UDC: 005.5:[659.2:004 Info M str. 38-43

RAZVOJ SOFTVERA ZA UPRAVLJANJE PROJEKTIMA ZASNOVAN NA RAZLIČITIM AGILNIM METODOLOGIJAMA
DEVELOPMENT OF PROJECT MANAGEMENT SOFTWARE BASED ON DIFERENT AGILE METHODOLOGIES

Ognjen Pantelić, Una Nikolić, Stefan Krstović

REZIME: U početku se za sve informatičke projekte koristio isti pristup razvoja softvera, tradicionalne metodologije. Međutim, zbog brzog rasta IT industrije, projekti su često bili neuspešni, pa samim tim počele su se razvijati i agilne metodologije. U prvom delu rada dat je pregled tradicionalnih i agilnih metodologija, sa teorijskog aspekta, kao i trenutna zastupljenost i jednih i drugih, a dalje u radu pažnja će biti usmerena samo na agilne metodologije. Metode nad kojima će se vršiti istraživanje su tri najpoznatije agilne metode: Skram, Kanban i Ekstremno programiranje. Sve tri metode biće posebno predstavljene zajedno sa njihovim vrednostima i osobinama, a zatim će biti predstavljena kombinacija njihovih vrednosti koja je osnov za razvoj softvera. Drugi deo rada se odnosi na praktični rad, odnosno na razvoj softvera za upravljanje projektima, u vidu veb aplikacije koji bi dao vizuelni prikaz i jasnije objasnio teorijski aspekt rada.

KLJUČNE REČI: agilne metodologije, upravljanje projektima, razvoj softvera, veb aplikacija, kombinacija metoda

ABSTRACT: In the beginning, the same software development approach, traditional methodologies, was used for all IT projects. However, due to the rapid growth of the IT industry, projects were frequently unsuccessful and therefore agile methodologies began to develop. The first part of the assignment will provide an overview of traditional and agile methodologies from a theoretical point of view, as well as their current representation, whereas further in the assignment attention will be focused on agile methodology. The research will be conducted on the three most common agile methods: Scrum, Kanban, and Extreme Programming. All three methods will be analyzed separately along with their values and characteristics, after which a combination of their values will be presented, which is the basis for software development. The second part of the assignment is related to the practical work, namely the development of project management software which would give a visual presentation and more distinctly explain the theoretical aspect of the paperwork.

KEY WORDS: agile methodologies, project management, software development, web application, a combination of methods

1. UVOD

Uspešnost projekata u svakom poslovanju, pa i u softverskom inženjerstvu, uslovljena je dobrom organizacijom. Informatički projekti postaju sve veći i složeniji, samim tim javlja se potreba za definisanjem formalnog procesa upravljanja projektima. Upravljanje projektima ima za cilj efikasnu upotrebu resursa, ravnomernu podelu poslova i formiranje plana kako bi se projekat realizovao u roku, kako je zamišljeno, sa predviđenim načinom izvršavanja i očekivanim rezultatima. Jedan od ključnih koncepata upravljanja projektima je sam proces razvoja softvera. U ovom radu fokus je na agilnim metodologijama razvoja softvera.

2. AGILNE METODOLOGIJE

Značajne specifičnosti koje odlikuju IT projekte, posebno projekte razvoja softvera, uslovile su potrebu za stvaranjem novih pristupa, procedura ili metodologija za efikasan rad na programiranju i razvoju softvera. Tradicionalni pristupi nisu bili pogodni, pre svega, zbog toga što se projekti razvoja softvera najčešće započinju bez čvrstih i nepromenljivih specifikacija i zbog toga što se najčešće zahtevaju česte promene, što takođe zahteva drugačiji i fleksibilniji pristup. Kod razvoja softvera veza sa klijentom je veoma čvrsta i klijent često nije siguran šta tačno očekuje kao rezultat, te kroz zahteve za promenama ide ka svom cilju. Moguće je da se rade faze ili delovi softverskog rešenja i kao takvi isporučuju klijentu, koji ocenjuje da li razvoj softvera ide u pravcu koji je za njega poželjan. Klijent ponekad zahteva da se proces vrati na prethodnu fazu i da se uvedu određene promene koje bi poboljšale softverski proizvod. Da bi se na ovaj način radilo, potrebno je da tim koji radi na razvoju softvera bude spreman na fleksibilan pristup u radu, na stalne promene i blisku saradnju sa klijentom. U pitanju je specifičan timski rad, neprekidna i brza komunikacija unutar tima i posebno bliska komunikacija sa klijentom kroz česte sastanke i brzu i efikasnu razmenu mišljenja i informacija kako bi se usaglasile želje i mogućnosti postizanja rezultata.[1]

Agilne metodologije razvoja softvera su nastale krajem dvadesetog veka. Promovišu dinamičan i disciplinovan timski rad. Termin 'agilne' je izabran u cilju da naznači sposobnost ovih metoda da odgovore na česte promene zahteva.

Kod agilnih metodologija planiramo onoliko koliko je neophodno i sa izgradnjom svakog dela, tim procenjuje uspešnost tog dela u saradnji sa klijentom. Kako klijent ima uvida u prototip, on je u mogućnosti da definiše ili pre definiše zahteve i objasni timu šta su zapravo zahtevi. Agilni metod obuhvata promene koje povećavaju uspešnost, a iterativnim razvojem smanjuje cenu tih izmena. Pravljenje izmena na malom delu projekta je mnogo jeftinije nego praviti izmene na već dovršenom projektu. [2]

3. PREGLED IZABRANIH METODOLOGIJA

Metodologije koje će biti dalje objašnjene su:

- Skram i Kanban (ili kombinacija ove dve) za praćenje menadžmenta
- Ekstremno programiranje (XP) za tehničku praksu

3.1 *Skram*

Skram metoda predstavlja iterativno-inkrementalni proces, koji se koristi za upravljanje razvojem i održavanjem softverskih proizvoda. Proces sadrži set menadžerskih preporuka, ali ne definiše aktivnosti samog razvojnog procesa. Iz tog razloga često se koristi u kombinaciji sa drugim procesima razvoja softvera. Zasnovan je na empirijskoj kontroli procesa, koju omogućavaju tri bazična koncepta: transparentnost (vidljivost značajnih aspekata procesa onima koji su odgovorni za rezultate), pregled (provera artifakata) i adaptacija (prilagođavanje procesa ili resursa ukoliko se utvrdi da rezultati nisu odgovarajući). U skladu sa tim, Skram meri proizvedene izlaze budućeg sistema nakon svake iteracije.

Odgovornosti u realizaciji skram aktivnosti podeljene su na sledeće uloge koje imaju pojedinci/timovi[3]:

1. Vlasnik proizvoda ima zadatak da prikuplja inpute od kupaca, krajnjih korisnika i članova razvojnog tima. Prikupljene inpute transformiše u zahteve i vrši njihovo vrednovanje sa aspekta prioriteta u razvoju. On je odgovoran da proizvod bude razvijen i isporučen u skladu sa zahtevima klijenta
2. Razvojni tim čini od pet do deset članova sledećih profila eksperata: analitičari, programeri, dizajneri i testeri. Razvojni tim ima autonomiju u donošenju odluka, kao i slobodu da vlasniku proizvoda dostavlja ideje za unapređenje proizvoda.
3. Skram master predstavlja specifičnu poziciju u razvoju softvera koja se odnosi na uspostavljanje posredništva između vlasnika proizvoda i članova razvojnog tima. On je odgovoran za uspešan razvoj krajnjeg proizvoda. Zadužen je za uspešnu implementaciju Skram metode na projektu, pružajući kontinuiranu pomoć i podršku članovima razvojnog tima.
4. Menadžer je odgovoran za konačno donošenje odluka, a učestvuje i u procesu postavljanja ciljeva i definisanju zahteva.
5. Klijent je osoba koja sa ostalim članovima tima učestvuje u procesu generisanja zahteva i definisanja funkcionalnosti koje budući sistem treba da ima. Takođe, učestvuje i u procesu provere proizvedenih rezultata i funkcionalnosti, pružajući povratne informacije razvojnom timu.

Ova metoda sugeriše da se rad na razvoju softvera odvija u kraćim ciklusima koji se nazivaju sprintovi, zatim se obavljaju neprekidne konsultacije sa klijentom i da se nakon određenog ciklusa vrši analiza i preispitivanje i eventualno uvedu željene i potrebne izmene. To uključuje obavezne sastanke pre i nakon svakog sprinta, radi razmatranja da li je sve urađeno prema zahtevima i da li je potrebno uvoditi neke promene. U određenoj situaciji moguće je vratiti se unazad i realizovati određeni sprint u skladu sa novim zahtevima.

Sprint predstavlja iteraciju koja može da traje najduže mesec dana, a za rezultat daje izvršni proizvod. [5]. Ključni događaj za pokretanje sprinta jeste planiranje sprinta. Na događaju planiranja sprinta precizira se šta u okviru sprinta treba biti urađeno tj. šta će biti rezultat inkrementa (proizvoda) i kako će se taj rezultat ostvariti. Vreme trajanja sprinta se ne može produžavati, pa je važna njegova realna procena tokom planiranja sprinta. Dnevni sastanci predstavljaju drugi bitan događaj u Skramu. Njihovo trajanje ne sme biti duže od petnaest minuta, te se iz tog razloga održavaju u stojećem položaju prisutnih. Dnevni sastanci obezbeđuju transparentnost u radu, jer je svaki član razvojnog tima u obavezi da podnese izveštaj o problemima koje je identifikovao, o urađenom poslu prethodnog dana i o rezultatima koje planira da ostvariti do sutrašnjeg dnevnog sastanka. Retrospektiva sprinta je događaj na kojem učestvuje ceo tim, a ima za cilj da identifikuje ispravnosti/defekte u radu izgrađenog softverskog rešenja. Na ovom događaju se putem table, ili neke druge vizuelne tehnike, u jednu kolonu evidentiraju ispravne funkcionalnosti rešenja, dok se u drugu vrši popis neispravnih funkcionalnosti.

Izgradnja softverskog rešenja Skram procesom realizuje se kroz tri razvojne faze [4] :

1. Predigra (planiranje i definisanje zahteva, dizajn/arhitektura visokog nivoa apstrakcije).
2. Igra - razvojna faza (razvoj, sprintovi - iterativni ciklusi, poboljšanja, nove verzije).
3. Post-igra (nema novih zahteva, testiranje, integracija).

3.2 *Kanban*

Ideja ove metode potekla je od tri inženjera Tojote: Kičiro Tojota, Eiji Tojoda i Taiči Ohno [6]. Osnovna ideja filozofije je da svaka organizacija osnovne kanban principe implementira na sopstveni način, kroz kontinuirano učenje iz sopstvenog iskustva. Iz tog razloga ne postoje utemeljene instrukcije za njihovu efikasnu realizaciju.

Kanban filozofija fokusirana je na smanjenje ukupnih troškova, na poboljšavanje sveukupnog kvaliteta i kvaliteta proizvoda koji se isporučuje korisnicima, skraćenje vremena isporuke proizvoda i na povećanje zadovoljstva korisnika.

Jedna od ključnih ideja Kanban metode je da se eliminiše višak. To se postiže korišćenjem kanban kartica i kanban table kako bi se vizualizovao način na koji se resursi kreću kroz proizvodni ciklus. Ovo svim učesnicima omogućava da budu potpuno uključeni u proces i pomaže menadžerima da spoznaju višak ili manjak u proizvodnji. Ova metoda takođe omogućava organizacijama da započnu sa njihovim postojećim tokom rada, polako uvodeći promene kako bi vremenom dostigli željeni nivo i postigli željene rezultate. To se takođe može postići ograničavanjem rada u toku (Work In Progress - WIP). Opšti termin za sisteme koji koriste Kanban metodu je kontinuirani tok, podrazumevajući da rad treba stalno da protiče kroz sistem, a ne da se deli u različite vremenske celine.

U nastavku će biti navedene i opisane aktivnosti koje su od suštinskog značaja za upravljanje Kanban sistemom. To su vizualizacija, ograničenje rada u procesu (*Work in progress*), upravljanje tokom rada, jasno definisana pravila i povratne informacije.

Kanban sistemi koriste mehanizme kao što je kanban tabla za vizuelizaciju rada i proces kroz koji rad prolazi. Definisano je nekoliko pokazatelja efikasne vizuelizacije. Neki od njih su tačka obaveze i tačka isporuke. Tačka obaveze nastaje kada se tim koji radi složi da izvrši određeni radni zadatak.

Tačka isporuke je kada tim isporučuje radni predmet kupcu. Zatim, efikasna vizuelizacija ogleda se u politikama koje određuju koji rad treba da se nalazi u određenoj fazi, a takođe i u ograničenjima rada u toku.

Kada se utvrde ograničenja u količini rada koji je u toku u sistemu i koriste se te granice da ako bi se znalo kada započeti novi proces, može se umanjiti protok rada i smanjiti vreme, poboljšati kvalitet i isporučivati češće.

Tok rada treba da maksimizira isporuku vrednosti, minimizira izgubljeno vreme i treba biti što predvidljiviji. Timovi koriste empirijsku kontrolu kroz transparentnost, inspekciju i adaptaciju kako bi uravnotežili potencijalno konfliktne ciljeve i kako do njih uopšte ne bi došlo.

Jasna pravila pomažu u objašnjavanju celokupnog procesa i različitih faza u toku procesa. Pravila treba da budu jednostavna, dobro definisana, vidljiva, uvek primenljiva i takva da se lako mogu promeniti.

Povratne informacije su suštinski element u bilo kom sistemu koji traži evolucionu promenu. Najbolji sistem za upravljanje životnog ciklusa Kanban metode je pomoću povratnih informacija. Razlog tome je taj što predmeti rada, zaposleni i drugi subjekti u procesu treba da protiču kroz sistem konstantno, a svaki sistem je različit na svoj način.

3.3 Ekstremno programiranje

Ekstremno programiranje (XP) je agilni okvir za razvoj softvera koji ima za cilj proizvodnju kvalitetnijeg softvera i produktivniji i kvalitetniji rad razvojnog tima. XP je najspecifičniji od agilnih metoda obzirom da se koristi konkretno za razvoj softvera. [7]

Ova metoda se vodi principom da je klijent član tima. On definiše ciljeve i prioritete u okviru korisničkih celina, konstantno sarađujući sa članovima tima, a to se sprovodi u cilju boljeg razumevanja zahteva korisnika od strane razvojnog tima. Korisničke celine služe kao referenca na nešto što će se dalje razmatrati i implementirati. Razvoj prolazi kroz kratke cikluse koji obuhvataju plan trenutne iteracije. Uspeh razvojnog procesa se meri sagledavanjem napretka. Svi detalji o korisničkim celinama se dokumentuju u obliku testova, radi lakšeg praćenja napretka i implementiranih zahteva. Potrebno je da produktivni kod pišu dva programera na jednoj razvojnoj jedinici. Jedan član piše kod dok ga drugi proverava i time prati proces implementacije. Poželjno je da se uloge često menjaju, kako bi se održao kvalitetan odnos između članova. Članovi parova se menjaju bar jednom dnevno. Na taj način je postignuta intenzivna saradnja između članova. Produktivni kod se piše sa ciljem da zadovolji testove koda. Razvijen softver je kolektivno vlasništvo članova tima. To znači da svako ima pravo da ga proveri i unapredi. Timovi koriste sisteme za upravljanje verzija koji olakšavaju čestu izmenu koda. Ekstremno programiranje ne preporučuje prekovremeni rad, osim možda poslednje nedelje razvoja. Na taj način se održavaju bolji odnosi i rasterećenost unutar tima, tim je motivisaniji što prouzrokuje smanjen broj grešaka. Tim bi trebalo da bude smešten u jednoj prostoriji, radi lakše komunikacije i saradnje. Suština planiranja je u podeli odgovornosti između korisnika i razvojnog tima. Korisnik odlučuje koje funkcionalnosti softvera treba implementirati dok tim odlučuje koliko to košta. Sistem treba dizajnirati što jednostavnije uz često refaktorisanje.

Metoda XP se vodi praćenjem dvanaest praksi, grupisanih u četiri kategorije [8]:

1. Povratne informacije (programiranje u paru, planiranje iteracije, razvoj vođen testovima (TDD))
2. Kontinuiran proces (zajednički direktorijum, osnovni kod koji se nadograđuje, refaktorisanje koda, često postavljanje na produkciju)
3. Zajedničko razumevanje (standardi kodiranja, zajednički kod, jednostavan dizajn)
4. Zaštita programera (održivi tempo – 40h nedeljno)

Uloge članova tima su slične definisane kao u Skram metodi. Glavna razlika je u tome što umesto Skram mastera postoji XP trener koji mora imati i tehničko znanje kako bi mogao da upravlja razvojnim timom.

Najveća prednost Ekstremnog programiranja je ta što omogućava kompanijama za razvoj softvera da uštede troškove, frustracije i vreme uklanjanjem neproduktivnih aktivnosti. To znači da prvenstveno nastoji da smanji rizike povezane sa neuspehom u samom projektu, a to omogućava programerima da se usredsrede na kodiranje.

Izgradnja softverskog rešenja XP procesom realizuje se kroz nekoliko razvojnih faza: istraživanje, planiranje, sprovođenje iteracija, produkcija, održavanje i terminiranje. [8]

4. KOMBINACIJA AGILNIH METODOLOGIJA

4.1. *Skramban*

Skramban kombinuje najbolje funkcionalnosti Skram i Kanban metode. Sastavlja prirodu perspektive Skrama i proces poboljšanja sposobnosti Kanbana, dozvoljavajući timovima da idu ka agilnom razvoju i konstantno poboljšavaju proces.

Kombinacija ove dve metode ima dosta prednosti. Može pomoći razvojnom timu da otkloni prekomerni stres i preopterećenost, poboljša efikasnost i zadovolji klijenta. Ali najbitnije je da doprinosi dostavljanju proizvoda visokog kvaliteta, kontinuiranom poboljšanju, smanjenju otpada i redukovanju vremena realizacije.

Skramban alati su veoma laki za korišćenje. Postoji dosta dodatnih alata koji olakšavaju i pomažu oko podešavanja Skramban table. Omogućavaju komunikaciju i saradnju tima u realnom vremenu, bilo kad i bilo gde tako što članovi tima dele zadatke, beleške, dokumente i komentare koristeći izabrani alat. Skramban alati su obično veb-orijentisani i predstavljaju se u vidu table ispunjene zahtevima klijenata koji su prikazani kroz zadatke dodeljene članovima tima. Ovakav način praćenja razvoja pomaže timu da održi kontinuitet u proizvodnji i razvoju i poveća produktivnost. Skramban kombinuje Skram i Kanban i sadrži najbolje prakse iz obe metodologije. Sa jedne strane koristi agilnu prirodu Skrama, a sa druge strane podstiče timove da konstantno poboljšavaju proces u skladu sa Kanbanovim principom kontinuiranog unapređenja.

Glavna razlika između Skram i Kanban table je u redosledu izvlačenja i dodavanja zadataka. U Skramu, jednom kada je sprint isplaniran i zadaci postavljeni na tablu, sve članovi tima mogu uraditi sa njima je povući ih, odnosno prebaciti u

sledeću kolonu koja označava da je rad na određenom zadatku započet. Međutim, u Kanbanu, nakon postavljanja zadataka na tablu, u odgovarajuće kolone, na članovima tima je da odaberu na kojim zadacima će prvo raditi. Samim tim Kanban je fleksibilniji u ovom pogledu. Skramban rešenje daje povećanje sposobnosti sistema, dopuštajući da se obrađuje više stavki. Praćenje vremena realizacije je lakše zahvaljujući održavanju balansa između kapaciteta tima i potražnje.

Obično se preporučuje primena WIP limita čak i na beklog, zato što smanjuje moguću grešku procene, odnosno estimacije koliko se može uraditi u određenoj iteraciji. Umesto toga, članovi tima mogu brže završiti sa tekućim zadacima i početi rad na sledećim. Na ovaj način se može vreme uštede na proceni može iskoristiti na pregled procesa i primenu novih ideja za unapređivanje procesa. Kako su Kanban i Skram procesi koji se lepo uklapaju i upotpunjuju, ono što se može dobiti od njihove kombinacije je mogućnost konstantnog usavršavanja procesa na jako jednostavan način.

4.2. *Skramban sa ekstremnim programiranjem*

Ekstremno programiranje podrazumeva jake tehničke prakse, uključujući razvoj vođen testovima, refaktorisanje koda, programiranje u paru i kontinuiranu integraciju. Ove prakse menjaju način na koji se sam posao izvršava i timovima trebaju meseci kako bi ih uključili u svoj proces rada. Skram i Kanban, sa druge strane, su metode koje ne menjaju sam proces izvršavanja i razvoja, već mogu promeniti način na koji je proces vođen i organizovan, i timovima je obično potrebno nekoliko dana da ih usvoje. Kombinacija ovih metoda se u praksi pokazala kao jako korisnom jer unapređuje sam proces proizvodnje sa jedne strane, zajedno sa poboljšanjem organizacije sa druge strane.[9]

Jedan od razloga zašto je XP metoda toliko popularna i korišćena je njena fleksibilnost. Zatim, XP se više svodi na izvođenje samih tehnika nego celokupnog procesa i zato se jako dobro uklapa sa metodama koje se više fokusiraju na unapređenje procesa, kao što su Skram i Kanban. U tom pogledu lako se mogu kombinovati funkcionalnosti Ekstremnog programiranja kada su neke od XP praksi pogodne za korišćenje u određenom procesu. XP metoda sama po sebi nije dovoljna i pogodna za implementaciju.

U svakom slučaju ne moraju se koristiti sve prakse Ekstremnog programiranja istovremeno, ali ako se tim odluči za neku praksu potrebno je sprovesti je i pridržavati se od početka do kraja. Ovo važi za bilo koji postupak koji je započet.

Najpopularnija hibrid metoda koja se trenutno koristi je Ekstremno programiranje u kombinaciji sa Skramom, a samim tim se Ekstremno programiranje lako može koristiti u kombinaciji sa Skramban metodom. [10]

Jedan od razloga zašto se XP ne svodi samo na tehničku praksu je taj što kao i Skram, i XP je baziran na vrednostima. XP vrednosti su komunikacija, jednostavnost, povratne informacije, hrabrost i poštovanje.

Povratne informacije se dosta vrednuju u Ekstremnom programiranju i dosta praksi je vezano za njih kao što je na primer planiranje nove verzije ili sama iteracija. Programiranje u paru i testiranje se takođe odnose na povratne informacije, u ovom slučaju o kvalitetu koda. Kontinuirana integracija se odnosi na informacije o tome gde će aplikacija biti integrisana. Dnevno izbacivanje nove verzije na produkciju se odnose na to da li će kod zaista raditi u realnom okruženju koje će biti korišćeno od strane klijenata.

Glavni događaj u obe metode je sprint, koji sa sobom nosi još dva sporedna događaja: planiranje i pregled sprinta. Sprint u Skramu nije parametar koji pokazuje kada će nešto biti pušteno na produkciju, već se koristi za planiranje. Isto to važi i u XP metodi, sprint se koristi za planiranje. U Skramu dužina sprinta je obično jedan mesec i kraće, dok se u XP-u ne dozvoljava da sprint traje duže od jedne nedelje. XP zahteva da tim radi održivim tempom 40 sati nedeljno, ali Skram timovi koji koriste XP prakse prilagođavaju zadatke i broj korisničkih priča, tako da ih mogu uraditi za vreme jednog sprinta zato što u tom slučaju rade 40 sati po sprintu.

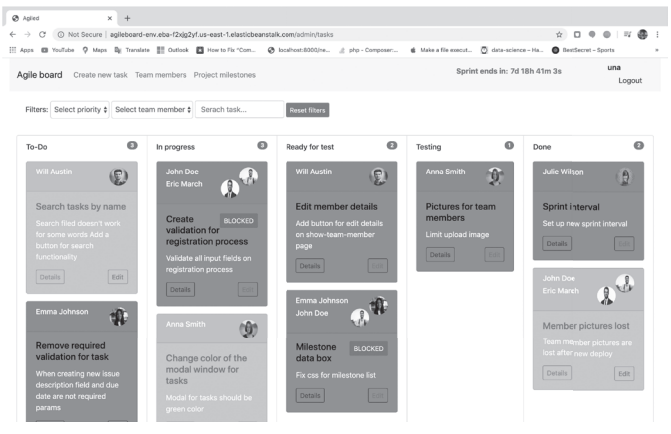
Pored sprinta, definisano je još nekoliko pojmova koji su od velikog značaja: beklog proizvoda, beklog sprinta, inkrement, definicija završenog, dnevni deploj na produkciju, programiranje u paru, desetominutni bild, razvoj vođen testovima, jedinstvenost koda i lokacija i raspored sedenja članova tima.

5. PREDSTAVLJANJE SOFTVERA ZA UPRAVLJANJE PROJEKTIMA KROZ VEB APLIKACIJU

Tehnologije koje su korišćene za razvoj serverskog dela aplikacije su PHP programski jezik, tačnije Laraver frejmvork i MySQL baza. Klijentski deo je urađen pomoću HTML5 i CSS3 tehnologija, koristeći Bootstrap klase, a dinamički sadržaj pomoću JavaScript biblioteke jQuery. Platforma na kojoj se nalazi aplikacija je Amazon AWS i celokupno postavljanje okruženja i konfiguracije je urađeno preko AWS Konzole.

Kao što je navedeno, jedna od najbitnijih prednosti bilo koje metode ili njihove kombinacije je vizualizacija rada koja se postiže korišćenjem table.

Ukoliko su članovi tima zajedno u prostoriji, obično se koriste lepljive sličice za prikaz i praćenje zadataka, ali ukoliko tome treba da pristupe i ostali ili ako su članovi tima fizički odvojeni onda je najbolje rešenje predstaviti tablu kroz aplikaciju. U tom slučaju članovi tima i sve interesne strane imaju mogućnost da u svakom trenutku pristupe aplikaciji i uvid u napredak i trenutno stanje projekta.

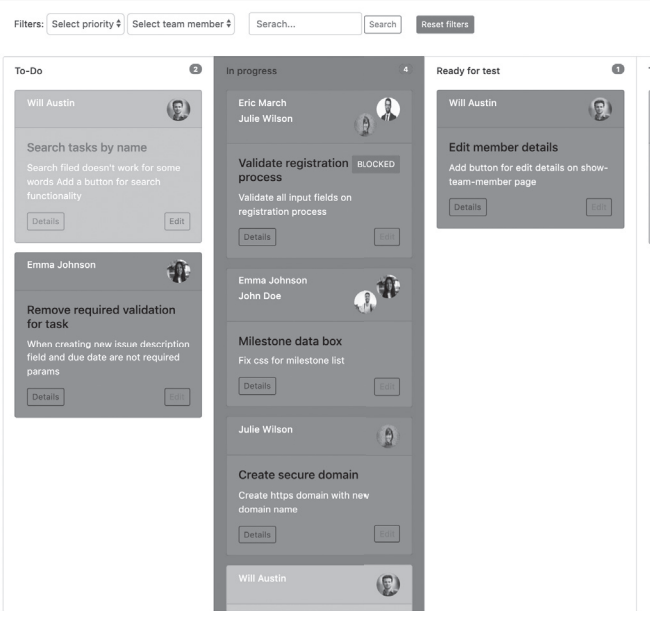


Slika 1 - *Agilna tabla*

Na Slici 3 prikazana je kreirana tabla koja prikazuje probleme i zadatke za jedan projekat. Jedna tabla se može koristiti za više projekata, kao što i jedan projekat može biti predstavljen na tabli koja može biti spoj Skram i Kanban table i ostalih agilnih metodologija. Na ovaj način se daje fleksibilnost kreiranju table u zavisnosti od potreba projekata.

Kreirano je pet kolona koje predstavljaju tok samog procesa. "*To-Do*" kolona predstavlja zadatke koji su iz bekloga izabrani i na kojima će članovi tima raditi. Kada se raspodele zadaci između članova tima i oni započnu svoj rad na njima, zadatak se prebacuje u kolonu "*In progress*". Ova faza se obično odnosi na sam razvoj proizvoda i podrazumeva rad programera ili operativca. Kada se razvoj završi zadatak se prebacuje u kolonu "*Ready for testing*" koja predstavlja početnu kolonu za članove tima koji testiraju kod. Kada krenu sa testiranjem, zadatak će prebaciti u kolonu "*Testing*" i tu će biti sve dok se testiranje ne završi. Ukoliko prilikom testiranja naiđu na neke probleme ili loše funkcionalnosti, zadatak će vratiti u kolonu "*To-Do*" i prebaciti ga na onoga koji taj problem treba da reši, bio to programer, menadžer ili bilo ko drugi. Ukoliko je testiranje prošlo kao što je očekivano i jedna celina je završena, zadatak se prebacuje u kolonu "*Done*" gde će stajati sve do puštanja na produkciju a zatim će se ukloniti sa table.

U veb aplikaciji je određeno da sprint traje dve nedelje, jer se smatra da projekat nije mnogo složen. Takođe iz Kanban metode je preuzet jedan od osnovnih principa a to je ograničenje WIP. To znači da će se mogućnost istovremenog rada na različitim zadacima ograničiti na 3 zadatka u okviru celog tima i odnosiće se uglavnom na članove tima koji razvijaju kod, jer će se njihova imena u najvećem slučaju i naći na zadacima iz ove kolone. Ovo pravilo se odnosi na zadatke koji su u koloni "*In progress*".

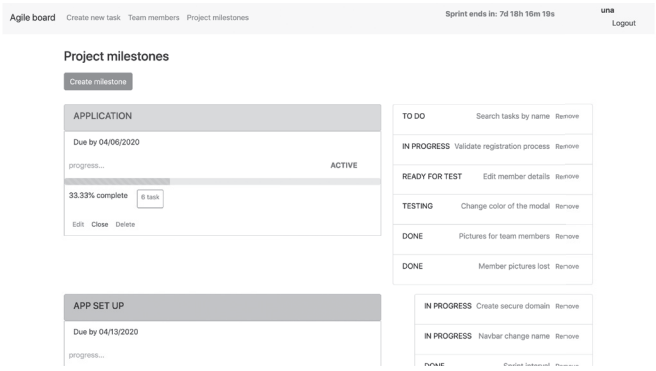


Slika 2 - *Označena "In progress" kolona sa WIP ograničenjem*

Majlstoun je pojam koji u projektu označava bitnu promenu, završenu celinu ili fazu u razvoju. Predstavlja bitnu komponentu za pravljenje plana i samog razvoja projekta. Svaka celina, odnosno ključni događaj koji je definisan predstavlja jedan majlstoun koji ima definisan krajnji rok.

Prilikom kreiranja majlstouna treba definisati početni i krajnji datum, odnosno vreme za koje će se zadaci iz majlstouna završiti i razviće se deo projekta koji je isporučiv i koji klijent može da pogleda. Kako to vreme direktno zavisi od zadataka koji su definisani, možemo reći da su glavna komponenta jednog majlstouna upravo zadaci. Moguće je prvo kreirati zadatke pa na osnovu njih odrediti majlstoune i dužinu trajanja, a moguće je i prvo kreirati majlstoun i rok pa onda u okviru njega zadatke koji trebaju biti završeni za to vreme. Koja će se od ovih metoda izabrati apsolutno je individualno i zavisi od zahteva i hitnosti projekta i njegovih komponenti.

Naravno, prilikom kreiranja majlstouna veoma je bitno odrediti cilj, odnosno definisati koji će deo projekta obuhvatati jednu celinu i biti završen, jer bez toga koncept majlstouna neće imati smisla. Nije nužno da se na kraju svakog majlstouna proizvod predstavi kupcu. To takođe zavisi od dogovora i cilja koji je definisan.



Slika 3 - *Majlstoun strana*

Koncepte ekstremnog programiranja koje će ovaj tim pratiti teže je predstaviti kroz aplikaciju. Neki će biti prikazani, a neki samo opisani uz pretpostavku njihovog korišćenja.

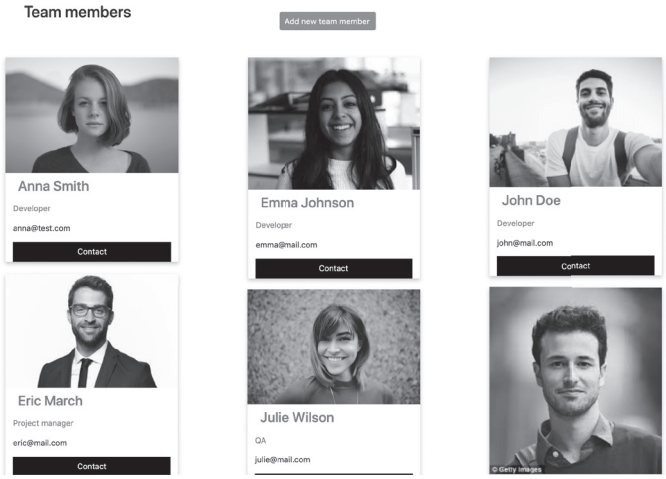
Jedna od glavnih vrednosti je ranije opisano programiranje u paru. Programiranje u paru je česta pojava kod timova koji se bave razvojem softvera i odnosi se na zajednički rad dva programera na istom računaru. Dok jedan radi i kuca kod, drugi ima ulogu da kontroliše. Na taj način se greške prilikom razvoja svode na minimum.

Još jedna vrednost XP metode je dostupnost zajedničkih direktorijuma i informacija svim članovima tima. Direktorijumi i kod su javni, tako da ih svako može pogledati, uključujući i strane interesne grupe, stejkholdere i vlasnike proizvoda, uloge koje ne utiču direktno na razvoj softvera.

Na strani 'TEAM MEMBER' prikazani su svi članovi tima i njihove uloge u timu. Od uloga definisane su:

1. Menadžer projekta – Umesto klasične uloge Skram mastera definisana je uloga menadžera koji je odgovoran za celokupnu organizaciju projekta i tima, a pored toga ima i tehničko znanje.
2. Vlasnik proizvoda - Vlasnik proizvoda je osoba unutar tima koja je u direktnoj vezi sa klijentom. Kao i ostali članovi tima, ima pristup aplikaciji i u svakom trenutku ima uvid u trenutno stanje projekta.

3. Razvojni tim - Sastoji se od nekoliko programera i nekoliko testera koji su u stalnoj komunikaciji, pomažu jedni drugima i rade zajedno na zadacima kako bi bili u korak sa zadatim rokovima i kako bi ostvarili dogovorene ciljeve.



Slika 4 - *Strana sa članovima tima*

Menadžer projekta ima najveće privilegije na aplikaciji i ima pristup svim stranicama i mogućnost da dodaje nove članove tima, kao i da menja i briše postojeće. Ostali članovi tima imaju ograničena prava i mogu vršiti izmene samo na svojim stranicama. Ova prava se odnose samo na deo za manipulisanje članovima tima, ostali delove aplikacije su u potpunosti dostupni svima, nezavisno od uloge u timu.

6. ZAKLJUČAK

Kod razvoja softvera veoma je važna uloga klijenta i stalna komunikacija sa klijentom, da bi se ispunili njegovi zahtevi i došlo do željenog krajnjeg rezultata – primenljivog softvera.

Predstavljanje i upoređivanje ovih metoda nikada se ne radi iz razloga kako bi se našla najbolja metoda, već kako bi se istražile razlike između njih i pronašli potencijalni rezoni koji govore o tome zašto i u kojim situacijama treba izabrati određenu metodu ili njihovu kombinaciju.

Ne postoji jedinstveni recept upravljanja i vođenja projekata zbog širokog spektara problema i zahteva koji se javljaju. Ne postoji najbolja metodologija koju treba koristiti. U zavisnosti od prirode problema odgovarajuća metodologiju će biti izabrana. Ukoliko projekat ima jasno definisane zahteve i ciljeve, koristiće se neku od tradicionalnih metodologija, ali za projekte sa nestabilnim zahtevima uvek će se koristiti neka od agilnih metodologije ili kombinacija, zato što su one prilagodljive.

7. LITERATURA

[1] Jovanović, A., Jovanović, F., Miletić, L., & Berić, I. (2016). Razvoj softvera primenom agilnih metodologija. U *Razvoj softvera primenom agilnih metodologija* (str. 896-899). Fakultet za projektni i inovacioni menadžment, Beograd.

[2] Milošević, D. (2016, Decembar 20). *Agilna metodologija*. Preuzeto sa Analitika i vizualizacija podataka: https://dusanmilosevic.com/agilna-metodologija/

[3] Schwaber, K., & Beedle, M. (2001). Agile Software Development with Scrum. U *Agile Software Development with Scrum* (str. 36-98). Prentice Hall PTRUpper Saddle River, NJUnited States.

[4] Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). U *Agile software development methods* (str. 18-36). Preuzeto sa https://www.vttresearch.com/sites/default/files/pdf/publications/2002/P478.pdf

[5] Schwaber, K., & Sutherland, J. (2013). *The Scrum Guide.* Preuzeto sa https://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf

[6] Womack, J., Jones, D., & Roos, D. (1990). The origins of Lean production. U *The Machine That Changed the World* (str. 19-46). Free Press.

[7] Beck, K., & Andres, C. (2004). U *Extreme Programming Explained: Embrace Change* (str. 17-51).

[8] Marić, M. (2015). XP. *Metodološko-radni okvir za razvoj softverske arhitekture poslovnog softvera u agilnim procesima*, 80-84.

[9] Rogers, P. (2017, August 30). *Better Together — XP and Scrum*. Preuzeto sa https://medium.com/agile-outside-the-box/better-together-xp-and-scrum-c69bf9bffcff

[10] *Extreme Programming (XP) vs Scrum*. (n.d.). Preuzeto sa Visual Paradigm: https://www.visual-paradigm.com/scrum/extreme-programming-vs-scrum/

dr Ognjen Pantelić, Vanredni profesor, Fakultet organizacionih nauka
Kontakt: ognjen.pantelic@fon.bg.ac.rs
Oblasti interesovanja: Informacioni sistemi, ERP sistemi, Logičko modelovanje IS

Una Nikolić, Master inženjer organizacionih nauka, Logispin
Kontakt: una9nikolic@gmail.com
Oblast interesovanja: Internet programiranje, objektno-orijentisano programiranje, Web servisi, Cloud platforme, razvoj web aplikacija

Stefan Krstović, Saradnik u nastavi, Fakultet organizacionih nauka
Kontakt: stefan.krstovic@fon.bg.ac.rs
Oblasti interesovanja: Informacioni sistemi, ERP sistemi, Process mining