Rational unified process metodologija razvoja softvera

Jakus Matešković, Vedrana

Sveučilište Jurja Dobrile u Puli

Fakultet ekonomije i turizma

„Dr. Mijo Mirković“

VEDRANA JAKUS MATEŠKOVIĆ

RATIONAL UNIFIED PROCESS METODOLOGIJA RAZVOJA SOFTVERA

Diplomski rad

Pula, Prosinac 2018.

Sveučilište Jurja Dobrile u Puli

Fakultet ekonomije i turizma

„Dr. Mijo Mirković“

VEDRANA JAKUS MATEŠKOVIĆ

RATIONAL UNIFIED PROCES METODOLOGIJA RAZVOJA SOFTVERA

Diplomski rad

JMBAG: 0303018010, redovni student

Studijski smjer: Poslovna informatika

Kolegij: Razvoj informacijskog sustava

Mentor: prof. dr.sc. Vanja Bevanda,

Pula, Prosinac 2018.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani _____, kandidat za magistra ekonomije/poslovne ekonomije ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, _____, _____ godine

IZJAVA

o korištenju autorskog djela

Ja, _____ dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom _____

_____

_____koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ (datum)

Potpis

_____

SAŽETAK/SUMMARY

Ovim radom se ukazuje na mogućnost primjene *Rational Unified process* metodologije od strane malog tima, odnosno od strane jedne osobe.

Metodologija pruža lak uvid u bazu znanja pojedinog tima, odnosno svi sudionici koriste zajedničku metodologiju te imaju isti pogleda na razvoj softverskog proizvoda. Detaljno su definirane odgovornosti unutar tima, te se metodologija često koristi i kao radni okvir lako prilagodljiv potrebama organizacije.

Jedna od glavnih karakteristika ovog pristupa je iterativni razvoj koji omogućava realizaciju korisničkih zahtjeva. Također, karakteristične su i četiri faze koje čine životni ciklus proizvoda, a to su: faza započinjanja, faza razrade, faza konstrukcije i faza tranzicije.

This master thesis shows that small team, or even one person can apply Rational Unified process methodology .

Methodology gives access to knowledge DataBase, which means that all team members use the same methodology and they have the same view on software product development. Team member responsibilities are defined in detail and the methodology is often used as work frame in accordance with organizational needs.

One of the main characteristics is iterative development which allows realization of users requests. Also, the main characteristic of this methodology are four phases of product life cycle: inception phase, elaboration phase, construction phase and transition phase.

SADRŽAJ:

1 UVOD

U novije je vrijeme razvoj aplikacija i softvera postao uobičajena praksa, a u svrhu olakšavanja i ubrzavanja poslovnih procesa. Kako se softveri razlikuju ovisno o tome za što su namijenjeni tako je i *Rational unified process*[1] metodologija pronašla svoje mjesto u njihovom razvoju. Naravno, valja naglasiti kako uporaba spomenute metodologije ovisi o razvijatelju softvera i njegovom timu, te potrebama koje klijent postavi pred njih.

Spomenuta se metodologija počinje koristiti još 90-ih godina 20. stoljeća kada je i kreirana, a kasnije se samo nadopunjavala kako bi na kraju predstavljala metodologiju pogodnu za upravljanje projektima i razvoj stvarnih softvera. Usmjerenost na arhitekturu i stvaranje slučajeva uporabe prema zahtjevima klijenata su prednosti koje ju ističu iz mase drugih pristupa razvoju softvera.

Početna namjena bila je usmjerena isključivo na velike i složenije projekte, no kasnija potreba za sve više manjih softvera (koji se danas češće koriste u svakodnevnom životu) doveli su do njezine prilagodbe manjim projektima.

U ovom će se radu razmatrati uporaba RUP metodologije u razvoju softvera, te je cilj prikazati njezinu primjenu s aspekta promatrača, a koji će se naći i u ulozi razvijatelja softvera. Također, jedan od ciljeva je i približavanje terminologije hrvatskom govornom području na kojem je takva literatura vrlo teško dostupna. Stručni radovi su također teško dostupni, te ne objašnjavaju detaljniju primjenu RUP metodologije.

Potrebno je naglasiti i kako je razina uspješno završenih softverski projekata veoma niska. Prema istraživanju tvrtke The Standish Group[2], čak 31,1% projekata se otkaže, što se uglavnom odnosi na tvrtke srednje veličine koje ostvaruju dobit između 200.000.000,00 i 500.000.000,00 dolara godišnje. Manje tvrtke, s godišnjom dobiti do 200.000.000,00 dolara, imaju veću stopu uspješno završenih softverskih projekata iz čega je proizašla i hipoteza ovog diplomskog rada.

[1] U daljnjem tekstu RUP

[2] The Standish Group Report, 2014., Dostupno na: https://www.projectsmart.co.uk/white-papers/chaos-report.pdf, Pristupljeno: 27.11.2018

Hipoteza rada je da se navedena metodologija može uspješno primijeniti na razvoj mobilnih aplikacija koje su danas sve više zastupljene, te da ju može primijeniti mali tim, odnosno samo jedna osoba.

Za razumijevanje spomenute metodologije bilo je potrebno najprije utvrditi temelje razvoja softvera i definirati osnovne modele za razvoj softvera. Neki od tih modela se koriste i u primjeni RUP metodologije.

Rad se sastoji od pet cjelina. Nakon uvoda u prvoj su cjelini definirani osnovni pojmovi programskog inženjerstva, te su u drugoj cjelini pojašnjeni temelji razvoja softvera. Odnosno, opisani su modeli i metode za razvoj softvera. Treća se cjelina kratko dotakla brzog razvoja aplikacija, a čija je namjena u današnje vrijeme sve rasprostranjenija. Nakon toga se četvrta i peta cjelina bave isključivo RUP metodologijom. Dok su u četvrtoj cjelini definirani osnovni pojmovi potrebni za razumijevanje spomenute metodologije, u petoj cjelini je metodologija primijenjena za razvoj aplikacije *ConstSoft* u sektoru građevine. Za prikaz primjene razvoja spomenute aplikacije korišteni su isključivo besplatno dostupni alati.

Na kraju rada nalazi se zaključak, popis literature, popis slika, tabela i grafova, te prilozi.

Metode korištene u istraživanju su metode dedukcije i indukcije, dok su za pisanje rada korištene metode deskripcije i komparacije. Veći dio literature koja je korištena za pisanje rada čine knjige, članci i materijali stranih autora.

2 PROGRAMSKO INŽENJERSTVO

NATO-va konferencija o softverskoj krizi 1968. godine predstavila je i popularizirala programsko inženjerstvo, te je ono od ideje koju je prakticirao mali broj ljudi naraslo u značajnu inženjersku granu. Uzrok je velikih istraživanja, ulaganja, ali i mnogih rasprava.

„Programsko inženjerstvo je disciplina čiji je cilj izrada softvera bez greške koji zadovoljava korisnikove potrebe i koji je dostavljen na vrijeme i unutar budžeta.“[3]

Fokus je usmjeren na isplativi razvoj softverskih sustava s naglaskom na kvalitetu, a njezinom primjenom softverski su proizvodi postali pouzdaniji, te lakši za održavanje. Također, uvođenjem ove discipline pojasnili su se koraci koji su potrebni za razvoj softverskog proizvoda, kao što su metode razvoja softvera (strukturna analiza, JSD[4]...), dizajn i implementacija.

Važno je naglasiti i kako programsko inženjerstvo nije usmjereno samo ne tehničke detalje, već objedinjuje znanstvene grane (Projektni management) i aktivnosti koje su potrebne za razvoj visokokvalitetnog softverskog proizvoda.

Izazovi s kojima se spomenuta disciplina susreće u 21. stoljeću su: heterogenost, dostava i povjerenje. Kako bi se odgovorilo na ove izazove, potrebno je uvođenje novih tehinika i alata, te inovativnih rješenja.

Proces razvoja softvera podrazumijeva ispunjenje skupa zadataka kako bi isti bio funkcionalan. Pri tome je potrebno definirati aktivnosti, ali i resurse koji će se koristiti kako bi se te aktivnosti izvršile. Resursi nisu neograničeni pa je stoga nužno definirati budžet koji će biti na raspolaganju za razvoj softvera, vremenski rok u kojem softverski proces mora biti završen, redoslijed aktivnosti kojima će se doći do krajnjeg rezultata, alate koji će biti na raspolaganju za razvoj željenog softvera i sl. Pametna raspodjela resursa i racionalno postavljanje ograničenja rezultira uspješnim softverskim proizvodom koji može biti razvijen za specifičnog korisnika ili tržište općenito.

[3] B. B. Agarwal, S. P. Tayal, Mahesh Gupta, Software Engineering and Testing, Jones and Bartlett Publishers, Massachusetts 2010., str 9

[4] Jackson system development

Na samim počecima informatičkog doba i s pojavom prvih računala pojavili su se i prvi vrlo jednostavni softveri. Takve je softvere bilo vrlo lako razviti jednostavnim pisanjem koda, bez prethodnog planiranja.

„Zbog povećanja složenosti softvera, dodavanje novih funkcionalnosti je postajalo sve teže, kao i pronalaženje i otklanjanje grešaka.“[5] Upravo je ta složenost istakla značaj timskog rada i zajedničkog donošenje odluka unutar pojedinog tima. Od velikog značaja za uspješnost softvera je bilo usklađivanje stavova i shvaćanja softverskog procesa među suradnicima zbog čega su se i počele razvijati softverske metodologije. Te su metodologije imale, a i danas imaju za cilj postizanje veće efikasnosti softverskog proizvoda i bolje predviđanje mogućih nedostataka i prednosti.

Bez obzira na korištenu metodologiju, razvoj softvera se odvija kroz takozvani životni ciklus koji podrazumijeva sljedeće faze:

1. Analiza i definiranje zahtjeva – početna faza koja obuhvaća utvrđivanje zahtjeva budućih korisnika te se definira lista korisničkih zahtjeva
2. Projektiranje i dizajn – definira se plan rješenja, glavne komponente sustava, arhitektura koja će biti korištena, te se prema tome izrađuje softver
3. Implementacija – označava realizaciju plana softverskog proizvoda
4. Testiranje – u ovoj se fazi definiraju uočene prednosti i nedostaci nakon što je softver implementiran, te predstavlja polazište za moguća poboljšanja i ispravljanje grešaka
5. Isporuka – nakon što je softver testiran i uklonjeni su svi nedostaci isporučuje se korisniku (naručitelju) te se vrši obuka korisnika.
6. Održavanje – najduža faza, obuhvaća ispravljanje svih nedostataka nakon što je proizvod isporučen, te služi kao temelj za razvoj novih poboljšanih verzija istog softvera ili razvoj potpuno novog.

„Teorijski promatrano, pod procesom razvoja softvera podrazumijeva se svaki opis razvoja softvera koji sadrži neke od nabrojanih faza, organiziranih tako da proizvode

[5] V. Tomašević, Razvoj aplikativnog softvera, Univerzitet Singidunum, Beograd 2012., str. 15.

odgovarajući ispravan i provjeren kod.“[6] Odnosno, svako definiranje korisničkih zahtjeva može rezultirati funkcionalnim i operativnim softverom ukoliko se njima pravilno upravlja.

„Modeliranje procesa razvoja softvera osigurava potpunu kontrolu i koordinaciju svih aktivnosti koje treba provesti da bi se proizveo željeni softver i ispunili ciljevi projekta.“[7] Ono omogućava lakšu komunikaciju među članovima tima te daje jasnu i idealiziranu sliku o cilju koji se želi postići. Osim toga razlozi za modeliranjem razvoja softvera su zajedničko definiranje ciljeva čime članovi tima mogu lako procijeniti usklađenost predviđenih aktivnosti i postavljenih zahtjeva. Omogućuje se brzo uočavanje mogućih nedostataka, te postizanje veće efikasnosti i kvalitete krajnjeg proizvoda.

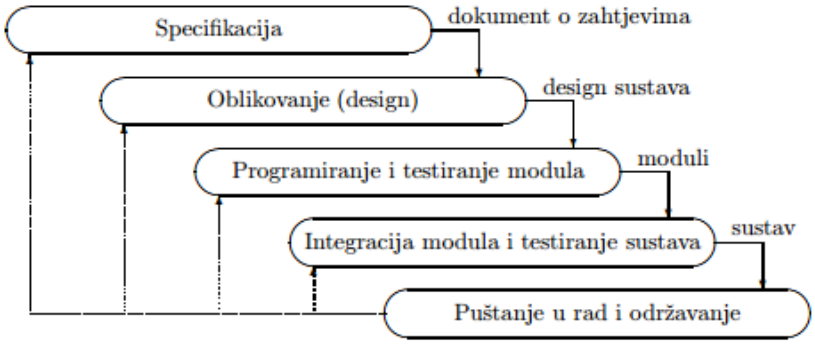
Modeliranjem se prezentira ideja u stvarnosti ovisno o konkretnoj situaciji, te nastaje podloga za modeliranje novog softverskog proizvoda uz određene preinake. Bez pravilnog modeliranja softverskog procesa krajnji proizvod može biti nevaljan, odnosno može doći do propadanja projekta.

S obzirom da je većina softverskih projekata vrlo zahtjevna potrebno je izabrati pravi model razvoja softverskog proizvoda, a neki od osnovnih modela su predstavljeni u nastavku.

Model vodopada je jedan od najstarijih modela životnog ciklusa softvera, pojavljuje se 1970-ih kada ga je prvi put spominjao Royce. Ovo je jedan od najzastupljenijih modela koji se danas koriste u praksi, a sastoji se od pet faza. Vrlo često služi i kao polazište za druge modele.

Sam naziv implicira povezanost razvojnih faza, odnosno na sljedeću se fazu prelazi tek kad je prethodna faza završila. Jednostavnije rečeno, izlaz iz jedne faze predstavlja ulaz u drugu fazu, odnosno faze se prelijevaju kao što je prikazano na slici 1.

[6] V. Tomašević, Razvoj aplikativnog softvera, Univerzitet Singidunum, Beograd 2012., str. 16.
[7] Ibid., str. 17.



Slika 1: Model vodopada

Izvor: R. Manger, Softversko inženjerstvo, skripta, nadopunjeno drugo izdanje, Zagreb, 2013., str. 13

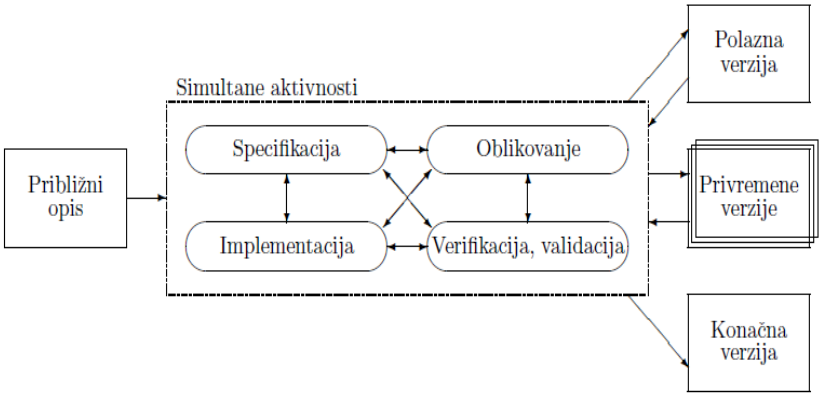
U svakoj se fazi definiraju kritične točke na temelju kojih se lako prati izvođenje projekta. Osim njih definiraju se i međuproizvodi, odnosno manje cjeline, koji pokazuju trenutnu fazu projekta.

Model vodopada omogućuje detaljno i jednostavno planiranje projekta i detaljnu podjelu poslova, te se upravo zbog toga može lako odrediti faza u kojoj se razvoj trenutno nalazi. Ono što ga čini nepoželjnim za korištenje je sporost razvoja softvera koja može dovesti do većih troškova zbog zastarjelosti i neažurnosti.

„Model evolucijskog razvoja (evolutionary development) nastao je kao protuteža modelu vodopada.“[8] Najčešće se koristi za razvoj polazne verzije koja će se prikazati korisniku, te za manje sustave s kraćim životnim vijekom. Povratne informacije koje dođu od krajnjeg korisnika često su polazište za uvođenje poboljšanja. Nakon određenog broja iteracije, te kada se ustanovi da su postignuta sva poboljšanja dobiva se završna verzija softvera.

Postupak razvoja ovakvog modela prikazan je na slici 2.

[8] R. Manger, Softversko inženjerstvo, skripta, nadopunjeno drugo izdanje, Zagreb, 2013, str. 14.



Slika 2: Model evolucijskog razvoja

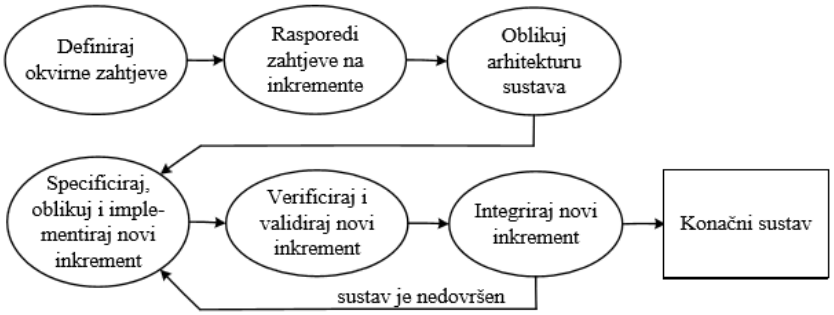
Izvor: R. Manger, Softversko inženjerstvo, skripta, nadopunjeno drugo izdanje, Zagreb, 2013., str. 14

Ovakav način razvoja se često naziva i istraživačko programiranje, s obzirom da se od korisnika nastoji izvući što više informacija. Model daje brze odgovore na postavljene zahtjeve, te omogućava razvoj i kod nedovoljno definiranih zahtjeva.

Ono što ga čini manje poželjnim je nedovoljna transparentnost u smislu procjene završetka projekta. Također, nije pogodan za održavanje, a razlog tome su česte promjene koje na kraju utječu na strukturu softvera.

Model inkrementalnog razvoja se često naziva hibridom modela vodopada i evolucijskog modela koji su ranije opisani i to upravo zbog razvoja u više iteracija. Ono što ga ipak razlikuje je to što se svakom iteracijom dodaje jedan novi dio, inkrement, a ne dotjeruje se već postojeći dio sustava. „Razvoj jednog inkrementa unutar jedne iteracije odvija se po bilo kojem modelu – na primjer kao vodopad.“[9] Model je prikazan na slici 3.

[9] R. Manger, Softversko inženjerstvo, skripta, nadopunjeno drugo izdanje, Zagreb, 2013, str. 15.



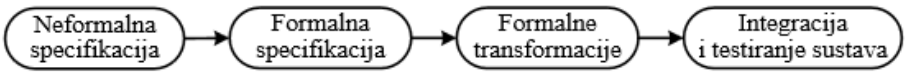
Slika 3: Model inkrementalnog razvoja

Izvor: R. Manger, Softversko inženjerstvo, skripta, nadopunjeno drugo izdanje, Zagreb, 2013., str. 15

Transparentnost ovog modela čini ga poželjnim za upotrebu jer se u svakom trenu može vidjeti do kojeg je inkrementa razvoj stigao. Korisnici su također zadovoljni ovim modelom razvoja sustava jer svakim inkrementom se zadovoljavaju njihove najnužnije potrebe, te lakše mogu dočekati završetak razvoja. S druge strane, korisnici pak mogu biti nedovoljno jasni u svojim zahtjevima što otežava razvoj smislenih inkrementa.

U praksi se ovaj model vrlo često koristi, a dokaz tome je i Microsoftov Office paket koji je razvijan upravo u inkrementima.

Model formalnog razvoja kako i sam naziv govori znači da se za razvoj koriste formalne metode. Zahtjevi se precizno definiraju na formalan način korištenjem nedvosmislene matematičke notacije nakon čega se transformiraju do završnog programa. Sam proces prikazan je koracima na slici 4.

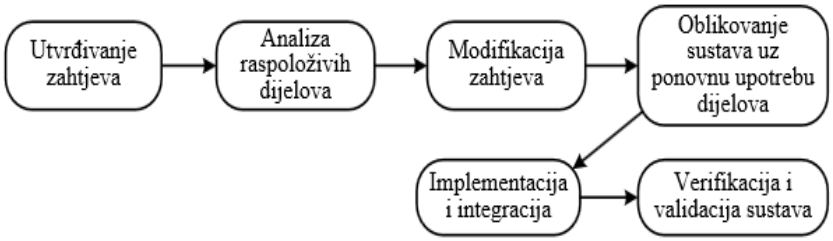


Slika 4: Model formalnog razvoja

Izvor: R. Manger, Softversko inženjerstvo, skripta, nadopunjeno drugo izdanje, Zagreb, 2013., str. 16

Formalna specifikacija je dobro polazište za daljnju automatizaciju ostalih faza razvoja, a njegova točnost smanjuje potrebu za testiranjem. Ipak, izrada formalne specifikacije zahtijeva puno znanja i truda. U praksi se najčešće može naići na ovaj model kod sustava kod kojih se zahtijeva velika pouzdanost i točnost.

Polazište Modela ponovne primjene postojećih resursa je ponovna upotreba već postojećih upotrebljivih softverskih cjelina. Cilj je razvoj novog sustava uz spajanje postojećih dijelova. Često se naziva modelom budućnosti jer korištenje gotovih rješenja je sve češće, a korisnici imaju sve manje vremena. Tijek ovog modela prikazan je slikom 5.



Slika 5: Model ponovne primjene postojećih resursa

Izvor: R. Manger, Softversko inženjerstvo, skripta, nadopunjeno drugo izdanje, Zagreb, 2013., str. 17

Velika prednost ovakvog modela je smanjenje količine softvera koji se mora razviti čime se smanjuje i upotreba ostalih resursa kao što su vrijeme i novac. Međutim, upravo zbog korištenja postojećih dijelova moguće je da sustav kao cjelina na kraju neće u potpunosti odgovoriti na potrebe korisnika. Djelomično je izgubljena kontrola nad razvojem sustava i to upravo zbog već razvijenih dijelova.

3 METODE RAZVOJA SOFTVERA

Metoda razvoja softvera je okosnica rada određenog softverskog tima koji u skladu s izabranom metodom razvija softverski proizvod. One određuju aktivnosti koje će se poduzimati u razvoju softvera, a koje će biti kombinirane u okviru jednog ili više softverskih modela koji su spomenuti ranije.

Metoda određuje način odvijanja aktivnosti i način na koji će ista biti dokumentirana. Metode će biti opisane u nastavku rada, a dijelimo ih u dvije osnovne skupine:

1. Klasične metode
2. Agilne metode.

3.1 Klasične/objektno orijentirane metode

Klasične metode javile su se 1970-ih godina 20. stoljeća zbog potrebe za bržim, točnijim i ekonomičnijim razvojem softvera. Njihov razvoj je potakao ponovno korištenje već nekih postojećih elemenata, a omogućile su lakše ispravljanje i modificiranje softvera.

Svoj su razvoj nastavile do danas, a karakteriziraju ih sljedeća svojstva:

- „Razvoj softvera promatra se kao pažljivo planirani proces koji ima svoj početak i kraj.
- Provodi se upravljanje projektom.
- Inzistira se na urednom dokumentiranju svih aktivnosti i svih proizvedenih dijelova.“[10]

Temelje se na objektno-orijentiranom pristupu koji se može prepoznati kroz sljedeće karakteristike:

- Identitet – podaci su organizirani u diskretne cjeline tzv. objekte.
- Apstrakcija – različiti aspekti sistema formiraju hijerarhiju koja pokazuje odnose među pogledima na sistem

[10] R. Manger, Softversko inženjerstvo, skripta, nadopunjeno drugo izdanje, Zagreb, 2013, str. 18.

- Klasifikacija – objekti sa zajedničkim svojstvima i ponašanjem se grupiraju u klase
- Enkapsulacija – klase enkapsuliraju ili skrivaju atribute i ponašanje objekata na način da se ne otkriju podaci implementacije
- Nasljeđivanje – klase su organizirane u hijerarhiju na osnovu sličnosti i razlika
- Polimorfizam – isto ponašanje kod različitih klasa se različito odražava na sistem
- Perzistencija – znači opstanak imena, stanja i ponašanja objekta u vremenu i nakon promjene objekta.

3.2 Agilne metode

U 80-im godinama 20. stoljeća smatralo se kako se kvalitetan softver može postići samo detaljnim planiranjem te korištenjem metoda koje će biti potpomognute CASE alatima.

Često su takvi softveri bili razvijani u velikim timovima koji su najčešće bili geografski raštrkani. Osim međusobne udaljenosti javljali su se i problemi kod koordinacije tima, reagiranja na kritične situacije, te općenito kod uključenosti velikog broja ljudi u životni ciklus pojedinog softvera. Upravo su ti problemi 1990-ih godina okupili nezadovoljne softverske inženjere koji su predstavili nove agilne metode. Taj je novi pristup omogućio usmjerenost na sam softver, a ne na njegov dizajn i dokumentaciju. Osnovna zamisao je bila i brže dostavljanje radne verzije softvera kupcu kako bi ga mogao prilagoditi svojim zahtjevima, a koji bi bili uzeti u obzir u kasnijim fazama razvoja (iteracijama).

Glavna i osnovna zadaća agilnih metoda je ubrzati razvoj softvera, smanjiti administraciju i birokraciju, te potaknuti brže odgovaranje softvera na promjene iz okruženja. „Agilno u pogledu razvoja softvera znači brz i fleksibilan odgovor na promjene.“[11]

Karakteristike agilnih metoda su:

- Razvoj softvera se promatra kao kontinuirani niz iteracija čiji se broj ne može predvidjeti.
- Aktivnosti se dogovaraju s korisnicima u kojima oni aktivno sudjeluju.

[11] D. Zima, Modern methods of software development (Online), Listopad 2015., Dostupno na: http://task.gda.pl/files/quart/TQ2015/04/tq419v-c.pdf, Pristupljeno 09.08.2016.

- Izbjegavaju se gotovo svi oblici dokumentacije.

Ono što još čvršće povezuje sve agilne metode je 12 principa, koje je sastavio tim od 17 autora kroz „*Manifesto for Agile Software Development*“, a to su sljedeći:

1. „Primarni cilj – zadovoljiti kupca brzom isporukom softvera
2. Promjene zahtjeva su dobrodošle, čak i u kasnijem stadiju razvoja.
3. Česta isporuka softvera koji radi, od nekoliko tjedana do nekoliko mjeseci.
4. Poslovni korisnici i developeri trebaju raditi zajedno na dnevnoj bazi.
5. Projekt izgraditi oko motiviranih pojedinaca.
6. Najučinkovitija metoda dobivanja razmjene informacija je komunikacija licem u lice.
7. Softver koji radi je primarna mjera napretka.
8. Promoviranje održivog razvoja kroz održavanje stalne suradnje sponzora, developera i korisnika.
9. Stalna fokusiranost na tehničku izvrsnost i dobar dizajn.
10. Jednostavnost je najbitnija.
11. Najbolja arhitektura, zahtjevi i dizajn proizlaze iz samoorganizirajućih timova.
12. Timovi u redovitim intervalima raspravljaju kako biti učinkovitiji, zatim se tome prilagođavaju.“[12]

Feedback, odnosno povratna informacija od korisnika, je još jedna od specifičnosti aglnih metoda koja omogućava uspješan razvoj softvera, njegovo poboljšanje i uspješnu isporuku.

Mnoga istraživanja pokazuju zastupljenost agilnih metoda u poslovanju, te njihov rast kroz godine. Tako npr. prema istraživanjima tvrtke *VersionOne* korištenje agilnih metoda je poraslo sa 84 % u 2012. godini na 95 % u 2015. godini. [13]

Kao najpoznatije agilne metode spominju se *Extreme Programming (XP), Scrum, Crystal, Dynamic Systems Development Method (DSDM)* i dr.

[12] M. Zekić-Sušac, 1. Pristup izradi mobilnih aplikacija – Agilne metode razvoja aplikacija, Dostupno na: http://www.efos.unios.hr/razvoj-poslovnih-aplikacija/wp-content/uploads/sites/228/2013/04/RPA_P1_Agilne-metode1.pdf, Pristupljeno: 09.08.2016.
[13] The 10th annual state of agile report, 2016, Dostupno na; http://www.agile247.pl/wp-content/uploads/2016/04/VersionOne-10th-Annual-State-of-Agile-Report.pdf, Pristupljeno: 10.8.2016.

Rational Unified Process (RUP) se također često spominje kao jedna od agilnih metoda.

4 BRZI RAZVOJ APLIKACIJA

Rapid application development[14], odnosno brzi razvoj aplikacija je pristup razvoju softvera kao procesu, te predstavlja više od samog opisivanja zahtjeva. Predstavlja odgovor na sve brži način života i pronalazak jednostavnih rješenja za svakodnevne obaveze i potrebe.

Ovaj pristup dodatno naglašava svojstva softvera kao što su fleksibilnost i mogućnost brze prilagodbe promjenama. „Metoda se primjenjuje za razvoj softvera koji omogućava izradu uporabljivog softvera.“[15] Omogućuje programerima i dizajnerima uporabu znanja koje su sakupili kroz proces razvoja, a u cilju poboljšanja ili potpune promjene postojećeg stanja softvera.

Metoda je nastala kao odgovor na već zastarjele modele za razvoj softvera, a razvio ju je James Martin 1991. godine.

Radi se o brzom dostavljanju softverskih rješenja koji udovoljavaju temeljnim zahtjevima, ali na kojima se još radi te nisu u potpunosti zadovoljeni. Primjenjuje se kod aplikacija koje nisu kritične te se izvode samostalno, a njihov razvoj može ubrzati korištenje drugih alata i tehnika.

Prilikom početka korištenja opisane metode najčešće se slijede četiri osnovna koraka:

1. Planiranje zahtjeva – razvijatelji softvera sakupljaju informacije o zahtjevima i opsegu projekta, koje će kasnije služiti za izradu prototipa softvera.
2. Korisnički dizajn – prikupljaju se povratne informacije od korisnika, ali s naglaskom na arhitekturu softvera. Te informacije su savršena baza za početak modeliranja i prototipiranja softvera. Ovaj se korak ponavlja dok se ne dođe do završne verzije i izgradi zadovoljavajuća arhitektura.

[14] U daljnjem tekstu RAD
[15] Procesi razvoja softvera, Dostupno na: http://web.efzg.hr/dok/inf/pozgaj/pisani%20materijali/T03%20Modeli%20razvoja%20softvera.pdf, Pristupljeno: 9.8.2016.

3. Brzi razvoj – u ovom je koraku naglasak na programiranju, izgradnji i integraciji. Kao i prethodni korak, ponavlja se dok se izmjenama ne dođe do krajnje verzije sa svim potrebnim komponentama .
4. *Cutover* (prijelaz) – posljednji je korak prijenos komponenti u verziju za uporabu u kojoj se može izvršiti testiranje i obuka.

RAD najčešće koristi već automatizirane alate, te kombinira sljedećih pet tehnika:

- Evolucijsko prototipiranje
- CASE alate
- Specijaliste s naprednim alatima (SWAT[16])
- Interaktivni JAD[17]
- *Timeboxing*.

Ovaj pristup može biti privlačan mnogim projektima i to zbog prednosti kao što su mjerljivost napretka, konstantne povratne informacije od korisnika, smanjeni rizik, povećana kvaliteta i brzo plasiranje na tržište. Sve navedene prednosti temeljene su na glavnoj karakteristici – brzini.

Brzi razvoj aplikacija često donosi i brojne propuste koji nisu poželjni u srži neke djelatnosti. Upravo zbog toga se kao mane ovakvog pristupa navode:

- Problem razvoja većih projekata
- Nekonzistentan GUI[18] dizajn
- Specijalizacija umjesto generičkih rješenja
- Manjkava dokumentacija
- Softver koji je težak za održavanje[19]

Pristup je najučinkovitiji kada postoji potreba za razvojem sistema koji može biti modeliran u kraćem vremenskom razdoblju, ne dužem od nekoliko mjeseci. Visina budžeta kao i

[16] Skilled Workers with Advanced Tools
[17] Joint Application Design
[18] Graphical user interface
[19] L. A. Maciaszek, Requirements analysis and system design, 3rd edition, Pearson Education Limited, 2007., str. 88.

dostupnost ostalih resursa, poput radne snage, u visokoj mjeri utječu na uspješnost softvera prilikom primjene ovakvog načina razvoja.

Kako bi RAD bio učinkovit moraju biti zadovoljena četiri osnovna aspekta, a to su:

1. Metodologija
2. Ljudi
3. Management
4. Alati

Ukoliko je samo jedan od ovih elementa neadekvatan ili se jedan od njih izostavi, razvoj neće biti u punoj brzini, odnosno usporit će se i produžiti životni ciklus razvoja softvera.

U brzom razvoju aplikacija često se koriste i neki od objektno orijentiranih programskih jezika poput Jave ili C++. Upravo to potvrđuje kako je kombinacija agilnih i objektno orijentiranih metoda najbolji recept za brzo dostavljanje softverskog rješenja, a čiji izbor ovisi o svrsi gotovog proizvoda.

4.1 Najzastupljenije metode razvoja mobilnih aplikacija

Razvija se sve više mobilnih aplikacija koje u svim namjenama pronalaze svoje mjesto i korisnike. Postavlja se samo pitanje: koje metode su najpogodnije za njihov brz i zadovoljavajući razvoj?

Najčešći odgovor na ovo pitanje su agilne metode i model vodopada, što je i potvrdila Mary Lotz u članku na web stranici *Seguetech*.[20]

Agilna metodologija se često spominje s naglaskom na SCRUM i to ne samo u razvoju mobilnih aplikacija nego u projektnom managementu uopće. „SCRUM je najčešće korištena poddomena agilne metodologije koja se brzo proširila na rukovanje velikim, kompliciranim projektima koji inače uzimaju puno vremena do dovršetka.“[21] Unutar

[20] Waterfall vs. Agile: Which is the Right Development Methodology for Your Project?, Dostupno na: https://www.seguetech.com/waterfall-vs-agile-methodology/, Pristupljeno: 14.9.2018.
[21] A Guide To Agile Scrum Methodology in Mobile App Development, Dostupno na: https://appinventiv.com/blog/agile-scrum-methodology-in-mobile-app-development, Pristupljeno: 14.9.2018.

SCRUM-a koraci koji se moraju odraditi dijele se na nekoliko ciklusa koji se još nazivaju i SPRINT. Svaki SPRINT ima zadano trajanje u kojem timovi moraju odraditi zadane komponente kako bi razvoj projekta bio uspješan. Ovakvim pristupom omogućava se bolja kvaliteta aplikacije, veće zadovoljstvo klijenata, transparentnost i brži povratak investicije.

Model vodopada kao jedna od najzastupljenijih metoda je već opisan u drugoj cjelini rada. „Unutar ove metodologije slijed koraka je:

1. Sakupljanje i dokumentiranje zahtjeva
2. Dizajn
3. Kodiranje i ispitivanje jedinica
4. Testiranje sustava
5. Testiranje prihvaćanja od strane korisnika
6. Ispravljanje grešaka
7. Dostava završnog proizvoda“[22]

Ono što je ključno za ovu metodologiju je to što svaki od navedenih koraka predstavlja zasebnu fazu razvoja softvera. Odnosno, jedna faza ne može postojati sama za sebe, niti započeti svoj tok prije nego prethodna završi.

Metoda koja se često spominje uz prethodne dvije je i RIPP[23] metoda Jamesa Martina, a koja se smatra jednom od najučinkovitijih i najpouzdanijih za brzi razvoj aplikacija. Metoda je zapravo samo inačica RAD-a. Karakteriziraju je brzina, iterativni razvoj, prototipiranje i vremenska ograničenost.

James Martin je smatrao kako softverski proizvodi mogu biti brzo i kvalitetno razvijeni kroz:

1. „Prikupljane zahtjeva putem radionica i fokus grupa
2. Prototipiranje i ponavljajuće testiranje dizajna
3. Ponovnu upotrebu softverskih komponenti
4. Stisnuti raspored koji odgađa poboljšanja za sljedeću verziju proizvoda

[22] Waterfall vs. Agile: Which is the Right Development Methodology for Your Project?, Dostupno na: https://www.seguetech.com/waterfall-vs-agile-methodology/, Pristupljeno: 14.9.2018.
[23] Rapid Iterative Production Prototyping – Brzo iterativno prototipranje proizvoda

5. Manju formalnost u recenzijama i timskoj komunikaciji.“[24]

4.2 Ekstremno programiranje

Ekstremno programiranje su 1999. godine razvili Kent Beck, Ward Cunningham i Ron Jeffries. Jedna je od najpopularnijih agilnih metoda, a ističe ju mogućnost primjene u velikim i malim tvrtkama neovisno o veličini projekta.

Poput drugih agilnih metoda omogućuje razvoj dugotrajnog softvera, te pruža mogućnost brze prilagodbe promjenama i zahtjevima. Osnova ove metode su jednostavnost, komunikacija, povratne informacije, hrabrost i poštovanje. Sam naziv je zapravo objedinio sve metode koje su se do nastanka Ekstremnog programiranja pokazale učinkovitima i uspješnima.

Glavna karakteristika je ipak timski rad, što znači da su manageri, razvijatelji softvera, programeri i korisnici jednaki i čine funkcionalan tim. Metoda dozvoljava timu da se zajednički dogovaraju oko rješavanja problema, te da se isti riješi na što uspješniji način te tako poveća produktivnost.

Bitno je naglasiti da je krajnji korisnik, u slučaju korištenja ove metode, aktivan sudionik koji u svakom trenu mora biti spreman pružiti povratnu informaciju ostatku tima.

Cilj je zadržati softver na što jednostavnijoj i jasnijoj razini kako bi svi u svakom trenu znali gdje se stalo s njegovim razvojem, te na čemu je potrebno dodatno poraditi.

Tijek razvoja softvera korištenjem metode ekstremnog programiranja je sljedeći:

1. Planiranje
2. Upravljanje
3. Dizajn
4. Kodiranje
5. Testiranje

[24] Rapid Application Development, Srpanj 2016, Dostupno na: http://searchsoftwarequality.techtarget.com/definition/rapid-application-development, Pristupljeno: 02.10.2016.

Planiranje se započinje zapisivanjem zahtjeva koje je korisnik postavio, te se dalje odvija u iteracijama. To znači u ovom slučaju da se rade manji planovi koji se ne izvršavaju istodobno nego jedan po jedan.

Upravljanje započinje svakodnevnim sastankom na kojem se formiraju manji timovi i vrši raspodjela radnika. Prije toga je potrebo osigurati radni prostor, a nakon čega se određuje željena održiva brzina razvoja softvera. Ono što je od ključne važnosti je da management uoči kada metoda nije funkcionalna, te da isto ispravi.

Kao što je i ranije spomenuto glavna karakteristika metode ekstremnoga programiranja je jednostavnost, pa se je toga potrebno držati i u dizajnu. Dizajn je potrebno dobro osmisliti, odrediti metode koje će se koristiti i rješenja za moguće rizike.

Kod kodiranja je najvažnije da je korisnik softvera u razvoju uvijek dostupan za eventualna pitanja. Kodiranje se vrši prema unaprijed dogovorenim standardima, te je poželjno što češće testiranje i integriranje.

Testiranje je posljednja faza. Odvija se testiranje svih segmenata, te proces nije završen dok svi testovi ne budu pozitivni. To podrazumijeva pronalazak eventualnih propusta koji se moraju ispraviti prije puštanja softvera u uporabu.

5 RATIONAL UNIFIED PROCESS

Rational Unified Process, poznatiji kao RUP, je pristup razvoju softvera koji su 1996. godine kreirali Jacobson, Rumbavgh i Booch. Kasnije je, 1998. godine, nadopunjen s poslovnim modeliranjem i disciplinom upravljanja promjenama i konfiguracijama, da bi 1999. godine bile mu dodane i discipline upravljanja projektima i tehnike za razvoj *real time* softvera. Do 2003. godine ovaj se pristup nalazio u sastavu IBM-a.

Radi se o pristupu koji je dobro definiran i strukturiran, usmjeren na arhitekturu, te vođen slučajevima uporabe. „Pruža discipliniran pristup dodijeljenim zadacima i odgovornostima unutar organizacije.“[25] Detaljno definira odgovornosti unutar tima, kako pojedine faze trebaju biti izvršene i u kojem trenutku. Također, koristi se kao radni okvir kojeg je moguće prilagoditi potrebama organizacije i projekta, a unutar kojeg su definirana pravila za razvoj na temelju preporuka i uputa koje pristup pruža.

Iako je na početku bio namijenjen većim projektima, primjena se proširila na srednje i male softverske projekte. Razlog tome je što RUP pruža lak uvid u bazu znanja pojedinog projekta koji je zasnovan na različitim uputama. „To doprinosi da svi članovi razvojnog tima, bez obzira na aktivnosti koje obavljaju, koriste zajedničku metodologiju, jezik i pogled na to kako treba razvijati softverski proizvod.“[26]

„RUP karakterizira iterativni razvoj, upravljanje zahtjevima, arhitektura zasnovana na komponentama, provjera kvalitete, vizualno modeliranje (UML) te kontrola promjena.“[27] Svi navedeni principi omogućuju razvoj kvalitetnih softverskih proizvoda i realizaciju kompletnih uputa.

Sastoji se od nekoliko osnovnih gradivnih elemenata ili blokova:

1. „Uloge (tko), koje definiraju skup povezanih vještina, sposobnosti i odgovornosti

[25] Rational Unified Process, Best practices for software development teams, Srpanj 2005, Dostupno na: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf, Pristupljeno: 16.8.2016.
[26] Metodologija razvoja informacionih Sistema, Dostupno na: http://www.ef.uns.ac.rs/Download/osi_master/2010-02-17_dodatni_materijal_za_ispit.pdf, Pristupljeno: 16.08.2016.
[27] K. Severović, Upravljanje odnosima s klijentima kao izvor informacija za oblikovanje I poboljšanje usluga, 2013, Dostupno na: http://services.foi.hr/thesis_phd/rad_severovic.pdf, Pristupljeno: 16.8.2016.

2. Proizvodi rada (što), koji predstavljaju rezultat nekog zadatka, uključujući modele, proizvode, dokumentaciju i sl.
3. Zadaci (kako), koji opisuju posao dodijeljen ulozi koji proizvodi neki koristan rezultat“[28]

Svaka se iteracija sastoji od zadataka koji su organizirani u devet disciplina, a koje dijelimo na inženjerske (šest) i one za podršku (tri), to su:

1. Poslovno modeliranje
2. Management zahtjeva
3. Analiza i dizajn
4. Implementacija
5. Razvoj
6. Testiranje
7. Projektni management
8. Management promjena
9. Okruženje

Ipak najvažnija je podjela RUP pristupa na faze životnog ciklusa, a koje će biti podrobnije objašnjene u praktičnom primjeru u drugom dijelu rada.

Faze životnog ciklusa su:

1. „Faza započinjanja (*Inception Phase*)
2. Faza razrade (*Elaboration Phase*)
3. Faza konstrukcije (*Construction Phase*)
4. Faza tranzicije (*Transition Phase*)“[29]

Kao prednosti ovog pristupa često se navode dobra dokumentiranost i visok nivo prilagodljivosti konkretnom projektu. Također, dostupnost raznih tutoriala za vježbanje čini ovaj pristup poželjnim i jednostavnim za korištenje, te je olakšano i upravljanje rizicima.

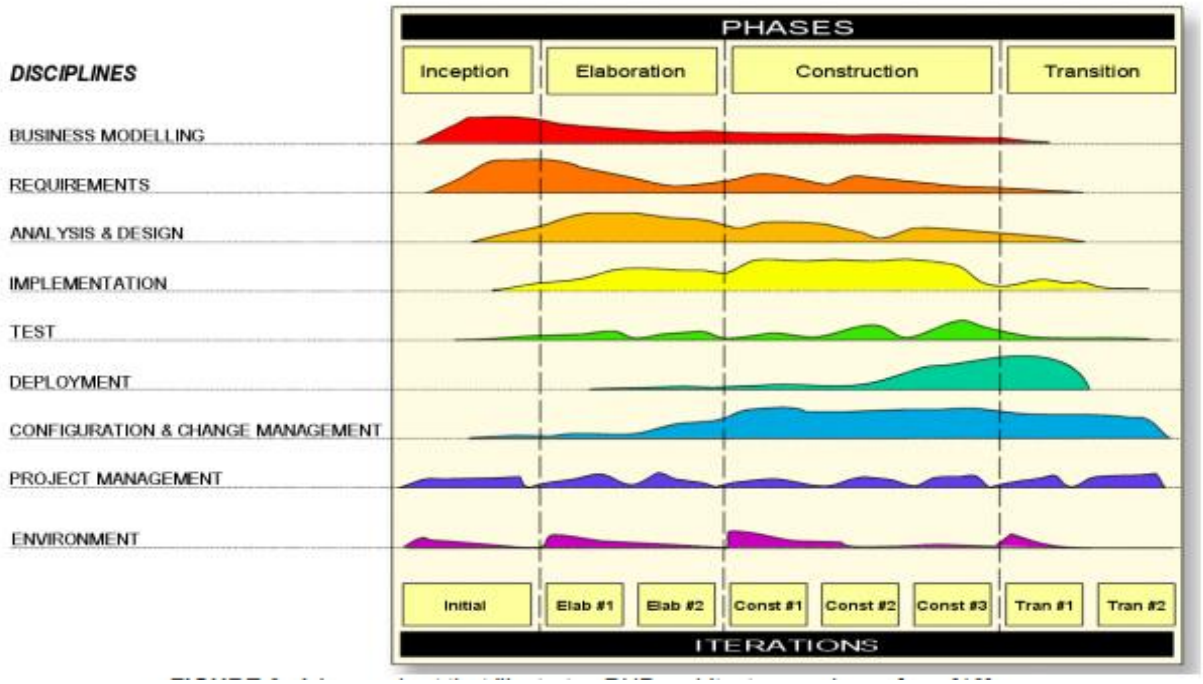
[28] V. Tomašević, Razvoj aplikativnog softvera, Univerzitet Singidunum, Beograd 2012., str. 30., 31.
[29] Ibid, str. 31.

S druge strane nedostaci RUP pristupa su kompleksnost procesa (puno koraka i procedura) što ga čini teže prilagodljivim manjim projektima.

Unatoč tome, RUP je jedna od boljih praksi u softverskom inženjerstvu, te se kao ključni koraci praktične perspektive ističe sljedećih šest koraka:

1. Iterativni razvoj softvera
2. Upravljanje zahtjevima
3. Korištenje komponenti baziranih na arhitekturi
4. Vizualno modeliranje softvera
5. Provjera kvalitete softvera
6. Kontroliranje promjena softvera

Često se spominje i kako se RUP pristup može promatrati kroz dinamičku, statičku i praktičnu perspektivu. Međutim, najčešće se spominje razvoj unutar dinamičke i statičke perspektive koje su detaljnije opisane u nastavku, te kroz faze prikazane na slici 6.



Slika 6: Statička i dinamička perspektiva

Izvor: Anwar, A., A Review of RUP (Rational Unified Process), str. 11

5.1 Dinamička dimenzija

Dinamička dimenzija[30] softver opisuje kao životni ciklus od četiri faze. Faze koje softver prolazi od ideje do gotovog proizvoda su:

1. faza uvođenja
2. faza elaboracije
3. faza konstrukcije
4. faza tranzicije

Svaka od tih faza odvija se u nekoliko iteracija, a do točnog broja iteracija dolazi se prilikom prilagodbe RUP-a potrebama softvera koji se želi proizvesti. Ono što je važno napomenuti je da se svakom fazom stvara inkrement u kojem se definiraju ključne točke i ciljevi kao smjernice za donošenje odluka o tome što će se proizvesti. Njihovo ostvarenje definira broj potrebnih iteracija. Ukoliko se ciljevi ne postignu u zacrtanoj iteraciji stvara se još jedna čime se produljuje vrijeme trajanja razvoja softvera. Upravo zato je važno dobro definirati što će se proizvesti u svakoj iteraciji te osigurati da se samo na tome i radi.

Kraj pojedine faze smatra se jednom od ključnih točaka u životnom ciklusu ili razvoju. Definirane ključne točke omogućuju predviđanje rezultata pojedine faze i razvojnog procesa općenito.

5.2 Statička dimenzija

Kako i sam naziv implicira statičku dimenziju definiraju dijelovi koji se ne mijenjaju odnosno oni koje je potrebno dobro postaviti i definirati na samom početku projekta. Dijelovi procesa kao što su aktivnosti, discipline, artefakti i uloge moraju biti logički povezani u smislene ključne procese, te je to glavno obilježje statičke dimenzije[31]. Detaljno opisuje tko što radi, na koji način i kada.

Uloga kao dio procesa predstavlja pojedinca ili više osoba koji obavljaju neki dio posla. To znači da više uloga može biti dodijeljeno jednoj osobi, te više osoba može imati istu

[30] Često se naziva horizontalnom ili vremenskom dimenzijom
[31] Naziva se i vertikalna dimenzija

ulogu. Međutim, odgovornosti su točno definirane među sudionicima projekta kao i način na koji se očekuje da rad bude izvršen.

Aktivnost određene uloge podrazumijeva posao koji se očekuje da bude izvršen, te ima jasno definiranu namjeru. Obično je potrebno nekoliko sati ili dana da bi se aktivnost izvršila. Najčešće uključuje jednu osobu, ali može utjecati na samo jedan artefakt u procesu ili na više njih. Najčešće se aktivnosti koriste kao okosnice za planiranje i napredak, te se mogu ponavljati. Ponavljanje se vrlo često dešava kada se prelazi iz jedne iteracije u drugu, odnosno kod širenja projekta to nužno ne znači da su isti pojedinci zaduženi za tu aktivnost.

Koraci su treći čimbenik statičke dimenzije, a više koraka čini jednu aktivnost. Oni mogu biti:

1. Koraci za razmišljanje – osoba koja ima određenu ulogu nastoji razumjeti prirodu zadatka, čimbenike koji utječu na njegovo izvršenje, te krajnji rezultat.
2. Koraci za izvedbu – uloge kreiraju i usavršavaju određene artefakte.
3. Koraci za analizu – uloge vrše analizu rezultat prema određenim kriterijima.

Cilj je što bolje izvršenje koraka kako bi se na kraju mogli pravilno vrednovati rezultati.

Dijelovi informacija koji su proizvedeni, prilagođeni i korišteni u procesu nazivaju se artefakti. Oni su elementi koji su proizvedeni tijekom projekta ili elementi koji su korišteni za razvoj krajnjeg proizvoda. Mogu se naći u puno različitih vrsta kao npr.: *use-case* model, klasa unutar modela, dokument, kod.

Gore navedeni dijelovi statičke dimenzije povezani u logičan slijed čine tijek posla. Tijek posla obuhvaća discipline i detalje unutar disciplina. Često se prikazuje UML dijagramima kao što su *sequence*, *collaboration* i *activity*.

Iako su ljudi također dio statičke dimenzije, oni se ne mogu smatrati strojevima pa tako i zacrtani tijek posla ovisi o njihovim postupcima, odnosno vrlo se često odvija drugačije od prvotno zacrtanog.

5.3 Vrste projekata

Kod RUP metodologije spominju se tri najčešće vrste projekata, koje opisuju razvoj softverskog proizvoda. Te tri vrste projekata oblikuju ideju razvoja bilo kakvog sustava kroz njegov životni ciklus. One također, daju teorijski okvir očekivanja u pojedinoj fazi, a opisane su niže.

Granymade je vrsta RUP projekta koja može biti *green field* ili *brown field.*

Green Field se odnosi na razvoj malih aplikacija koje se razvijaju od samog početka tj. osnovne ideje i sve što se koristi za njihov nastanak mora biti razvijeno.

Brown field se odnosi na razvoj novih verzija već postojećih aplikacija. Pogodan je za unaprjeđenje performansi postojećeg sustava.

Osim tri osnovne vrste projekata, postoje i dvije vrste koje ih upotpunjuju.

Mars kao vrsta projekta nadovezuje se na već spomenuti *green field,* međutim fokusiran je na razvoj velikih i kompleksnijih sustava.

Jupiter je evolucijski ciklus već postojećih aplikacija i to onih s kompliciranijim sustavom. Najčešće obuhvaća više RUP projekata te se najčešće koristi za unapređenje postojećih sustava.

Iako postoji još mnogo vrsta projekata i kombinacija više njih u jedan, ranije spomenute vrste su najčešće korištene za razumijevanje RUP metodologije i njezinog ciklusa.

5.4 Objektno orijentirana ili agilna metoda

Često se RUP metodologija spominjala u kontekstu objektno orijentiranih metoda, no u novije se vrijeme sve češće spominje kao sastavni dio agilnih metoda.

Kako je već ranije navedeno agilne metode karakterizira pokret, aktivnost, brzina, te spremnost na promjene. Sve to karakteristika je i RUP metodologije, odnosno adaptivnog procesa koji se može krojiti prema potrebama organizacije, dostupnosti resursa i karakteru tima.

RUP je prihvatio i implementirao mnoge koncepte na kojima je razvijen agilni pristup, te je usvojio mnoge nove agilne tehnike. On je zapravo okvir koji sadrži najbolje vještine informacijskih tehnologija, a to znači da sadrži karakteristike agilnih i tradicionalnih objektno orijentiranih metoda.

Za konstantni razvoj i rast zaslužno je prihvaćanje mnogih vještina iz različitih izvora. Dakle, možemo slobodno reći da RUP usvaja najbolje od svih metoda s kojima se susreće, te da ovisi o timu da li će biti primijenjen kao agilna ili objektno orijentirana metoda.

5.5 IBM Rational Rose

Na sam spomen RUP metodologije zasigurno je većini znalaca prva asocijacija *IBM Rational Rose*. Radi se o objektno orijentiranim UML softverskim alatima koji se nerijetko nazivaju i obitelji, a služe za izgradnju arhitekture softvera te za njegovo dizajniranje. U skupini ovih alata mogu se naći svi oni koju su potrebni za izgradnju softvera, te koji razvijatelju pomažu u analizi, razvoju, dizajnu, testiranju, te na kraju dostavljanju gotovog rješenja u stvarnom vremenu.

Može se reći da IBM-ovi alati kreiraju vizualni okvir aplikacije kroz klase, sudionike i druge elemente koji su potrebni za rad aplikacije, te veze među njima. Radi na principu povuci i ispusti, te se pomoću programskih jezika na temelju dokumentiranih UML dijagrama generira programski kod. Ovu skupinu alata karakterizira i razvoj aplikacije u iteracijama, što znači da izlazni „proizvod“ jedne iteracije predstavlja ulazni „proizvod“ za sljedeću iteraciju.

Ono što ih također ističe je to što dozvoljavaju tzv. *round-trip engineering*. To znači da se u bilo kojem trenutku razvijatelj aplikacije može vratiti na bilo koju fazu i ažurirati dio sustava koji nije konzistentan i na taj način poboljšati stabilnost cjelokupne aplikacije.

U tabeli 1 nalazi se popis IBM-ovih *Rational Rose* alata.

Tabela 1:IBM Rational Rose family

Ime proizvoda	Programski jezik	Poslovno modeliranje	Arhitektura softevra i dizajn	Programiranje i izgradnja	Modleiranje podataka
IBM Rational Rose Modeler	N/A	Podržano	Podržano		
IBM Rational Rose Data Modeler	N/A	Podržano	Podržano		Podržano
IBM Rational Rose Developer for Java	Java, CORBA	Podržano	Podržano	Podržano	
IBM Rational Rose Developer for Visual studio	ANSI C++, C++, CORBA, MSVB (microsoft visual basic,MSVC++(microsoft visual c++)	Podržano	Podržano	Podržano	Podržano uz Rose C++
IBM Rational Rose Enterprise	Ada, ANSI C++, C++; CORBA, Java, MSVB, MSVC++	Podržano	Podržano	Podržano	Podržano

IZVOR: http://www.predictiveanalyticstoday.com/ibm-rational-rose/, 21.03.2017.

Iz tabele 1 je vidljivo da *IBM Rational Rose Modeler* ne podržava programske jezike, ali se koristi za poslovno modeliranje i izradu arhitekture softvera te dizajna.

IBM Rational Rose Data Modeler također ne podržava programiranje, ali uz poslovno modeliranje i mogućnost izrade arhitekture i dizajna omogućava i modeliranje podataka.

IBM Rational Rose Developer for Java se razlikuje od prethodna dva alata po tome što omogućava i programiranje unutar samog alata i to u Java-i kako i sam naziv govori.

IBM Rational Rose for Visual Studio i *IBM Rational Rose Enterprise* razlikuju se samo u programskim jezicima koji se mogu koristiti unutar njih samih.

Iako bi prikaz RUP metodologije bio najjasniji uporabom upravo ovih alata, besplatne verzije su vrlo teško dostupne, a naišlo se i na nekoliko izvora za preuzimanje koji nisu bili valjani. Upravo iz tog razloga će se primjena RUP metodologije u ovom radu prikazati uporabom drugih besplatnih alata.

6 PRIMJENA RUP METODOLOGIJE ZA RAZVOJ APLIKACIJE

U ovom će poglavlju na primjeru razvoja manje aplikacije biti prikazane karakteristike RUP metodologije, te životni ciklus aplikacije od ideje do završnog proizvoda. Aplikacija će prikazati najjednostavniju primjenu navedene metodologije od strane samo jedne osobe, odnosno od strane autora ovog rada.

6.1 Analiza područja primjene

Najvažniji čimbenik koji je potrebno istražiti prije nego se krene s razvojem aplikacije je tržište, odnosno područje za koje će se nova aplikacija razviti. Osim tržišta važno je istražiti koje su potrebe korisnika u odabranom području, kakva je konkurencija, te koja je mogućnost plasmana zamišljene aplikacije i projekta na stvarnom tržištu.

Sektor koji je izabran za implementaciju aplikacije, koja je osmišljena u sklopu ovog diplomskog rada, je građevinarstvo. Na temelju podataka državnog zavoda za statistiku uočeno je kako je građevinarstvo brzorastuća grana Hrvatskog gospodarstva („Obujam građevinskih radova u siječnju ove godine bio je za 6,8 posto veći nego u istom lanjskom mjesecu...“[32]).

„Građevinarstvo je izuzetno važna djelatnost za svako gospodarstvo jer zapošljava znatan dio radne snage i uz sebe veže velik broj pratećih djelatnosti, od industrije građevinskog materijala, kemijske industrije, drvne industrije, metalne industrije, industrije stakla i nemetala do proizvodnje namještaja i dijela uslužih djelatnosti.“[33] Ono je jedan od prvih pokazatelja gospodarskog rasta, ali isto tako i njegove stagnacije. Unatrag nekoliko godina ovaj je sektor bio pogođen gospodarskom krizom i u Republici Hrvatskoj, što je usporilo njegov tehnološki razvoj.

„U 2016. je godini, prvi put nakon 2008. godine, zabilježen rast BDV-a građevinarstva.“ No, unatoč tome i danas (2018.) se primjećuje zaostatak u korištenju naprednih

[32] Najveći rast građevinskih radova od kraja 2016., Travanj 2018, Dostupno na: https://lider.media/aktualno/biznis-i-politika/hrvatska/najveci-rast-gradevinskih-radova-od-kraja-2016/ , Pristupljeno: 05.06.2018.
[33] Građevinarstvo u 2016. Godini, Ožujak 2017, Dostupno na: https://www.hgk.hr/documents/gradevinarstvo-rh-u-201658e619733cb43.pdf, Pristupljeno: 12.02.2018.

tehnologija u odnosnu na druge države iz regije. „Zato naši vodeći građevinari više ne mogu preuzimati financijski i tehnološki zahtjevne projekte poput Pelješkog mosta, već oni u pravilu odlaze stranim kompanijama.“[34] Upravo je zbog ovakvih činjenica i navoda građevinski sektor odabran kao polazište za prikaz primjene RUP metodologije.

Trenutno u građevinskom sektoru postoji mnoštvo aplikacija koje na razne načine mogu pomoći u procesu građevinskog projekta, ali i onih koje isti u potpunosti bilježe. Zastupljenost takvih aplikacije u Republici Hrvatskoj je još uvijek niska te se upravo zbog toga može naći prostora za nove aplikacije koje će biti problemski orijentirane, odnosno one koje će biti fokusirane na specifične zahtjeve. Također, od svih aplikacija koje se koriste najmanje je onih koje se mogu koristiti na mobilnim uređajima i pametnim telefonima, a upravo takve su poželjne zbog dostupnosti na terenu.

Neke od aplikacija koje se trenutno koriste su AutoCAD, MS Project i Gala Construction.

AutoCAD je najpoznatiji alat za projektiranje, a omogućuje dvodimenzionalno ili trodimenzionalno projektiranje. Osnovne karakteristike su sofisticiranost i visoka preciznost, te se koristi za stvaranje objekata, precizno crtanje, baratanje objektima, baratanje razinama, te kotiranje. Unutar njega moguće je koristiti neograničen broj radnih listova.

*MS Projec*t ili softver za vođenje projekata razvijen je od strane Microsofta, te se može koristiti na Windows operativnim sustavima. Cilj je olakšavanja posla voditeljima projekata. Njegovo korištenje omogućuje pravilno i jednostavno definiranje zadataka, vremena, resursa i troškova. Omogućuje izradu gantograma, kalendara, putokaza i sl., te se mogu prikazati veze među zadacima i izvještaji o pojedinim fazama projekta. Naravno, potrebno je naglasiti da su njegovi rezultati točni koliko i uneseni ulazni podaci.

„*Gala Construction* software je programsko rješenje za graditeljsku struku koje u potpunosti rješava pitanja kalkulacija, planiranja, kontrole utrošaka, obračuna izvedenih radova.“[35] Fokus je usmjeren na planiranje, kontrolu količina i financija, te u bilo kojem

[34] Građevinarstvo u 2016. Godini, Ožujak 2017, Dostupno na: https://www.hgk.hr/documents/gradevinarstvo-rh-u-201658e619733cb43.pdf, Pristupljeno: 12.02.2018.
[35] MS Windows (Desktop) verzija, Dostupno na: https://www.gala-construction-software.com/desktop.html, Pristupljeno 24.09.2018.

trenutku pruža uvid u sve informacije vezane za projekt. Omogućuje komunikaciju s drugim aplikacijama kao što su *AutoCAD* ili *ArchiCAD*[36]. Zasnovan je na Microsoft tehnologijama te korisniku omogućuje nesmetan rad i korištenje. Zanimljivost ovog softvera je i to što je u njegov razvoj uključen Građevinski fakultet u Zagrebu.

6.1.1 Potreba i korištenje mobilnih aplikacija u RH

Kako bi se potreba za mobilnim aplikacijama u RH i njihovo trenutno korištenje što točnije prikazalo anketirano je 10 ljudi iz građevinskog sektora. Za područje istraživanja izabran je Novigrad u Istri, kao jedan od najrazvijenijih gradova u Republici Hrvatskoj prema Finininoj analizi iz 2017. godine. U tom gradu trenutno djeluje 7 ovlaštenih, te 10 neovlaštenih građevinskih inženjera.

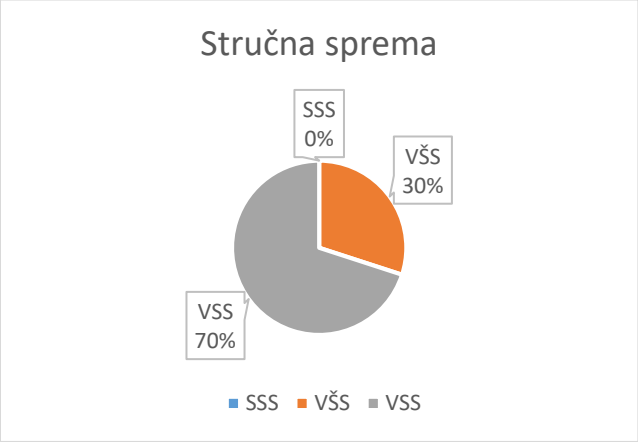
Primjer ankete priložen je na kraju rada (Prilog 2).

U nastavku su prikazani rezultati anketiranja, te dana dodatna pojašnjenja za pojedine segmente.

Graf 1 prikazuje stručnu spremu ispitanika, te je vidljivo da od 10 ispitanika 7 ih je označilo kako imaju visoku stručnu spremu, 3 ispitanika imaju višu stručnu spremu, dok niti jedan ispitanik nema srednju stručnu spremu.

[36] Softver koji omogućuje 3D dizajn, te služi za stvaranje virtualnih građevinskih objekata.

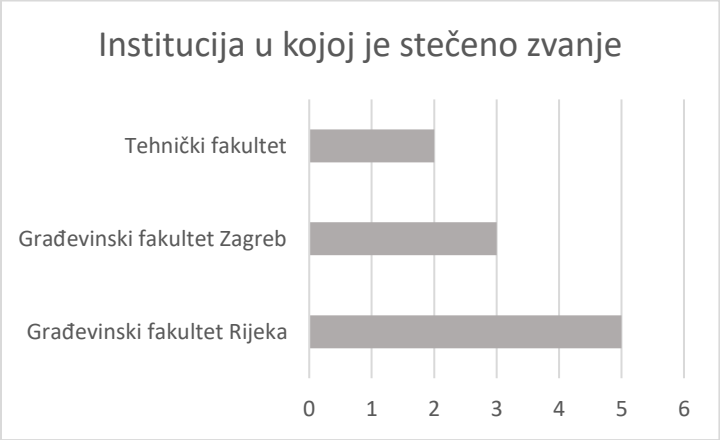
Graf 1:Stručna sprema



Izvor: Izrada autora

Čak 5 ispitanika stručnu spremu je steklo na Građevinskom fakultetu u Rijeci, njih 3 na Građevinskom fakultetu u Zagrebu, dok su 2 ispitanika završila Tehnički fakultet. Rezultati prema Instituciji obrazovanja prikazani su na Grafu 2.

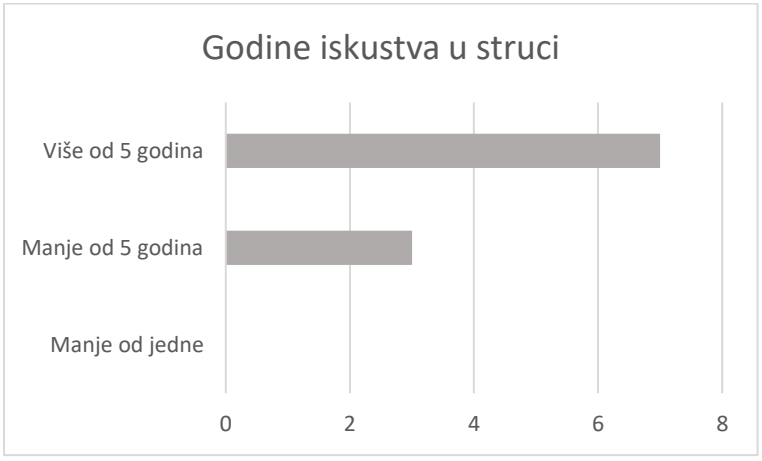
Graf 2: Institucija stečenog zvanja



Izvor: Izrada autora

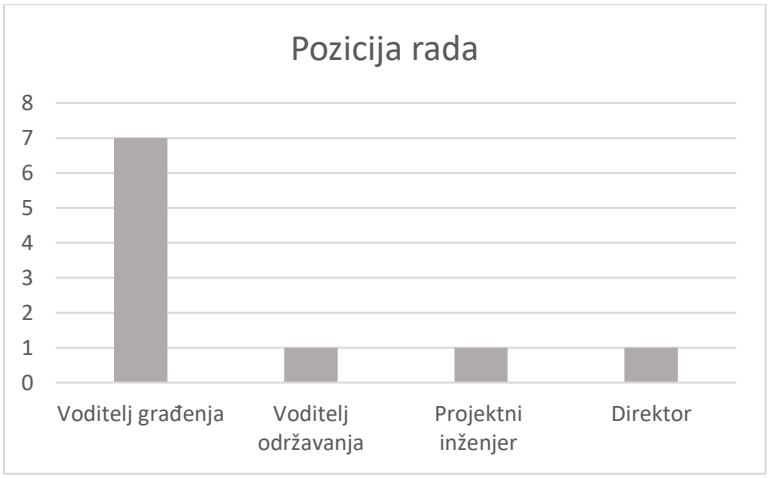
Iz Grafa 3 je vidljivo kako 7 ispitanika ima više od 5 godina iskustva u struci, te se može reći da je taj rezultat usko povezan sa Grafom 4, odnosno radnom pozicijom gdje je također 7 ispitanika kao radnu poziciju označilo “voditelj građenja”. Na trenutnoj poziciji 5 ispitanika radi više od 5 godina, a isto je prikazano u Grafu 5.

Graf 3: Iskustvo u struci



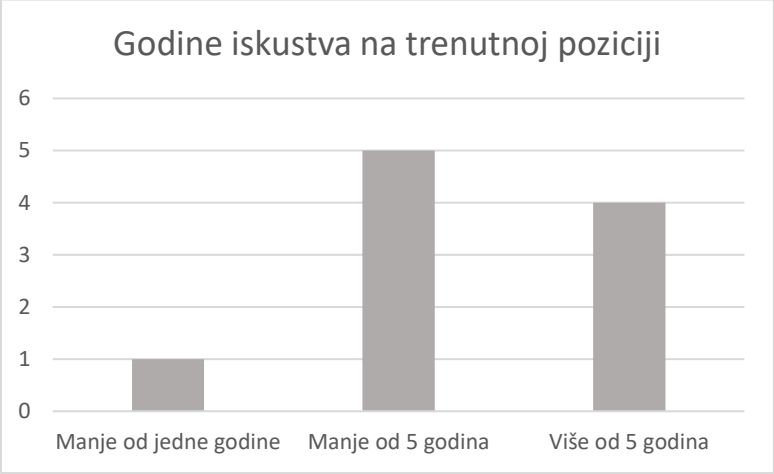
Izvor: Izrada autora

Graf 4: Radna pozicija



Izvor: Izrada autora

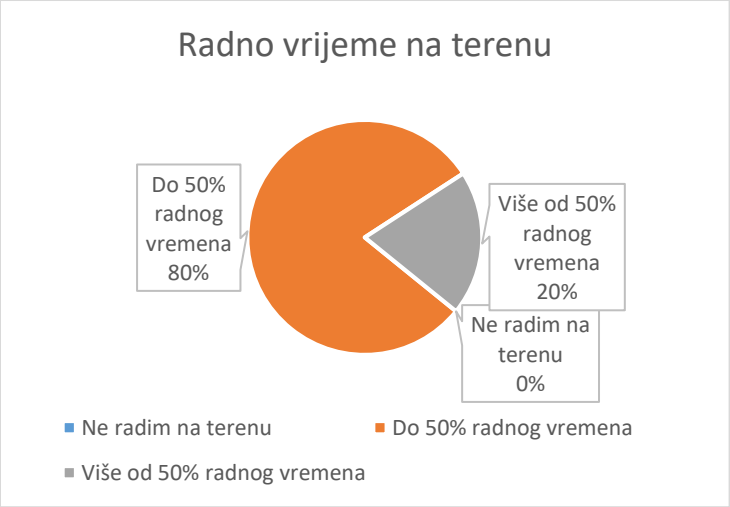
Graf 5: Iskustvo na trenutnoj poziciji



Izvor: Izrada autora

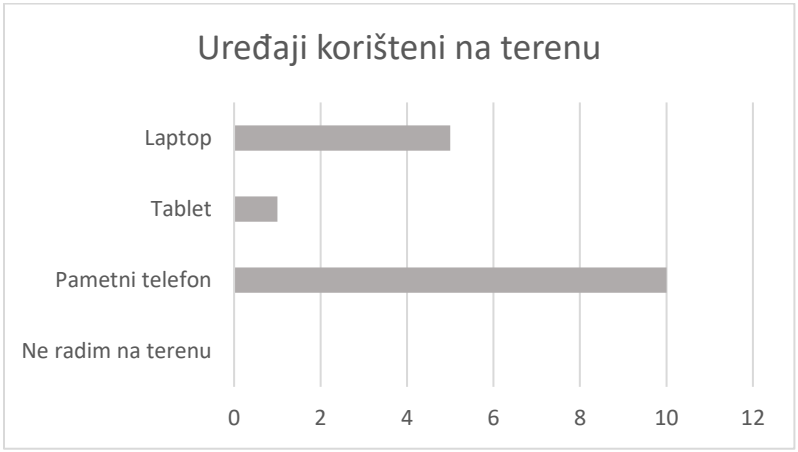
S obzirom da rad na terenu čini 50 % radnog vremena za 8 ispitanika od 10, ne čudi činjenica da je najzastupljeniji uređaj na terenu pametni telefon, nakon čega slijedi laptop, te tablet. Ovi se podaci mogu iščitati iz Grafa 6 i Grafa 7.

Graf 6: Radno vrijeme provedeno na terenu



Izvor: Izrada autora

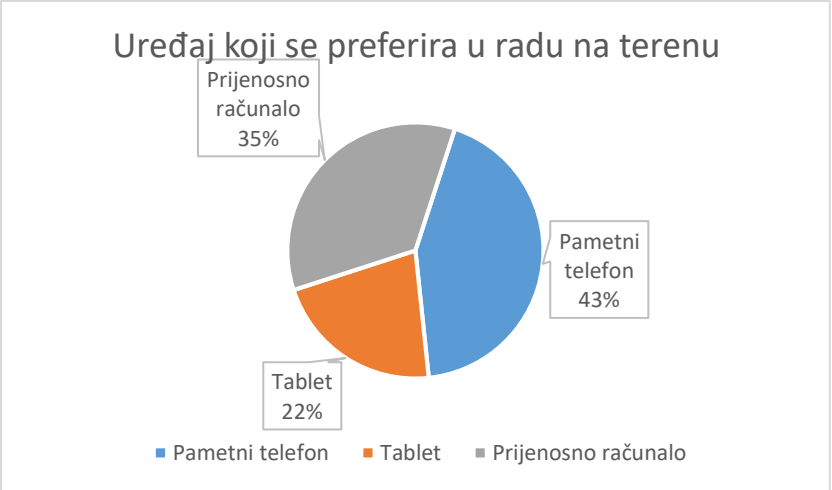
Graf 7: Uređaji koji se koriste na terenu



Izvor: Izrada autora

Sukladno prethodno navedenom te prema Grafu 8 na terenu se najviše preferiraju uređaji koju su lako prenosivi, pa tako 43 % njih je prvo izabralo pametni telefon, zatim 35 % prijenosno računalo, a 22 % tablet.

Graf 8: Preferencija uređaja

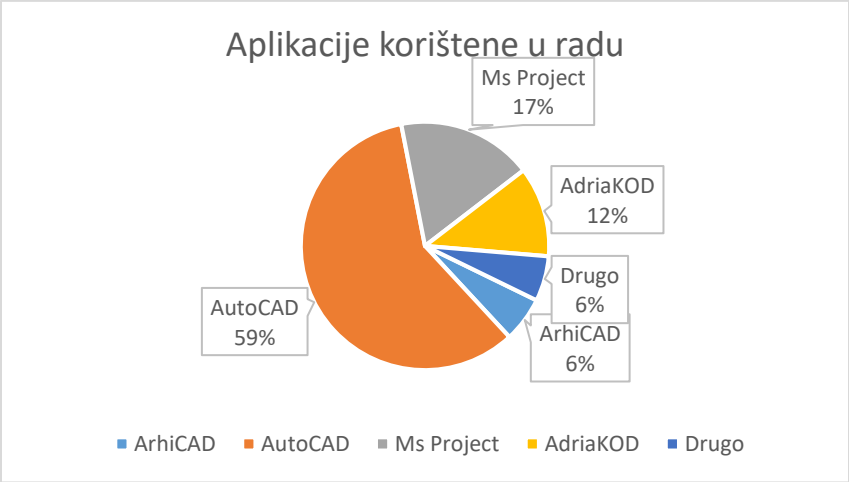


Izvor: Izrada autora

Iz ankete su dobiveni i rezultati da rad s više uređaja na terenu predstavlja problem za 50 % ispitanika (5 od 10), te da 100 % ispitanika (10 od 10) želi da mu je sve potrebno za rad dostupno samo na jednom uređaju.

Neke od aplikacija navedenih u Grafu 9 mogu se koristiti i na mobilnim uređajima, što je potvrdilo 6 ispitanika od 10, međutim samo 2 ispitanika su potvrdila da ih zaista i koriste na mobilnim uređajima svaki dan. Samo 5 ispitanika povremeno koristi navedene aplikacije na mobilnim uređajima, dok 3 ispitanika ih nikada ne koriste na mobilnim uređajima.

Graf 9: Korištene aplikacije

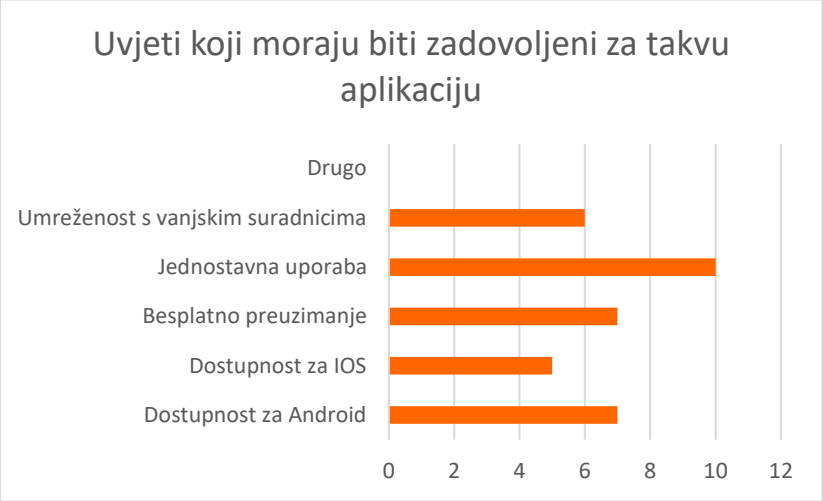


Izvor: Izrada autora

Svi ispitanici su potvrdili da komuniciraju s vanjskim suradnicima, odnosno 50 % njih to radi na tjednoj razini, a 50 % na dnevnoj razini. Da aplikacije koje trenutno koriste omogućuju komunikaciju s vanjskim suradnicima potvrdila su samo 2 ispitanika, dok je 8 njih to negiralo. Sukladno tome, njih 7 od 10 je potvrdilo da bi aplikacija koja omogućuje komunikaciju s vanjskim suradnicima ubrzala tijek posla, a njih 3 se ne slaže s tom tvrdnjom.

Aplikacija koja omogućuje komunikaciju s vanjskim suradnicima i objedinjuje potrebe posla na jednom uređaju mora zadovoljiti određene uvjete kako bi bila od koristi. Ti uvjeti su prikazani u Grafu 10, a ispitanici su najviše glasova dali za jednostavnu uporabu (10 od 10 ispitanika), dok nakon toga slijedi besplatno preuzimanje i dostupnost za Android operativne sustave.

Graf 10: Uvjeti koje mora zadovoljiti nova aplikacija



Izvor: Izrada autora

Svi ispitanici (10 od 10) su potvrdili da bi bilo od koristi da nova aplikacija omogućuje izračun količina materijala, te pohranu istih.

Također, 10 od 10 ispitanika bi koristilo takvu aplikaciju.

6.1.2 Zaključak analize

Iako su aplikacije spomenute u prethodnom dijelu trenutno najzastupljenije, nedostatak je to što se zbog svoje opsežnosti one najčešće koriste na prijenosnim računalima i stolnim računalima u uredima, a ne na terenu.

Aplikacije kao što su MS Project, AutoCAD i AdriaKod najčešće se na mobilinim uređajima koriste samo za iščitavanje informacija koje su već unesene za određeni projekt, a rjeđe za stvaranje novog projekta direktno na terenu.

Upravo iz tog razloga lako se može naći prostora za razvoj mobilne aplikacije koja bi bila orijentirana na određeni problem, a samim time i dovoljno jednostavna što bi privuklo veći broj korisnika.

S obzirom na sve navedeno jasno je da je za potrebe razvoja aplikacije unutar ovog rada odabran sektor građevine, u kojem će se prikazati primjena RUP metodologije na primjeru mobilne aplikacije pogodne za Android operativne sustave.

Aplikacija koja će se razviti uz korištenje spomenute metodologije imat će za cilj olakšati rad na terenu građevinskim inženjerima i drugim djelatnicima koji mogu imati koristi od njenog korištenja. Predstavljat će jednostavnu i brzu pomoć na terenu, a uz to može biti temelj za smanjenje popratne dokumentacije, smanjiti trajanje i trošak ukupnog projekta, ubrzati komunikaciju među djelatnicima i vanjskim suradnicima.

6.2 Opis aplikacije

Kako je već rečeno aplikacija je osmišljena kao vrlo jednostavan alat za olakšavanje rada u građevinskom sektoru na terenu.

Za njezino pokretanje je dovoljno posjedovanje pametnog telefona s Android operativnim sustavom te mobilne internetske veze. Aplikacija se jednostavnim preuzimanjem od razvijatelja softvera instalira na pametni telefon korisnika, te se može započeti s njezinim korištenjem za što nije potrebno nikakvo informatičko predznanje, niti dodatne licence.

Aplikacija će služiti za izračun kubature prostora, odnosno materijala potrebnog za rad u tom prostoru (beton), a na osnovu parametara koje unosi korisnik (inženjer građevine/voditelj gradilišta) na terenu.

Aplikacija ima vrlo jednostavno sučelje koje obuhvaća sve potrebne elemente za lako korištenje i navigaciju unutar nje same. Osim unosa i potvrde slanja parametara korisnik na terenu nije dužan poduzimati nikakve druge korake. Odnosno, zamisao je da se uneseni parametri prenose u administraciju tvrtke odakle se vrše daljnje narudžbe materijala.

Završna verzija aplikacije predstavlja polazište za razvoj većih sustava koji bi bili od koristi u ovom sektoru. Tako se primjerice aplikacija može u dogovoru s dobavljačima integrirati i u njihov sustav čime bi se izbjegao kontakt s administrativnim odjelom tvrtke koja naručuje beton. To bi značilo da korisnik unosom parametara šalje direktno podatke tvrtki od koje želi naručiti materijal, te bi od nje mogao dobiti procijenjeno vrijeme isporuke materijala.

Napominje se kako će biti razvijena samo osnovna verzija aplikacije, te će se prikazati ključni koraci za razumijevanje RUP metodologije bez implementacije u stvarnom vremenu.

6.3 Početna faza (inception)

Prva faza u razvoju aplikacije RUP metodologijom određuje što će se razviti, tko su sudionici u razvoju, koji su ključni zahtjevi koji se moraju ispuniti, te je potrebno definirati slučajeve 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.

Na kraju faze je poželjno da budu definirani alati i procesi razvoja aplikacije.

Kako je već napomenuto razvijat će se aplikacija u građevinskom sektoru za korištenje na terenu. Bit će razvijana od strane jedne osobe, autora ovog rada, koja predstavlja razvijatelja aplikacije, administrativnog pomoćnika i projekt menadžera. U tom specifičnom slučaju jedna će osoba biti prisutna u svim fazama razvoja te će sudjelovati u razvoju svih slučajeva korištenja. Razvoj aplikacije bit će predstavljen u obliku dnevnika.

S obzirom da se radi o vrlo jednostavnoj aplikaciji u početnoj će se fazi odviti samo jedna iteracija, odnosno definirat će se što se želi razviti i to u što je moguće kraćem roku.

Nepoznati termini, rizici i budući potencijal aplikacije bit će u detalje objašnjeni.

Prvi dan: Nakon istraživanja građevinskog sektora i provedenog anketiranja ispitanika iz struke utvrđeno je kako se zaposlenici bore s previše dokumentacije koju je potrebno obrađivati na terenu. Informacije do kojih dolaze na terenu potrebne su za kasnije ispunjavanje dokaznica o projektu te za obavljanje raznih izračuna, što znači da moraju biti točne i pravovremene. U tu se svrhu koriste razne tehnologije, no trenutno dostupne aplikacije previše su opsežne i zahtijevaju mnogo više od pametnog telefona. Dakle, može se reći kako ustaljena procedura usporava posao te zahtijeva adekvatan radni prostor na terenu koji nije uvijek dostupan i omogućen.

Kako bi se generirala prava ideja, ustanovili korisnički zahtjevi i definirala poslovna pravila obavljen je intervju s voditeljem gradilišta iz privatne tvrtke u Istri koji se u daljnjem tekstu navodi kao naručitelj. Intervju se nalazi na kraju rada u prilogu 1.

Intervjuom su definirani slijedeći korisnički zahtjevi:

1. Aplikacija će izračunavati potrebnu količinu materijala koji treba naručiti (beton)
2. Vršiti će se pohrana podataka o gradilištu (naručen materijal i količina)

Temeljem korisničkih zahtjeva definirani su i zahtjevi sustava, a to su:

1. Pohrana šifre gradilišta
2. Izračun potrebnog materijala temeljem unesenih dimenzija (kubature)
3. Pohrana količine potrebnog materijala na temelju unesenih dimenzija
4. Grupiranje količina po šifri gradilišta

Aplikacija koja će ispunjavati navedene zahtjeve zvati će se *ConstSoft*.

Vizija je korištenje na terenu u svakom trenutku, smanjenje nepotrebne dokumentacije i bilješki čime će posao biti tečniji, te ubrzanje komunikacije među zaposlenicima i s trećim stranama.

Aplikacija će osim brzine obrade gradilišta, doprinijeti i modernizaciji građevinske firme. Kao takva pružat će mogućnost pohrane podataka na jednom mjestu čime će se omogućiti jednostavniji i brži uvid u podatke pojedinog gradilišta, a što još jednom ukazuje na prednost ubrzanja procesa obrade gradilišta.

S obzirom na postavljenu viziju i ciljeve predviđeno je 20 dana rada na alatu do njegove prve verzije. U planiranom vremenu razvoj aplikacije će proći i kroz preostale tri životne faze koje su predviđene RUP metodologijom, a to su faza elaboracije, faza konstrukcije i faza tranzicije.

Plan rada prikazan je u Tabeli 2.

Tabela 2:Plan razvoja ConstSoft-a

1. dan	2. dan	3.-18. dan	19. dan	20. dan
Početna faza: - Vizija - Tjedni plan - Rizici - Troškovi - Poslovni slučaj korištenja	Faza elaboracije: - Prototip - Smanjenje rizika	Faza konstrukcije: - Dizajn - Kodiranje - Testiranje	Faza tranzicije: - Rad na poboljšanjima - Prikaz prve verzije	Dostavljanje naručitelju

Izvor: izrada autora

Zamišljena je vrlo jednostavna arhitektura pa se ne predviđaju nikakvi troškovi za razvijatelja aplikacije. Planira se korištenje besplatnih alata za razvoj aplikacije za Android operativne sustave.

Tijekom razgovora s naručiteljem odmah su primijećeni prvi rizici poput spajanje na bazu podataka naručitelja, ulaganje u novu opremu i *smartphone*, te plaćanje mobilnog interneta za sve zaposlenike koji će koristiti alat.

Jedan od nedostataka aplikacije i jedan od većih rizika, koji je uočen već u prvoj fazi je prostor za pohranu podataka. Međutim, moguće rješenje problema biti će obrađeno u budućim fazama s naglaskom na hipotetsko rješenje kao što je spremanje na *cloud*. Prilikom toga potrebno je obratiti pozornost na povjerljivost podataka.

Osim rizika koji alat predstavlja za naručitelja, primijećen je i problem za razvijatelja softvera, a to je pronalazak dobrih besplatnih alata za razvoj softvera.

Osim naručitelju, od kojeg je i krenula ideja razvoja, aplikacija bi bila pogodna za sve manje tvrtke u građevinskom sektoru. Ista bi se mogla implementirati i u većim tvrtkama. Osnovna ideja je razvoj jednostavne aplikacije za potrebe ovog rada te se neće ići u razradu većih funkcionalnosti i detalja.

Naručitelj je naveo kako želi osigurati svoje podatke od gubitka i zlouporabe te će se morati poraditi i na *backup* pohrani svih podataka.

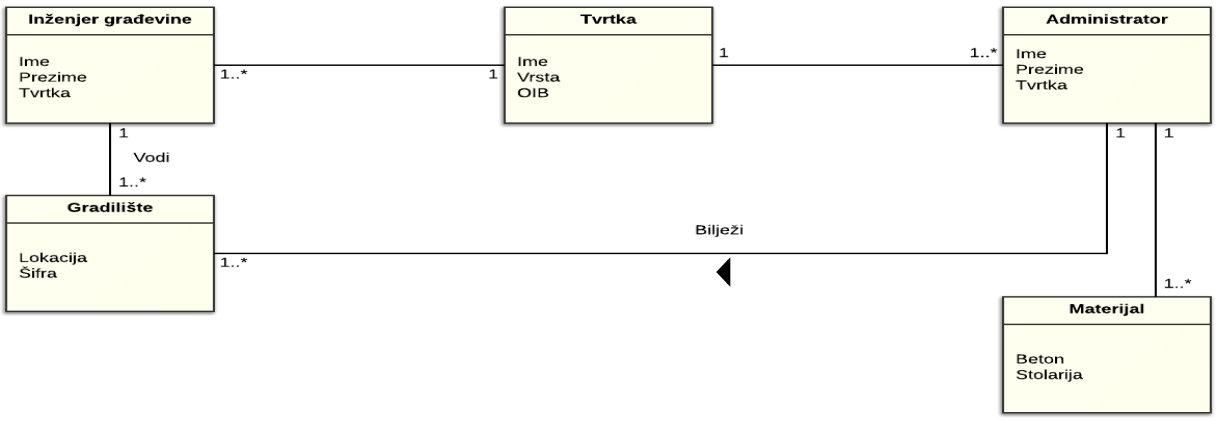
Nakon proučavanja ideje i potreba naručitelja definirani su slijedeći slučajevi korištenja, a koji će biti razrađeni u sljedećoj fazi:

1. Otvaranje gradilišta

2. Obrada gradilišta
3. Narudžba betona

Kao najkritičniji slučaj korištenja ističe se Narudžba betona za koji će biti potrebno povezati pametne telefone korisnika na terenu sa sustavom tvrtke kako bi informacije tekle bez poteškoća te kako bi se posao odvio u zacrtanom vremenskom roku.

Osnovne funkcionalnosti prikazane su na slici 7 u nastavku:



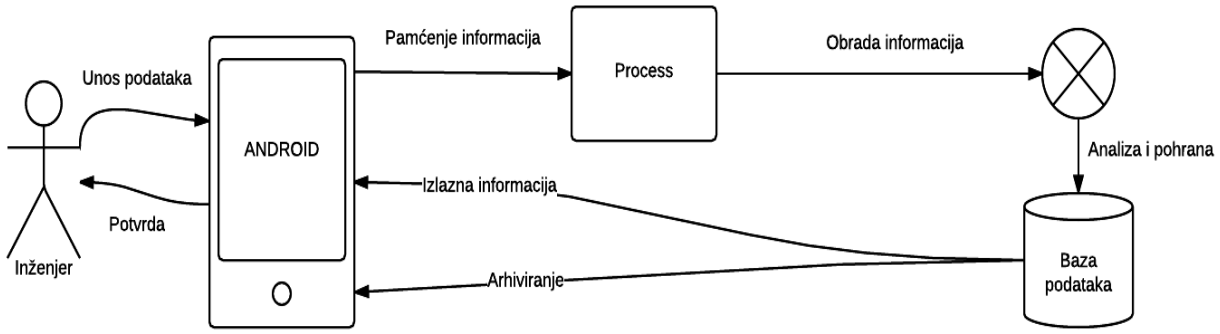
Slika 7:Osnovni dijagram ConstSoft

Izvor: izrada autora

Slika prikazuje klasni dijagram, a na kojem se mogu vidjeti klase sustava, s odgovarajućim kardinalnostima koje definiraju vrstu veze među klasama.

Vidljivo je da u tvrtki rade inženjer i administrator. Inženjer je zadužen za vođenje gradilišta za koje se bilježi lokacija i šifra pod kojom se ono prati unutar tvrtke. Administrator bilježi podatke o gradilištu na osnovu kojih dalje vrši narudžbu materijala.

Niže, na slici 8 je prikazana prva verzija arhitekture aplikacije.



Slika 8: Arhitektura aplikacije CostaSoft

Izvor: izrada autora

Vidljivi su osnovni elementi arhitekture te se mora izvršiti testiranje baze u koju će se pohranjivati podaci. Također, bit će potrebno testirati brzinu mobilnog interneta koji će naručitelj pribaviti za svoje zaposlenike, te općenito pratiti protok informacija unutar aplikacije.

6.4 Faza elaboracije (*elaboration*)

U fazi elaboracije detaljnije se proučavaju zahtjevi naručitelja kako bi se što bolje shvatile njegove želje. „Po završetku ove faze trebali bi imati model zahtjeva sustava, koji može biti skup UML slučajeva korištenja, opis arhitekture i razvojni plan za softver.“[37] Faza je temelj za fazu konstrukcije koja slijedi nakon nje, a upravo je zato važno uspostaviti stabilnu arhitekturu koja zadovoljava zahtjeve i umanjuje rizike.

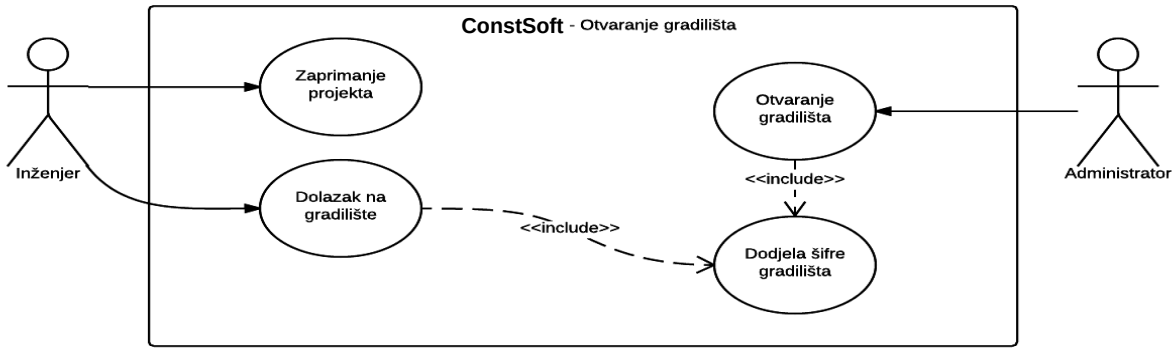
Na temelju dosadašnjih zahtjeva definirani su već spomenuti slučajevi korištenja (Otvaranje gradilišta, Obrada gradilišta, Narudžba betona) koji će se u nastavku detaljnije dokumentirati. Izradit će se prototip aplikacije kako bi se predočilo kako će ona u stvarnosti i izgledati, te će se ista razviti pomoću alata Xamarin Forms[38] unutar programa Visual Studio 2017.

[37] I. Sommerville, SOFTWARE ENGINEERING; Ninth edition, Pearson, 2011., str. 51
[38] Alat za višeplatformski razvoj mobilnih aplikacija

S obzirom da se radi o vrlo maloj aplikaciji predviđa se da će biti dovoljna samo jedna iteracija u ovoj fazi. Dizajn, implementacija i testiranje manjeg broja kritičnih scenarija bit će dovoljni da bi se identificirao tip arhitekture koji će se primijeniti.

Na kraju će se definirati precizniji plan rada ukoliko se primijete neke nelogičnosti ili nedostaci u dosadašnjem planu. Također, na kraju elaboracijske faze vrši se dovoljan broj testiranja kako bi se vidjelo jesu li ublaženi svi rizici vezani za arhitekturu.

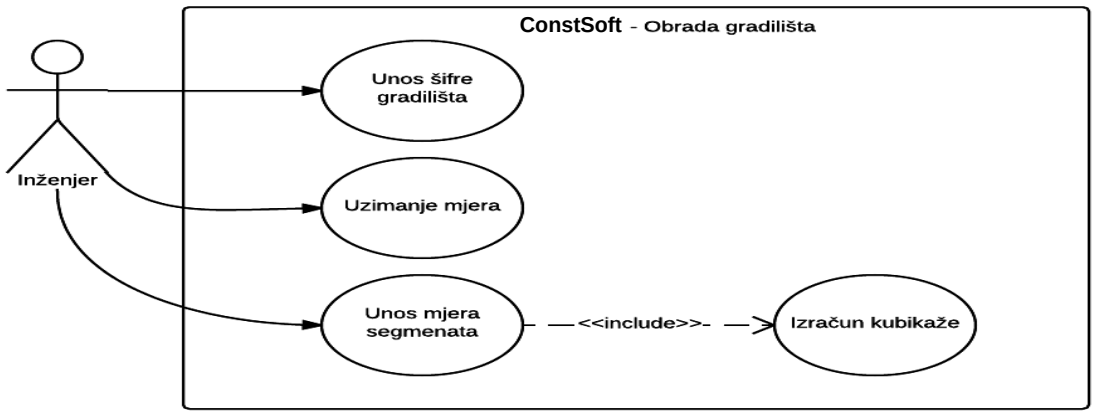
Drugi dan: Analizom osnovnog klasnog dijagrama iz kojeg su vidljive klase te veze među njima, definirani su slučajevi korištenja koji će se upotrebom aplikacije pojednostaviti. U nastavku su prikazana i opisana sva tri slučaja korištenja koja su već ranije spomenuta.



Slika 9: Otvaranje Gradilišta

Izvor: izrada autora

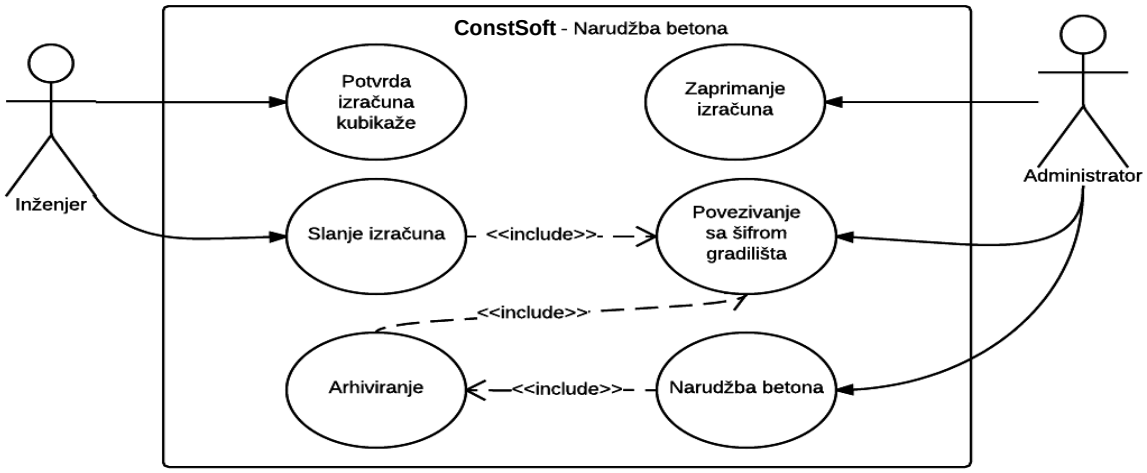
Na slici 9, možemo vidjeti prvi slučaj korištenja koji prikazuje kako teče proces otvaranja gradilišta. Nakon što inženjer zaprimi projekt dolazi na mjesto gradnje te gradilištu dodjeljuje šifru. S druge strane Administrator u bazi podataka otvara novo gradilište kojem dodjeljuje već zadanu šifru od strane inženjera. Već se ovdje primjećuje da se prilikom izgradnje arhitekture aplikacije treba paziti na povezivanje inženjera i administratora. To je veza od ključne važnosti za nesmetan i pravovremen protok informacija, te će za to biti potrebna stabilna internetska veza na obje strane i dovoljno prostora za pohranu.



Slika 10: Obrada gradilišta

Izvor: izrada autora

Obrada gradilišta je drugi slučaj korištenja, te je prikazan slikom 10. Obuhvaća unos šifre gradilišta koja se dodjeljuje svakom novom projektu. Nakon proučavanja projekta uzimaju se mjere s gradilišta, te koje se unose u aplikaciju. Samim unosom mjera izračunava se ukupna količina potrebnog betona. Kako bi aplikacija funkcionirala u skladu s glavnim ciljem biti će potrebno napraviti arhivu podataka kako bi se isti čuvali i bili dostupni u nekom kraćem vremenskom razdoblju. Za to je ponovno potrebno osigurati dovoljno prostora za pohranu.



Slika 11:Narudžba betona

Izvor: izrada autora

Posljednji slučaj korištenja prikazan je na slici 11, a odnosi se na samu narudžbu betona. Inženjer provjerava unesene mjere pojedinih segmenata te potvrđuje iznos koji je dobio unosom tih mjera. Šalje izračun u bazu kojeg zaprima administrator te se povezuje s gradilištem po ranije dodijeljenoj šifri. Administrator naručuje beton nakon čega se količina arhivira po šifri gradilišta.

Što s tiče arhitekture bit će potrebno povezati aplikaciju s bazom u kojoj će se podaci čuvati, te odakle će biti dostupni za daljnje postupke i narudžbe.

Svi koraci definirani u prethodna tri slučaja korištenja i uočene potrebne stavke podloga su za stvaranje prototipa arhitekture na kojoj će aplikacija biti izrađena. Ustanovljeno je kako će prethodno definirana arhitektura biti dovoljna za neometan rad aplikacije, te neće iziskivati dodatne troškove od razvijatelja aplikacije ni od naručitelja aplikacije.

Potrebno je ulaganje u pametne telefone s Android operativnom sustavom te ulaganje u stabilnu vezu između pametnih telefona i baze na koju će se spremati podaci. Osim toga podaci će se izračunavati unutar aplikacije, dok će se analiza i pohrana vršiti na nekoj odabranoj bazi. Svi izračuni biti će zabilježeni u bazi podataka tvrtke, a na kojoj su također spremljeni svi podaci o svakom gradilištu. S baze se u aplikaciju vraća povratna informacija o potvrdi narudžbe, odnosno narudžba se arhivira i unutar aplikacije.

Analizom Arhitekture utvrđeno je kako će pružiti potrebnu stabilnost i performanse za siguran budući rad aplikacije. No, isto tako primijećeno je da će biti potrebno poraditi na boljoj povezanosti s bazom podataka tvrtke, naročito ukoliko se bude planirao razvoj druge verzije aplikacije *ConstSoft*. Za sada će ova arhitektura biti dovoljna, te je dovoljno pouzdana za predviđene funkcionalnosti aplikacije.

S obzirom na odabrani alat za razvoj aplikacije, ustanovilo se kako će faza konstrukcije ipak duže trajati zbog čega je i definiran novi plan rada. Faza konstrukcije će trajati 10 dana više od prvotno predviđenih 16 dana. Novi plan je prikazan u tabeli 3.

Tabela 3: Plan razvoja ConstSoft-a

1. dan	2. dan	3.-28. dan	29. dan	30. dan
Početna faza - Vizija - Tjedni plan - Rizici - Troškovi - Poslovni slučaj korištenja	Faza elaboracije - Prototip - Smanjenje rizika	Faza konstrukcije - Dizajn - Kodiranje - Testiranje	Faza tranzicije - Rad na poboljšanjima - Prikaz prve verzije	Dostavljanje gotove verzije naručitelju

Izvor: izrada autora

6.5 Faza konstrukcije (*construction*)

Treća faza RUP životnog ciklusa je faza konstrukcije u kojoj je kao i u prethodnim fazama vrlo važno definirati ciljeve. Radi se o fazi u kojoj se snažan fokus stavlja na detaljan dizajn aplikacije, implementaciju i testiranje kako bi se dobio kompletan sustav. Ova faza uzima najviše vremena i to najčešće više od 50 % ukupnog trajanja projekta, te ima najviše troškove. Iako se u fazi elaboracije koja joj je prethodila definirala većina rizika, u ovoj se fazi stalno pojavljuju novi kako ona napreduje. Ukoliko je dobro definirana arhitektura u fazi elaboracije rizici u trenutnoj fazi ne bi trebali biti od većeg značaja, te bi njihov utjecaj na cjelokupnu aplikaciju trebao biti minimalan.

Tijekom faze konstrukcije potrebno je optimizirati upotrebu resursa kako bi se postigao najveći mogući nivo kvalitete i učinkovitosti. Iz tog se razloga u ovoj fazi najviše komunicira s naručiteljem, te se pokušavaju što je moguće točnije definirati zahtjevi.

Zahtjeve je potrebno i uskladili s arhitekturom aplikacije, odnosno treba voditi računa da se dostavi gotov proizvod koji će udovoljiti naručiteljevim željama.

Ova se faza odvija u više iteracija, a ukoliko se radi o manjim sustavima to su najčešće dvije iteracije. One su podijeljene tako da se u svakoj odvija implementacija određenog broja slučajeva korištenja, a broj ovisi o uočenim rizicima, o čemu na kraju i ovisi broj iteracija.

Članovi tima tijekom ove faze raspravljaju u eventualnim problemima u arhitekturi, te daju kritičko mišljenje o dizajnu i implementaciji. „Po završetku ove faze, trebali bi imati

softverski sustav koji radi i prateću dokumentaciju koja je spremna za dostavu korisnicima.“[39]

Obično se rad u fazi konstrukcije dokumentira unutar jednog od CM[40] alata kako bi se izbjegle eventualne pogreške ili izgubio dio posla. Unutar CM alata izrađuje se snimka svake iteracije kako bi se dokumentirao tijek razvoja i definirali dnevni zadaci.

Treći – dvadest i osmi dan: Za potrebe ovog diplomskog rada aplikacija će biti razvijena pomoću Xamarin Forms alata unutar Visual Studio 2017. Pošto alat nije ranije korišten od strane razvijatelja aplikacije predviđeno je nekoliko dana za instalaciju alata i potrebnih komponenti kako bi isti bio operativan. Također, nekoliko je dana utrošeno i na edukaciju o alatu preko internet preglednika i radionice u organizaciji Radiokluba Pazin koja se održala 10. Studenog 2018. godine.

Izradom slučajeva uporabe u prethodnoj fazi uočili su se još neki rizici koje je potrebno testirati, te će se kroz jednu iteraciju prikazati komponente i potrebni testovi (tabela 4).

Tabela 4: Prva iteracija faze konstrukcije

Zahtjevi	Komponente	Testovi
Detaljizirana su 3 slučaja uporabe	Definirano je 14 komponenti sustava	Pokretanje aplikacije i izbor opcija iz početnog izbornika
	14 komponenti je gotovo u potpunosti implementirano	Slanje sume segmenata
	Sustav je funkcionalan	Provjera zaprimanja informacija na bazu podataka tvrtke
	Implementiran je kod aplikacije	

Izvor: izrada autora

Pošto se radi o primjeru aplikacije za potrebe ovog diplomskog rada izvedena su samo prva dva testa iz prethodne tabele, te su oba uspješno izvršena. Zaključeno je kako nije potrebno daljnje poboljšavanje trenutne verzije sustava.

[39] I. Sommerville, SOFTWARE ENGINEERING; Ninth edition, Pearson, 2011., str. 51
[40] Configuration Management

Također, nakon što svi podaci budu dokumentirani tijek razvoja aplikacije bit će prikazan pomoću grafova.

6.6 Faza tranzicije (Transition)

Posljednja se faza odnosi na uvođenje sustava kod korisnika. Izvodi se tzv. Beta test kako bi se vrednovala očekivanja korisnika te na temelju toga ispravile eventualne pogreške i nepravilnosti. U ovoj se fazi također vrši obuka korisnika aplikacije, te se sve priprema kako bi sustav mogao biti implementiran. Cilj je u potpunosti zadovoljiti želje korisnika, te se ova faza izvodi u jednoj do dvije iteracije ovisno o složenosti sustava koji se radi.

Mnogo se pažnje posvećuje poduzimanju svih mjera za uspješno upravljanje novim sustavom. Fokus je naravno i na mogućim budućim performansama projekta, a koje su uočene kroz razvoj.

Dvadeset i deveti dan: Na početku radnog dana potrebno je Beta verziju dostaviti naručitelju. Prilikom instalacije na *smartphone* s Android operativnim sustavom, uputiti će se inženjera u osnove korištenja.

Na kraju radnog dana konačna verzija je spremna za uporabu, te je to kraj rada na prvoj verziji aplikacije *ConstSoft*.

Trideseti dan: Aplikacije je spremna za isporuku naručitelju, te je prikazana u nastavku s koracima za njezinu uporabu.

Aplikaciju *ConstSoft* je potrebno instalirati na *smartphone* naručitelja nakon čega se pritiskom na ikonu ona i pokreće.

Otvara se prvi prozor prikazan na slici 13, a na kojem se nalazi ekran s izbornikom.

Korištenje je potrebno započeti unosom šifre gradilišta nakon što se izabere prvo dugme iz izbornika „Gradilište“. Otvara se novi ekran prikazan na slici 14, te se unosi šifra gradilišta. Šifru se može opozvati pritiskom na dugme za povratak u izbornik, a može se i potvrditi što automatski korisnika vodi na slijedeći ekran prikazan na slici 15. Ekran „Mjere“ može se otvoriti i direktno iz izbornika. Unose se mjere po segmentima koje se mogu potvrditi ili opozvati pritiskom na jedan od dugmadi.

Ukoliko se mjere potvrde korisnik će se naći na ekranu „Izračun“ koji je prikazan na slici 16. Tamo će mu biti prikazan zbroj za gradilište za koje je upisana šifra, odnosno bit će mu prikazana potrebna količina betona s obzirom na ranije unesene mjere. Ako se korisnik slaže s dobivenim izračunom može ga potvrditi nakon čega će mu biti prikazan ekran sa slike 17 „Potvrda narudžbe“. Korisnik može potvrditi narudžbu što će ga vratiti na izbornik ili može opozvati potvrdu što će ga vratiti korak unatrag, odnosno na ukupan zbroj.

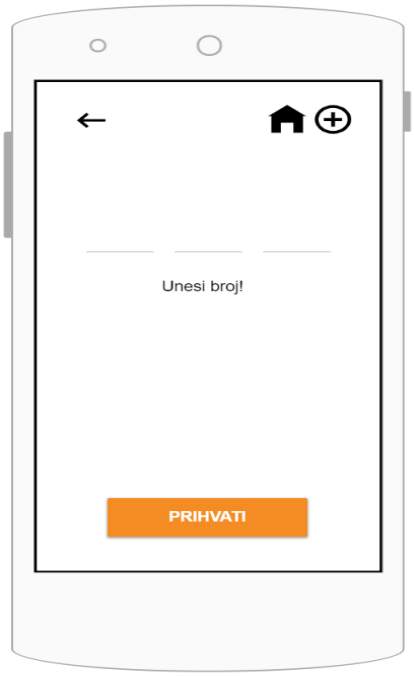
Sva dugmad za povratak vraća korak u natrag, dok ikona *home* na vrhu ekrana vraća na izbornik.

Slika 12: Izborni ekran

Izvor: izrada autora

Slika 1312: Unos šifre gradilišta

Izvor: izrada autora



Slika 14: Unos mjera segmenata

Izvor: izrada autora

Slika 135: Potvrda izračuna

Izvor: izrada autora

Slika 14: Potvrda narudžbe

Izvor: izrada autora

6.7 Analiza razvoja aplikacije

Sam razvoj aplikacije započeo je nakon što je proučena literatura o uporabi i primjeni RUP metodologije.

Nakon što je shvaćena osnovna ideja kako primijeniti RUP metodologiju na razvoj jednostavne aplikacije dogovoren je intervju s naručiteljem. Tijekom i nakon intervjua stvorena je osnovna ideja te se krenulo u potragu za dostupnim besplatnim alatima koji će se koristiti u razvoju aplikacije. Pronalazak dobrih alata bio je dosta dugotrajan te težak. Većina dobrih alata zahtijeva stvaranje računa te plaćanje određenog iznosa za njihovu uporabu. No, za potrebe ovog diplomskog rada korišteni su besplatni alati, te programi unutar *MS Office* paketa.

Razvoj aplikacije započeo je analizom odabranog tržišta, definiranjem osnovnih zahtjeva, definiranjem vizije aplikacije, te mogućih rizika. Slikama su se vizualno prikazali osnovni procesi aplikacije te je to vrlo jednostavno učinjeno pomoću alata *LucidChart*[41].

U tabeli 5 je prikazan tijek aplikacije, odnosno koliko je vremena utrošeno na pojedinu fazu, te koliko je rizika uočeno prilikom razrade svake faze. Isto je prikazano i na grafu 11.

Tabela 5: Utrošeno vrijeme i rizici

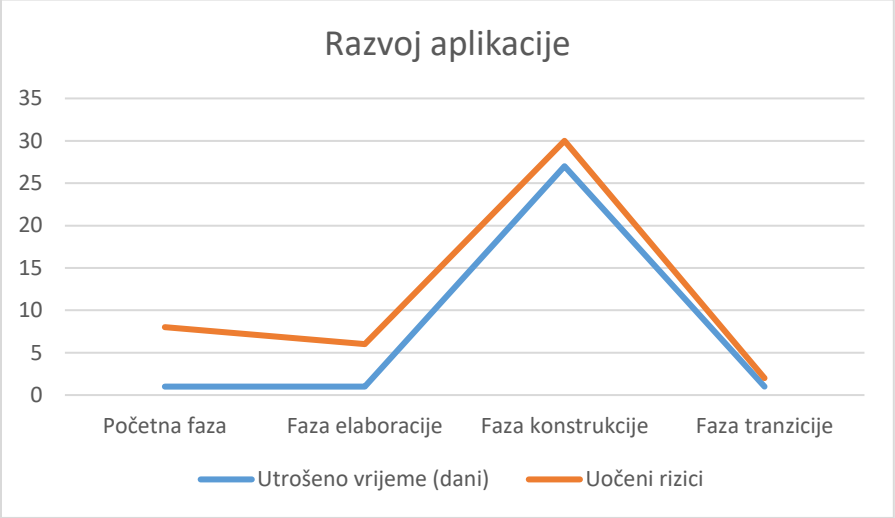
ConstSoft	**Početna faza**	**Faza elaboracije**	**Faza konstrukcije**	**Faza tranzicije**
Utrošeno vrijeme (dani)	1	1	27	1
Uočeni rizici	7	5	3	1

Izvor: izrada autora

Iz tabele 5 vidi se da je faza konstrukcije trajala najduže, te su unutar te faze uočena 3 rizika. Da se primijetiti kako je najviše rizika uočeno u početnoj fazi, te da su rizici opadali s napretkom razvoja.

[41] Softveru za dijagrame toka, organizacijske grafikone, UML dizajn, mentalne mape i druge vrste dijagrama koji korisnicima omogućava suradnju i zajednički rad.

Graf 11: Prikaz razvoja aplikacije



Izvor: Izrada autora

Grafom 11 je još jednom prikazano utrošeno vrijeme i primijećeni rizici tijekom pojedinih faza, te se još jednom potvrdilo kako je najviše vremena utrošeno unutar Faze konstrukcije koja izisikuje najviše znanja i pažnje od razvijatelja aplikacije.

Pomoću MS Project-a prikazan je proces razvoja s detaljima o trajanju i vezi između faza unutar projekta, te pojedinih aktivnosti, što je vidljivo na slikama 19, 20, 21 i 22.

Task Mode	Task Name	Duration	Start	Finish
	CONSTASOFT	**30 days**	**Mon 1.10.18**	**Tue 30.10.18**
	Početna faza	**1 day**	**Mon 1.10.18**	**Mon 1.10.18**
	Interviju s naručiteljem	2 hrs	Mon 1.10.18	Mon 1.10.18
	Definiranje vizije I zahtjeva	2 hrs	Mon 1.10.18	Mon 1.10.18
	Definiranje rizika	3 hrs	Mon 1.10.18	Mon 1.10.18
	Izrada plana razvoja	1 hr	Mon 1.10.18	Mon 1.10.18
	Klasni dijagram	*Milestone*	Mon 1.10.18	Mon 1.10.18

Slika 15: Detalji početne faze

Izvor: izrada autora

	Faza elaboracije	**1 day**	**Tue 2.10.18**	**Tue 2.10.18**
	Definiranje slučajeva uporabe	3 hrs	Tue 2.10.18	Tue 2.10.18
	Prototipitanje arhitekture	4 hrs	Tue 2.10.18	Tue 2.10.18
	Izrada novog plana razvoja	1 hr	Tue 2.10.18	Tue 2.10.18
	Prototip arhitekture	*Milestone*	Tue 2.10.18	Tue 2.10.18

Slika 168: Detalji faze elaboracije

Izvor: izrada autora

	Faza konstrukcije	**27 days**	**Wed 3.10.18**	**Mon 29.10.18**
	Programiranje aplikacije	23 days	Wed 3.10.18	Thu 25.10.18
	Testiranje rizika	3 days	Fri 26.10.18	Sun 28.10.18
	Testiranje arhitekture	1 day	Mon 29.10.18	Mon 29.10.18

Slika 1917: Detalji faze konstrukcije

Izvor: izrada autora

	Faza tranzicije	**1 day**	**Tue 30.10.18**	**Tue 30.10.18**
	Beta test	3 hrs	Tue 30.10.18	Tue 30.10.18
	Ispravljanje grešaka	3 hrs	Tue 30.10.18	Tue 30.10.18
	Obuka naručitelja	2 hrs	Tue 30.10.18	Tue 30.10.18
	Prva verzija aplikacije	*Milestone*	Tue 30.10.18	Tue 30.10.18

Slika 20: Detalji faze tranzicije

Izvor: izrada autora

Graf 12: Proces razvoja aplikacije prema fazama

Izvor: izrada autora

Graf 12 je još jedna potvrda kako se u razvoju aplikacije *ConstSoft* najviše vremena utrošilo na fazu konstrukcije, odnosno na programiranje same aplikacije.

7 ZAKLJUČAK

Danas se način života svodi na uporabu brojnih tehnologija koje bi trebale imati zadaću ubrzati svakodnevne procese, odnosno koje bi svakom čovjeku omogućile maksimalnu iskoristivost vremena bilo privatno ili poslovno. Poslovi oduzimaju sve više vremena zbog čega se često radi prekovremeno, odnosno čovjek ima sve manje vremena za svoje privatne potrebe.

Upravo iz tih razloga i zbog dostupnosti raznih softverskih metodologija znanstvenici i informatičari bi trebali raditi na tome da se razviju softverski alati koji bi skratili i ubrzali poslovne procese. To bi omogućilo brži protok informacija, bržu obradu podataka, a s ciljem obavljanja poslovnih zadataka u zadanom vremenskom roku i unutar radnog vremena.

Kako već sada postoji mnoštvo mobilnih aplikacija, smatra se kako bi mobilna aplikacija u građevinskom sektoru donijela veliki napredak tog sektora u Republici Hrvatskoj, ali i u svijetu ukoliko je njezina namjena prilagodljiva.

Uporabom RUP metodologije, koja je predstavljena u ovom diplomskom radu, uočeno je da je ta metoda lako prilagodljiva na mnoge softverske sustave neovisno o njihovoj veličini i namjeni. Primjena ove metodologije u sektoru građevine pokazala se kao vrlo mudrim i dobrim izborom s obzirom da se kroz faze aplikacija konstantno može unaprjeđivati i poboljšavati. Svaka faza generirala je verzijom aplikacije, od prve faze koja je dala samo privid onog što se želi razviti do posljednje faze tranzicije u kojoj je prikazano kako bi ona stvarno izgledala i u uporabi.

RUP metodologijom su se pri razvoju prototipa ove aplikacije na vrijeme uočili rizici koji su već postojali, te oni potencijalni, što se omogućilo njihovo sprječavanje odnosno uklanjanje.

Iako aplikacija nije implementirana u stvarnom vremenu prikazom svih koraka njezinog razvoja moglo se uočiti kako je faza konstrukcije najzahtjevnija te zahtijeva najviše znanja od razvijatelja softvera. Isto tako unutar ove faze odvijaju se ključni potezi kako bi aplikacija zaživila i zadovoljila zahtjeve korisnika.

Nakon izvršenog istraživanja i analize potvrđeno je da i tim od jedne osobe može uspješno primijeniti spomenutu metodologiju u razvoju mobilne aplikacije.

8 POPIS LITERATURE:

Knjige:

1. Kroll P., Kruchten P. (2003.), Rational Unified Process Made Easy: A Practitioner's Guide to the RUP, Adison Wesley
2. Maciaszek L. A. (2007.), Requirements analysis and system design, 3rd edition, Pearson Education Limited
3. Manger R. (2013.), Softversko inženjerstvo, skripta, nadopunjeno drugo izdanje, Zagreb.
4. Sommerville I. (2011.), SOFTWARE ENGINEERING, Ninth edition, Pearson
5. Tomašević V. (2012.), Razvoj aplikativnog softvera, Univerzitet Singidunum, Beograd

Članci:

1. Anwar, A. (2014.), A Review of RUP (Rational Unified Process), College of Arts & Sciences Department of Computer Science & MIS University of Atlanta Atlanta, GA, USA

Internet:

1. A Guide To Agile Scrum Methodology in Mobile App Development, Dostupno na: https://appinventiv.com/blog/agile-scrum-methodology-in-mobile-app-development (Pristupljeno Rujan 2018.)
2. GALA građevinski software,troškovnici,mrežno planiranje (Online), Dostupno na: http://sur.ly/i/gala-construction-software.com/ (Pristupljeno: Siječanj 2018.)
3. Građevinarstvo u 2016. godini, HGK, (Online), Ožujak 2018., Dostupno na: https://www.hgk.hr/documents/gradevinarstvo-rh-u-201658e619733cb43.pdf (Pristupljeno: Veljača 2018.)
4. K. Severović, Upravljanje odnosima s klijentima kao izvor informacija za oblikovanje I poboljšanje usluga (Online), 2013, Dostupno na: http://services.foi.hr/thesis_phd/rad_severovic.pdf (Pristupljeno: Kolovoz 2016.)

5. Metodologija razvoja informacionih Sistema (Online), Dostupno na: http://www.ef.uns.ac.rs/Download/osi_master/2010-02-17_dodatni_materijal_za_ispit.pdf (Pristupljeno: Kolovoz 2016.)
6. Najveći rast građevinskih radova od kraja 2016. (Online), Travanj 2018., Dostupno na: https://lider.media/aktualno/biznis-i-politika/hrvatska/najveci-rast-gradevinskih-radova-od-kraja-2016/ (Pristupljeno: Lipanj 2018.)
7. Pristup izradi mobilnih aplikacija – Agilne metode razvoja aplikacija, M. Zekić-Sušac (Online), Dostupno na: http://www.efos.unios.hr/razvoj-poslovnih-aplikacija/wp-content/uploads/sites/228/2013/04/RPA_P1_Agilne-metode1.pdf (Pristupljeno: Kolovoz 2016.)
8. Procesi razvoja softvera (Online), Dostupno na: http://web.efzg.hr/dok/inf/pozgaj/pisani%20materijali/T03%20Modeli%20razvoja%20softvera.pdf (Pristupljeno: Kolovoz 2016.)
9. Rapid Application Development (Online), Srpanj 2016, Dostupno na: http://searchsoftwarequality.techtarget.com/definition/rapid-application-development (Pristupljeno: Listopad 2016.)
10. Rational Unified Process, Best practices for software development teams (Online), Srpanj 2005, Dostupno na: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf (Pristupljeno: Kolovoz 2016.)
11. The 10th annual state of agile report (Online), 2016, Dostupno na: http://www.agile247.pl/wp-content/uploads/2016/04/VersionOne-10th-Annual-State-of-Agile-Report.pdf (Pristupljeno: Kolovoz 2016.)
12. The Stadish Group Report (Online), 2014, Dostupno na: https://www.projectsmart.co.uk/white-papers/chaos-report.pdf (Pristupljeno: Studeni 2018)
13. Waterfall vs. Agile: Which is the Right Development Methodology for Your Project? (Online), 2018, Dostupno na: https://www.seguetech.com/waterfall-vs-agile-methodology/ (Pristupljeno: Rujan 2018.)

9 POPIS SLIKA, TABELA I GRAFOVA

10 PRILOZI:

Prilog 1: Intervju s magistrom inženjerom građevinarstva

UPITNIK O POTREBAMA TVRTKE U GRAĐEVINSKOM SEKTORU

1. Kojom vrstom građevine se bavi tvrtka?

- Pretežno niskogradnjom i elektrifikacijom, ali i visokogradnjom.

2. Na kojem području djelujete?

- Pretežito na području Istarske županije.

3. Koliko ljudi je zaposleno u vašoj tvrtki?

- Pedeset ljudi.

4. Koliko je inženjera građevinarstva zaposleno u vašoj tvrtki?

- Troje.

5. S koliko vanjskih tvrtki usko surađujete?

- Trenutno s nekoliko.

6. Koji su najčešći kanali komunikacije s partnerima i kooperantima?

- E-mail i telefonski razgovori.

7. S kojim problemima se u vašoj tvrtki inženjeri građevine najčešće susreću?

- U građevinarstvu je to veoma širok pojam. Zna se desiti i do desetak problema dnevno, a sve u vezi izvođenja radova.

8. Koji se resursi koriste za rješavanje problema?

- Koriste se napredni softweri, ali najčešće vlastita pamet.

9. Koje je trenutno optimalno vrijeme obrade jednog gradilišta u početnoj fazi?

- Ovisno o vrsti. Od par dana do nekoliko tjedana.

10. Koje je željeno optimalno vrijeme obrade gradišta u početnoj fazi?

- Tjedan dana, po mogućnosti i ranije.

11. Na koji način ste povezani s administrativnim dijelom tvrtke?

- Na svakodnevnoj bazi, osobno, telefonski i e-mailom.

12. Jeste li zadovoljni informatičkom opremom s kojom raspolaže vaša tvrtka?

- Djelomično, jer je uvijek postoji mogućnost napretka.

13. Biste li voljeli imati aplikaciju koja će ubrzati obradu gradilišta?

- Naravno, jer tako ostaje prostora za rješavanje druge tekuće problematike.

14. Biste li voljeli biti povezani s administrativnim dijelom putem pametnog telefona?

- Svakako da da.

15. Imate li neki prijedlog?

- Bilo bi dobro kad bi jednim klikom na pametnom telefonu mogao naručiti beton, armaturu i slično. Na taj način bi odmah slao potrebe materijala gradilišta, i ne bi ulazio u procedure narudžbe već bi taj dio rješavali administratori u tvrtki.

16. Koji materijal najčešće naručujete?

- Pretežito beton i armaturu.

17. Da li bi Vam aplikacija za narudžbu betona ubrzala i olakšala posao?

- Svakako da.

18. Što biste zahtijevali od takve aplikacije?

- Automatski izračun materijala, pohranu podataka o svakom gradilištu, dobru povezanost s administrativnim dijelom tvrtke.

Prilog 2: Anketa

Anketa o potrebi i korištenju mobilnih aplikacija u građevinskom sektoru

Anketu je potrebno ispuniti na način da se zaokruži jedan ili više odgovora (ako je to izričito navedeno) ispod pitanja. Ako nema ponuđenih odgovora, odgovor je potrebno upisati na crtu ispod postavljenog pitanja.

Ukoliko je potrebno bodovati odgovore od najlošijeg do najboljeg, tada je broj bodova potrebno upisati na crtu ispred svakog ponuđenog odgovora sukladno uputama navedenim u pitanju.

1. Zaokružite vašu stručnu spremu:
    - SSS
    - VŠS
    - VSS

2. U kojoj ste instituciji stekli svoje zvanje?
    - _____

3. Koliko imate iskustva u radu u svojoj struci?
    - Manje od jedne godine
    - Manje od 5 godina
    - Više od 5 godina

4. Na kojoj poziciji radite?
    - _____

5. Koliko dugo radite na trenutnoj poziciji?
    - Manje od jedne godine
    - Manje od 5 godina
    - Više od 5 godina

6. Koliko radnog vremena provodite na terenu?
    - Ne radim na terenu
    - Do 50% radnog vremena
    - Više od 50% radnog vremena

7. Ukoliko radite na terenu, koje od slijedećih uređaja pritom koristite (zaokružite jedan ili više):
    - Ne radim na terenu
    - Pametni telefon
    - Tablet
    - Laptop

8. Uređajima koji su navedeni niže dodijelite broj bodova s obzirom na to koji najviše preferirate koristiti u radu. Jedan bod dajte uređaju koji najmanje preferirate, dva boda uređaju koji srednje preferirate, a tri boda uređaju koji najviše preferirate.

    _ Pametni telefon

    _ Tablet

    _ Prijenosno računalo

9. Da li vam rad s više uređaja predstavlja problem?
    - DA
    - NE

10. Biste li voljeli da vam je sve potrebno dostupno samo na jednom uređaju?
    - DA
    - NE

11. Koje od slijedećih aplikacija koristite u radu (zaokružite jedan ili više)?
    - ArhiCAD ili sl.
    - AutoCAD ili sl.
    - MS Project ili sl.
    - AdriaKOD ili sl.
    - Drugo: _____

12. Mogu li se aplikacije koje ste ranije zaokružili koristiti I na mobilnim uređajima?
    - DA
    - NE

13. Koliko često navedene aplikacije koristite na mobilnim uređajima?
    - Nikada
    - Povremeno
    - Svaki dan

14. Koliko često komunicirate s vanjskim suradnicima, kao što je dobavljač materijala?
    - Nikada
    - Svaki tjedan
    - Svaki dan

15. Omogućuje li neka od navedenih aplikacija komunikaciju s vanjskim suradnicima?
    - DA
    - NE

16. Smatrate li da bi aplikacija koja omogućuje komunikaciju s vanjskim suradnicima ubrzala vaš tijek posla?
    - DA
    - NE

17. Koje uvjete bi trebala ispunjavati takva aplikacija (zaokružite jedan ili više)?
    - Dostupnost za Android mobilne uređaje
    - Dostupnost za IOS mobilne uređaje
    - Besplatno preuzimanje
    - Jednostavna uporaba
    - Umreženost s vanjskim suradnicima
    - Drugo:_____

18. Da li bi Vam bilo od koristi da unutar aplikacije postoji mogućnost izračuna količina materijala?
    - DA
    - NE

19. Da li bi Vam bilo od koristi da aplikacija može pohraniti izračunate količine?
    - DA
    - NE

20. Da li biste koristili takvu aplikaciju?
    - DA
    - NE