Peraković D., Periša M., Informacijski sustavi mrežnih operatora. [Prezentacija] Podjele, vrste i elementi informacijskog sustava. Fakultet prometnih znanosti Sveučilišta u Zagrebu. 2018.
2. Mesarić J., Šebalj D., Oblikovanje i implementacija informacijskih sustava. [Prezentacija] Osnove sistemskog pristupa. Ekonomski fakultet u Osijeku Sveučilišta J.J. Strossmayera u Osijeku, 2020.
3. Big Water Consulting. Software Development Life Cycle (SDLC). Preuzeto s: https://bigwater.consulting/2019/04/08/software-development-life-cycle-sdlc/ [Pristupljeno: 01. srpnja 2022.]
4. Vresk A. Agilne metode za upravljanje projektima. Završni rad. Sveučilište u Zagrebu, Fakultet organizacije i informatike; 2020. Preuzeto s: https://repozitorij.foi.unizg.hr/en/islandora/object/foi:6341 [Pristupljeno: 07. srpnja 2022.]
5. Langer AM. Guide to Software Development, Designing and Managing the Life Cycle. Second Edition. New York: Springer Nature, Springer-Verlag London Ltd; 2012. Preuzeto s: https://b-ok.xyz/book/1247104/a8856d [Pristupljeno: 10. srpanj 2022.]
6. Javanmard M., Alian M., Comparison between Agile and Traditional software development methodologies. Second National Conference on Applied Research in Computer Science and Information Technology. Tehran: Payam Noor University; 2015. Preuzeto s: https://www.researchgate.net/publication/274918013_Comparison_between_Agile_and_Traditional_software_development_methodologies [Pristupljeno: 10.srpnja 2022.]
7. Virtasant. SDLC Methodologies: From Waterfall to Agile. Preuzeto s: https://www.virtasant.com/blog/sdlc-methodologies [Pristupljeno: 15.srpnja.2022.]
8. Agile Manifesto. Manifesto for Agile Software Development. Preuzeto s: https://agilemanifesto.org/iso/en/manifesto.html [Pristupljeno: 20.07.2022]
9. Kroll P., Kruchten P., Rational Unified Process Made Easy: A Practitioner`s Guide to RUP. Boston: Addison Wesley; 2003. Preuzeto s: https://www.semanticscholar.org/paper/The-Rational-Unified-Process-Made-Easy-A-Guide-to-Kroll-Kruchten/0543cf186bbea1ca6111265fb838520b1d5b42a0 [Pristupljeno: 22.srpnja.2022]
10. Ashraf A. A Review of RUP (Rational Unified Process). International Journal of Software Engineering (IJSE), 2014;5(2). Preuzeto s: https://www.academia.edu/11981553/A_Review_of_RUP_Rational_Unified_Process_-_2014 [Pristupljeno: 01.kolovoza.2022]
11. Mrvelj Š., Analiza i modeliranje prometnih sustava. [Prezentacija] Sustav i model sustava. Fakultet prometnih znanosti Sveučilišta u Zagrebu. 10. rujna 2020.
12. Lethbridge TC., Laganiére R. Object – Oriented Software Engineering, Practical software development using UML and Java. Second Edition. Berkshire: McGraw-Hill Education; 2005. Preuzeto s:

https://www.academia.edu/8751249/Object_Oriented_Software_Engineering_Practical_Software_Development_using_UML_and_Java [Pristupljeno: 04. kolovoza.2022]
13. Botonjić A. UML – Unified Modeling Language – Use Case Diagram. Završni rad. Sveučilište Jurja Dobrile u Puli, Fakultet ekonomije i turizma "Dr. Mijo Mirković". 2015. Preuzeto s: https://zir.nsk.hr/islandora/object/unipu:333/datastream/PDF/view [Pristupljeno: 03.kolovoza 2022.]
14. UML diagrams. Preuzeto s: https://www.uml-diagrams.org/uml-25-diagrams.html#google_vignette [Pristupljeno: 04. kolovoza 2022.]
15. IBM Developer. An introduction to the Unified Modeling Language. Preuzeto s: https://developer.ibm.com/articles/an-introduction-to-uml/?mhsrc=ibmsearch_a&mhq=Rational%20Unified%20Process [Pristupljeno: 12. kolovoza 2022.]
16. Jakus Matešković V. Rational Unified Process metodologija razvoja softvera. Diplomski rad. Sveučilište Jurja Dobrile u Puli, Fakultet ekonomije i turizma "Dr. Mijo Mirković". 2018. Preuzeto s: https://dabar.srce.hr/islandora/object/unipu%3A3039 [Pristupljeno: 12. Kolovoza 2022.]
17. Manger R., Softversko inženjerstvo. [Prezentacija] Uvod u UML i UP. Prirodoslovno matematički fakultet – matematički odsjek Sveučilišta u Zagrebu. 13. rujna 2021.

POPIS GRAFIČKIH PRIKAZA

Sveučilište u Zagrebu
Fakultet prometnih znanosti
Vukelićeva 4, 10000 Zagreb

IZJAVA O AKADEMSKOJ ČESTITOSTI I SUGLASNOSTI

Izjavljujem i svojim potpisom potvrđujem da je DIPLOMSKI RAD
(vrsta rada)
isključivo rezultat mojega vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju upotrijebljene bilješke i bibliografija. Izjavljujem da nijedan dio rada nije napisan na nedopušten način, odnosno da je prepisan iz necitiranog rada te da nijedan dio rada ne krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Svojim potpisom potvrđujem i dajem suglasnost za javnu objavu završnog/diplomskog rada pod naslovom Analiza agilnih metodologija za razvoj informacijskih sustava, u Nacionalni repozitorij završnih i diplomskih radova ZIR.

Student/ica:

U Zagrebu, 12.09.2022

Stanislav
(ime i prezime, *potpis*)