EKONOMSKE PARADIGME RAZVOJA SOFTVERA

ECONOMIC PARADIGM OF SOFTWARE DEVELOPMENT

Radomir Brzaković[1)], mr Zoran Marjanović[2)]

Rezime: *Iako većina kompanija ne proizvodi softver on je neophodan za njihovo poslovanje. Imajući u vidu ovu činjenicu sa jedne strane i razvoj softvera,kao najdinamičniju industriju, sa druge ovaj rad predstavlja pokušaj sagledavanja ekonomskih paradigmi u softverskoj industriji. Razmotrene su paradigme različitih poslovnih modela, odnosno različitih modela razvoja i upotrebe softvera. U radu su prikazane prednosti i nedostaci razmatranih modela kroz određeni broj najbitnijih parametara.*
Ključne reči: *razvoj softvera, softverska industrija, ekonomska paradigma, poslovni modeli*

Abstract: *Although most companies do not produce software it is necessary for their business. Taking into account this fact on the one hand and software development, as the most dynamic industry, on the other this paper an attempt of viewing the economic paradigm in the software industry. Considered the paradigm of different business models, or different models of development and use of software The paper presents the advantages and disadvantages of the model considered by a number of the most important parameters.*
Key words: *software development, software industry, economic paradigm, business models*

1. UVOD

Softverska industrija je mlada. Za nešto više od 60 godina koliko postoji, postala je jedna od najvećih industrija na svetu, kao i jedna od najdinamičnijih. Prihodi koji se globalno ostvaruju u ovoj industriji su veći od $800 milijardi godišnje. Poslovni model predstavlja okvir ostvarivanja vrednosti. Ovaj termin se koristi za široki opseg neformalnih i formalnih opisa koje koriste kompanije za prikaz različitih aspekata svog poslovanja, uključujući ciljeve, ponudu, strategije, infrastrukturu, organizacione strukture, prakse trgovanja i operativne procese i politike.

Informacione tehnologije su promenile način poslovanja svih industrija. Šta više, i dalje ga menjaju, iz dana u dan. Poslovni procesi postaju efikasniji i sadržajniji, kao i komunikacija, deljenje informacija i zabava. Dinamična oblast IT-a se menja i razvija iz dana u dan, a svojim napretkom pokreće i sve ostale sektore ljudskog života i rada. Posledično, informacione tehnologije menjaju i same sebe, unapredjuju i svoju industriju iz dana u dan. Novi poslovni modeli se bez sumnje primećuju pri svakoj značajnoj promeni u svetu računarstva - kao primere možemo navesti pojavu personalnih računara, interneta ili razvoja otvorenog koda.

Poslovni modeli u softverskoj industriji su još specifičniji zbog same prirode softvera, koji mnogi nazivaju „tek za nijansu različitim od čiste ideje“. Načini na koje se danas prihodi ostvaruju od softvera kreću se u širokom rasponu od modela baziranih na prodaji proizvoda, preko onih koji na tržište plasiraju softverske usluge, do originalnih kombinacija sa drugim industrijama kakve su advertajzing ili proizvodnja hardvera. Sve poslovne modele i tokove ostvarivanja prihoda u ovoj oblasti je teško izolovati i klasifikovati, kao što je teško odabrati kompanije koje predstavljaju najboljeg predstavnika konkretnog modela.

Da bi kompanija u ovako dinamičnom sistemu opstala, neophodno je odlično poznavanje trenutnog stanja nauke (tzv. state-of-the-art) u oblasti informacionih tehnologija. Ipak, ovo često nije dovoljno, već je potrebno poznavati istoriju i trenutno stanje same industrije. Mnoge kompanije vode računa o tome kada je trenutak da promene poslovni model, da krenu u osvajanje novog segmenta tržišta ili odustanu od prevaziđenog.

2. EKONOMSKE PARADIGME RAZVOJA SOFTVERA

2.1. Ekonomska funkcija softvera u poslovanju

Važno je pomenuti da većina kompanija nije u poslu proizvodnje softvera. One prodaju cipele, automobile, usluge, ili bilo šta drugo. Ipak, svim, pa čak i najmanjim kompanijama je neophodan

1) Radomir Brzaković, dipl. inf., Fiat automobili Srbija, Kragujevac, email: brzijax@yahoo.com
2) mr Zoran Marjanović, dipl. ing., Zastava automobili, Kragujevac, email: z.marjanovic74@yahoo.com

softver da bi radile. Bez njega bi njihovo poslovanje bilo manje efikasno ili nemoguće. Samo treba pokušati zamisliti finansijsko planiranje bez spreadsheet-ova ili korespodenciju bez email-a. Danas nam je softver potreban u poslovanju u toj meri da čak iako većina firmi ne prodaje softver, svaka firma sa više od 50 ljudi će verovatno zaposliti programera, web dizajnera ili sistem administartora za skript programiranje. Za takve poslove, softver je esencijalna operativna tehnologija, a ne njihov proizvod.

Operativna tehnologija je esencijalna za poslovanje, ali to nije ono što ta firma prodaje. Na primer, ako kompanija prodaje knjige, knjige su izvor profita, a softver je trošak, neizbežan trošak poslovanja. Postoje dve osnovne vrste operativne tehnologije koja je trošak: diferencirajuća i nediferencirajuća. Diferencirajuća je ona tehnologija koja proizvod konkretne kompanije čini privlačnijim za potrošača u odnosu na proizvode konkurenta. Npr, ako posetimo web sajt Amazon.com i tražimo knjigu, Amazon će nam takođe reći koje su još knjige kupili ljudi koji su gledali knjigu koju mi trenutno gledamo. I vrlo često, knjige koje se tu navode jesu dovoljno zanimljive da i mi poželimo da ih kupimo. Na sajtu Barnes and Noble ne postoji ova opcija, pa i nije iznenađenje što Amazon prodaje više knjiga online. Tako, za Amazon, softver za preporučivanje jeste diferencijator poslovanja. Očigledno, bilo bi greška otvoriti kod poslovnih diferencijatora, zato što bi onda konkurenti mogli da ih iskoriste da bi postali podjednako poželjni potrošaču.

Sa druge strane, našem poslovanju ne bi smetalo kada bi konkurenti videli kako radi svaki deo našeg ne-diferencirajućeg softvera. Naprotiv, konkurent bi u tom delu mogao da bude najbolji kolaborator (ako je partnerstvo ograničeno na rad na ne-diferencirajućem softveru), zato što su njihove potrebe najsličnije vašim. Ova činjenica se demonstrira svaki dan u svetu otvorenog koda, u kome su HP i IBM partneri u razvoju softvera koji pomaže prodaju sistema oba snabdevača, dok ostaju ljuti konkurenti na višim niovima softvera gde je diferencijacija izmedju njih moguća i efektivna.

Možda 90% softvera u bilo kom poslovanju je ne-diferencirajući. Većina tog softvera se naziva i smatra infrastrukturom, odnosno bazom na kojoj se gradi diferencirajuća tehnologija. U kategoriju infrastrukture spada operativni sistem, web serveri, baze podataka, serveri za Java aplikacije i ostali middleware, grafičke radne površine korisničkog interfejsa i generalni alati za GUI radne površine kao što su web čitači, email klijenti, spreadsheet procesori i prezentacione aplikacije. Bilo koji softver koji pruža diferencirajuću vrednost nesoftverskoj kompaniji se gradi preko jedne ili više ovih infrastrukturnih komponenti.

Važan indikator toga da li je softver diferencirajući ili ne je da li konkurent može doći do istog softvera. Ni Windows ni Linux ne mogu pomoći da se kompanija diferencira jer su dostupni svima. Oni se razlikuju među sobom, ali oni ne mogu biti prednost nekog drugog biznisa. Jedan ili drugi može uštedeti nešto novca ili firmu učiniti efikasnijom, ali generalno ne čine poslovanje privlačnijim potrošačima.

Drugi važan indikator da li je softver diferencirajući je da li potrošač može da vidi efekte tog softvera. Potrošača ne interesuje koji operativni sistem kompanija koristi, osim ako sistem nije oboren sve vreme. Takodje ih ne zanima da li koristi Microsoft Office ili OpenOffice. Kompanija možda ima puno razloga da bira jedan od njih, ali potrošače to ne zanima jer ih ne vide.

Tako, da bi kompanija učinila svoje poslovanje poželjnijim potrošačima, treba da troši više na diferencirajući softver koji čini njene proizvode poželjnijim, a manje na softver koji ne diferencira samu kompaniju od konkurenata. Jedan od načina trošenja manje na nediferencijatore je korišćenje softvera otvorenog koda, jer se time distribuira cena i rizik sa samo jedne kompanije na više kompanija koje sarađuju.

Naravno, za ovo je potrebna iscrpna analiza toga šta je diferencirajuće, a šta nije u softveru koji kompanija koristi, a ovo nije uvek lako. Postoji puno kompanija sa NNO („Nije Napravljeno Ovde“) sindromom. To je kada menadžeri i tehničko osoblje nisu voljni da razmatraju rad ljudi van kompanije, jer ne veruju da je dobar kao onaj koji oni prave. Ovaj sindrom je skup i izazvaće dupliranje napora koji postoje na drugim mestima, umesto provođenja vremena na diferenciranje softvera koji je najbitniji za konkretan posao ([1]).

2.2. Ekonomske paradigme razvoja softvera

Kako je softver neophodan za poslovanje, mora da bude razvijen na neki način. Osnovne ekonomske paradigme razvoja softvera su kupovina, razvoj unutar kompanije ili sklapanje ugovora sa nekom drugom kompanijom da to uradi za nas, zatim kolaboracija bez licenci otvorenog koda, i najzad razvoj u zajednici otvorenog koda.

Svaka od navedenih se od ostalih razlikuje u tome kako distribuira cenu razvoja, kako distribuira rizik od neuspeha, po efikasnosti u finansiranju samog razvoja softvera u poredjenju sa transakcionim troškovima procesa dobijanja softvera, kao i po stepenu u kom drugi mogu biti ograničeni u korišćenju softvera. Ovi faktori odredjuju koja paradigma je odgovarajuća za korišćenje

konkretne vrste softvera. Ovde je bitno istaći da je ultimativni izvor finansiranja razvoja softvera potrošač a ne proizvodjač, i on može da usmeri svoja sredstva u razvoj korišćenjem bilo koje paradigme da bi dobio softver kakav želi.

2.2.1. Paradigma kupovine gotovog softverskog proizvoda

Paradigma kupovine gotovog softvera je najbliža prosečnoj osobi, a samo 30% softvera se razvija i prodaje na ovaj način. Ovde cenu razvoja softvera obično iznosi jedan proizvodjač. Proizvodjač teži da povrati novac uložen u razvoj, plus da ostvari profit, prodajom završnog proizvoda. Zbog toga je sav trošak razvoja neophodno uložiti unapred, pre nego što proizvođač može da počne da ostvaruje prihod. Rizik neuspešne proizvodnje profitabilnog proizvoda je kompletno na proizvođaču.

Ako se proizvod pokaže kao uspešan na tržištu, cena razvoja se distribuira potrošačima. Kako paradigma prodaje ne može direktno da distribuira cenu i rizik dok proizvod ne sazri, proizvođači često moraju da se okrenu tržištu investicija kao spoljnom mehanizmu distribuiranja cene i rizika.

Potreba proizvođača za spoljnim kapitalom je dugoročna, jer mogu biti potrebne godine da bi se razvio softver i tržište za njega, i dodatno vreme pre nego što cena razvoja može biti amortizovana i dok proizvod počne da ostvaruje profit. Berze povećavaju likvidnost investicije tako što omogućavaju investitorima da monetizuju percepciju budućeg potencijala kompanije (reflektovano u ceni akcije kompanije), umesto refleksije na prihode kompanije kojih još uvek nema. Uspešne kompanije mogu reinvestirati profit u razvoj sledećeg softverskog proizvoda, radije nego da ga vrate na tržište investicija. Transakcioni troškovi tradicionalnog modela paradigme prodaje softvera (tzv. modela "cigla i malter") su izuzetno veliki, toliko veliki da se procenjuje da manje od 10% novca koji plaća krajnji korisnik za softverski proizvod ide u razvoj softvera, dokumentaciju i marketing proizvoda. Majkrosoft je potrošio 16.8% svojih prihoda iz 2004. na istraživanje i razvoj proizvoda. Ostatak prihoda je otišao na stavke od kojih potrošač nema direktnih koristi, odnosno transakcione troškove kao što su advertajzing, dizajn i proizvodnja atraktivnih pakovanja, plaćanje maloprodajama za mesto na policama, osoblje za prodaju i naravno na profit. Cifra od 16.8% ne uključuje marže prodavaca, tako da se dobija da je odnos troškova razvoja softvera i cene proizvoda i manji od 10%.

Neefikasnost postoji i na strani kupovine, zbog nepoklapanja izmedju zahteva potrošača/ kupca i funkcionalnosti softvera koji se kupuje. Kupac često kupuje softver za koji se, pri detaljnijoj proceni, ispostavlja da nije dovoljno koristan za željenu primenu. Potrošač skoro nikad ne može da povrati uloženu cenu ovog proizvoda sa police. Gubici postoje i jer softverske kompanije propadaju ili prestaju da izdaju nove verzije proizvoda, pa ukidaju korisničku podršku. Kako generalno izvorni kod nije dostupan ni za proizvode koji se ne dorađuju, a nema ni podrške, potrošač može samo da otpiše svoju investiciju. Kombinovanjem ovih faktora rizika, dolazimo da procene da se svega 50% kupljenog softvera stvarno i koristi, ili se ne koristi efektivno pa je kupovina neuspešna. Ako ovo pomnožimo efikasnošću od (manje od) 10% paradigme kupovine softvera, nalazimo da je efikasnost finansiranja razvoja softvera kupovinom gotovog proizvoda manja od 5%. Ova procena se odnosi na ukupnu ekonomsku efikasnost paradigme, a ne na efikasnost gledanu iz ugla potrošača. Najvažnija implikacija ove izuzetno male efikasnosti je da paradigma kupovine ekonomski može da bude korišćena samo za kreiranje proizvoda za masovna tržišta. Masovna tržišta će maskirati neefikasnost paradigme, zato što svaki potrošač može platiti relativno malu cenu u poređenju sa cenom razvoja softvera. Ukupno, oni će i dalje plaćati više nego 20 puta cenu razvoja softvera. Naravno, ukoliko funkcionalnosti ovog softvera zadovoljavaju zahteve kupca, kupac ima računicu da softver kupi. Postoji mnogo važnih softverskih proizvoda koji jednostavno ne mogu biti kreirani korišćenjem paradigme prodaje softvera, jer nemaju dovoljno veliko tržište da amortizuju cenu razvoja i velike transakcione troškove paradigme. Šta više, mnogi novi proizvodi koji bi možda kad-tad mogli da izgrade veliko tržište neće biti razmatrani, jer kompanije i investitori ne mogu biti ubeđeni da bi se to tržište zaista razvilo, ili smatraju da je rizik neuspeha previše visok. Ovo u neku ruku onemogućuje inovaciju u softverskoj industriji koja koristi ovu paradigmu. Jedan od primera neuspeha inovacije kod paradigme prodaje softvera je činjenica da je najvažnija inovacija softverske industrije (govorimo o komercijalnoj, ne naučnoj inovaciji) u poslednjoj dekadi, web server i web čitač, su razvijeni u open source zajednici, u istraživačkim laboratorijama državnih univerziteta. Nijedna od kompanija koja je mogla da kompletira posao nije bila ubeđena da će moći da zaradi novac na ovim projektima. Šta više, jedina kompanija koja je investirala značajna sredstva u razvoja web-a (Autodesk, investiranjem u Xanadu projekat Teda Nelsona), je prekinula projekat jer je model prihoda koji je Xanadu predvideo za web (uplate proizvođačima sadržaja za svaku reč, sa komplikovanim sistemom praćenja izvedenih radova i referenci) bio previše

komplikovan za razvoj.

Tim Berners-Lee, koji se često naziva liderom i idejnim tvorcem uspešnog web-a kakav znamo danas, uopšte nije razmišljao o razvijanju bilo kakvog kanonskog modela prihoda, već je ostavio korisnicima da smisle načine za zaradu na web-u.
Takodje, paradigma prodaje softvera mora da ponudi proizvod za što širi skup kupaca, da bi generisala najveći mogući profit. Zbog ovoga je proizvod generalno dostupan svima, pa je teško ili nemoguće iskoristiti ga za diferenciranje poslovanja u odnosu na konkurente.

2.2.2. Razvoj unutar kompanije ili ugovoreni razvoj

Razvoj unutar kompanije obavljaju programeri zaposleni u kompaniji, dok ugovoreni razvoj obavlja spoljna kompanija, koja ekskluzivno biva unajmljena za dogovoreni posao. U oba slučaja, programeri su generalno plaćeni za posao pisanja softvera, pre nego za softver kao proizvod, što je bio slučaj kod paradigme kupovine. Drugi oblik ugovorenog razvoja je detaljno proširenje i specijalizacija proizvoda snabdevača koji je u osnovnoj varijanti svima dostupan. Na primer, srednja ili veća kompanija može da kupi standardno okruženje web servera, a onda da plati proizvodjaču za specijalizaciju tako da odgovara konkretnim potrebama te kompanije.
Potrošač ima odličnu kontrolu nad razvojem unutar kompanije i nad ugovornim razvojem, zato što programeri neće biti plaćeni ako ne urade ono što kupac želi. Kupac generalno može da diktira i da li će softver biti ekskluzivno samo njegov ili će biti dostupan ostalima. Pošto kupac može da kontroliše distribuciju, ovo je izuzetna paradigma za razvoj diferencirajućeg softvera. Šta više, verovatno je ovo jedina paradigma koja zaista omogućava razvoj softvera koji će diferencirati poslovanje kupca. Generalno, kontraktor će sigurno pokušati da iskoristi neki deo posla koji je obavio za jednog korisnika da brže obavi posao za drugog korisnika, odnosno da ponovo proda posao za koji je već bio plaćen. Koliko je kontraktor uspešan u ovome, zavisi od uslova ugovora i njegove iskrenosti. Ako je kontraktor samopouzdan da može da preproda rad koji obavlja za jednog kupca, on može da ga naplati manje, i kupac može da ima koristi od distribucije cene i rizika. Ipak, ovo generalno nije zgodan scenario za diferencirajući softver: ako kontraktor preprodaje kupčeve diferencijatore posla, kupčevo poslovanje može da bude u problemu.
U zamenu za potpunu kontrolu, potrošač/kupac generalno snosi kompletnu cenu i rizik razvoja. Zbog toga, ovo možda nije najefektivniji metod, po pitanju cene, za razvoj nediferencirajućeg softvera. Ako kupac plaća 100% cene razvoja novog softvera koji duplira funkcije postojećeg softvera koji bi bio dostupan potrošačima za manje, to se naziva izmišljanjem vruće vode i predstavlja uludo trošenje sredstava. Ako kupac može da podnese da se njegov softver distribuira drugima, i nije mu potrebna apsolutna kontrola nad razvojem, paradigma kupovine ili paradigma razvoja u zajednici otvorenog koda može biti efektivnija po ceni za taj softver. Razvoj unutar kompanije ili ugovaranjem je razumno efikasan u usmeravanju velikog procenta ulaganja ka samom razvoju softvera. Efikasnost u ovome je 50% do 80%, za razliku od 10% kod paradigme kupovine. Glavni izvori neefikasnosti su cene nalaženja novih potrošača, cene održavanja ekspertize koja se ne koristi sve vreme i sl.
Odredjeni procenat projekata razvijenih ovom paradigmom nikad ne uspe da proizvede softver koji se koristi u poslovanju kupca i da postigne ciljeve koji su planirani. Kao procenat uspeha se može uzeti 50%, slično ukupnom procentu paradigme prodaje.

2.2.3. Kolaboracija bez licenci otvorenog koda

Standardan način saradnje među kompanijama na razvoju softvera je bio konzorcijum. Zatvoreni konzorcijumi za razvoj softvera imaju istoriju velikih neuspeha. U skorije vreme, zatvoreni konzorcijumi vredni milione dolara su zamenjeni uspešnijim projektima otvorenog koda. Kao primer mogu se navesti Taligent i Monterey, 2 konzorcijuma okupljena u nameri kreiranja zamene za Unix. Linux ih je prestigao i zamenio. Takodje, Common Desktop Environment je zamenjen open source GNOME radnom površinom u većini kompanija koje su podržavale CDE. Pre nekoliko godina, velika agrikulturna korporacija Cargill je osnovala konzorcijum sa namerom korišćenja prednosti razvoja otvorenog koda, dok u isto vreme pruža tajnost i deli benefite softvera samo sa članovima konzorcijuma. Dve godine kasnije, Cargill je napustio projekat koji je finansirao. Zatvoreni konzorcijum se jednostavno pokazao kao pogrešna struktura za razvoj nediferencirajućeg softvera. Ima smisla širom otvoriti vrata kada ne štitite diferencijaciju, i primiti članove koji mogu da korisno doprinesu čak i kad ne mogu da pomognu u finansiranju. Takodje konzorcijum košta više od projekta otvorenog koda zato što ima manje članova koji dele cenu i rizik, a ima mnogo više strukture i transakcionih troškova nego što bi bilo u ekvivalentnom projektu otvorenog koda. Kod zatvorenih konzorcijuma, ljudi se postavljaju da rade u interesu onoga ko ih plaća, a kod open source-a je to obično po pokazanim tehničkim zaslugama u razvoju. Zato se može desiti da član

radi na štetu celog projekta, ako to njemu koristi. Produkciono planiranje se često svede na nerešive rasprave medju kompanijama, zato što svaka ima različitu marketinšku ideju i argumente. Kada se uzme u obzir nesrećna istorija razvoja u konzorcijumima, i sa druge strane visok stepen uspeha velikih projekata otvorenog koda, čak i kad su u pitanju iste kompanije, izgleda da je red koji se uvodi licencama otvorenog koda esencijalna komponenta efektne kolaboracije izmedju velikog broja strana sa različitim interesima.

2.2.4. Paradigma razvoja otvorenog koda

Kod paradigme razvoja otvorenog koda, više entiteta (individue, kompanije, akademske institucije) se okuplja u cilju razvoja softverskog proizvoda. Generalno, inicijalni razvoj obavlja jedan entitet kao razvoj unutar kompanije, a softver se pušta u javnost, odnosno kod se otvara čim postane koristan drugima, a najčešće pre nego što se može smatrati završenim proizvodom i samim tim mnogo ranije nego što bi bio pušten da se radi o paradigmi kupovine. Kada je softver jednom upotrebljiv i koristan, drugi entiteti počinju da ga upotrebljavaju. Tek tada paradigma razvoja otvorenog koda počinje da funkcioniše, jer pre toga drugi entiteti nemaju interesa da koriste i proširuju softver. Proširivanjem softvera se bave individue ili zaposleni entiteta koji koristi softver. Inkremetalna cena dodavanja karakteristike je mnogo manja nego cena celog razvoja. Strane koje kreiraju modifikacije imaju interes da te modifikacije pišu tako da one budu prihvaćene od strane drugih programera sa projekta, i da budu unešene u zvaničnu verziju izvornog koda. Ako do ovog spajanja ne dođe, cena održavanje te promene kroz naredne verzije je mnogo veća, jer je onda sa svakom novom verzijom neophodno dodavati ih i održavati kompatibilnost sa bazom koda koja se stalno menja. Zbog toga otvoreni kod okuplja zajednicu zainteresovanih programera koji doprinose korisnom proizvodu. Cena i rizik razvoja proizvoda se distribuiraju među programerima, a bilo koja kombinacija tih programera može da nastavi da radi na projektu ako ostali odu. Distribucija cene i rizika počinje čim je projekat dovoljno zreo da izgradi zajednicu širu od inicijalnog programera. Projekat otvorenog koda razvijaju direktno njegovi krajnji korisnici. Npr, funkcionalnosti Apache web servera dodaju kompanije kojima su te funkcionalnosti potrebne da bi izvršavali svoje web sajtove, ili ih dodaju kontraktori koji rade za te kompanije. Korisnici konkretnog proizvoda otvorenog koda se generalno sami identifikuju: oni traže proizvod u direktorijumu open source softvera, onda ga download-uju i testiraju. Ako su zadovoljni, počinju da ga koriste. Time stiču kontinuirani interes za proizvod, a u nekom trenutku, ako požele dodatne funkcionalnosti imaju interes da postanu ko-programeri softvera koji koriste. Kompanije koje se pridruže razvoju projekta otvorenog koda, koriste taj softver kao nediferencirajući, odnosno on im je trošak a ne izvor prihoda. Ovim kompanijama nije bitno što razvoj otvorenog koda ne donosi profit: njihovi izvori profita su neke druge stvari, a softver je tehnologija koja omogućava rad. Da bi mogli da nastave sa ostvarivanjem prihoda iz svojih izvora, oni moraju da investiraju u infrastrukturu. U slučaju diferencirajućeg softvera, verovatno će morati da koriste razvoj unutar kompanije ili ugovoreni razvoj, jer će morati da spreče da diferencijator dođe u ruke konkurenta. Za nediferencijatore, postoji izbor izmedju kupovine ili paradigme razvoja otvorenog koda. Pokušaću da odgovorim koji je metod od ova 2 efikasniji, u smislu opšte ekonomske efikasnosti, kao i u smislu pogodnosti sa strane kupca. Kako se korisnici sami identifikuju, open source nema neefikasnosti paradigme prodaje, koja mora da koristi advertajzing i druge skupe mehanizme da bi privukla korisnike. U najmanjoj meri je podjednako efikasna u usmeravanju sredstava u razvoj softvera u odnosu na transakcione troškove kao razvoj unutar kompanije ili ugovoreni razvoj. Zbog ovoga open source paradigma može biti korišćena za razvoj proizvoda koji nisu namenjeni masovnom tržištu, i koji iz tog razloga ekonomski ne bi mogli biti razvijeni paradigmom kupovine. Ipak, od konkurencije na tržištu konkretnog proizvoda, ukoliko isto postoji, zavisi da li je za kupca isplativije da kupi gotovi proizvod ili se uključi u razvoj u zajednici. Naravno, ne uspeju svi projekti otvorenog koda. Većina nezrelih projekata umre na samom početku, ili zauvek ostane solo projekat. Ipak, trošak takvih projekata je jako mali, jer oni nikad i ne privuku značajan broj korisnika ili programersku zajednicu, pa te projekte možemo isključiti iz razmatranja. Zreliji projekti otvorenog koda mogu prestati da žive kada naiđe nešto bolje, ali je često moguće iskoristiti izvorni kod, podatke i veštine iz jednog projekta za drugi, pa tako investicija u razvoj projekta nije izgubljena. Cena učestvovanja u zrelom projektu otvorenog koda se veoma razlikuje od cena kupovine ili razvoja unutar kompanije. Glavni izdatak su vreme i cena zaposlenih koji učestvuju. Detaljnije, u ovo cifru ulazi osoblje koje procenjuje i testira softver, oni koji prilagodjavaju softver, odnosno rade na razvoju i oni koji rade kao podrška korisnicima unutar kompanije. Rizik predstavlja mogućnost da se ulaže u softver koji će bit zamenjen zbog nemogućnosti da se prilagodi ili da se isprate potrebe korisnika. Maksimalna cena ovakvog

razvoja je u slučaju kada nema programerske zajednice osim tog konkretnog korisnika, i u tom slučaju je cena ista kao cena razvoja unutar kompanije ili ugovaranjem, gde jedan potrošač finansira ceo razvoj. Stvarna cena u skladu sa ovim zavisi od broja aktivnih participanata i potrebnog rada. Izgubljena investicija je vreme osoblja. Uzimajući sve ovo u obzir, opšta ekonomska efikasnost je velika barem kao kod paradigme razvoja unutar kompanije i ugovaranjem, a puno veća nego kod paradigme kupovine.

3. ZAKLJUČAK

Paradigma razvoja otvorenog koda ima nekoliko značajnih ekonomskih prednosti nad ostalim paradigmama: ona kombinuje efikasnost u usmeravanju resursa u sam razvoj softvera sa boljom distribucijom cene i rizika. Nju je moguće primeniti na razvoj proizvoda koje nije moguće razviti korišćenjem paradigme kupovine jer nema masovnog tržišta.

Open source distribuira cenu i rizik proizvoda, dok razvoj unutar kompanije fokusira svu cenu i sav rizik na jedan entitet. Open source takođe ne zahteva korišćenje investicionih tržišta za distribuiranje cene i rizika, a ditribucija počinje mnogo ranije u odnosu na paradigmu prodaje. Otvoreni kod daje korisniku komplentu kontrolu nad specijalizacijom i prilagodjavanjem proizvoda. Ipak, ne daje korisniku kontrolu nad tim ko ima pristup proizvodu. Zbog toga open source paradigma nije dobar mehanizam za razvoj diferencirajućeg softvera. Jedina odgovarajuća paradigma za razvoj diferencirajućeg softvera je razvoj unutar kompanije ili ugovoreni razvoj. Poslovanja koja zahtevaju nediferencirajući softver (a to su ustvari sva poslovanja) mogu da prebace deo svoje infrastrukture i usmere razvoj na open source projekte, i time će bez sumnje postići veću ekonomsku efikasnost. Ovo je dobra poslovna strategija jer time ostaje više sredstava za ulaganje u diferencirajući softver koji utiče na profit. Ipak, kod konkretne odluke treba uzeti u obzir i ponude na tržištu odgovarajućih gotovih proizvoda.

Paradigma	Efikasnost	Stopa neuspeha	Distribucija cene	Distribucija rizika	Štiti diferencijaciju korisnika	Štiti diferencijaciju snabdevača	Potrebna veličina tržišta
Kupovina gotovog proizvoda	manje od 10%	50%	kasno, nekad nakon što počne prodaja	Ne	Ne	Da	Masovno tržište 100,000 ili više, za specijalizovana tržišta manje
Unutar kompanije i ugovaranjem	60% do 80%	50%	Ne	Ne	Da	Možda	1
Konzorcijumi i saradnja zatvorenog koda	60% do 80%	Možda 90%, neprihvatljivo visoka	Da	Da	Možda	Možda	5 i više
Razvoj otvorenog koda	60% do 100%	50% ako isključimo nezrele projekte	Rano, tokom razvoja	Da	Ne	Ne	5 i više

Tabela 1. Poređenje ekonomskih paradigmi razvoja softvera

LITERATURA

[1] The Emerging Economic Paradigm of Open Source, Bruce Perens, 2005.
[2] The Magic Cauldron, Eric S. Raymond, 2002.
[3] Producing open source software, Karl Fogel, 2005.
[4] The business of software, Michael Cusumano, 2004.
[5] Open source software development case studies, Robert L. Greenberg, 2003.
[6] http://www.answers.com/topic/software-industry
[7] http://www.microsoft.com
[8] http://www.ibm.com
[9] http://www.google.com/corporate
[10] http://www.mysql.com