Linije programskih proizvoda

Bartaković, Ivan

Sveučilište Jurja Dobrile u Puli

Odjel za informacijsko-komunikacijske tehnologije

IVAN BARTAKOVIĆ

LINIJE PROGRAMSKIH PROIZVODA

Završni rad

Pula, rujan 2016.

Sveučilište Jurja Dobrile u Puli

Odjel za informacijsko-komunikacijske tehnologije

IVAN BARTAKOVIĆ

LINIJE PROGRAMSKIH PROIZVODA

Završni rad

JMBAG: 0303046234; redoviti student

Studijski smjer: Informatika

Predmet: Softversko inženjerstvo

Mentor: doc. dr. sc. Tihomir Orehovački

Pula, rujan 2016.

IZJAVA O AKADEMSKOJ ČESTITOSTI

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

Student

_____

U Puli, _____, 2016. godine

IZJAVA

o korištenju autorskog djela

Ja, Ivan Bartaković dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom „Linije programskih proizvoda“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

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

U Puli, _____ (datum)

Potpis

_____

Sadržaj

Uvod

U današnjici se razvoj programskih proizvoda svodi na to tko će proizvesti brže i kvalitetnije. Iz tog razloga ponovna upotreba je postala veoma čest način razvoja programskih proizvoda. Ponovna upotreba se temelji na iskorištavanju dijelova već proizvedenih softvera za razvoj novih. Razlog prijelaza s tradicionalnog razvoja programskih proizvoda na razvoj pomoću ponovne upotrebe je želja organizacija da smanje troškove proizvodnje i održavanja. Osim što organizacijama donosi uštedu na troškovima proizvodnje, razvoj pomoću ponovne upotrebe može i povećati kvalitetu programskih proizvoda te ubrzati proces dostavljanja proizvoda na tržište. Tijekom godina razvijene su različite tehnike razvoja programskih proizvoda pomoću ponovne upotrebe te se ovisno o strategiji organizacije koja se bavi razvojem programskih proizvoda bira najprigodnija tehnika. Razvoj pomoću linija programskih proizvoda je jedna od tehnika ponovne upotrebe. Linije programskih proizvoda su skupovi softverskih sustava koji imaju sličan skup upravljačkih značajki koje zadovoljavaju neke specifične potrebe za specifično tržište te su razvijeni iz zajedničkog skupa komponenti koje se ponovno upotrebljavaju na unaprijed propisani način. Kod linija programskih proizvoda postoje četiri perspektive razvoja kojima organizacija mora pristupiti da bi uspješno razvila jednu programsku liniju. Jedna od tih perspektiva je poslovna perspektiva u kojoj se određuju poslovne strategije razvoja programskih proizvoda na kojima se temelji razvoj cijele linije programskih proizvoda. Organizacijama je tehnika razvoja pomoću linija programskih proizvoda postala privlačna upravo zato što takav način razvoja koristi strategijski pristup razvoju softvera koji na više načina pozitivno utječe na poslovanje organizacije. Ako se razvoj pomoću linija programskih proizvoda koristi na ispravan način to organizacijama donosi velike uštede truda i novaca kod proizvodnje te dovodi do poboljšanja kvalitete proizvoda. Cilj ovog završnog rada je da se detaljno opiše razvoj programskih proizvoda pomoću ponovne upotrebe tehnikom razvoja linija programskih proizvoda te kako takav razvoj programskih proizvoda utječe na organizaciju koja se bavi takvim razvojem.

Ovaj rad je podijeljen u 6 poglavlja, u kojem nakon obrade teme slijedi zaključak, literatura te popis slika.

U prvom poglavlju pod nazivom „Razvoj programskih proizvoda pomoću ponovne upotrebe“ govori se o prednostima i problemima ponovne upotrebe, tehnikama ponovne upotrebe te ključnim faktorima kod odabira tehnika ponovne upotrebe.

Slijedi drugo poglavlje s naslovom „Općenito o linijama programskih proizvoda“ u kojem se govori o potencijalu ponovne upotrebe kod linija programskih proizvoda, potencijalnim problemima kod takve upotrebe te o BAPO modelu koji govori o četiri glavne perspektive razvoja koje organizacije moraju uzeti u obzir kod razvoja linije programskih proizvoda. Te su te četiri perspektive obrađene u sljedećih nekoliko poglavlja.

Treće poglavlje nosi naslov „Poslovna perspektiva razvoja linija programskih proizvoda“ te se u njemu govori o tržištima linija programskih proizvoda, strategijama definiranja proizvoda, strategijama tržišta, o životnom ciklusu linije programskih proizvoda i njenom opsegu te se na kraju obrađuje ekonomičnost linija programskih proizvoda.

U sljedećem poglavlju pod nazivom „Arhitekturalna perspektiva kod razvoja linija programskih proizvoda“ govori se o tehnikama realizacije varijabilnosti u referentnoj arhitekturi, te o evaluaciji i evoluciji referentne arhitekture. Na kraju poglavlja prikazan je jedan primjer referentne arhitekture.

Peto poglavlje „Procesna perspektiva razvoja linije programskih proizvoda“ govori o dva životna ciklusa razvoja linije programskih proizvoda odnosno o domenskom i o aplikacijskom razvoju.

Šesto i posljednje poglavlje „Organizacijska perspektiva razvoja linije programskih proizvoda“ govori o nekoliko vrsta struktura organizacije te o određivanju uloga i zadaća u organizaciji.

Nakon toga slijede zaključna razmatranja ovog rada, literatura i popis slika.

1. Razvoj programskih proizvoda pomoću ponovne upotrebe

Prije nego što se upustimo u detaljniju obradu linija programskih proizvoda trebamo se upoznati s razvojem programskih proizvoda pomoću ponovne upotrebe. Razlog nastanka ovakve metode razvoja programskih proizvoda je potražnja za boljom kvalitetom i bržom izradom programskih proizvoda uz manje troškove razvoja i održavanja. Razvoj pomoću ponovne upotrebe funkcionira tako da se programski proizvod pokušava razviti korištenjem što više dijelova postojećih sustava. Kada se u obzir uzme veličina dijelova koja se ponovno upotrebljava, razlikuje se nekoliko vrsta ponovne upotrebe a to su (Manger, 2013., str. 139):

- Ponovna upotreba cijelog aplikacijskog sustava:

  Kod ponovne upotrebe cijelog aplikacijskog sustava cijeli sustav se može ponovno iskoristiti tako što se integrira u druge sustave bez nekih većih promjena te se konfigurira da bi odgovarao novim potrebama korisnika. Ovaj način će se detaljnije razraditi u potpoglavlju 1.4.

- Ponovna upotreba dijela sustava:

  Kod ponovne upotrebe dijela sustava iskorištava se samo dio postojećeg sustava da bi se izgradio novi sustav ili se iskorištavaju softverske komponente iz postojećih sustava koje su dizajnirane na takav način da se mogu ponovno upotrebljavati.

- Ponovna upotreba objekta ili funkcije:

  Sommerville (2007., str. 416) navodi da se ponovna upotreba objekta ili funkcije temelji oko standardnih biblioteka tj. iskorištavaju se komponente koje imaju implementiranu jednu funkciju ili jedan objekt te da je to bila uobičajena praksa korištenja ponovne upotrebe u zadnjih četrdesetak godina.

Također osim vrsta po veličini, postoji i podjela s obzirom na to što se upotrebljava te ona po Mangeru (2013., str. 140) glasi:

- Ponovna upotreba programskog koda:

  Za izgradnju novog sustava koristi se već napisani programski kod iz postojećeg softvera. Tu se može raditi o nasljeđivanju objekata, pozivanju web servisa, korištenja biblioteka ili uključivanju komponenata.

- Ponovna upotreba koncepta, modela ili dizajnerskog rješenja:

  Kod takvog pristupa ponovne upotrebe koristi se već poznati obrazac za oblikovanje, ali se programski kod ne iskorištava već se piše novi programski kod na osnovu tog obrasca za oblikovanje. Također, ako se radi o metodi grafičkog razvoja može se koristiti već izrađeni UML dijagram iz kojeg se programski kod kreira automatski.

Razvoj pomoću ponovne upotrebe donosi ima nekoliko prednosti nad tradicionalnim razvojem programskih proizvoda. No ako se takav razvoj ne izvodi pažljivo može nastati nekoliko problema.

1.1. Prednosti i problemi kod korištenja ponovne upotrebe

Jedna od najvećih prednosti korištenja ponovne upotrebe bi bili smanjeni troškovi proizvodnje. Također postoje i druge prednosti korištenja ponovne upotrebe a to su (Sommerville, 2007., str. 416):

- Povećanje pouzdanosti programskog proizvoda:

  Programski proizvod koji je već razvijen, testiran i korišten u postojećim sustavima ima prednost nad novim programskim proizvodima zato što je on već testiran u fazi dizajna i implementacije te su njegove greške pronađene i ispravljene. Iz toga razloga kada u svrhu ponovne upotrebe koristimo komponente tog već razvijenog programskog proizvoda možemo biti više pouzdaniji u njega nego što bi bili kod tradicionalnog načina razvoja.

- Smanjenje rizika poslovnih procesa:

  Kada se razvija novi programski proizvod, troškovi razvijanja su nepredvidivi te uvijek postoji neizvjesnost dali su predviđeni troškovi ili rokovi dostavljanja proizvoda dobro određeni, dok je cijena programskog proizvoda kojeg planiramo ponovno koristiti već poznata te su zbog toga predviđeni troškovi puno bliži stvarnom trošku.

- Bolja iskoristivost specijalista:

  Programeri, softverski inženjeri ili ostali specijalisti umjesto da razvijaju slične programske proizvode svaki puta ispočetka, mogu iskoristiti svoje znanje tako da razviju programski proizvod koji će u budućnosti opet moći koristiti te tako uštede na trudu, vremenu i troškovima.

- Usklađenost sa standardima:

  Ako se komponente ponovno iskorištavaju na više aplikacija, npr. korisničko sučelje, onda takva komponenta postaje pouzdanija za korisnike jer su upoznati s tom komponentom kroz prijašnje aplikacije.

- Ubrzani razvoj proizvoda:

  Ponovno korištenje programskih proizvoda je povoljno za brže dostavljanje proizvoda na tržište zato što se smanjuje vrijeme razvoja i validacije samog proizvoda.

Iako su prednosti korištenja ponovne upotrebe poprilično velike, u pojedinim okolnostima postoje i velike mane:

- Povećani troškovi održavanja:

  Može se dogoditi da dokumentacija korištenog programskog proizvoda nije ažurirana ili nam nije dostupan izvorni kod te je teže držati korištene dijelove programskog proizvoda kompatibilne s novim promjenama.

- Nedostatak potrebnih alata za razvoj programskih proizvoda:

  Ponekad nema dostupnih softverskih alata pomoću kojih bi se ubrzala implementacija raspoloživih resursa.

- Poteškoće kod pronalaženja komponenti:

  Ako se potrebne komponente ne pronađu, moraju se ili ponovno definirati zahtjevi pa opet krenuti u potragu za komponentama koje će odgovarati novim zahtjevima ili će se odustati od ideje ponovnog korištenja.

- Problemi s inženjerima:

  Ponekad softverski inženjeri smatraju da bi vlastitim razvojem programskih proizvoda napravili bolji proizvod te ne žele koristiti metodu ponovne upotrebe.

Uz sve prednosti i mane, metoda razvoja programskih proizvoda pomoću ponovne upotrebe je isplatljiva ako se pažljivo i strategijski sagledaju sve opcije te se odabere prava tehnika ponovne upotrebe te se napravi dugoročni plan razvijanja programskih proizvoda tom tehnikom.

1.2. Tehnike za razvoj programskih proizvoda pomoću ponovne upotrebe

Prema Sommerville-u (2007., str. 420) u zadnjih nekoliko desetljeća razvijeno je mnoštvo tehnika za razvoj programskih proizvoda pomoću ponovne upotrebe. Kao što već znamo, možemo ponovno upotrijebiti cijeli sustav, dio sustava ili samo jednu funkciju ili klasu sustava. S obzirom na to postoje sljedeće tehnike za ponovnu upotrebu:

- Obrasci za oblikovanje:

  Kod obrazaca za oblikovanje se radi o ponovnoj upotrebi provjerenih generičkih rješenja za probleme u oblikovanju koji se pojavljuju u mnoštvu različitih konteksta. Moramo naglasiti da to nije gotov obrazac koji se može

direktno pretvoriti u izvorni ili strojni kod. Nego ih možemo opisati kao predložak za rješavanje problema koji se mogu koristiti u mnoštvu različitih situacija.

- Razvoj programskih proizvoda pomoću komponenti:

  Razvoj programskih proizvoda pomoću komponenti je tehnika ponovne upotrebe koja koristi gotove komponente koje su razvijene u svrhu ponovne upotrebe te se te komponente zajedno integriraju i implementiraju da bi činile novi sustav.

- Aplikacijski okviri:

  Aplikacijski okvir se može definirati kao kolekcija konkretnih ili apstraktnih klasa koje zajedno grade arhitekturu za obitelj sličnih aplikacija. Aplikacijski okvir je upravo to, samo okvir, te da bi od toga razvili aplikaciju, mora se nadograditi ili proširiti njegov kod.

- Ponovna upotreba baštinjenih (eng. *legacy*) sustava:

  Kod ponovne upotrebe baštinjenih sustava koristi se stari baštinjeni sustav koji se koristi u suvremenom okruženju tako da se pomoću posebno razvijenih omotača (eng. *wrapper*) definiraju sučelja koja pružaju pristup starom sustavu.

- Razvoj programskih proizvoda pomoću web servisa:

  Kod razvoja pomoću web servisa sustav se izrađuje tako da se koriste javno dostupni servisi s interneta iz registra servisa te se s njima komunicira pomoću sučelja za komuniciranje preko određenih protokola.

- Linije programskih proizvoda:

  Linije programskih proizvoda su skupovi softverskih sustava koji imaju sličan skup upravljačkih značajki koje zadovoljavaju neke specifične potrebe za specifično tržište te su razvijeni iz zajedničkog skupa komponenti koje se ponovno upotrebljavaju na unaprijed propisani način.

- Ponovna upotreba COTS (eng. *commercial off-the-shelf*) sustava:

  COTS sustavi su javno dostupni sustavi koji se mogu prilagoditi korisničkim potrebama bez mijenjanja izvornog programskog koda. Umjesto mijenjanja izvornog koda koriste se javno dostupni softverski paketi koji se ili integriraju međusobno ili se konfiguriraju za specifične potrebe korisnika.

- Promjenjive vertikalne aplikacije:

  Kod promjenjivih vertikalnih aplikacija razvija se generički sustav koji se može mijenjati prema nekim specifičnim zahtjevima korisnika.

- Ponovna upotreba pomoću standardnih biblioteka:

  Kod ponovne upotrebe pomoću standardnih biblioteka kod razvoja novog sustava koriste se gotove funkcije ili klase iz pojedinih biblioteka.

- Ponovna upotreba pomoću generatora programa:

  Po Mangeru (2013., str. 140) ponovna upotreba pomoću generatora programa se koristi u usko specijaliziranim domenama tako da postoje alati koji iz specifikacije automatski generiraju odgovarajuće programe.

Svaka od ovih tehnika može doprinijeti uspjehu organizacije koja razvija programski proizvod no potrebno je na pravilan način izabrati koja tehnika je najpovoljnija za razvoj njihovih sustava. Sljedeće poglavlje se bavi ključnim faktorima kod odabira tehnike za ponovnu upotrebu.

1.3. Ključni faktori za odabir tehnike ponovne upotrebe

Da bi se minimalizirao nastanak problema treba strateški odabrati tehniku za ponovnu upotrebu koja će najbolje funkcionirati u specifičnom okruženju organizacije. Kada se odabire tehnika za ponovnu upotrebu postoje ključni faktori koje treba uzeti u obzir a to su (Sommerville, 2007., str.419):

- Rok za isporuku programskog proizvoda:

  Ako proizvod treba biti dostavljen na tržište u što kraćem roku poželjno je koristiti ponovnu upotrebu cijeloga sustava, što bi značilo da se trebaju koristiti COTS sustavi. Takvi sustavi neće u potpunosti odgovarati postavljenim zahtjevima, ali zahtijevaju najmanje vremena za razvijanje.

- Očekivani životni vijek programskog proizvoda:

  Ako se razvija programski proizvod koji bi trebao biti dugoročan, glavni fokus bi trebao biti na olakšavanju održavanja tog proizvoda. Tada treba odabrati dali se isplati razvijati programski proizvod s ponovnom upotrebom ili krenuti samostalnim razvojem. Ako se ipak ide putem ponovne upotrebe preporuka je da se izbjegava korištenje komponenti i sustava od vanjskih dobavljača zbog toga što ne možemo biti sigurni da će ti dobavljači moći pružati podršku tim komponentama.

- Znanje i iskustvo razvojnog tima:

  Razvoj programskih pomoću tehnika ponovne upotrebe je poprilično kompliciran, te zahtjeva mnogo vremena za proučavanje prije nego što se te tehnike mogu efektivno koristiti. Ako razvojni tim već ima neko prethodno znanje i iskustvo rada s nekom od tehnika za ponovnu uporabu onda ta tehnika ulazi u uži izbor.

- Specifični nefunkcionalni zahtjevi i pouzdanost programskog proizvoda:

  Ako postoje specifični nefunkcionalni zahtjevi gdje se traži visoka pouzdanost ili visoke performanse treba izbjeći korištenje tehnika kod kojih nemamo dostupan izvorni kod jer bez izvornog koda ne možemo garantirati pouzdanost.

- Aplikacijska domena:

  Kod nekih specifičnih domena postoje generički proizvodi koji su lako modificirani te ih možemo nadograditi prema vlastitim zahtjevima te se tako razvija novi proizvod.

- Platforma na kojoj će sustav raditi:

  Sustavi razvijeni pomoću komponenti najčešće su prilagođeni specifičnoj platformi tako da ako koristimo razvoj pomoću komponenti vezani smo uz tu specifičnu platformu.

Kod razvoja programskih proizvoda u skoro svakoj situaciji se može iskoristiti neka tehnika ponovne upotrebe, ali to nije uvijek slučaj. Menadžeri odlučuju kojim pristupom će se razvijati programski proizvod, te ponekad oni nisu dovoljno upoznati s prednostima i rizicima ponovne upotrebe te će radije koristiti samostalni razvoj iako takav pristup ima veći rizik (Sommerville, 2007., str. 421).

1.4. Ponovna upotreba cijelog aplikacijskog sustava

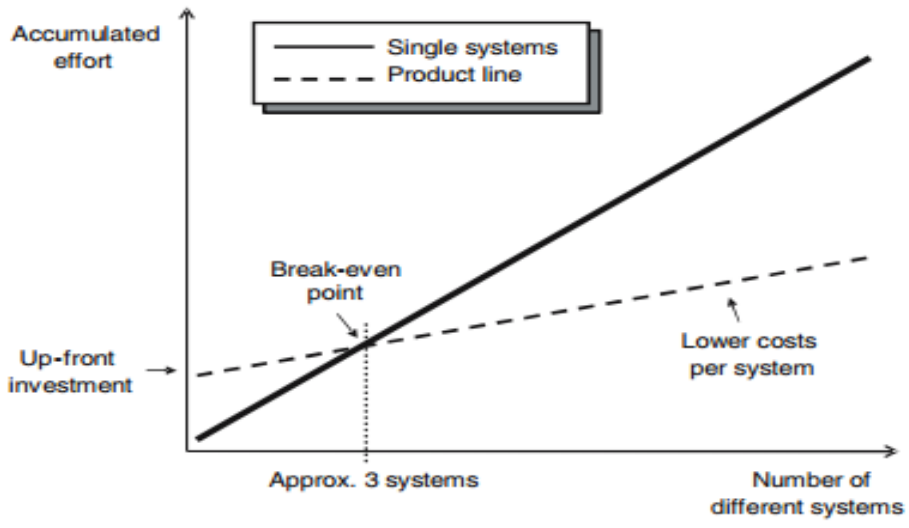
Kao što je spomenuto u uvodu poglavlja ponovna upotreba cijelog aplikacijskog sustava se izvodi tako da se iskorištavaju cijeli sustavi tako da se modificiraju za specifične zahtjeve ili se više sustava integrira da bi se razvio novi sustav. Ovakva vrsta ponovne upotrebe je najisplativiji način korištenja ponovne upotrebe upravo zato što se iskorištavaju cijeli sustavi koji se mogu u kratkom vremenu modificirati da bi se razvio novi sustav.

Postoje dva glavna načina razvoja programskih proizvoda pomoću ponovne upotrebe cijelog aplikacijskog sustava. Jedan od tih su COTS sustavi. Kao što smo spomenuli kod tehnika ponovne upotrebe, COTS sustavi su javno dostupni sustavi koji se lako prilagođavaju specifičnim potrebama korisnika jer sadrže mnoštvo mogućnosti i funkcija. Drugi način su linije programskih proizvoda. Linije programskih proizvoda su skupovi sustava koji imaju zajedničku arhitekturu te se razvoj novih sustava temelji na toj arhitekturi te se dodaju ili modificiraju komponente da bi se zadovoljili specifični zahtjevi proizvoda. Linije programskih proizvoda je glavna tema ovog završnog rada te je detaljnije obrađena u sljedećim poglavljima.

2. Općenito o linijama programskih proizvoda

Linija programskih proizvoda je skup softverskih sustava koji imaju sličan skup komponenti koje zadovoljavaju neke specifične potrebe za neko određeno tržište te su razvijeni iz zajedničkog skupa bazičnih sredstava na već unaprijed propisani način (Bass i sur., 2003., str. 356). Linije programskih proizvoda imaju veliki potencijal za organizacije koje se bave razvojem programskih proizvoda. No, treba napomenuti da ako se razvoju linije programskih proizvoda ne pristupi ispravno nastaju pojedini problemi. Potencijal linija programskih proizvoda i potencijalni problemi opisani su u poglavlju 2.1.

Razvoj linija programskih proizvoda ima velike ekonomske prednosti kao što su smanjeni troškovi i vrijeme dostavljanja proizvoda na tržište te su to jedni od glavnih razloga zašto kompanije prelaze s tradicionalnog razvoja proizvoda na ovakav pristup. Na slici 1. možemo vidjeti točku isplativosti razvoja linija programskih proizvoda. Na slici se jasno vidi da razvoj linija programskih proizvoda postaje isplativ nakon tri razvijena proizvoda. Više o ekonomičnosti linija programskih proizvoda obradili smo u potpoglavlju 3.3.

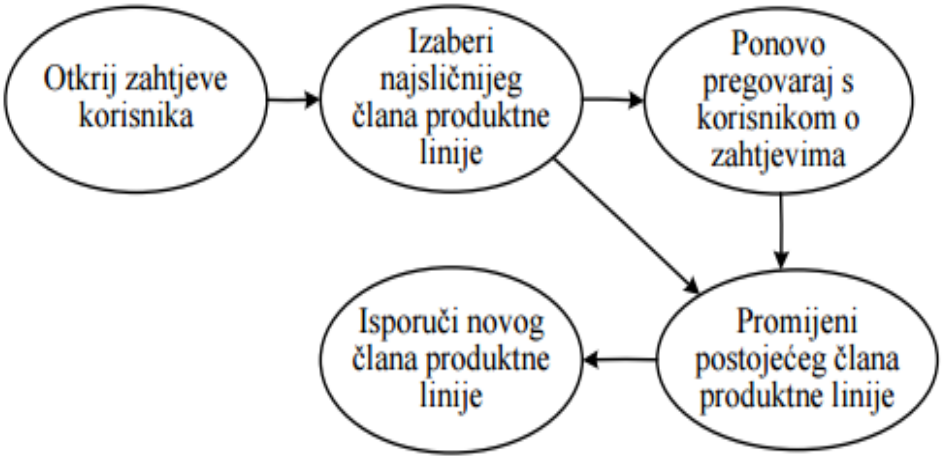


Slika 1.: Ekonomičnost linija programskih proizvoda

Izvor: Linden i sur. (2007., str. 4)

Razvoj linija programskih proizvoda se sastoji od dva ciklusa razvoja. Prvi ciklus je domenski razvoj (eng. *domain engineering*) tj. razvoj za ponovnu upotrebu, dok je drugi ciklus aplikacijski razvoj (eng. *application engineering*) tj. razvoj uz ponovnu upotrebu. Domenski razvoj služi za razvoj aplikacije koja će služiti kao platforma za sve proizvode u proizvodnoj liniji. Ta platforma se razvija tako da uključuje zajedničke komponente koje će se koristiti kroz cijelu liniju proizvoda te se dizajnira tako da određene komponente budu varijabilne kako bi se mogle modificirati za specifičan proizvod. Aplikacijski razvoj je razvoj individualnih aplikacija koje se temelje na već razvijenoj platformi. Prikupljaju se korisnički zahtjevi za novi proizvod te ako proizvod u potpunosti spada u proizvodnu liniju, platforma se modificira prema novim zahtjevima te je proizvod spreman za prodaju. ako postoji neka funkcionalnost koja nije razvijena na platformi, provjerava se dali je platforma kompatibilna za takav zahtjev te se zatim nadodaje ta funkcionalnost i proizvod je gotov. Treba napomenuti da kod dizajniranja platforme treba razmišljati o potencijalnim funkcionalnostima koje se neće uključiti u platformu, ali će možda zatrebati kod specifičnih proizvoda te se treba dizajnirati platformu tako da bude kompatibilna s tim potencijalnim funkcionalnostima. Ovi razvojni procesi su obrađeni u poglavlju 3.

Treba se napomenuti da je razvoj individualnih proizvoda pomoću linija programskih proizvoda uvelike jednostavniji nego što je to kod tradicionalnih načina razvoja. Na slici 2. možemo vidjeti da se postupak razvijanja novog proizvoda u proizvodnoj liniji odvija u nekoliko koraka. Prvi korak je da se otkriju zahtjevi korisnika. Nakon toga slijedi biranje najsličnije aplikacije koja već postoji u liniji te se ta aplikacija modificira da bi odgovarala novim zahtjevima korisnika. Jedan od koraka je i ponovno pregovaranje o zahtjevima s korisnikom ako neki zahtjev nije uključen u postojećoj aplikaciji. Pregovaranje nam služi za minimaliziranje promjena koje moramo napraviti na postojećoj aplikaciji. Ako je pregovaranje uspješno te se s korisnikom napravi kompromis sa zahtjevima nova verzija aplikacije se može brže i jeftinije izraditi. Tako izgradnja novog proizvoda postaje više slaganje postojećih komponenti nego kreiranje novih, te je umjesto programiranja glavna aktivnost integracija postojećih komponenti.



Slika 2.: Razvoj novog člana u liniji programskih proizvoda

Izvor: Manger (2013., str. 145)

Linden i sur. (2007., str. 16) navode da postoje četiri perspektive razvoja kojima se organizacija mora posvetiti da bi uspješno razvila liniju programskih proizvoda. Postoje poslovna perspektiva, arhitekturalna perspektiva, procesna perspektiva te organizacijska perspektiva razvoja. Ovakav pristup razvoju je poznatiji kao BAPO model te se o njemu govori više u poglavlju 2.3.

Kao što smo već spomenuli korištenjem linija programskih proizvoda mogu se postići velike uštede u proizvodnji, no i puno više od toga. U sljedećem poglavlju obradit ćemo potencijal ponovne upotrebe kod linija programskih proizvoda.

2.1. Potencijal ponovne upotrebe kod linija programskih proizvoda

Linije programskih proizvoda imaju veliki potencijal te su (Bass i sur., 2003., str. 358) naveli listu potencijalnih prednosti:

- Analiza zahtjeva:

  Većina zahtjeva će uvijek biti slična zahtjevima koje imaju već razvijeni sustavi te ih možemo ponovno upotrijebiti i uštedjeti na analizi zahtjeva.

- Arhitektura sustava:

  Razvijanje arhitekture sustava programskog proizvoda zahtijeva velike vremenske investicije najiskusnijih stručnjaka u organizaciji, no kada se koristi ponovna upotreba za novi sustav taj se korak može preskočiti jer arhitektura već postoji, te nema rizika od potencijalno krive arhitekture jer je već testirana u prošlim sustavima.

- Komponente programskih proizvoda:

  Komponente programskog proizvoda se mogu primijeniti na svim proizvodima. Ponovna upotreba komponenata uz ponovno korištenje izvornog koda uključuje i ponovnu upotrebu dizajna. Dijelovi dizajna koji su bili uspješni na prijašnjim proizvodima se iskorištavaju , a neuspješni se izbjegavaju.

- Modeliranje i analiza:

  Umjesto da se izrađuju nanovo, modeli performansi, analize rasporeda, problemi distribuiranih sustava, sheme za greške, itd. se prenose s proizvoda na proizvod te se na tome može značajno uštedjeti.

- Testiranje:

  Sva testiranja uključujući testne planove, procese, sve testne slučajeve i podatke te komunikacijski putevi za prijavljivanje i popravljanje grešaka su već izrađeni te se mogu preuzeti od prijašnjih proizvoda.

- Planiranje projekta:

  Zbog prijašnjih iskustava lakše je predvidjeti budžet i vremensko trajanje projekta. Također upravo zbog tih prijašnjih iskustava lakše se odrede timovi i njihova veličina.

- Procesi, metode i alati:

  Cijeli proces razvijanja proizvoda je već bio korišten tako da se sva dokumentacija i procesi odobrenja, svi alati, razvijanje sustava i distribucija procesa, svi standardi kodiranja i ostale svakodnevne radnje kod razvijanja programskog proizvoda mogu prenijeti s jednog proizvoda na drugi.

- Ljudska snaga:

  Kao što smo već spomenuli, aplikacije u liniji programskih proizvoda su veoma slične te upravo iz tog razloga zaposlenici se mogu lako premještati između različitih projekata jer se njihovo iskustvo može primijeniti na bilo koji proizvod u proizvodnoj liniji.

- Eliminacija grešaka:

  Svaki novi sustav koji se razvije u liniji programskog proizvoda biti će sve više kvalitetniji. Razlog tome je to što svaki novi sustav koji se razvija iskorištava prednosti eliminacije grešaka kod prošlih sustava. Također valja napomenuti da što je sustav više kompliciran, više se isplati riješiti inženjerske probleme kao što su problemi kod performansi i pouzdanosti upravo zato što se je onda taj problem riješio za cijelu produktnu liniju.

Iz ove liste možemo zaključiti da je potencijal korištenja linija programskih proizvoda jako velik, ali ga nije lako iskoristiti. Kao što već znamo linije programskih proizvoda se temelje na ponovnoj upotrebi, a primjena ponovne upotrebe ima dugu ali ne tako uspješnu povijest u softverskom inženjerstvu. Primjerice, kada se koristi ponovna upotreba stvara se baza sredstava za ponovno korištenje. Ta baza se puni komadićima prethodnih projekata, te svaki put kada programeri razvijaju novi programski proizvod, prije nego počnu razvijati nove elemente, moraju prvo provjeriti da li postoji sličan element u bazi. Problem je u tome što ako je ta baza sredstava premalena, programeri će teško naći nešto korisno u njoj, a ako je prevelika, teško će biti pronaći ono što im je potrebno. Također, ako su komponente koje oni žele iskoristiti premaleni, ne isplati im se modificirati postojeću komponentu iz baze sredstava već će je radije razviti nanovo. Ako je komponenta prevelika, onda je jako

teško razumjeti kako ta komponenta zapravo funkcionira, te takve komponente skoro nikad neće u potpunosti odgovarati novoj aplikaciji jer su takve komponente najčešće bile razvijene za aplikaciju s drugačijom arhitekturom nego ona koju programer razvija. Linije programskih proizvoda izbjegavaju sve ove probleme. One koriste ponovnu upotrebu na drugačiji, veoma striktan i isplaniran način. Kod linija programskih proizvoda unaprijed se definiraju i arhitektura sustava i funkcionalnosti komponenata. U njihovu bazu zajedničkih sredstava se ne stavlja ništa što nije bilo korišteno u toj liniji. Upravo to što linije programskih proizvoda koriste ponovnu upotrebu na striktan i planiran način je ključan razlog zašto takav razvoj uspijeva.

2.2. Potencijalni problemi kod linija programskih proizvoda

Linije programskih proizvoda mogu organizaciji dovesti napretke u produktivnosti rada, kvaliteti proizvoda te uvelike povećati brzinu isporuke proizvoda na tržište. No, moguće je i da organizacija pogriješi kod razvoja linije programskih proizvoda te postoje određeni faktori koji mogu dovesti do tih pogreški (Bass i sur., 2003., str. 361):

- Nedostatak vođe odnosno ekspertnog stručnjaka koji će kontrolirati razvoj linije programskih proizvoda
- Nesposobnost organizacije da pruža podršku za razvoj programske linije
- Nesposobnost identificiranja poslovnih ciljeva za korištenje linije programskih proizvoda
- Odustajanje organizacije od takvog razvoja proizvoda čim se naiđe na prvu prepreku koju ne mogu odmah riješiti
- Nesposobnost organizacije da pripremi i obrazuje zaposlenike za promjenu u pristupu razvoja softvera s individualnog razvoja na razvoj proizvodnih linija

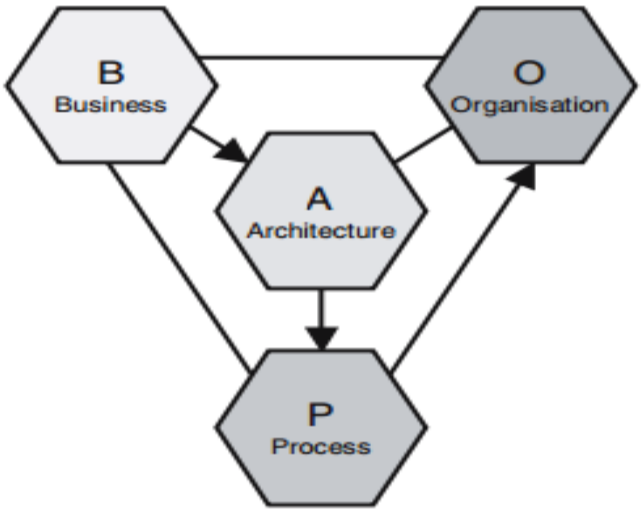
Ovih nekoliko problema može biti kobno za razvoj linija programskih proizvoda. No postoje strategije koje služe da se ovi potencijalni problemi zaobiđu. Jedna takva strategija bi bila da se u organizaciji razvije probna linija programskih proizvoda.

Takva probna linija demonstrira sve prednosti takvog načina razvoja softvera te se tako odredi struktura timova i riješe procesni problemi prije nego što se uopće počne s pravim razvojem linije programskih proizvoda (Bass i sur., 2003., str. 361).

2.3. BAPO model

BAPO (eng. *Business, Architecture, Process and Organisation*) model je kratica za četiri glavne perspektive razvoja kojima se organizacija mora posvetiti da bi razvila uspješnu liniju programskih proizvoda (Linden i sur., 2007., str. 16):

- Poslovna perspektiva razvoja linije programskih proizvoda
- Arhitekturalna perspektiva razvoja linije programskih proizvoda
- Procesna perspektiva razvoja linije programskih proizvoda
- Organizacijska perspektiva razvoja linije programskih proizvoda



Slika 3.: Glavni faktori kod razvoja linija programskih proizvoda

Izvor: Linden i sur. (2007., str. 17)

Na slici 3. možemo vidjeti da su sva četiri faktora usko povezana. Poslovna perspektiva razvoja je najutjecajniji faktor u ovom modelu te je odabir poslovne strategije prvo što se radi kod pokretanja razvoja proizvodnih linija. Arhitektura se oslanja na već određene poslovne strategije. Razvojni procesi omogućavaju izradu proizvoda te se temelje na arhitekturi. Te na kraju se na procesnu perspektivu nadovezuje organizacijska perspektiva razvoja koja se brine o raspodjeli zadataka i timova koji će raditi na razvoju proizvodne linije.

U sljedećih nekoliko poglavlja posebno će se obraditi svaki faktor BAPO modela te je prva na redu poslovna perspektiva razvoja linija programskih proizvoda.

3. Poslovna perspektiva razvoja linija programskih proizvoda

„Razvoj linija programskih proizvoda ima veliki utjecaj na poslovne uspjehe organizacija. To se odnosi i na proizvode koja organizacija trenutno proizvodi i za nova tržišta na koja želi ući.“ (Linden i sur., 2007., Str.21)

Svaka tehnika razvoja teži nekim poboljšanjima kvalitete i uštedama kod proizvodnje, no to je posebno istaknuto kod razvoja linija programskih proizvoda. Cilj linija programskih proizvoda je da se razvije platforma na kojoj će se temeljiti niz proizvoda koji će utjecati na cijelo tržište. Upravo zbog toga razvoj linije programskih proizvoda direktno utječe na reputaciju organizacije na tržištu.

Linden i sur. (2007., str. 21) navode poslovna pitanja pomoću kojih se oblikuje razvoj linija programskih proizvoda:

- Da li se isplati pokrenuti razvoj linije programskih proizvoda?
- Kako će linija programskih proizvoda utjecati na poslovne performanse?
- Koje proizvode bi trebali razvijati kao dio linije?
- Koje će biti glavne karakteristike tih proizvoda?
- Koji će se tajming odabrati za ulazak na tržište?

- Na koji način će proizvodi biti proizvedeni?
- Koje funkcionalnosti će biti razvijene kao dio linije, tj. u platformi, a koje će biti razvijene individualno za svaki proizvod?
- Na koji način će se pokrenuti razvoj linije unutar organizacije?
- Kako možemo unaprijediti liniju nakon nekog određenog vremena?

Da bi se na ta pitanja moglo odgovoriti prvo se trebaju utvrditi ciljevi te linije i tržišta kojima će se linija baviti te se mora ispitati ekonomičnost linije koje se želi razviti.

3.1. Tržišta linija programskih proizvoda

Kod tržišta linija programskih proizvoda postoje četiri faktora koja utječu na poslovnu strategiju linija programskih proizvoda. Linden i sur. (2007., str. 22) navode ta četiri faktora:

- Strategija definiranja proizvoda
- Strategija tržišta linija programskih proizvoda
- Životni ciklus linije programskih proizvoda
- Veza strategije tržišta i linije programskih proizvoda

3.1.1. Strategija definiranja proizvoda

Razvoj programskih proizvoda se može podijeliti u dvije klase. Prva klasa bi bila razvoj vođen korisničkim zahtjevima, dok bi druga bila razvoj vođen organizacijom. Kod razvoja vođenim korisničkim zahtjevima proizvodi se definiraju i razvijaju po specifičnim zahtjevima korisnika. U takvom razvoju teško je unaprijed odrediti konstantne zahtjeve za platformu te u tom slučaju platforma mora biti fleksibilna za

razvoj po specifičnim zahtjevima. Takav razvoj se naziva razvoj masovne prilagodbe. Druga klasa je razvoj vođen organizacijom te u tom slučaju organizacija definira većinu proizvoda. Ovakav razvoj se koristi kada se proizvodi razvijaju za masovna tržišta gdje je svaki proizvod prodan u stotinama tisućama primjeraka (Linden i sur., 2007., str 22.).

Linija programskih proizvoda podržava oba pristupa razvoju te odabir koji će se pristup koristiti ovisi o veličini uštede proizvodnje koje bi organizacije imala s pojedinim pristupom.

3.1.2. Strategije tržišta linije programskih proizvoda

Strategija tržišta omogućuje organizacijama odabir u kojem svijetlu se žele prikazati na tržištu. Postoje tri tipa organizacijske strategije tržišta (Linden i sur.,2007., str. 23):

- Strategija vođena uštedama:
- Diferencijacija:
- Fokusiranje:

Kod strategije vođene uštedama organizacijski ciljevi su da razvijaju proizvode s najmanjim mogućim troškovima proizvodnje. Kod diferencijacije organizacijski ciljevi su da se može bitno razlikovati njihove proizvode od proizvoda njihove konkurencije. Diferencijacija se može postići na tri načina. Prvi način bi bio poboljšanje nekog dijela proizvoda da se taj dio proizvoda bitno razlikuje od konkurentskog proizvoda. Drugi način je da se proizvod koji se dostavlja na tržište inovativan što znači da je organizacija inovativna te poboljšava javno mišljenje o njoj. Te je treći način brzina, odnosno da se nove tehnologije dostupne na tržištu primjene na vlastitim proizvodima u što kraćem roku, tj. prije konkurencije. Treća organizacijska strategija za tržište bi bila fokusiranje. Kod strategije fokusiranja organizacija se specijalizira u specifičnom području te su njeni proizvodi u tom području detaljnije razvijeni od konkurencije.

Linije programskih proizvoda mogu koristiti bilo koju od ovih strategija zasebice te u nekim slučajevima mogu kombinirati više strategija da bi maksimizirale učinak svoje linije.

3.1.3. Životni ciklus linije programskih proizvoda

Da bi mogli shvatiti životni ciklus linije programskih proizvoda prvo ćemo opisati faze životnog ciklusa jednog proizvoda (Linden i sur., 2007., str . 24.):

- Ulazak proizvoda na tržište:

  U ovom stadiju proizvod je tek razvijen te je početna prodaja veoma niska jer je proizvod još uvijek nepoznat.

- Rast proizvoda:

  U stadiju rasta proizvoda on postaje sve poznatiji na tržištu te prodaja raste.

- Zrelost proizvoda:

  Kod zrelosti proizvoda rast prodaje se smanjuje te se moraju smanjiti cijene da bi se ponovno osvojilo tržište

- Zasićenost proizvoda:

  Kada tržište postane zasićeno proizvodom to znači da je postignuta maksimalna prodaja proizvoda te je vjerojatno konkurencija postala velika.

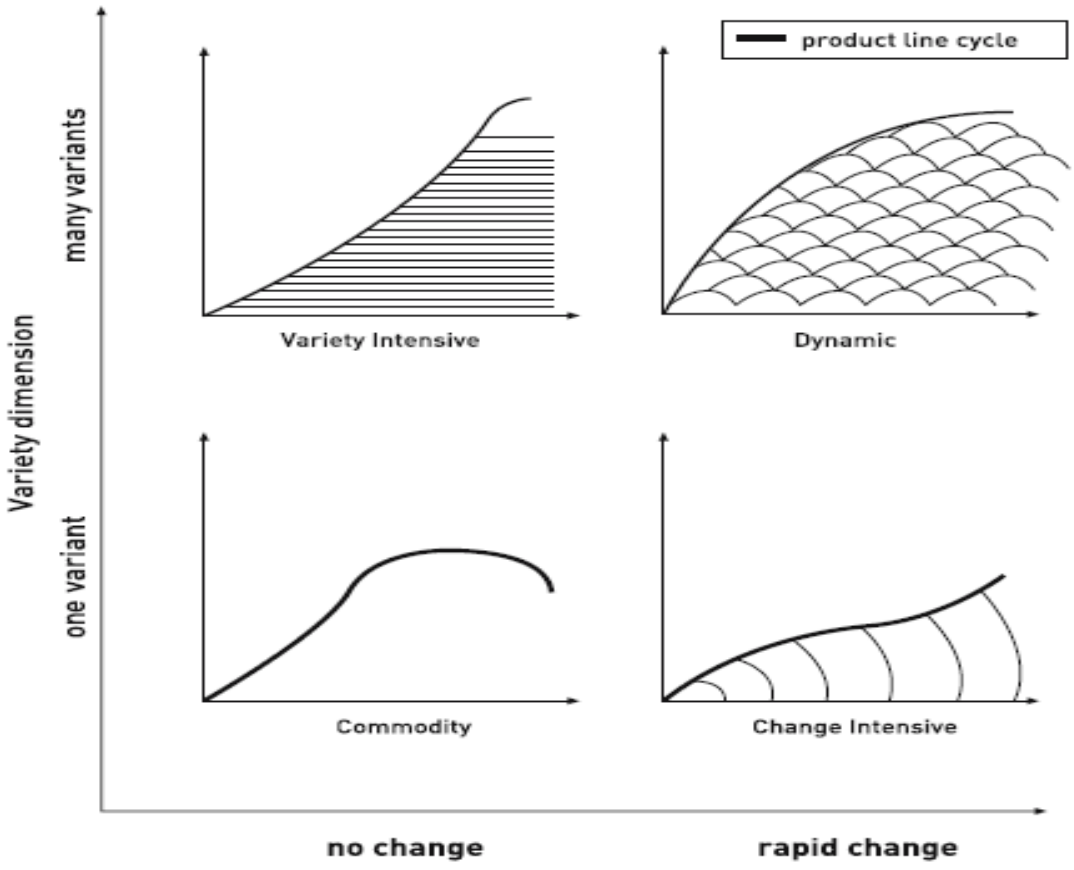
- Degeneracija proizvoda:

  U fazi degeneracije proizvoda prodaja je značajno pala te je proizvod zamijenjen novijima. Treba spomenuti da ti novi proizvodi mogu biti razvijeni od strane iste organizacije.

Kada se želi odrediti životni ciklus linije programskih proizvoda treba uzeti u obzir da se linija sastoji od više proizvoda. S obzirom na to postoje dva slučaja. Prvi slučaj je da na tržištu u isto vrijeme postoji više proizvoda iz iste linije. U tom slučaju ti proizvodi si međusobno smanjuju prodaju te se to naziva kanibalizacija linije programskih proizvoda (Linden i sur., 2007., str 24.). Drugi slučaj je kada se na

tržištu različiti proizvodi iz iste linije zamjenjuju nakon nekog vremenskog roka. S obzirom na ova dva slučaja postoji više mogućih životnih ciklusa linije programskih proizvoda.

Na slici 4. možemo vidjeti da se mogući životni ciklusi mogu podijeliti u dvije dimenzije, varijacijsku dimenziju i dimenziju serijskih promjena. Varijacijska dimenzija nam pokazuje koliko različitih proizvoda iz programske linije postoji na tržištu u isto vrijeme, dok nam dimenzija serijske promjene pokazuje kojom se brzinom proizvodi zamjenjuju. Kao što vidimo na slici kroz ove dvije dimenzije postoje četiri kategorije životnog ciklusa linije te organizacija mora istražiti tržište na koje želi ući s proizvodnom linijom da bi mogla odrediti koji od ovih životnih ciklusa je najprikladniji za njihovu liniju.



Slika 4.: Mogući životni ciklusi linija programskih proizvoda s obzirom na varijaciju i brzinu promjene

Izvor: Linden i sur. (2007., str. 25)

3.1.4. Veza strategije tržišta i linije programskih proizvoda

„U prošlosti, organizacije su realizirale sve marketinške strategije s različitim razvojnim pristupima od obične ponovne upotrebe do razvoja jednog sustava. No linije programskih proizvoda mogu funkcionirati sa strategijom tržišta u uspješnoj simbiozi.“ (Linden i sur., 2007., str 26.)

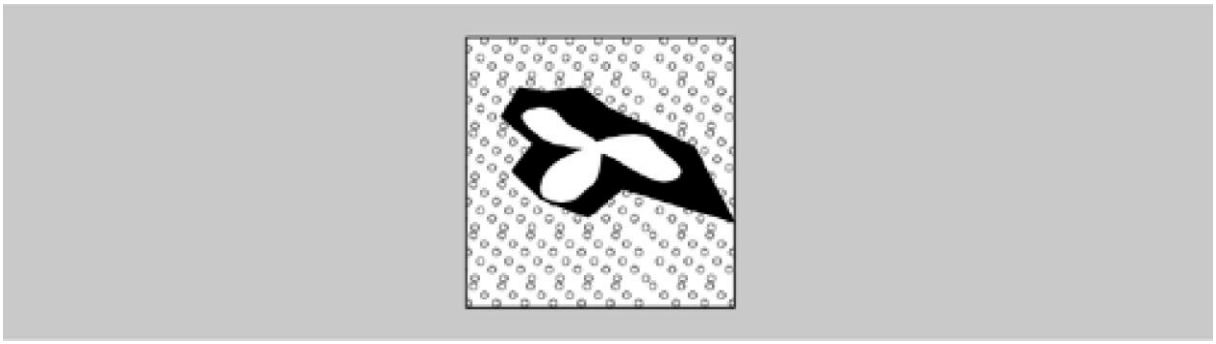
Ono što je veoma bitno za shvatiti kod linija programskih proizvoda je to da ona može podržati tržišnu strategiju. Razlog tome je to što linija programskih proizvoda može uvelike smanjiti troškove i vrijeme koje je potrebno za dostavljanje novog proizvoda. Upravo zbog toga je linija programskih proizvoda odlična za organizacije koje su odabrale poslovanje pomoću strategije vođenom uštedama jer im linija omogućuje da razvijaju proizvode po puno nižim cijenama nego kod tradicionalne proizvodnje. Također treba napomenuti da je i strategija diferencijacije veoma korisna jer organizacije mogu dostaviti više proizvoda na tržište s manjim troškovima od konkurencije koja koristi tradicionalan razvoj.

Treba napomenuti i to da organizacije moraju odabrati ispravnu strategiju za svoju liniju. Pošto kod linije programskih proizvoda imamo generičku platformu koja nam je baza za sve proizvode, mora se koristiti tržišna strategija koja odgovara našoj platformi. Razlog tome je što je jako teško razviti proizvod koji nije kompatibilan s razvijenom platformom te je od izrazite važnosti da je tržišna strategija usko povezana s našom linijom programskih proizvoda. Da bi tržišna strategija mogla biti usko povezana s našom linijom prvo se mora napraviti opseg linija programskih proizvoda.

3.2. Opseg linija programskih proizvoda

Da bi se linija programskih proizvoda uspješno razvila to zahtjeva planiranje opsega te linije unaprijed, na organizacijskom nivou. Opseg jedne linije bi bio unaprijed izrađen plan u kojem su definirani koje vrste proizvoda će biti izrađene a koje neće. Organizacija mora strateški odrediti opseg prije nego što se krene u razvoj jedne

linije programskih proizvoda da bi se moglo odabrati strategiju tržišta. Opseg linije programskih proizvoda opisat ćemo pomoću sljedeće slike:



Slika 5.: Primjer opsega linije programskih proizvoda

Izvor: Bass i sur. (2003., str. 360)

Na slici 5. prikazan je prostor svih mogućih sustava. Prostor je podijeljen na tri područja. Područje u sredini prostora prikazano bijelom bojom predstavlja sve sustave koje bi organizacija mogla izraditi jer spadaju u njen predodređeni opseg. Područje koje je šareno obilježeno predstavlja sustave koji su izvan opsega. To bi značilo da organizacija nije dovoljno pripremljena za izradu takvih sustava unutar vlastite linije programskih proizvoda. Područje koje je obilježeno crnom bojom predstavlja sustave koji ne spadaju direktno u opseg linije programskih proizvoda ali bi mogli biti razvijeni ako za njihov razvoj postoje dobri razlozi. Bass i sur. (2003., str. 360) su dali primjer u liniji sustava za automatizaciju ureda. U takvoj liniji planer za konferencijsku sobu bi bio u bijelom području tj. bio bi uključen u liniju, simulator leta bio bi u šarenom području tj. izvan opsega linije, dok bi u crnom području bila specijalizirana intranet tražilica koja bi možda jednog dana bila razvijena u liniji ako bi se mogla razviti u razumnom vremenu i ako bi za njen razvoj postojali strateški razlozi kao na primjer neko očekivanje da bi budući klijenti željeli takav proizvod.

Opseg linije programskih proizvoda može se opisati i kao predviđanje organizacije koje proizvode će klijenti tražiti od njih u budućnosti. Za determiniranje tog opsega organizacija koristi strateške planere, marketinško osoblje, analiste područja kojih su im u interesu te ekspertne tehnološke stručnjake.

Treba se napomenuti da je određivanje opsega jedan od ključnih faktora kada se želi pokrenuti razvoj linija programskih proizvoda te ako se pogriješi u ovom dijelu razvoja cijela linija može propasti. Opseg se može pogrešno odrediti na dva načina. Prvi način bi bio da se opseg postavi preusko odnosno proizvodi u liniji se razlikuju u malom broju funkcionalnosti tj. preslični su. Tada organizacija neće moći razviti dovoljan broj proizvoda koji će opravdati investicije za razvoj linije programskih proizvoda. Drugi način bi bio da se opseg odredi preopširno odnosno proizvodi koje organizacija planira proizvesti se previše razlikuju u funkcionalnostima. Tada se ne isplati imati zajedničku bazu sredstava iz koje bi se razvila linija aplikacija jer će trud za doradu funkcionalnosti kod svake aplikacije biti prevelik da bi se takva linija isplatila. Bass i sur. (2003., str. 360) navode da problem kod definiranja opsega nije u pronalasku zajedničkih značajki kod različitih sustava, jer će kreativni arhitekt uvijek pronaći zajedničke značajke između bilo koja dva sustava već da je ključno pronaći zajedničke značajke koje se mogu iskoristiti tako da bitno smanje troškove razvoja sustava koje organizacija želi izraditi.

Spomenuli smo da kod određivanja opsega linije programskih proizvoda organizacije uz specijalizirane stručnjake koriste i marketinško osoblje. To treba naglasiti jer je istraživanje tržišta i vrsta kupaca veoma ključno kod određivanja opsega linije. Bass i sur. (2003., str. 360) navode primjer nizozemskog proizvođača elektronike Philips. Philips ima različite proizvodne linije za elektroničke video sustave te sustave za digitalnu video komunikaciju. Philips je mogao razviti jednu produktnu liniju za oba dva sustava, no uz pomoć marketinškog osoblja i istraživanja kupaca, zaključili su da su elektronički video sustavi na jako velikom tržištu te da kupci imaju malo znanja o takvim sustavima, dok je kod digitalne video komunikacije tržište manje a kupci su bolje upoznati s takvim sustavima. Upravo iz tih razloga Philips je odlučio razviti dvije posebne proizvodne linije za oba tržišta.

Kao što smo naveli u uvodu drugog poglavlja, linije programskih proizvoda rezultiraju ekonomičnijom proizvodnjom te je to tema sljedećeg poglavlja.

3.3. Ekonomičnost linija programskih proizvoda

Kod razvoja linija programskih proizvoda ekonomičnija proizvodnja se ostvaruje tako što se programski proizvodi razvijaju iz zajedničkog skupa sredstava koji su već razvijeni, za razliku od razvijanja programskog proizvoda samostalno na tradicionalan način. Upravo zbog tih razloga je takav pristup razvijanju softvera privlačan. Linden i sur. (2007., str. 27.) navode listu prednosti koje razvoj proizvodne linije može pružiti:

- Smanjeni troškovi razvoja:

  Pošto se kod razvoja novog proizvoda u liniji većina dijelova ponovno iskorištava, razvoj novog proizvoda je puno jednostavniji te je zbog toga smanjen trošak razvoja novog proizvoda.

- Smanjeno vrijeme razvoja:

  Zbog toga što je razvoj novog proizvoda puno jednostavniji to također znači i da će se taj proizvod razviti u punom kraćem roku od tradicionalnog načina.

- Poboljšana korisnost proizvoda:

  Korištenjem jednakih korisničkih sučelja kroz proizvodnu liniju poboljšava se korisnost proizvoda zato što se korisnici ne moraju privikavati na novo sučelje kod svakog novog proizvoda.

- Jednostavnija portabilnost:

  Prebacivanje proizvoda s jedne platforme na drugu se u slučaju proizvodnih linija se može postaviti kao točka varijabilnosti na generičkoj aplikaciji te tada portabilnost postaje jednostavnija za izvršiti.

- Lakše održavanje:

  Pošto proizvodi koriste većinu istih komponenti od platforme, održavanje postaje lakše jer je jednostavnije održavati samo platformu nego svaki proizvod zasebno.

Uobičajeno je da organizacija odabere neke od ovih potencijalnih prednosti te s obzirom na te prednosti odabere pripadajuću strategiju koja će im pomoći ispuniti te prednosti.

U ovom poglavlju smo utvrdili da je poslovna strana linije programskih proizvoda veoma bitna za poboljšanje uspjeha linije te da je za uspjeh ključno pronaći vezu između poslovne strategije i tehničke strane izvedbe proizvodne linije. Kao što smo već spomenuli po BAPO modelu se na poslovnu perspektivu razvoja linije programskih proizvoda oslanja arhitekturalna perspektiva razvoja te je to tema sljedećeg poglavlja.

4. Arhitekturalna perspektiva kod razvoja linija programskih proizvoda

U prošlom poglavlju smo spomenuli da za razvoj uspješnih linija programskih proizvoda organizacije trebaju odrediti poslovne strategije na kojima će se temeljiti razvoj proizvodne linije. Arhitektura se nadovezuje na te poslovne strategije te se izrađuje sukladno s njima. Prema SEI-u (eng. Software Engineering Institute) razvoj proizvodnih linija se temelji na arhitekturi te uspjeh svih ostalih dijelova ovisi o dizajnu arhitekture.

Arhitektura se mora razviti tako da služi kao podloga svim proizvodima u liniji. Takva djeljiva arhitektura se može nazvati referentna arhitektura (eng. *reference architecture*). To je arhitektura koja je generalizirana tako da pruža podršku različitim funkcionalnostima koje koriste proizvodi unutar proizvodne linije. Referentna arhitektura se razvija u domenskom razvoju te se zatim koristi u aplikacijskom razvoju kao podloga za razvoj individualnih proizvoda.

Jedan od izazova kod razvoja arhitekture linije je to što postoje različiti korisnički zahtjevi za individualne proizvode. Cilj je pronaći način da se pruži podrška tim varijabilnostima proizvoda kroz referentnu arhitekturu. To se može postići kroz nekoliko tehnika koje su opisane u sljedećem poglavlju.

4.1. Tehnike realizacije varijabilnosti u referentnoj arhitekturi

Linden i sur. (2007., str. 40) navode tri tehnike realizacije varijabilnosti u arhitekturi:

- Tehnika adaptacije:

  Kod tehnike adaptacije postoji samo jedan način implementacije pojedine komponente, ali postoje sučelja koja mogu prilagoditi ponašanje tih komponenti za potrebe pojedinih proizvoda.

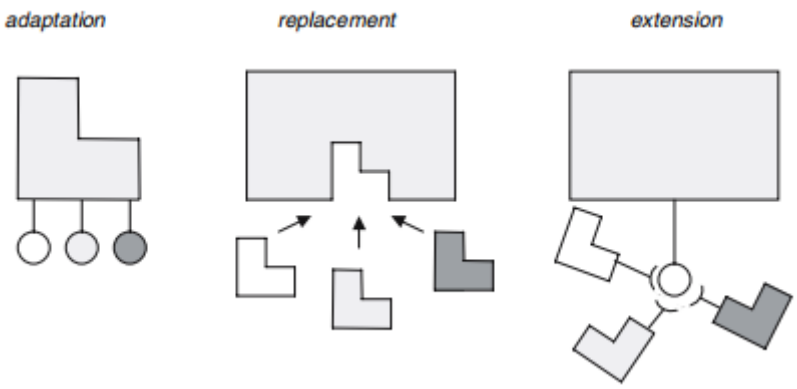
- Tehnika zamjene:

  Kod tehnike zamjene dostupno je nekoliko implementacija pojedine komponente. Zatim se u aplikacijskom razvoju proizvoda ili odabire jedna od postojećih implementacija ili se razvija specifična implementacija za taj proizvod.

- Tehnika proširenja:

  Kod tehnike proširenja dodaju se nove komponente. Da bi se ova tehnika mogla koristiti referentna arhitektura mora biti dizajnirana tako da podržava dodavanje novih komponenti. Novo dodane komponente mogu biti generičke komponente koje će nam poslužiti kod drugih proizvoda ili mogu biti specifične za jedan proizvod.

  Na slici 6. možemo vidjeti vizualan prikaz tih tehnika.



Slika 6.Tehnike za realizaciju varijabilnosti u arhitekturi

Izvor: Linden i sur. (2007., str.41)

4.2. Evaluacija referentne arhitekture

Bass i sur. (2003., str 364.) navode da je evaluacija referentne arhitekture veoma važan korak u razvoju jedne linije s obzirom na to da veliki broj sustava ovisi o toj arhitekturi. Evaluacija arhitekture se svodi na procjenu robusnosti te da li ta arhitektura može služiti kao podloga za izradu proizvoda koji su uključeni u opseg linije. Glavni fokus evaluacije mora biti na točkama varijacije. Mora se potvrditi da su točke varijacije dobro određene, da su dovoljno fleksibilne da pokriju cijeli opseg linije, da dozvoljavaju da se proizvodi razviju brzo te da ne nameću visoke troškove razvoja. Također čest je slučaj da su maksimalne performanse proizvoda u proizvodnoj liniji nepoznate te se u tom slučaju evaluacijom postavljaju maksimalne performanse koje takva arhitektura može postići. Evaluacijom se također utvrđuju potencijalni problemi s arhitekturom te se zatim može na vrijeme reagirati s određenim strategijama za rješavanje tih problema.

Ako se razvija proizvod koji ne spada u prvobitni opseg linije programskih proizvoda, arhitektura se mora ponovno evaluirati da se ispita hoće li taj proizvod moći biti razvijen na osnovu te arhitekture. Ako se može razviti, opseg linije biti će proširen s tim proizvodom, dok ako se ne može razviti s takvom arhitekturom, evaluacija će pomoći kod određivanja modifikacija za arhitekturu da bi odgovarala razvijanju novog

proizvoda. Modifikacija arhitekture se također naziva i evolucija arhitekture te je to tema sljedećeg poglavlja.

4.3. Evolucija referentne arhitekture

Nakon nekog vremena postojanja linije programskih proizvoda pojavit će se proizvod s novijim zahtjevima. Spomenuli smo da se tada prvo evaluira sposobnost arhitekture da pruži podršku tim zahtjevima. Kada arhitektura ne može pružiti podršku tim zahtjevima tada referentna arhitektura mora evoluirati te se to zove namjerna evolucija (eng. *intentional evolution*).

No evolucija se može dogoditi i bez nekog cilja. Takva evolucija se zove nenamjerna evolucija (eng. *unintentional evolution*) te takva evolucija može dovesti do dva problema. Prvi problem je razlika između dokumentirane arhitekture i njene implementacije. Linden i sur. (2007., str 44.) navode da ako se nenamjerna evolucija ne nadgleda i kontrolira doći će do toga da se programski proizvod neće moći razumjeti pomoću dokumentacije. Drugi problem može biti arhitekturalna erozija (eng. *architectural erosion*). Arhitekturalna erozija znači da se dizajn programskog proizvoda nenamjerno uništio zbog puno malih promjena. Treba napomenuti da se problem arhitekturalne erozije može spriječiti putem refaktoriranja (eng. *refactoring*) tj. unaprjeđuje se dizajn arhitekture bez da se mijenjaju funkcionalnosti.

U sljedećem poglavlju prikazan je primjer arhitekture programskog proizvoda.

4.4. Primjer referentne arhitekture

Manger (2013., str 143.) nam navodi primjer linije programskih proizvoda tj. referentnu arhitekturu linije sustava za upravljanje inventarom.

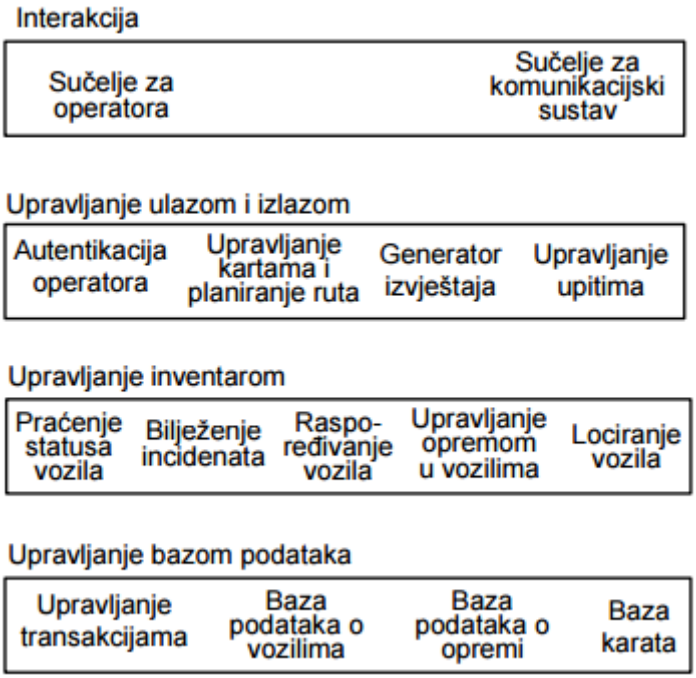
Slika 7.: Referentna arhitektura linije programskih proizvoda za upravljanje inventarom

Izvor: Manger (2013., str. 144)

Na slici 7. je prikazana referentna arhitektura linije sustava za upravljanje inventarom. U ovom primjeru prati se raspored komada opreme unutar organizacije. Ako se pretpostavi da je u ovom primjeru riječ o elektrani i generatorima i transformatorima što bi bila njezina oprema onda oprema ne mijenja svoju lokaciju te ova arhitektura odgovara takvom sustavu. Dok ako se primjerice isti taj sustav želi koristiti za praćenje opreme na fakultetu mora se dodati novi modul koji će služiti za premještanje računala iz jednog kabineta u drugi. Na ovaj način se razvija novi

proizvod u liniji programskih proizvoda. U ovom slučaju za razvoj novog proizvoda dodali smo samo jedan modul. (Manger, 2013., str. 143)

U sljedećem primjeru kojeg možemo vidjeti na slici 8. je arhitektura sustava za raspoređivanje vozila hitne pomoći koja je dosta kompliciranija nego arhitektura polazne aplikacije na slici 7. Ovaj primjer nam pokazuje kako se od referentne arhitekture izrađuje novi proizvod. Koriste se oni moduli arhitekture koji odgovaraju specifičnim zahtjevima novog proizvoda te se dodaju novi moduli koji će zadovoljiti te nove zahtjeve. U ovom slučaju dodani su novi specifični moduli koji služe za rad s kartama, planiranja ruta vozila, za sučelje prema komunikacijskom sustavu, itd., te su postojeći moduli modificirani da bi odgovarali novim zahtjevima primjerice modul praćenja inventara je postao modul koji služi za praćenje statusa vozila.



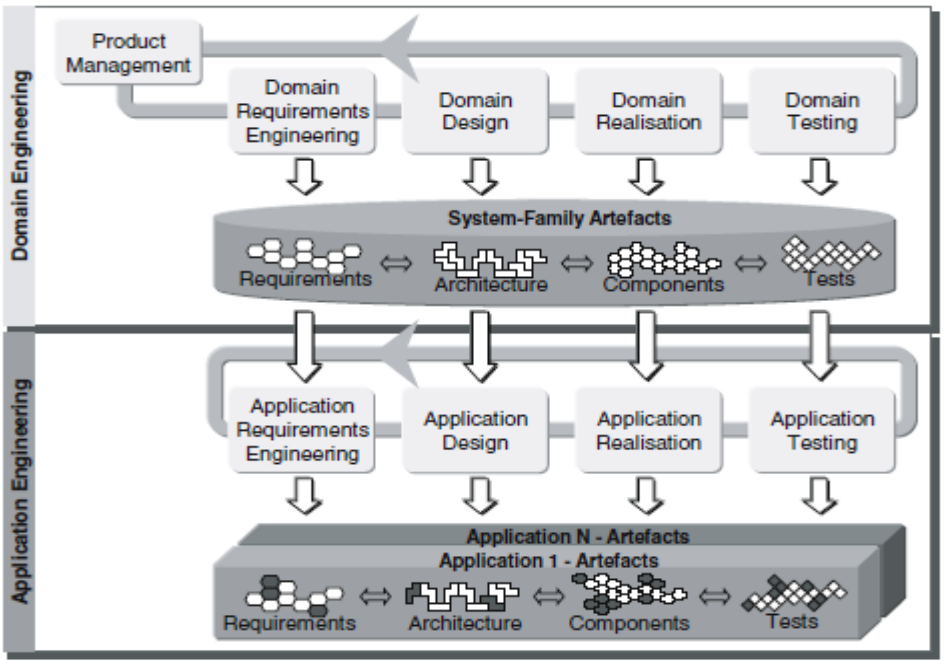
Slika 8.: Arhitektura jednog proizvoda u liniji

Izvor: Manger (2013., str. 144)

5. Procesna perspektiva razvoja linije programskih proizvoda

Razvojni proces linije programskih proizvoda se može podijeliti u dva životna ciklusa. Domenski razvoj i aplikacijski razvoj. U domenskom razvoju se razvija platforma koja će služiti kao podloga razvoju individualnih proizvoda u aplikacijskom razvoju. Zatim se u aplikacijskom razvoju razvijaju individualni proizvodi u liniji.

Na slici 9. Može se vidjeti da unutar ova dva životna ciklusa postoji devet procesa. Osam procesa zapravo čine četiri para procesa pošto se određivanje zahtjeva, dizajniranje, realizacija i testiranje radi i u domenskom razvoju i u aplikacijskom razvoju te su ti procesi međusobno povezani. Procesi u domenskom razvoju nam daju kao rezultat komponente koje će biti zajedničke za sve proizvode, dok nam procesi u aplikacijskom rezultiraju korištenjem tih komponenti za razvoj novog proizvoda te povratnim informacijama koje se prosljeđuju u domenski razvoj te poboljšavaju razvoj platforme.



Slika 9.: Dvociklusni model linije programskih proizvoda

Izvor: Linden i sur. (2007., str. 48)

Kroz sljedeća dva poglavlja detaljnije će se obraditi domenski razvoj i aplikacijski razvoj.

5.1. Domenski razvoj

Cilj domenskog razvoja je da se napravi platforma koja će se temeljiti na referentnoj arhitekturi te će sadržavati komponente koje će biti zajedničke svim proizvodima. U prošlom primjeru na slici 9. se moglo vidjeti da se životni ciklus domenskog razvoja sastoji od pet procesa.

Prvi proces je upravljanje proizvodima te se u njemu moraju odrediti tržišne strategije i opseg proizvoda linije. Nakon što se odredio opseg proizvoda mogu se prepoznati komponente koje će biti zajedničke te koje će funkcionalnosti biti varijabilne kroz pojedine proizvode. U ovom procesu se također obavlja dokumentacija proizvoda te se razvija planska mapa koja opisuje buduće proizvode i datume njihovih izlaska na tržište.

Drugi proces je određivanje domenskih zahtjeva. Taj proces služi za određivanje i upravljanje zahtjevima koji će služiti za izgradnju referentne arhitekture. Zahtjevi koji se određuju moraju uključivati sve moguće zahtjeve koji će biti zajednički kroz sve aplikacije u proizvodnoj liniji.

Linden i sur. (2007., str 49) navode pet osnovnih faza određivanja domenskih zahtjeva:

- Prikupljanje zahtjeva:

  U ovom koraku se analiziraju potrebe korisnika i investitora.

- Dokumentacija zahtjeva:

  Kod ovog koraka se zapisuju proizvodni zahtjevi na veoma precizan način.

- Pregovaranje o zahtjevima:

  Pregovaranje s investitorima o zahtjevima.

- Validacija i verifikacija zahtjeva:

  Cilj validacije i verifikacije je da zahtjevi budu jasni, potpuni, ispravni i razumljivi.

- Upravljanje zahtjevima:

  Ova faza se bavi održavanjem zahtjeva tijekom cijelog životnog ciklusa domenskog razvoja. Kada se otkriju novi zahtjevi onda se tijekom ove faze često se vraća na neke od prijašnjih faza.

Treći proces je dizajn domene. Nakon što su se odredili zahtjevi kreira se referentna arhitektura koja se temelji na tim zahtjevima. Ta arhitektura onda služi kao podloga za proizvode kod aplikacijskog dizajna.

Četvrti proces je realizacija domene. Cilj realizacije domene je da se odrede komponente koje će se koristiti kroz sve proizvode. Linden i sur. (2007., str. 51) navode da postoji četiri načina kako doći do komponente:

- Napraviti komponentu:

  Komponentu izrađuje organizacija koja razvija proizvodnu liniju. Prednost samostalnog razvijanja komponente je to što organizacija ima potpunu kontrolu nad specifikacijom i implementacijom komponente.

- Kupiti komponentu:

  Komponenta se kupuje kao gotov proizvod. Primjer može biti kupnja operativnog sustava za računalo.

- Rudarenje komponente:

  Koristi se komponenta koja već postoji u nekom proizvodu unutar linije programskih proizvoda.

- Iznajmljivanje komponente:

  Specifikacije komponente se kreiraju u vlastitoj organizaciji ali se ta komponenta daje na izradu drugoj kompaniji.

Peti i zadnji proces je testiranje domene. Testiranje domene je proces otkrivanja grešaka u platformi. Ovo je veoma važan korak u izrađivanju platforme jer ako se neka greška izostavi, ta greška će postojati na svim proizvodima koji koriste komponentu na kojoj je greška. Linden i sur. (2007., str. 52) navode nekoliko tipova testiranja:

- Dinamičko testiranje:

  Dinamičko testiranje se bavi testiranjem individualnih komponenti i sučelja, te sustavnim testiranjem za cijelu platformu.

- Regresijsko testiranje:

  Regresijsko testiranje se bavi provjeravanjem implementacije komponente tj. dali razvijena komponenta odgovara predviđenim specifikacijama.

- Testiranje prihvatljivosti:

  Testiranje prihvatljivosti se radi kada je organizacija nabavila komponente od neke druge organizacije te se mora provjeriti dali ta komponenta odgovara platformi.

Nakon što su se izvršili ovi procesi, uzimaju se razvijene komponente te ih se zatim koristi u aplikacijskom razvoju za izradu individualnih proizvoda.

5.2. Aplikacijski razvoj

U aplikacijskom razvoju uzimaju se kreirane komponente iz domenskog razvoja te ih se koristi za razvoj pojedinih proizvoda. U životnom ciklusu aplikacijskog razvoja postoje četiri procesa koja se nadovezuju na procese iz domenskog razvoja.

Prvi proces je određivanje aplikacijskih zahtjeva. Taj proces se nadovezuje na zahtjeve koji su određeni u domenskom razvoju te ih se nadopunjuje se specifičnim zahtjevima klijenata ili investitora. U ovom procesu se ispituje koliko je novih zahtjeva koji nisu ispunjeni komponentama platforme. Ako postoji novi zahtjev koji nije ispunjen treba se odlučiti dali se isplati zadovoljavati zahtjev. Ako se ne isplati

pregovara se s investitorima da se izmijeni zahtjev tako da se može ispuniti s komponentama koje postoje u platformi ili se odbacuje taj zahtjev.

Treba napomenuti da ako se često pojavljuje jednaki novi zahtjev kod mnogo aplikacija treba se dati povratna informacija domenskom razvoju. Domenski razvoj bi zatim trebao implementirati taj zahtjev u sljedećoj verziji platforme.

Drugi proces je dizajn aplikacije. Kod dizajna aplikacije uzima se referentna arhitektura koja je razvijena u dizajnu domene te se nadograđuje ta arhitektura s komponentama koje će zadovoljavati nove specifične zahtjeve.

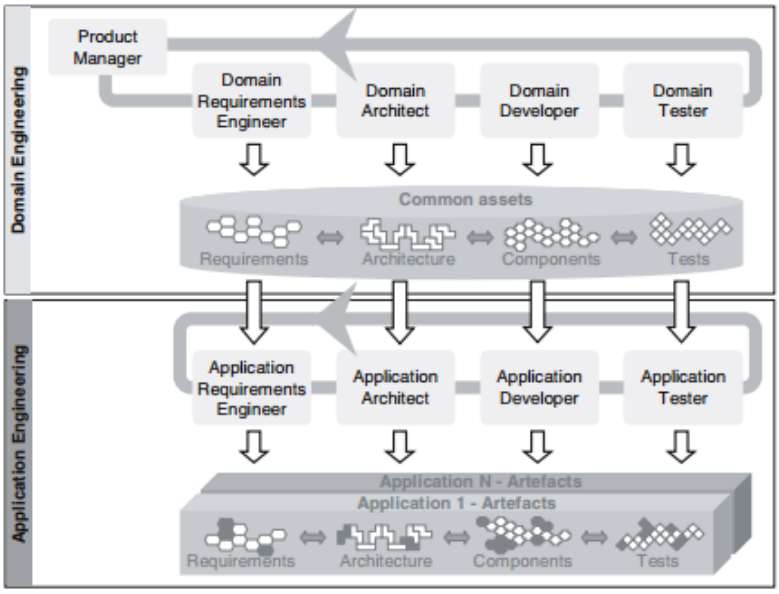
Treći proces je realizacija aplikacije. U tom procesu se implementiraju proizvodi. Koriste se komponente platforme koje su potrebne za razvoj pojedinog proizvoda te se implementiraju nove koje su određene u dizajnu aplikacije.

Četvrti proces je testiranje aplikacije. Testiranje aplikacije se radi da bi se utvrdilo da li je razvijena aplikacija dovoljno kvalitetna za tržište. Iako su se neke komponente testirale u domenskom razvoju, moraju se opet testirati jer se često dogodi da su te komponente modificirane da bi odgovarale novim komponentama.

U ovom poglavlju obradili smo razvojne procese koje služe za razvijanje linije programskih proizvoda te smo utvrdili da postoje dva životna ciklusa razvoja linije, domenski i aplikacijski razvoj. U domenskom razvoju izrađuje se platforma s komponentama koje će biti zajedničke budućim proizvodima, dok se u aplikacijskom razvoju koriste te komponente za izradu individualnih proizvoda. U sljedećem poglavlju bavit ćemo se organizacijskom perspektivom razvoja linije programskih proizvoda koja služi za raspodjelu uloga i zadataka kroz ove obrađene procese.

6. Organizacijska perspektiva razvoja linije programskih proizvoda

U prošlom poglavlju obradili smo procesnu perspektivu razvoja jedne linije programskih proizvoda. Da bi se ti procesi mogli uspješno izvršiti organizacija treba ispravno odrediti i mapirati aktivnosti i uloge zaposlenika u razvojnom procesu.



Slika 10.: Uloge zaposlenika u razvoju linija programskih proizvoda

Izvor: Linden i sur. (2007., str. 62)

Organizacija dodjeljuje zadatke i uloge na osnovu razvojnih procesa. Ti zadatci i uloge mogu biti dodijeljeni jednoj ili više osoba ovisno o veličini organizacije. Na slici 10. možemo vidjeti da postoji devet glavnih uloga koje kontroliraju devet razvojnih procesa a to su:

- Menadžeri proizvoda (eng. *product managers*):

  Zadatak menadžera proizvoda je da odrede koji će se proizvodi razvijati, kako će se razvijati i zašto će se razvijati. Menadžeri proizvoda evaluiraju postojeće proizvode te planiraju razvoj budućih proizvoda tj. njihove funkcionalnosti i njihove poslovne vrijednosti. Za organizaciju su poslovne vrijednosti posebno važne jer to uključuje planiranje budućih troškova i profita pojedinih proizvoda.

Također treba napomenuti da je menadžer proizvoda taj koji pokreće proizvodnju novog proizvoda, da svi ostali zaposlenici usko rade s njim te on uz pomoć inženjera aplikacijskih zahtjeva definira funkcionalnosti novog proizvoda.

- Inženjeri domenskih zahtjeva (eng. *domain requirements engineers*):

  Posao inženjera domenskih zahtjeva se sastoji od razvoja zahtjeva koji će biti zajednički kroz cijeli proizvodnu liniju te od utvrđivanja varijabilnih zahtjeva koji će se koristiti za pojedine proizvode.

- Arhitekti domene (eng. *domain architects*):

  Domenski arhitekti razvijaju i održavaju referentnu arhitekturu koja se temelji na zahtjevima koje su utvrdili inženjeri zahtjeva.

- Programeri domene (eng. *domain developers*):

  Posao domenskih programera je da za cijelu proizvodnu liniju razvijaju i održavaju ponovno upotrebive komponente i sučelja koja su dizajnirana u referentnoj arhitekturi.

- Ispitivaču domene (eng. *domain testers*):

  Ispitivači razvijaju i održavaju testne komponente za cijelu liniju programskih proizvoda. Uz to također moraju napraviti sustavne testove i testove integracije te pripremaju testove varijabilnosti za aplikacijske ispitivače.

- Inženjeri aplikacijskih zahtjeva (eng. *application requirements engineers*):

  Uloge inženjera aplikacijskih zahtjeva su razvoj i održavanje zahtjeva individualnog proizvoda. Njihov rad se temelji na definiranim domenskim zahtjevima. Ako postoje neki novi zahtjevi koji nisu određeni u domenskom razvoju njihov je posao da ih odrede i održavaju te je moguće da se ti novi zahtjevi uključe u domenske zahtjeve te u tom slučaju oni usko surađuju s inženjerima domenskih zahtjeva.

- Arhitekti aplikacije (eng. *application architect*):

  Arhitekti aplikacije moraju razviti i održavati arhitekturu individualnog proizvoda. Ta arhitektura se temelji na referentnoj arhitekturi koju su izradili arhitekti domene. Ako postoje neki problemi s referentnom arhitekturom arhitekti aplikacije prijavljuju te probleme domenskim arhitektima. Također prosljeđuju i informacije o komponentama koje će koristiti u novom proizvodu te koje novo razvijene komponente bi mogle biti kandidati za ulazak u referentnu arhitekturu.

- Programeri aplikacije (eng. *application developers*):

  Uloga programera aplikacije je da razviju i održavaju komponente koje su potrebne za novi proizvod. To uključuje i ponovnu upotrebu komponenti koje su razvili domenski programeri te uključuje i razvoj novih komponenti koje su specifične za novi proizvod. Ako postoje problemi s ponovno korištenim komponentama programeri aplikacije prijavljuju te probleme domenskim programerima. Također prosljeđuju domenskim programerima komponente i sučelja koje bi mogle postati dio platforme.

- Ispitivači aplikacije (eng. *application testers*):

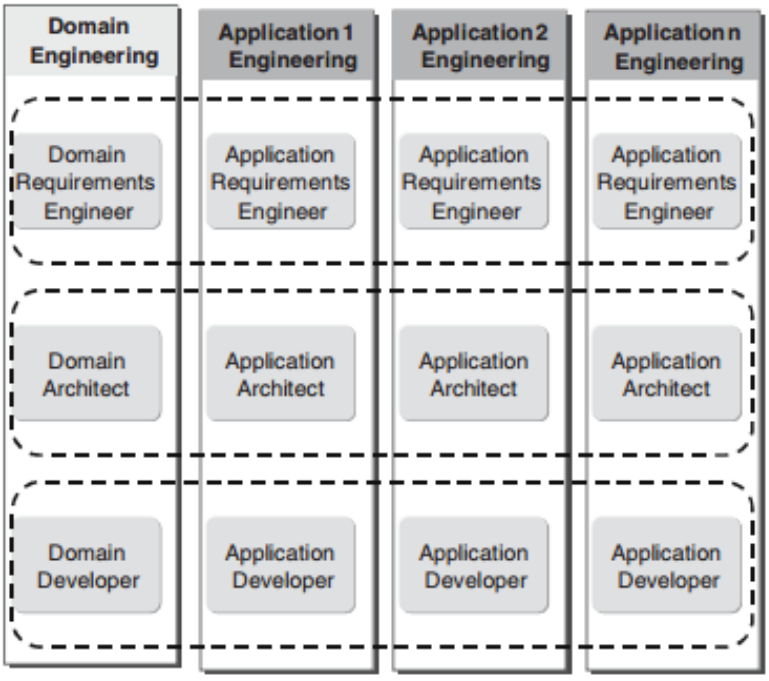
  Ispitivači aplikacije testiraju aplikaciju. Oni koriste testove koje su im pripremili ispitivači domene te rade sustavne testove i testove integracije.

Jedna dodatna uloga koja nije prikazana na slici 10. je menadžer domenskih komponenti. Menadžer domenskih komponenti je odgovoran za održavanje svih verzija i varijacija svih komponenti u proizvodnoj liniji.

S obzirom na povezanost ovih uloga i zadataka u organizaciji razlikuju se tri strukture organizacije koja razvija proizvodnu liniju a to su (Linden i sur. 2007., str. 66): proizvodno orijentirana struktura organizacije, procesno orijentirana struktura organizacije i matrix struktura organizacije.

6.1. Proizvodno orijentirana struktura organizacije

Proizvodno orijentirana struktura organizacije (eng. *product-oriented organisation*) je najčešće korištena struktura organizacija koje se bave razvojem linija programskih proizvoda. Na slici 11. možemo vidjeti da je proizvodno orijentirana struktura organizacije struktura u kojoj postoji jedan odjel koji je odgovoran za domenski razvoj te više odjela koji su odgovorni za aplikacijski razvoj. Prednosti ovakve strukture su to što su domenski i aplikacijski razvoj odvojeni po odjelima te se uloge i zadaće mogu lako raspodijeliti. Odjel domenskog razvoja je odgovoran za razvoj platforme koja se sastoji od ponovno upotrebljivih komponenti, dok su odjeli aplikacijskog razvoja odgovorni za razvoj i dostavljanje vlastitog proizvoda na tržište.

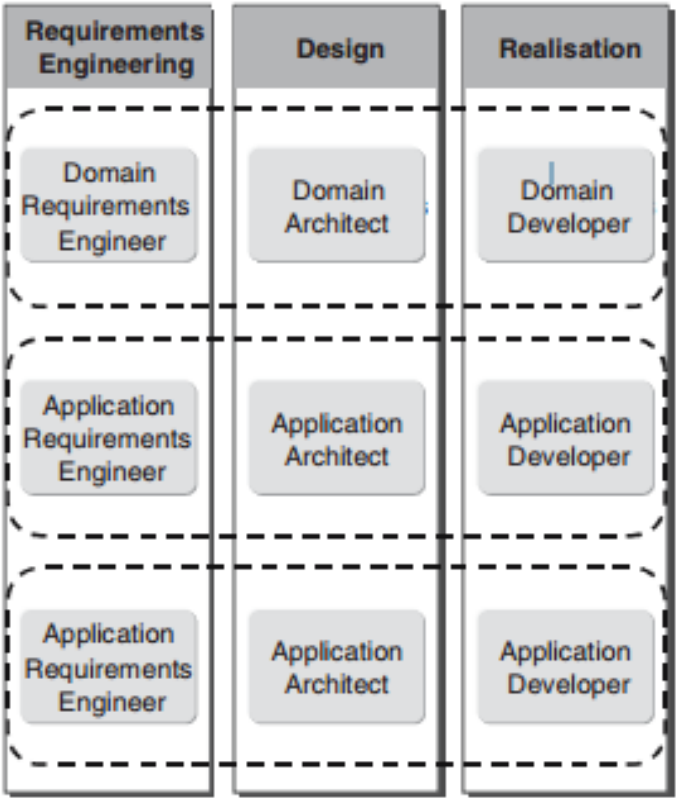


Slika 11.: Proizvodno orijentirana struktura organizacije

Izvor: Linden i sur. (2007., str. 67)

6.2. Procesno orijentirana struktura organizacije

Procesno orijentirana struktura organizacije (eng. *process-oriented organisation*) koristi jednu osobu za više razvojnih procesa domene ili aplikacije kao što možemo vidjeti na slici 12. U takvim organizacijama ključni faktor je komunikacija između različitih faza razvoja da bi se proizvod mogao dostaviti na vrijeme. Prednosti ovakve strukture su da je jednostavnije premještati zaposlenike s razvoja jednog proizvoda na drugi jer će na svakom proizvodu raditi na sličnim funkcionalnostima. Mana ovakve strukture je da su razvojni procesi jednog proizvoda udaljeni tj. loša je komunikacija između zaposlenika koji rade na njima. Iz tog razloga se ovakva struktura koristi samo u manjim organizacijama te se tako izbjegavaju problemi s komunikacijom.

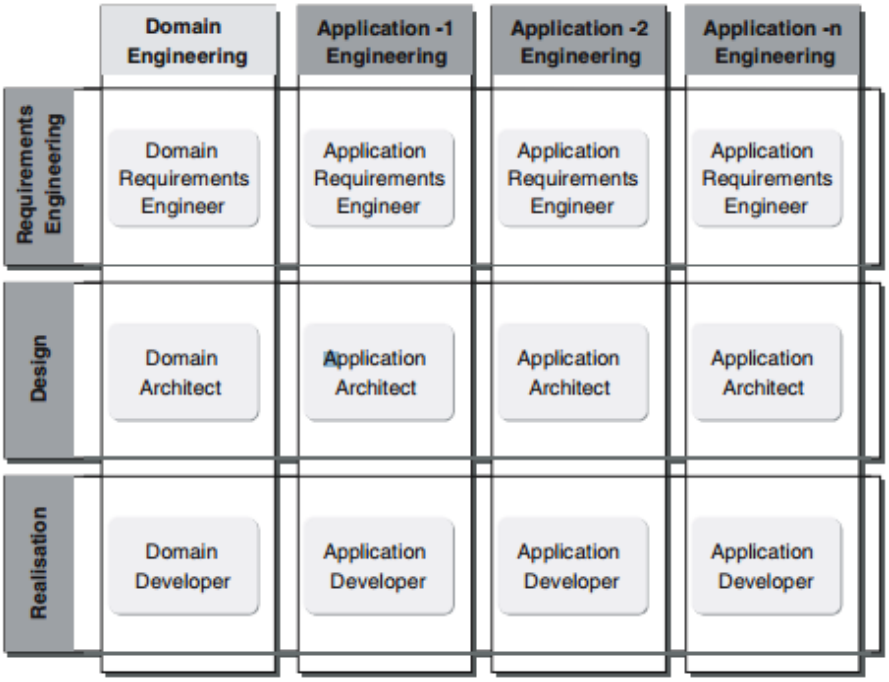


Slika 12.: Proizvodno orijentirana struktura organizacije

Izvor: Linden i sur. (2007., str. 69)

6.3. Matrična struktura organizacije

Matrična struktura (eng. *matrix organisation*) se koristi kod organizacija koje žele napraviti kompromis između funkcionalnih i proizvodnih potreba. Kod matrične strukture svaki programer ima odgovornosti prema nekom proizvodu te odgovornosti prema drugim zaposlenicima koji rade na tom proizvodu. Matrična struktura može biti dobra podloga za razvoj proizvodnih linija, no također može se dogoditi da je struktura previše kompleksna te se onda teže rješavaju novonastali problemi. Prikaz matrične strukture možemo vidjeti na slici 13.



Slika 13.: Matrična struktura organizacije

Izvor: Linden i sur. (2007., str. 71)

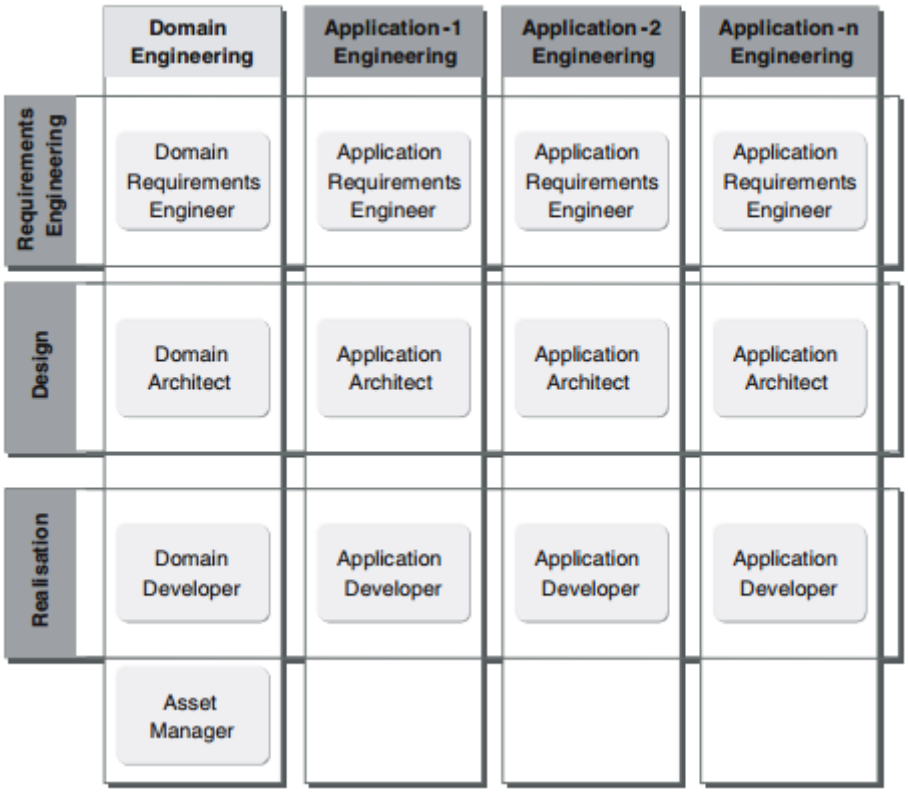
6.4. Određivanje uloga testiranja u organizaciji

Proces testiranja se nadovezuje na sve ostale razvojne procese te zato ima posebnu ulogu u organizaciji. Kod proizvodno orijentirane strukture organizacije testiranje se podijeli na domensko testiranje koje obavlja domenski odjel te aplikacijsko testiranje koje obavljaju aplikacijski odjeli tj. svaki odjel za svoj proizvod. Kod procesno orijentirane strukture organizacije postoji poseban odjel samo za testiranje te se s tim olakša ponovna upotreba testova kroz proizvodnu liniju. Kod organizacija koje koriste matričnu strukturu testiranje se dodaje kao novi red u matricu (slika 13.).

Linden i sur. (2007., str.71) navode da neke organizacije koriste hibridni pristup testiranju. Hibridni pristup testiranju bi bio takav da je struktura razvoja proizvodno orijentirana ali testiranje dobije poseban odjel. Taj odjel testira i platformu i sve proizvode. Prednosti takvog pristupa je što postoji kooperacija između domenskog i aplikacijskog testiranja dok je mana to što s ovim pristupom testiranje ponekad traje predugo. No to vrijeme testiranja se može smanjiti tako da se odjel za testiranje uključi u ostale procese razvoja programskog proizvoda.

6.5. Određivanje uloga za upravljanje komponentama

Uz devet glavnih uloga koje upravljaju s devet glavnih razvojnih procesa, postoje i menadžeri domenskih komponenti (eng. *asset managers)*. Njihov zadatak je da upravljaju komponentama koje se razvijaju u domenskom razvoju. Na slici 14. može se vidjeti da će se u organizacijama s proizvodno orijentiranom te matričnom strukturom, upravljanje komponentama samo nadodati kao dodatni zadatak u domenskom razvoju.

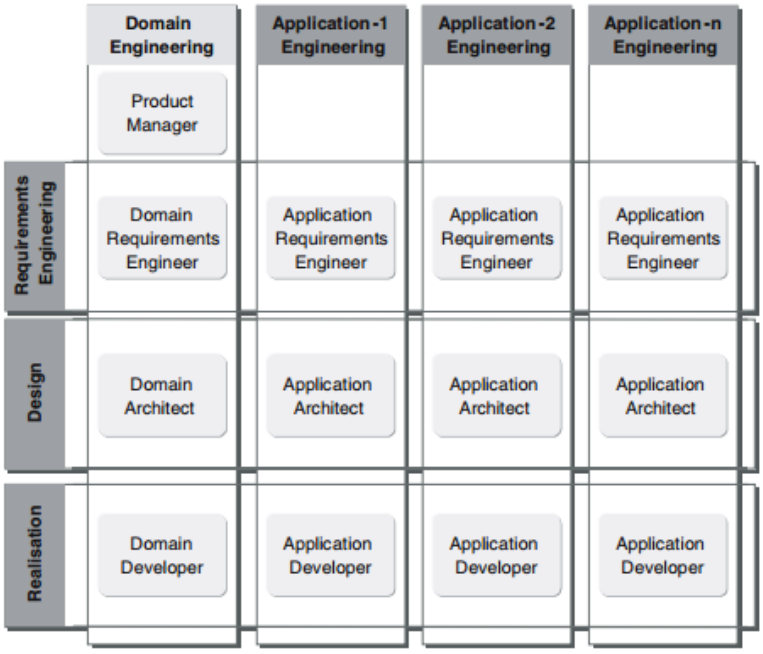


Slika 14.: Upravljanje komponentama u domenskom razvoju

Izvor: Linden i sur. (2007., str. 73)

6.6. Određivanje uloga za upravljanje proizvodima

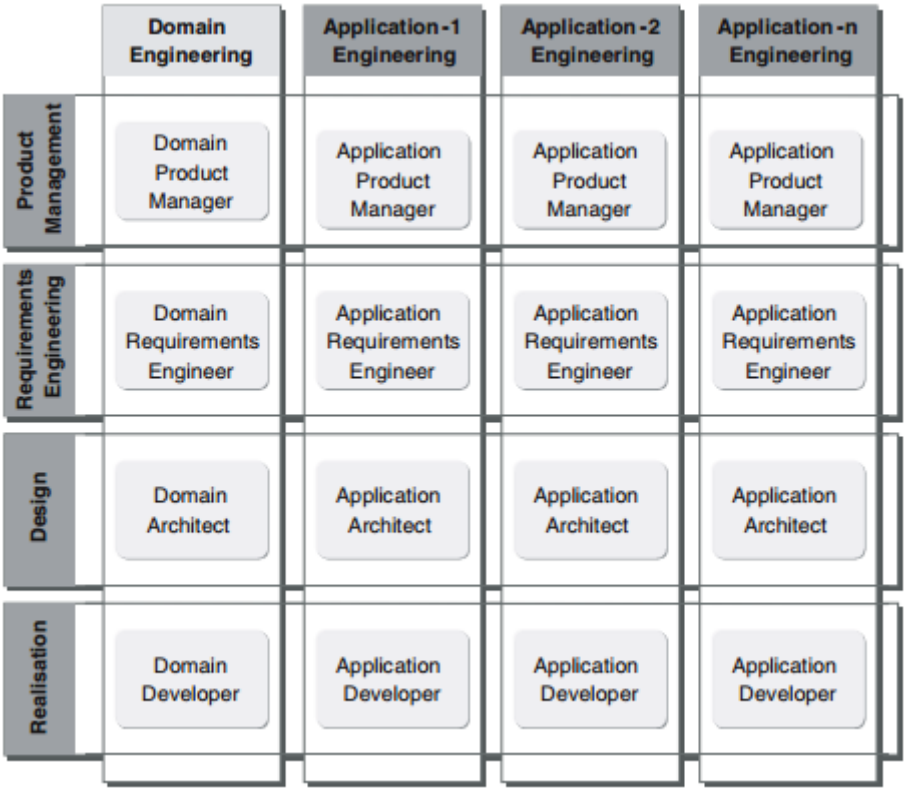
Menadžeri proizvoda (eng. *product managers*) određuju zajedničke komponente i varijacije u liniji programskih proizvoda. Organizacija može rasporediti uloge upravljanja proizvodima na dva pristupa. Prvi način bi bio da se uloge raspodjele kroz odjel domenskog razvoja kao što možemo vidjeti na slici 15.



Slika 15.: Upravljanje proizvodima u odjelu domenskog razvoja

Izvor: Linden i sur. (2007., str. 75)

No menadžeri proizvoda također moraju biti u komunikaciji sa zaposlenicima odjela aplikacijskog razvoja zato što je razvoj novog proizvoda pokrenut od strane menadžera proizvoda. Mana ovog pristupa sa slike 15. je ta što je domenski razvoj udaljen od krajnjeg korisnika te menadžeri proizvoda imaju otežanu komunikaciju s njima te se može dogoditi da menadžeri proizvoda krivo odrede funkcionalnosti novog proizvoda te iz tih razloga proizvod neće uspjeti na tržištu. Kao što možemo vidjeti na slici 16. drugi pristup je da organizacija raspodjeli uloge menadžera proizvoda kroz jedan odjel domenskog razvoja i nekoliko odjela aplikacijskog razvoja. Tako menadžer proizvoda u domenskom razvoju može planirati zajedničke komponente za platformu dok menadžeri u aplikacijskom razvoju planiraju široki spektar aplikacija. Da bi ovakav pristup funkcionirao mora postojati dobra komunikacija između menadžera u domenskom razvoju i menadžera u aplikacijskom razvoju (Linden i sur., 2007., str. 75).



Slika 16.: Upravljanje proizvodima u odjelima domenskog i aplikacijskog razvoja

Izvor: Linden i sur. (2007., str. 76)

Iz ovog poglavlja možemo zaključiti da je ispravna raspodjela uloga i zadataka u organizaciji važan dio razvoja linije programskih proizvoda te da je potrebna dobra komunikacija između različitih odjela u organizaciji bez obzira kakvu strukturu rada koristi ta organizacija.

Zaključak

Razvoj programskih proizvoda pomoću ponovne upotrebe je u današnjici veoma zastupljen. Linije programskih proizvoda su donekle zaslužne za to. Organizacije koje na ispravan način pristupe razvoju linije programskih proizvoda uživaju u velikim prednostima koje im one pružaju. Organizacije uvelike štede na trudu, vremenu i troškovima proizvodnje te uz to povećavaju kvalitetu svojih programskih proizvoda. Spomenuto je u radu da se razvoj linija programskih proizvoda temelji na četiri perspektive razvoja. Važno je naglasiti da se mora ispravno pristupiti svakoj perspektivi razvoja da bi razvoj linije bio uspješan. U poslovnoj perspektivi važno je odrediti poslovne strategije kojima će se linija programskih proizvoda razvijati. U arhitekturalnoj perspektivi važno je razviti referentnu arhitekturu na kojoj će se temeljiti razvoj svih pojedinih proizvoda u liniji. Procesna perspektiva se temelji na dva životna ciklusa koji se nazivaju domenski i aplikacijski razvoj. Kod domenskog razvoja najvažnije je razviti platformu koja je temeljena na referentnoj arhitekturi te odrediti varijabilnosti koje ta platforma mora podržavati dok je kod aplikacijskog razvoja ključno da se na ispravan način implementiraju te varijabilnosti. Organizacijska perspektiva se bavi raspodjelom uloga i zadataka po zaposlenicima u organizaciji. Postoji nekoliko organizacijskih struktura te organizacije moraju odabrati njima najprikladniju strukturu.

Linije programskih proizvoda se još uvijek smatraju novom tehnologijom u inženjerstvu programskih proizvoda. Neke organizacije se pribojavaju koristiti razvojem linija programskih proizvoda jer još uvijek postoje neka ograničenja. Tijekom zadnjih desetak godina nastalo je dosta strategija razvoja pomoću linija programskih proizvoda, no napravljeno je malo detaljnih analiza takvih strategija da bi se moglo jasno naglasiti prednosti i mane svake strategije. Organizacije koje su koristile razvoj linija programskih proizvoda su većinom imale pozitivna iskustva i uživali su u brojnim prednostima koje takav razvoj pruža no nisu uspjeli unaprijed točno prognozirati te prednosti. Neke organizacije su vođene striktno po poslovnim ciljevima, te su skeptične za prijelaz na razvoj pomoću linija programskih proizvoda upravo zato što nije razvijeno dovoljno detaljnih ekonomskih modela takvog razvoja.

Razvoj programskih proizvoda sve više teži za većom prilagodbom i fleksibilnošću proizvoda. Možemo zaključiti da linije programskih proizvoda postaju sve popularnija opcija kada se organizacije nalaze u situaciji da proizvode veliki broj proizvoda koji se razlikuju u malom broju funkcionalnosti. Linije programskih proizvoda donekle omogućavaju fleksibilnost i prilagodbu proizvoda kroz varijabilnost referentne arhitekture te su sve popularniji način razvoja programskih proizvoda. Organizacije koje razvijaju mali broj sličnih programskih proizvoda bi se trebale držati tradicionalnog razvoja jer se razvoj pomoću linija programskih proizvoda isplati samo kada će se razvijati veliki broj sličnih aplikacija. Ako organizacije koje se bave tradicionalnim razvojem imaju nekoliko sličnih proizvoda, te u budućnosti planiraju razvijati slične proizvode, preporučljivo je da pokrenu vlastitu liniju programskih proizvoda.

Literatura

1. Bass, L., Clements, P. i Kazman, R. (2003): *Software architecture in practice,* second edition*.* USA*:* Addison Wesley.
2. Bockle, G. , Linden, F. i Pohl, K. (2005): *Software product line engineering: foundations, principles, and techniques,* Berlin: Springer-Verlag Berlin Heidelberg.
3. Bosh, J. i Lee, J. (2010): *Software product line: going beyond,* 14th International Conference, Lancaster: Lancaster University
4. Eriksson, M.: An introduction to software product line development, <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.63.9209&rep=rep1&type=pdf >.
   pristupljeno 17. kolovoza 2016.
5. Linden, F., Schmid, K. i Rommes, E. (2007): *Software product lines in action,* Berlin: Springer-Verlag Berlin Heidelberg.
6. Manger, R. (2013): Softversko inženjerstvo, nadopunjeno drugo izdanje skripte, Prirodoslovno Matematički Fakultet Sveučilišta u Zagrebu.
7. McGregor, J.(2014): Software product lines,
   < http://www.jot.fm/issues/issue_2004_03/column6.pdf >.
   pristupljeno 18. kolovoza 2016.
8. Northrop, L. (2008): Software product lines essentials,
   < http://www.sei.cmu.edu/library/assets/spl-essentials.pdf >.
   pristupljeno 15. kolovoza 2016.
9. Sommerville, I. (2007.): *Software Engineering,* 8th edition, Harlow, England: Addison-Wesley
10. Sommerville, I. (2015): Software Engineering, 10th Edition, Boston: Pearson Education.

Popis slika

Sažetak

Ponovna uporaba je veoma isplativ način razvoja programskih proizvoda. Linije programskih proizvoda omogućile su jako veliki napredak u razvoju programskih proizvoda. Pomoću razvoja linija programskih proizvoda organizacije mogu uvelike uštedjeti na trudu i troškovima proizvodnje programskih proizvoda te se uvelike može smanjiti vrijeme koje je potrebno da se proizvodi dostave na tržište. Razvoj linija programskih proizvoda temelji se na četiri perspektive razvoja. U poslovnoj perspektivi razvoja određuju se strategije razvoja programskog proizvoda. Arhitekturalna perspektiva razvoja se nadovezuje na poslovne strategije te se kod nje razvija arhitektura linije programskih proizvoda. Procesna perspektiva razvoja ima zadaću implementirati dizajniranu arhitekturu dok se organizacijska perspektiva razvoja bavi raspodjelom uloga i zadataka za liniju programskih proizvoda kroz cijelu organizaciju.

Ključne riječi: ponovna uporaba, linije programskih proizvoda, poslovna perspektiva razvoja, arhitekturalna perspektiva razvoja, procesna perspektiva razvoja, organizacijska perspektiva razvoja

Summary

Reuse is a very cost effective way of developing software products. Software product lines have enabled very great progress in the development of software products. With the development of software product line organizations can greatly save on labor and production costs of software products and can greatly reduce the time it takes to get products to market. The development of software product line is based on four perspectives of development. In the business development perspective strategies of software product lines development are determined. Architectural perspective of development is linked to the business strategy and with it develops the architecture of software product lines. Process development perspective is tasked to implement designed architecture while organizational development perspective deals with the distribution of roles and tasks for the software product lines across the organization.

Keywords : reuse, software product lines, business perspective of development, architectural perspective of development, process perspective of development , organizational perspective of development