Kvalifikacijski rad

UPRAVLJANJE PROJEKTIMA RAZVOJA SOFTVERA ZA POTPORU POSLOVNIM PROCESIMA TVRTKE

Perino Krneta
Email:perino.krneta@ris.hr

Sažetak

U ovom radu je dan pregled metodologija/metodika za upravljanje projektima kao i onih za projektiranje i razvoj informacijskih sustava. To su usko povezana znanja o projektima razvoja poslovnog softvera koja se međusobno preklapaju i nadopunjuju. Za uspješnu realizaciju jednog projekta razvoja softvera potrebno je pravilno koristiti i vladati znanjima o jednima i drugima.

Ključne riječi: upravljanje projektima, projektiranje informacijskih sustava, MIRIS, agilne metode

Uvod

Postoji niz pretpostavki za uspješno vođenje projekta razvoja IS kao što su korištenje metodologija projektiranja informacijskih sustava, metodologija upravljanja projektima, alata za razvoj programske podrške, SUBP, formiranje i upravljanje razvojnim timom i dr. U ovom je radu dan pregled stanja metodologija upravljanja projektima i pobrojane su i metodologije razvoja informacijskih sustava, s ciljem mogućeg objedinjavanja jednih i drugih u jednu jedinstvenu metodiku za razvoj i upravljanje razvojem poslovnog softvera. No prije nego što se nastavi s pregledom stanja, bit će definirani osnovni pojmovi.

Što je projekt? Projekt je zajednički posao koji poduzima pojedinac ili grupa ljudi koji za cilj ima proizvesti jedinstveni proizvod ili uslugu i ima ograničeno trajanje/1/. Kada se govori o projektima razvoja poslovnog softvera, može se dati i konkretnija definicija. Zajednički je posao u tom slučaju razvoj softvera (projektiranje, programiranje, testiranje …), jedinstveni proizvod ili usluga je poslovni softver (aplikacija), a ograničeno trajanje period je od najviše 16 mjeseci, projekti s dužim trajanjem u današnjem poslovnom okruženju su upitne korisnosti.

Što je softver? To je bilo koji skup instrukcija koji stroj može pročitati i uputiti procesor na izvršavanje neke specifične radnje/25/. Softver dijelimo na:

- **Sistemski softver** – on upravlja hardverom i predstavlja platformu za izvršavanje aplikativnog softvera. U njega ubrajamo operativne sustave, boot firmware, drivere za različite dijelove hardvera, i sl./25/
- **Programski softver** - je obično alat koji koriste programeri u davanju instrukcija kompjutoru koristeći neki programski jezik. Postoje razni alati za razvoj aplikativnog softvera, npr. Jdeveloper, Clarion, OracleForms, Apex, MS Visual studio i sl.
- **Aplikativni softver** – još se zove i korisnički softver, njega kompjutor koristi za obavljanje korisnih funkcija. U njega se ubrajaju razne desktop aplikacije kao što su web preglednici, MS office, poslovni softver, edukacijski softver, baze podataka, SUBP i sl. /25/, /26/. Ne može se samostalno koristiti, za njegovo pokretanje potreban je sistemski softver.

Što je poslovni softver? To je posebna vrsta aplikativnog softvera koji koristimo kao programsku potporu poslovnim procesima tvrtke.

Što je metodologija upravljanja projektima? Možemo ju definirati na sljedeće načine /21/:

- Proces koji dokumentira skup koraka i procedura kako bi projekt uspješno doveli do kraja.
- Definiranje procesa kako bi dostigli završetak projekta.
- Niz ili skup koraka kroz koje projekt napreduje.
- Skup metoda, procedura i standarda koji definiraju zajednički razvojni i menadžerski pristup s ciljem isporuke proizvoda, usluge ili rješenja.
- Integrirani skup zadataka, tehnika, alata, uloga i odgovornosti za isporuku projekta

Što je upravljanje projektima? Upravljanje projektima je primjena znanja, vještina, alata i tehnika u projektnim aktivnostima da bi se ispunili projektni zahtjevi /1/.

Različiti pristupi upravljanju projektima.

U klasifikacijama metodologija za upravljanje projekatima vlada i dosta velika zbrka, nešto se nekada zove metodologija, nešto okvir za rad (Fremework), nešto metoda ili tehnika, a nešto proces. Za potrebe ovog rada dat ću svoj pogled na podjelu

metodologija za upravljanje projektima. Načelno ih možemo sagledati sa dva osnovna pristupa, a to su tradicionalni pristup i moderni pristup. I jedan i drugi imaju svoje prednosti i mane, a razlikuju se i okolnosti u kojima su se pokazali boljima. Tradicionalni pristup koristi konvencionalne prokušane metode i generalno je bolji kada se govori o tipovima projekata tzv. „printing by numbers“, odnosno o većim projektima koji uključuju jasno definirane aktivnosti i faze, npr. građevinski projekti ili implementacija novog operativnog sustava u tvrtku. Njega karakteriziraju stroža pravila (za komuniciranje, dokumentiranje, izmjene plana,…), veća uloga project managera, a jednom kad se napravi plan projekta, on se uglavnom ne mijenja do kraja. Za razliku od njega, moderan ili agilni ili iterativni pristup bolji je za tzv. R&D projekte, odnosno projekte kod kojih nisu jasno definirani zahtjevi, ne zna se točno koji su rezultati projekta, a ti projekti kraće traju, npr. razvoj novog uređaja za komunikaciju sa samo jednom tipkom (iphone), ili razvoj nove platforme za društvene mreže, ili razvoj nove Internet platforme za isporuku SaaS aplikacija (Thin@). Moderan pristup karakterizira manje dokumentacije, manje formalna komunikacija, veći angažman krajnjeg korisnika, manji timovi, više komunikacije među članovima tima, veće dijeljenje znanja, tzv. učešće iteracije/2/, /3/, /4/, /5/, /6/ i /22/. Ovdje se, kao i u drugima granama poslovanja i života, previše često odbacuju neke tradicionalne, a odabiru moderne, samo zato jer su moderne. Kod jednog i drugog pristupa upravljanju projektima postoji niz metodologija koje su njegovi predstavnici, a ovdje će biti navedeni značajniji. Metodologije upravljanja projektima možemo podijeliti na sljedeći način:

- Tradicionalni (The traditional approach) ili fazni pristup ili „Metodologije teške kategorije“
  - Critical chain project management
  - Event chain methodology
  - PRINCE 2
- Moderni (Modern project management) ili iterativni pristup ili „Metodologije lake kategorije“
  - Agile project management
  - SCRUM
  - Lean project management
  - Extreme project management
  - Benefits realisation management

Tradicionalni ili fazni pristup

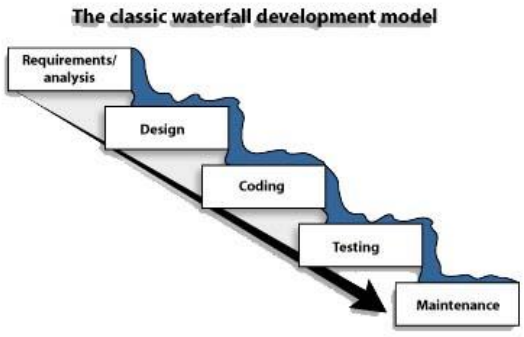
Tradicionalni ili fazni pristup identificira niz koraka/faza koje treba izvršiti kako bi projekt završili. Tradicionalni pristup obično podrazumijeva 5 faza:

- Faza iniciranja projekta
- Faza planiranja i razvoja
- Faza izvršavanja
- Faza nadzora i kontrole
- Faza zatvaranja

Ove faze se uglavnom koriste sekvencijalno, ali se mogu i preklapati. Različite industrije koriste različite varijante ovih projektnih faza. Npr. u razvoju softvera ovaj pristup je znan kao model vodopada. Model vodopada se pokazao korisnim za manje dobro definirane projekte (iako smatram da takvih nema), ali nije dobar za veće projekte kod kojih nisu jasno definirani funkcionalni zahtjevi softvera. Kako bi se razriješili ti problemi, kasnije su objavljene razne modificirane metode vodopada /3/.

Tradicionalni (čisti) model vodopada

Model vodopada (slika 1) je u softversku industriju preuzet iz proizvodnje i građevinarstva. Obje ove industrije imaju strukturirano fizičko okruženje i bilo kakve naknadne promjene nakon isporuke proizvoda su gotovo nemoguće. Tijekom početka industrije proizvodnje softvera, nisu postojale formalne metodologije razvoja softvera, stoga je preuzet ovaj model.



Slika 1. Model vodopada /30/

Model vodopada podrazumijeva sljedeće faze u razvoju softvera:

Analiza zahtjeva, razvoj (dizajn i programiranje), testiranje, verifikacija i održavanje. Ove faze su sekvencijalne, što znači da se tek nakon završetka jedne faze prelazi na drugu fazu. Odnosno nakon što softver prijeđe iz jedne u drugu fazu ne može se više vratiti u prethodnu. To je ujedno i najveća mana ovog modela, što je dovelo do razvoja modificiranih modela /7/, /3/.

Modificirani model vodopada

Kao što je već rečeno, nastanak ovog modela je posljedica nedostataka tradicionalnog modela vodopada. Faze su iste kao kod tradicionalnog modela, glavna razlika je ta da se faze mogu preklapati čime je model dobio na fleksibilnosti. To omogućava da se niz zadataka iz različitih faza izvršava istovremeno, što dalje omogućava da se greške u razvoju eliminiraju u fazi razvoja, a ne tek u fazi implementacije ili održavanja kad su troškovi za to ogromni. Isto tako to omogućava i promjene u dizajnu tijekom programiranja ili testiranja.

Prednost ovog modela je i smanjenje količine dokumentacije, što omogućava programerima više rada na programiranju.

Nedostatak ovog modela je otežana kontrola napretka jer postoji mogućnost da su u jednom trenutku „aktivne“ tri ili četiri faze. Pored toga, i dalje postoje neke zavisnosti među fazama koje su iste kao i kod tradicionalnog modela. Te zavisnosti mogu dodatno zakomplicirati projekt kod velikog preklapanja faza pa se može javiti potreba da se razvojni tim počne vraćati u prethodnu fazu kako bi je detaljno definirao, a to dovodi do značajnog utroška vremena i kašnjenja na projektu. To može biti riješeno na način da se postave mjerila završetka za svaku fazu Unatoč ovim nedostacima, ovaj model se široko koristi u industriji proizvodnje softvera/7/.

Critical chain project management

Critical Chain Project Management (CCPM) je razvio Dr Eli Goldratt i prvi put je predstavio u knjizi Theory of Constraints book “Critical Chain” 1997. g. Ova metoda je nastala kao odgovor na problem kašnjenja velikog broja projekata u odnosu na plan, pri čemu su imali veće troškove od planiranih i manje funkcionalnosti nego što je predviđeno /24/. Kod izrade projektnog rasporeda (project schedule) projekta najčešće korištene metode su CPM i PERT koje su u svojoj osnovi imale definiranje aktivnosti, definiranje redoslijeda izvođenja aktivnosti i određivanje trajanja aktivnosti. Upravo kod određivanja trajanja pojedine aktivnosti se u obzir uzimalo tzv. sigurnosno vrijeme koje je služilo kako bi projektni raspored (project schedule) bio što realističniji i ne bi „procurio“ tijekom izvođenja projekta. Takav način dobivanja projektnog rasporeda rada ima nekoliko manjkavosti koje su dovodile do kašnjenja projekata. Točnije, javljaju se sljedeća tri problema /24/:

- Zadatak/aktivnost ne izvršavamo sve dok krajnje vrijeme za njegov početak (Student Syndrome)
- Zadatke/aktivnosti izvršavamo na način da ih stignemo završiti do roka, iako ih možemo završiti i ranije (Parkinson’s Law)
- Biramo samo zadatke za čiji završetak ima dovoljno vremena tj. laganije zadatke (Cherry picking tasks)

Dakle, sigurnosno vrijeme koje smo uključili u izvršavanje aktivnosti je protraćeno. Pored toga, menadžment često forsira ljude da rade više zadataka odjednom (nekada iz potpuno nebitnih razloga). Na taj način ljudi skaču sa zadataka/aktivnosti ne druge i produžuju vrijeme koje je predviđeno planom: radeći tako treba im puno više vremena za izvršenje pojedine aktivnosti, nego da ih rade jednu po jednu. Pored toga, ljudi često ne žele prijaviti da su zadatak završili prije vremena jer im se idući sličan zadatak planira prema prethodnom iskustvu, znači skraćuje im se vrijeme planirano za buduće slične zadatke /24/.

CCPM pokušava riješiti ove probleme na sljedeći način:

- **Critical Chain** – Definira se Critical chain (CC), to je najdulji lanac (ne put, kao u CPM) zavisnih zadataka. U ovom slučaju zavisnost se odnosi na resurse, na dijeljenje resursa za istu aktivnost i na logičku zavisnost aktivnosti. To je glavna razlika u odnosu na CPM, gdje se gleda samo logička zavisnost aktivnosti.
- **Procjene vremena aktivnosti** – kako bi se smanjilo traćenje vremena povezanog s prevelikim sigurnosnim vremenom, CCPM predlaže smanjenje vremena trajanja aktivnosti za 50%.
- **Sigurnost** – CCPM koristi sigurnosne buffere kako bi upravljao nesigurnostima tijekom izvršavanja projekta. Sigurnosno vrijeme koje je bilo definirano za svaki zadatak, sad je dignuto na projektnu razinu. Postoje tri tipa sigurnosnih buffera koji osiguravaju sigurnost realizacije projekata:
  - **Projektni buffer** – vrijeme dodano na kraju između zadnjeg zadatka i datuma završetka projekta. Bilo koje kašnjenje u najdužem lancu će konzumirati nešto od tog buffera, ali će datum završetka ostati netaknut. Preporuka je da on bude polovica trajanja ukupnog sigurnosnog vremena svih aktivnosti. Na taj način ukupno trajanje projekta je na 75% od prvotno planiranog.
  - **Feeding buffer** – Kašnjenja na putovima koji ulaze u CC ili najduži lanac mogu prouzročiti kašnjenje projekta, i to tako što će kasniti neka od podaktivnosti u kritičnom lancu. Kako bi se to onemogućilo dodaju se feeding buffer-i

između zadnjeg zadatka na ulaznom putu i CC. Preporuka je da oni budu polovica vremena sigurnosnog vremena svih aktivnosti ulaznog puta.
    - **Resursni buffer** – Može se postaviti pored CC kako bi se osigurao dovoljan broj ljudi i vještina koji su potrebni za rad na CC.
- **Prioriteti** – Svim resursima na projektu su dani jasni i definirani prioriteti vezani uz održavanje CC, i to u ovisnosti o određenom bufferu, kao i o projektu u cijelosti. Resurs koji ima više od jednog otvorenog zadatka, prije nego završi bilo koji zadatak na putu koji ulazi u CC, bit će dodijeljen onom zadatku koji ugrožava CC.
- **Završetak** – Resursi se ohrabruju da aktivnosti završavaju na najbrži mogući način pritom ne ugrožavajući kvalitetu. Zadaci se ne ostavljaju napola dovršenim, kako bi se izbjegao mulitasking. Micanje sigurnosnog vremena „tjera“ resurse na predaniji rad i eliminaciju ranije spomenutih Studentskog sindroma i Parkinsovog zakona.
- **Upravljanje buffer-ima** – količina konzumacije svakog buffer-a u odnosu na trajanje projekta nam govori koliko kašnjenja utječu na rok. Ako su varijacije na projektu ravnomjerno raspoređene, onda će i konzumacija buffer-a biti linearna. Rezultat će biti projekt završen uz iskorišteni cijeli buffer. Ukoliko je konzumacija buffer-a veća nego napredak projekta, onda PM mora poduzeti korektivne mjere.
- **Preostalo vrijeme** – Aktivnosti se kontroliraju u odnosu na vrijeme koje je potrebno za završetak (koliko dana do završetka zadatka), a ne na postotak završenosti. Na taj se način kontroliraju buffer-i i njihova iskorištenost /24/.

Event chain methodology

Nadopunjuje CPM i CCPM. To je tehnika modeliranja nesigurnosti i mrežnog planiranja koja se fokusira na identificiranje u upravljanje događajima koji mogu utjecati na projektne rokove. Pomaže umanjiti negativni utjecaj raznih događaja na izvršavanje projektnih aktivnosti i omogućava lagano modeliranje nesigurnosti u mrežnom planu /11/. Bazira se na sljedećim principima /12/:

- Vjerojatnost pojave rizika u određenom trenutku izvršavanja pojedine aktivnosti
- Lanac događaja (Event Chain). Neki događaj može inicirati neki drugi događaj, ovaj pak neki novi, itd., što dovodi do lanca događaja koji mogu značajno utjecati na tijek izvršavanja projekta.
- Kritični događaj ili lanac događaja. Događaj ili lanac događaja koji imaju najveći utjecaj na projekt se određuje analizom.
- Praćenje napretka projekta preko događaja. Uključivanje analize događaju u izvještaje o napretku projekta.
- Vizualizacija lanca događaja putem dijagrama.

PRINCE 2

Akronim za PRojects IN Controlled Environments, verzija 2. Razvijen je od strane vlade Ujedinjenog kraljevstva i predstavlja standard za upravljanje javnim projektima u UK. PRINCE2 je nastao iz ranije metode pod nazivnom PROMPTII i iz PRINCE metode upravljanja projektima koja je inicijalno razvijena 1989. od strane Central Computer and Telecommunications Agency (CCTA) kao standard vlade UK za upravljanje IT projektima /8/.

PRINCE2 je bazirana na 7 principa, 7 tema i 7 procesa. Principi su kontinuirana poslovna opravdanost projekta, učenje iz iskustva, jasno definirane uloge i odgovornosti, upravljanje po fazama, upravljanje prema izuzecima, fokusiranje na proizvod i prilagođavanje projektnom okruženju. Sedam tema su: business case, organizacija, kvaliteta, plan, rizik, promjena i napredak.. Principi i teme se realiziraju kroz procese. Procesi su /8/:

- Započinjanje projekta (Starting up a project - SU)
- Inicijacija projekta (Initiating a project - IP)
- Upravljanje projektom (Directing a project - DP)
- Kontrola faza (Controlling a stage - CS)
- Kontrola dosega faza (Managing stage boundaries - SB)
- Upravljanje isporukama proizvoda (Managing product delivery - MP)
- Zatvaranje projekta (Closing a project - CP)

Moderni ili iterativni pristup

Glavni predstavnik modernog pristupa su tzv. agilne metodologije. S ciljem boljeg razumijevanja osnova modernog pristupa treba opisati kako je došlo do pojave tih metodologija. Ukratko, one su nastale prirodnom potrebom za povećavanjem uspješnosti projekata. Naime, veliki broj projekata kojima je upravljano na tradicionalan način, imao je problema. Poznato istraživanje Standish Group CHAOS Studies /27/ je pokazalo da mnogi IT projekti ne uspijevaju zadovoljiti rokove projekata uz predviđeni trošak, a često ne uspijevaju ostvariti očekivane koristi. Ove probleme su potvrdile mnoge organizacije, npr.

Američko ministarstvo obrane DoD je imalo 75% projekata razvoja softvera koji nisu nikada zaživjeli ili su prekinuli razvoj prije kraja projekta. I druga znanstvena istraživanja su dovela u pitanje tradicionalne metode upravljanja projektima razvoja softvera. 1998. godine Harvard Business School academics Robert D. Austin and Richard L. Nolan su istraživali velike softverske projekte. Njihovo istraživanje/20/ koje je ispitivalo mnoge temeljne ideje upravljanja projektima razvoja softvera, dovelo je do sljedećih zaključaka:

- Prva pogrešna pretpostavka kod upravljanja projektima razvoja softvera je ta da je moguće planirati veliki projekt razvoja softvera.
- Druga pogrešna pretpostavka je ta, da je moguće zaštititi se od kasnijih izmjena (naknadnih zahtjeva)
- Treća pogrešna pretpostavka je ta da uopće ima smisla „zaključati“ veliki projekt rano, već u fazi planiranja.

Watts Humphrey, ugledni istraživač iz IBM-a popratio je ovu studiju sa člankom/28/ u kojem je naveo njegov „Princip nesigurnosti zahtjeva“ (Requirements Uncertainty Principle), koji kaže slijedeće: „za novi softver zahtjevi neće biti do kraja poznati sve dok ga korisnik ne počne koristiti“ . Hadar Ziv sa sveučilišta u Kaliforniji (University of California) je nedugo zatim dao svoj princip nesigurnosti u softverskom inženjerstvu (Uncertainty Principle in Software Engineering), koji kaže: „Nesigurnost je nerazdvojno i neizbježno vezana uz proces razvoja softvera i softverski proizvod“ /31/

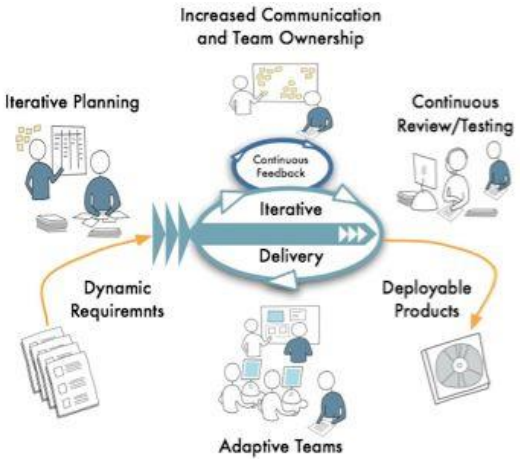
Veza između ovih ideja i koncepata agilnog pristupa razvoju projekta je očigledna. Ako korisnici ne mogu reći što žele, dok to ne vide, ako predviđanje i planiranje velikih IT projekata nije moguće i ako se ne možemo zaštititi od zahtjeva za izmjenom tijekom faze razvoja softvera, tada su koncepti i ideje koje stoje iza tradicionalne metode vodopada očigledno neupotrebljive. Dalje se može zaključiti da će metode koje se baziraju na inkrementalnom razvoju i prototipskom pristupu donijeti značajne koristi. Godine 2001. Alan MacCormack objavio je članak pod nazivom Evolutionary Model of Software Development Methods, /32/ koji se bazira na istraživanju koje je proveo s još dvojicom kolega na 29 završenih projekata razvoja softvera iz 17 različitih tvrtki. U članku konstatira problem postojećih metoda upravljanja projektima razvoja softvera, pored toga ponudio je četiri smjernice (dobre prakse) koji bi bili alternativa ili nadopuna tradicionalnih metoda. Prema njemu uspješnost projekta povećavaju:

- Rano izbacivanje dizajna proizvoda prema korisniku
- Dnevno dodavanje novog programskog koda i brza povratna informacija
- Iskusan i vješt razvojni tim
- Veće ulaganje u razvoj programske infrastrukture.

Nakon svega ovoga Agile manifesto /29/je bio samo kulminacija ovih novih teorija. Napisala ga je grupa zagovornika metoda inkrementalnog razvoja softvera, postao je temeljni dokument agile pokreta i utvrdio temeljne koncepte agilnog upravljanja projektima. Među potpisnicima su bili neki od utemeljitelja poznatih agilnih metodologija: Singer Kent Beck je razvio Extreme programming, Alistar Cockburn je jedan od razvijača Crystal Methods, a Jim Highsmith je preveo agile koncepte razvoja softvera u agile projektnu metodologiju /20/, /21/. Nabrojimo sada najznačajnije predstavnike modernog pristupa.

Agile project management

To je iterativna i inkrementalna metoda za dizajn i izgradnju projektnih aktivnosti u informacijskoj tehnologiji, tehničkim projektima, projektima za proizvodnju novog proizvoda ili usluge i vrlo fleksibilnim projektima. Shema agilnog procesa dana



je na slici 2.

Slika 2. Agilni proces /17/

Najbolji učinak ima kod projekata koji su previše kompleksni za specifikaciju funkcionalnosti prije samog testiranja prototipa. Agile je nastao kao odgovor kontinuiranim i konstantnim zahtjevima za promjenama tijekom projekata razvoja softvera. Umjesto provođenja tjedana ili mjeseci u definiranju detaljne specifikacije zahtjeva korisnika za cijeli projekt i nakon toga još duže na razvoju proizvoda, pa

na testiranju tijekom kojeg nalazi stotine grešaka, Agile specificira manje segmente koji se mogu samostalno koristiti (inkrementi), razvija ih i testira u ciklusima od dva do četiri tjedna /18/. Agile je danas jedan od najpopularnijih pristupa u upravljanju projekata razvoja softvera. Agile project management je nastao iz Agilnih metodologija razvoja softvera, a definirao ju je jedan od potpisnika Agile manifesta Jim Highsmith. Kod Agile upravljanja projektima cijeli tim je odgovoran za upravljanje timom, a ne samo voditelj projekta. Kad se radi o procesima i procedurama, često je zdrava logika ispred pisanih politika i procedura. Na taj se način osigurava nesmetan tijek projekta, tj. brže donošenje odluka. Pored toga što je menadžer, agile PM mora pokazivati osobine vođe i motivatora, tako se podiže moral tijekom projekta i održava disciplinu. Agile PM nije „šef" razvojnog tima, nego koordinator aktivnosti i resursa potrebnih za brzi i kvalitetni razvoj softvera. Možemo reći da je agile PM mentor i zaštitnik projektnog tima prije nego menadžer /19/.

SCRUM

Od svih agilnih metodologija SCRUM je jedinstvena po empirijskom procesu kontrole. Scrum koristi stvarnu procjenu napretka projekta, ne koristi prognoze napretka projekta ili optimistične procjene. Kod Scrum-a projekti su podijeljeni u kratke radne cikluse koji se zovu sprintovi (sprints), koji obično traju jedan, dva ili tri tjedna. Na kraju svakog sprinta, steakholderi i članovi tima se sastaju i procjenjuju napredak projekta i planiraju slijedeći sprint. Na taj način smjer projekta se kontinuirano oslanja i određuje u odnosu na odrađeni posao, a ne spekulacije i predviđanja. Nije teško pretpostaviti da je zbog ovoga ova metodologija popularna kod project managera i developera, a bilo bi zanimljivo saznati koliko je popularna kod project sponzora ili višeg menadžmenta. Scrum ima samo tri osnovne projektne role Vlasnik proizvoda (Product owner), Scrum gospodar (Scrum master) i Član tima (team member) /23/.

- **Vlasnik proizvoda**: On je odgovoran za prijenos vizije proizvoda prema timu, predstavlja interes kupca tako da definira zahtjeve na proizvod i prioritete. To je uloga sa najvećom odgovornošću, a samim tim ima i najveći autoritet. On je odgovoran ako projekt propadne, ili ako ne ispunjava očekivanja kupca. Njegov najveći problem je naći pravu razinu involviranosti u projekt, s jedne strane ima veliku odgovornost i autoritet, s druge ne smije se previše petljati u posao jer onda cijeli pristup baziran na samostalnosti članova tima pada u vodu.
- **Scrum gospodar skupa**: On je moderator između vlasnika proizvoda i članova tima. On ne upravlja timom, on otklanja zapreke koje onemogućuju timu u dostizanju ciljeva sprinta. Ukratko, pomaže timu ostati kreativan i produktivan i ujedno osigurava da se odrađeni posao pravilno i transparentno prezentira vlasniku proizvoda.
- **Član tima**: Odgovorni za završetak posla. Prema nekim preporukama idealan broj članova tima je 7+/- 2 različitih stručnjaka. Npr. kod softverskih projekata tipičan tim bi sadržavao programere, projektante, analitičare, testere, UI dizajnere, administratore baze podataka. Tim određuje kako realizirati sprint što im daje dosta veliku autonomiju i slobodu u razvoju, ali samim time i odgovornost za ispunjavanje ciljeva sprinta /23/.

Lean project management

Svodi se na primjenu lean koncepata (construction, manufacturing, management) u projektni menadžment. Lean management je sustav upravljanja poslovanjem pomoću definiranih principa, dobre poslovne prakse i alata za proizvodnju kvalitetnijih usluga i proizvoda sa što manje grešaka, koristeći manje rada, prostora, kapitala i vremena /14/. To ukratko znači proizvodnja sa što manje „otpada (škarta)", odnosno zadržavanje samo onih koraka (aktivnosti) u proizvodnim i drugim procesima koji dodaju vrijednost, a to su: one koje je kupac spreman ih platiti, one koje mijenjaju proizvod ili mu dodaju nužne informacije, one koje su pravno ili ugovorno obvezujuće /14/. Sukladno navedenom, Lean project management ima za cilj stvarati više vrijednosti sa što manje otpada u projektnom smislu /15/. Lean je nastao (odnosno, prvi je puta značajnije implementiran) u Japanu 60-ih godina prošlog stoljeća u tvornici Toyota, kao odgovor na specifičnu situaciju (plaćanje unaprijed, manjak prostora i kvalificirane radne snage) u kojoj se proizvođači automobila u Japanu našli nakon 2. svjetskog rata /13/. Poznat je pod imenom Toyota Production Sistem (TPS). Nakon što su se TPS principi (Toyota je imala 14 principa) pokazali uspješnim u proizvodnoj industriju, proširili su se i u druge grane gospodarstva /13/. U knjizi Lean Thinking /citat/ James Womack i Daniel Jones su definirali 5 ključnih Lean principa: Identificiraj vrijednosti (Identify Value), Napravi mapu toka vrijednosti (Map the Value Stream), Kreiraj tijek (Create flow), Utvrdi napor (Establish Pull), Traži izvrsnost (Seek Perfection). Ako ih primijenimo na upravljanje projektima to bi značilo /15/:

- **Identify Value** – Pažljivo razlomimo projekt kako bi odredili koji dijelovi projekta su nebitni i mogu se eliminirati.

- **Map the Value Stream** – Analizirajmo tim/timove i pažljivo odredimo tijek projekta kako bismo vidjeli koji su nam timovi neophodni za projekt i u koje vrijeme, na taj način optimiziramo prijenos posla i smanjujemo mogućnost stvaranja uskog grla.
- **Create Flow** – Razlomimo projekt u manje zadatke (aktivnosti) kojima je lakše upravljati. Mjerimo performanse, promatrajmo kako se tim ili pojedinci snalaze u pojedinim situacijama i u kojima su bolji, a u kojima su lošiji kako bismo im mogli davati zadatke u kojima su najbolji.
- **Establish Pull** – Prije nego se odlučimo na koji ćemo način realizirati aktivnosti i koje će rezultate projekt proizvesti, verificirajmo ih sa projektnim sponzorima. Bolje je kasnije krenuti i onda odraditi sve, nego nekoliko puta počinjati i nakon toga se vraćati. Ukoliko je moguće radimo samo ono za što smo sigurni da treba napraviti. (Decide late, deliver fast ili poznatije tri puta mjeri jednom sijeci).
- **Seek Perfection** – Dajmo timu mogućnost odlučivanja, dajmo im više slobode u odlučivanju, ali i odgovornosti za njihove odluke. Kontinuiranom i jasnom komunikacijom s članovima tima promovirajmo važnost kontinuiranog učenja i poboljšanja.

Extreme project management

Extreme project managment (XPM) je metoda upravljanja vrlo složenim i nesigurnim projektima. Kod XPM fokus upravljanja projektima je na ljudima, a ne na metodama i tehnikama izrade plana aktivnosti i upravljanju realizacijom plana aktivnosti. Kod ovog pristupa nema jasno definiranih projektnih faza, ne postoje jasne smjernice kako realizirati pojedinu projektnu aktivnost. Kod XPM-a se prilagođavamo projektnoj aktivnosti i izvršavamo je na najbolji mogući način. Nema dugačkih rokova, definirani rokovi su vrlo kratki, često i kraći od 2 tjedna. Članovi tima imaju veliku slobodu u donošenju odluka kako realizirati pojedinu aktivnost, ali zato nose i odgovornost za rokove i kvalitetu te aktivnosti. Iz svega proizlazi izuzetno veliki utjecaj ljudskog faktora kod ovakvog načina upravljanja projektima. Članovi projektnog tima kod XPM-a imaju potpuno drugačije uloge i odgovornosti nego kod tradicionalnog pristupa, tako de je glavni izazov projektnog menadžera kod XPM-a promjena načina na koji članovi tima razmišljaju. U tom smislu za PM-a je važna promocija određenih vrijednosti u promjeni sustava razmišljanja članova tima. To su /16/:

- Normalno je da su zahtjevi i projektne aktivnosti kaotični.
- Nesigurnost je jedina sigurna karakteristika XPM-a.
- Takve tipove projekta nije moguće u potpunosti kontrolirati.
- Promjenu treba prihvatiti, ne joj se suprotstavljati.
- Osjećaj sigurnosti se povećava s popuštanjem kontrole projekta.

Benefits realisation management

Benefits realisation management je projektni pristup koji se bavi koristima koju projekt ima za organizaciju u kojoj se realizira. Prema nekim autorima ovo nije dio projektnog menadžmenta, ali svakako ga treba istaknuti kao jedan od novijih pristupa upravljanju projektima. Nastao je iz razloga što su mnogi projekti proglašeni uspješnima, ali njihov učinak na okolinu je bio minimalan. Prema ovom pristupu jedna od uloga PM-a tijekom realizacije projekta je i učinkovitost projekta, a ne samo rokovi i rezultati projekta. To znači da tijekom projekta PM tijesno surađuje sa korisnikom projekta kako bi osigurao da se proizvod ili usluga koju projekt proizvodi snažno ugradi i prihvati od strane krajnjeg korisnika. Neke od projektnih aktivnosti koje tome doprinose su /17/:

- Sudjelovanje u prezentacijama i demonstracijama proizvoda ili usluge.
- Održavanje radionica i obuka.
- Priprema marketinškog materijala.
- Organiziranje lansiranja proizvoda.
- Organiziranje i vođenje sastanaka.
- Traženje kreativnih rješenja za probleme korisnika.
- Traženje uzroka.
- Poticanje promjena .

Kako bi se postigli učinci, moramo prije toga nešto mijenjati. Glavno načelo Benefit Realisation pristupa je: učinak dolazi samo sa promjenom i promjena mora biti podržana sa učinkom. Ako nema učinka, onda nema smisla ni promjena. Kako bi se implementirao Benefit realization pristup, ljudi moraju promijeniti način razmišljanja, rada i upravljanja, što nije nimalo lagan zadatak. Međutim, bez toga naš projekt se vrlo lako može pridružiti dugačkoj listi „uspješnih“ projekata čiji rezultati nikada nisu ostvarili planirani učinak. Glavna poruka ovog pristupa je: Ne dopustite da se vaši projekti realiziraju i „umru“, razmatrajte učinke na početku projekta i osigurajte ih na kraju projekta /17/.

Metodologije projektiranja IS-a

Metodologije projektiranja IS su još jedan element kojeg trebamo za uspješan razvoj softvera. One su danas u manjim informatičkim tvrtkama, ali i ne samo njima, prečesto zanemarene. Razvoj softvera se uglavnom radi bez projekta, korisnički zahtjevi se ne pretvaraju u modele procesa i podataka, nego direktno u programski kod. To je ujedno i jedan od razloga manje uspješnosti projekata razvoja softvera. Neke od značajnijih metodologija projektiranja informacijskih sustava koje se danas koriste (ili ne koriste.) su: SSADM (Structured systems analysis and design method), Oracle CASE Method, MIRIS (MetodIka za Razvoj Informacijskih Sustava), Information Requirement Analysis/Soft systems methodology.

Zaključak

Možemo zaključiti da danas postoji veliki broj metodologija upravljanja softvera i metodologija za projektiranje informacijskih sustava. Tradicionalne (teške - velike) metodologije su dosta složene, dugo se uče i teško ih je primjenjivati jer su dosta općenite i treba dosta prakse u njihovoj primjeni kako bi ih se savladalo. Nasuprot njima, moderne agilne metodologije se prikazuju kao rješenje svih problema upravljanja projektima. Za razliku od tradicionalnih, vrlo su primamljive jer nisu tako opsežne, ne zahtijevaju puno dokumentiranja, obećavaju rezultate bez puno učenja. Međutim, ni one ne donose željene rezultate.

Vrlo je česta situacija da svatko iz dostupnih metodologija izvuče neku svoju derivaciju koja je više ili manje uspješna i na takav način radi.

Gdje je onda rješenje problema, ako ga uopće ima? Da li je možda rješenje u pravilnoj kombinaciji metodologija projektiranja i razvoja s metodologija upravljanja projektima. Naime, svaka od metodologija se samostalno poučava i predstavlja kao jedino rješenje problema, je li to ispravno?

Potrebno je ljudima/softverskim tvrtkama koje se bave proizvodnjom aplikativnog softvera dati konkretan alat za izradu softvera, pomoću kojeg bi svoje projekte uspješno završavali.

Literatura:

1) A Guide to the Project Management Body of Knowledge (PMBOK® Guide)—Fifth Edition, Project Management Institute, 2013
2) http://www.virtualprojectconsulting.com/pmmethodologies/
3) http://en.wikipedia.org/wiki/Project_management#Approaches
4) http://www.changedesigns.net/public/team/leading_teams/project-management-standards.html
5) Scott Berkun, The Art of Project Management, O'Reilly, 2005
6) Susan Snedaker, How to Cheat at IT Project Management, Syngress, 2005
7) Bhakti Satalkar , http://www.buzzle.com/articles/modified-waterfall-model.html, 2010
8) http://en.wikipedia.org/wiki/PRINCE2.
9) http://en.wikipedia.org/wiki/Critical_chain_project_management
10) http://pinnacle-strategies.com/critical-chain.html
11) http://en.wikipedia.org/wiki/Event_chain_methodology
12) http://www.sixsigmaonline.org/six-sigma-training-certification-information/project-management-%E2%80%93-event-chain-methodology.html
13) http://www.intergraph.com/assets/pdf/LeanConstructionWhitePaper.pdf
14) http://www.sqs.com/de-group/_download/White_Paper_Lean_PM_EN.pdf
15) http://blog.backbase.com/2935/going-with-the-flow-the-lean-approach-to-successful-project-management/
16) http://www.tutorialspoint.com/management_concepts/extreme_project_management.htm
17) http://www.projectsmart.co.uk/what-is-benefits-realisation.html
18) http://www.versionone.com/agile-project-management/
19) http://www.tutorialspoint.com/management_concepts/agile_project_management.htm
20) http://www.techrepublic.com/blog/tech-decision-maker/the-roots-of-agile-project-management/
21) http://hbswk.hbs.edu/item/2201.html
22) Modern Project Management: Essential Skills and Techniques, Iman Attarzadeh, Siew Hock Ow, Department of Software Engineering, Faculty of Computer Science & Information Technology, University of Malaya.
23) http://scrummethodology.com/
24) http://www.goldratt.co.uk/resources/critical_chain/
25) http://en.wikipedia.org/wiki/Software
26) http://bs.wikipedia.org/wiki/Softver
27) http://www.standishgroup.com/services.php

28) http://www.sei.cmu.edu/news-at-sei/columns/watts_new/2003/1q03/watts-new-1q03.htm
29) http://agilemanifesto.org/
30) http://rootsitservices.com/CustomPages/sdlifecycle.aspx
31) http://www.ics.uci.edu/~ziv/papers/icse97.ps
32) http://hbswk.hbs.edu/item/2201.html