Na osnovu člana 17. Zakona o Vijeću ministara Bosne i Hercegovine ("Službeni glasnik BiH", br. 30/03, 42/03, 81/06, 76/07, 81/07, 94/07 i 24/08), a u skladu sa članom 3. Odluke o usvajanju Politike softvera u institucijama Bosne i Hercegovine ("Službeni glasnik BiH", broj 143/07), na prijedlog Ministarstva komunikacija i prometa Bosne i Hercegovine, Vijeće ministara Bosne i Hercegovine na _____. sjednici održanoj _____ 2013. godine, donijelo je

O D L U K U
o usvajanju Dokumenta o modalitetima planiranja, razvoja i implementacije programskih rješenja u institucijama Bosne i Hercegovine

Član 1.
(Predmet odluke)

Ovom Odlukom usvaja se Dokument o modalitetima planiranja, razvoja i implementacije programskih rješenja u institucijama Bosne i Hercegovine.

Član 2.
(Djelokrug primjene)

Dokument o modalitetima planiranja, razvoja i implementacije programskih rješenja u institucijama Bosne i Hercegovine primjenjuje se u svim institucijama Bosne i Hercegovine.

Član 3.
(Dokument)

Dokument o modalitetima planiranja, razvoja i implementacije programskih rješenja u institucijama Bosne i Hercegovine je sastvani dio ove Odluke.

Član 4.
(Stupanje na snagu)

Ova Odluka stupa na snagu narednog dana od dana objavljivanja u "Službenom glasniku BiH".

VM broj___/13
. 2013. godine
Sarajevo

PREDSJEDAVAJUĆI
Vijeća ministara BiH
Vjekoslav Bevanda

Obrazloženje

Pravni osnov

Pravni osnov za donošenje ove Odluke sadržan je u članu 17. Zakona o Vijeću ministara Bosne i Hercegovine ("Službeni glasnik BiH", br. 30/03, 42/03, 81/06, 76/07, 81/07, 94/07 i 24/08) kojim je propisano da Vijeće ministara Bosne i Hercegovine u ostvarivanju svojih prava i dužnosti između ostalih akata, donosi i odluke.

Pravni osnov je također sadržan i u članu 3. Odluke o usvajanju Politike softvera u institucijama Bosne i Hercegovine ("Službeni glasnik BiH", broj 143/07) kojom je Ministarstvo komunikacija i prometa Bosne i Hercegovine zaduženo da pripremi i podnese na usvajanje Vijeću ministara Bosne i Hercegovine dokumente koji su definirani u poglavlju 3. Politike softvera u institucijama Bosne i Hercegovine, između ostalih i dokument Modalitet planiranja, razvoja i implementacije programskih rješenja u institucijama Bosne i Hercegovine.

Razlozi za donošenje

Vijeće ministara Bosne i Hercegovine je na 69. sjednici održanoj 16. novembra 2004. godine usvojilo Strategiju razvoja informacionog društva u Bosni i Hercegovini, Politiku razvoja informacionog društva u Bosni i Hercegovini i Akcioni plan razvoja informacionog društva u Bosni i Hercegovini.

Na osnovu navedenih dokumenata, Vijeće ministara Bosne i Hercegovine je na 23. sjednici, održano 20. septembra 2007. godine, donijelo Odluku o usvajanju Politike softvera u institucijama Bosne i Hercegovine. Odluka je stupila na snagu danom donošenja.

U skladu sa članom 3. ove Odluke, Ministarstvo komunikacija i prometa Bosne i Hercegovine je zaduženo da u roku od godinu dana od dana stupanja na snagu Odluke, pripremi i podnese na usvajanje Vijeću ministara Bosne i Hercegovine dokumente koji su definirani u poglavlju 3. Politike softvera u institucijama Bosne i Hercegovine.

Jedan od dokumenata iz poglavlja 3. Politike softvera u institucijama Bosne i Hercegovine (tačka 3.2.) je i dokument – Modalitet planiranja, razvoja i implementacije programskih rješenja.

Ovaj dokument specificira standarde i preporuke za vođenje procesa planiranja, razvoja i implementacije programskih rješenja u institucijama Bosne i Hercegovine. Izrada ovog propisa predviđena i Programom rada Ministarstva komunikacija i prometa Bosne i Hercegovine za 2013. godinu. Tekst navedenog dokumenta dat je u prilogu ove Odluke .

Zbog svega navedenog predlaže se donošenje predmetne Odluke u predloženom tekstu.

Obrazloženje predloženih rješenja

Članom 1. - utvrđuje se predmet odluke
Članom 2. – utvrđuje se djelokrug primjene
Članom 3. – utvrđuje se prilog odluke
Članom 4. – utvrđuje se stupanje na snagu odluke

Finansijska sredstva

Za provođenje ove Odluke nije potrebno obezbjediti dodatna finansijska sredstva u Budžetu institucija Bosne i Hercegovine i međunarodnih obaveza Bosne i Hercegovine za 2013. godinu.

Zavisno od modaliteta programskih rješenja za koje se odluče institucije Bosne i Hercegovine ovisiće potrebna sredstva.

MODALITET PLANIRANJA, RAZVOJA I IMPLEMENTACIJE PROGRAMSKIH RJEŠENJA U INSTITUCIJAMA BOSNE I HERCEGOVINE

Dokument „Modalitet planiranja, razvoja i implementacije programskih rješenja u institucijama Bosne i Hercegovine“ daje smjernice i definira način na koji institucije Bosne i Hercegovine pristupaju planiranju, razvoju i implementaciji programskih rješenja koja su izvodljiva i pouzdano rade unutar zadanih granica, odnosno koja zadovoljavaju poslovne ciljeve, prema zahtjevima korisnika, u prihvatljivom vremenu i po opravdanoj cijeni, a u skladu sa najboljim praksama i poznatim metodama koje se koriste u svijetu.

Ovaj dokument u metodološkom smislu definira način na koji institucija Bosne i Hercegovine pristupa planiranju, razvoju i implementaciji programskih rješenja kroz specifikaciju:

- strukture timova koji učestvuju u razvojnom procesu i pojedinačnih uloga,
- vještina koje moraju posjedovati akteri u razvojnom procesu,
- produkata razvojnog procesa,
- procesa koji se odvijaju tokom razvojnog ciklusa,
- aktivnosti koje se izvršavaju,
- kontrolnih tačaka u razvojnom procesu,
- preporučenih tehnika i alata i
- standarda kvaliteta.

Standardi i preporuke koji su navedeni u ovom dokumentu daju glavne smijernice o općem pristupu razvojnom procesu, metodologiju strateškog planiranja IT, metodologiju razvoja i eksploatacije informacionih sistema i metodologiju vođenja projekata.

Ovaj dokument daje integralni pogled na procese, kao što su:

1. Planiranje i razvoj programskih rješenja
2. Svakodnevni menadžmenta IT
3. Metodologija razvoja informacionih sistema
4. Eksploatacija programskih rješenja i organizacije promjene usljed uvođenja IT baziranih rješenja
5. Vođenje promjene tokom razvoja i implementacije programskih rješenja
6. Pravila za izgradnju sigurnijih programa.

U smislu ovog dokumenta pojmovi imaju slijedeće značenje:

1. **Informacija** je rezultat obrade, manipulacije, organiziranja i interpretacije podataka koji daju određeno znanje primatelju.
2. **Informacioni sistem** predstavlja sistem za prikupljanje, obradu, prenos, pohranu i distribuciju informacija, te ih čini dostupnim i upotrebljivim.
3. **Informacione tehnologije** podrazumijevaju različite elemente i vještine za kreiranje, obradu, razmjenu i pohranu informacija upotrebom elektronskih računara i pripadajućeg softvera.
4. **Internet** je globalna računarska mreža, sačinjena od velikog broja međusobno povezanih računarskih mreža i uređaja, koja omogućava razmjenu podataka između računara.
5. **Interoperabilnost** predstavlja sposobnost informacionih i komunikacionih sistema i poslovnih procesa da podrže protok podataka i omoguće razmjenu informacija i znanja.
6. **Komunikacija** predstavlja razmjenu ili prenošenje informacija između određenog broja sudionika, putem javno dostupnih elektronskih komunikacionih mreža i usluga.
7. **Podatak** označava svako predstavljanje činjenica, informacija ili koncepata u obliku koji je pogodan za njihovu obradu u računarskom sistemu.
8. **Proces** predstavlja skup povezanih, struktuiranih i koordinisanih aktivnosti koje kombinuju raspoložive resurse, kako bi na osnovu određenih ulaznih parametara proizveli određene izlaze krajnjim korisnicima.
9. **Softver** predstavlja uređeni skup naredbi koji služe za upravljanje radom računara, kao i za rješavanje određenog zadatka pomoću računara.
10. **Standard** je dokument za opću i višekratnu upotrebu, donesen konsenzusom i odobren od priznatog tijela, koji sadrži pravila, smjernice ili karakteristike aktivnosti ili njihove rezultate i koji ima za cilj postizanje optimalnog nivoa uređenosti u datom kontekstu.
11. **Životni ciklus razvoja sistema (engl. System Development Life Cycle – SDLC**) predstavlja proces kroz koji stručnjaci različitih profila i korisnici informacionog sistema razvijaju informacioni sistem.

1. OPĆI PRISTUP RAZVOJNOM PROCESU

1.1.Planiranje i razvoj softverskog rješenja

Planiranje i razvoj softverskog rješenja se bazira na grafičkom mapiranju podataka i informacija sa softverskim aplikacijama i procesima koji sakupljaju, upravljaju i transformišu te podatke i informacije. Kako bi dobili najadekvatnija rješenja koja će zadovoljiti naše zahtjeve, potrebno je da se tačno odrede razlozi zbog kojih treba planirati programska rješenja.
Polazna tačka planiranja razvoja programskog rješenja treba obezbijediti:

- jasnu definiciju čime će se programsko rješenje baviti
- zadatke i ciljeve programskog rješenja
- uloge unutar timova za razvoj programskog rješenja
- potrebne aplikacije i podatke koje se razvijaju, koriste ili planiraju koristiti unutar sistema
- raspložive resurse (osoblje, tehnička sredstva, tehnologija).

Institucije Bosne i Hercegovine trebaju na vrijeme izvršiti adekvatno planiranje i izbor odgovarajućih metodologija razvoja programskih rješenja kako slijedi:

A. Vlastiti razvoj programskih rješenja

Razvoj vlastitim informatičkim snagama podrazumjeva osposobljavanje i angažiranje IT osoblja, kao i povremeno ili dugoročno angažovanje spoljnih saradnika. Prednosti ovakvog pristupa su fleksibilnost, kreativnost i povećanje stručnosti vlastitog osoblja. Nedostaci su da ovaj pristup zahtjeva značajno vrijeme i intenzivni rad, razvoj je skuplji i dugotrajniji i može povećati gomilanje zaostalog posla.
Preporučuje se da institucije Bosne i Hercegovine koriste vlastita programska rješenja kada se radi o programskoj podršci koja je specifična njihovim potrebama a da kao takva ne postoje gotova rješenja na tržištu. Postoje dodatni ili posebni razlozi kao što su povećana tajnost podataka i poslovnih procesa ili povećana zaštita IS.

Programska rješenja mogu biti bazirana na modelima, kao što su:

1. **Kaskadni model (waterfall model)** je linearni model procesa razvoja programske podrške gdje se pojedina faza mora dovršiti prije pokretanja nove faze (nema povratne veze između susjednih faza).

   Kaskadni model razvoja programske podrške se sastoji od pet faza:

   - **Faza definicije i analiza zahtjeva korisnika:** gdje se utvrđuju zahtjevi koje sistem treba da zadovolji. Pri analizi zahtjeva moraju se uzeti u obzir aktivnosti i ograničenja koji postoje u sistemu.
   - **Faza oblikovanje sistema:** gdje se generiše sistem koji daje plan rješenja. Plan uključuje komponente i algoritme koji će biti korišteni, kao i arhitekturu sistema.
   - **Faza implementacija i testiranje jedinica sistema:** gdje programeri pišu i testiraju programski kod prema utvrđenim zahtjevima klijenta.
   - **Faza integracije i testiranje sistema:** gdje se vrši integraciono testiranje tokom kog se moduli povezuju u jednu cjelinu. Na kraju ove faze provodi se

završno testiranje u kome se provjerava da li sistem ispunjava postavljene zahtjeve korisnika.

- **Faza puštanje u rad i održavanje:** gdje se sistemi isporučuju i instaliraju u radnom okruženju i obavlja se obuka neposrednih korisnika. Održavanje u ovoj fazi podrazumijeva ispravljanje grešaka u sistemu koje se javljaju nakon njegove isporuke.

U svakoj fazi se definišu kontrolne tačke na kojima dolazi do slaganja između projektnog tima i kupaca ili projektnih dioničara u vezi isporuke i cilja projekta. Osim kontrolnih tačaka, u svakoj fazi mogu se definisati i međuproizvodi, na osnovu kojih se dobija uvid u trenutni stepen izvršenja projekta.
U procesu definisanja i stvaranja softverskog proizvoda uključeno je više učesnika koji mogu da imaju sljedeće uloge:

1. **Kupac** je organizacija, kompanija ili pojedinac koji naručuje programsko rješenje i finansira njegov razvoj.
2. **Korisnik** je osoba ili više osoba koje će koristit sistem.
3. **Razvojni tim** čine ga softverski inženjeri specijalizovani za različite aspekte razvoja. Uloge koje učestvuju u procesu razvoja softvera metodologijom zasnovanom na kaskadnom modelu su:
    - **Analitičar** utvrđuje funkcionalnosti i osobine koje korisnik želi i dokumentuje njegove zahtjeve, također radi zajedno sa projektantima na generisanju opisa funkcija sistema.
    - **Projektant** projektuje sistem prema zadatim funkcionalnim zahtjevima i prosljeđuje ih programerima.
    - **Programer** piše programski kod na odgovarajućem programskom jeziku koristeći predviđeno razvojno okruženje.
    - **Inženjer** za tesiranje: detaljno testira programski kod koji je napisao programer te vrši varifikaciju i validaciju sistema.
    - **Inženjer** za isporuku i održavanje: isporučuje i instalira softver u radnom okruženju, obučava korisnike za operativno korištenje sistema i bavi se poslovima održavanja sistema nakon njegove isporuke.

Ova metoda se preporučuje kod velikih projekata (investicija), za dobro definisano okruženje, gdje postoje razrađene procedure ručne obrade ili računarski sistem koji treba unaprijediti. Nedostaci ovog modela su izraženi u slučaju grešaka ili novih/promijenjenih zahtjeva. Razvijeni sistem se smatra neupotrebljiv dok nije u potpunosti gotov.

2. **Spiralni model:** Ovaj model se preporučuje prilikom upravljanje rizicima razvoja programskog proizvoda.

Spiralni model objedine dobre osobine razvoja odozgo-nadole što se smatra kao projektovanje sistema njegovom dekompozicijom na podsisteme sa postepenim prelaskom na detalje i odozdo-nagore, sto se smatra prototipsko projektovanje manjih cjelina koje se poslije nadograđuju i povezuju u kompletan sistem.

Implementacija ovog modela treba da se razvija u okviru 4 procesne faze:

a) **Faza postavljanje cilja** gdje se identifikuju specifični ciljevi (opcije/alternative i ograničenja) koji direktno utiču na nivoima rizika i njihovo umanjenje ili eliminaciju.
b) **Faza procjena rizika i njegovo smanjenje** -analiziraju se rizici koji su utvrđeni u u opcijama i alternativama, te preslikavanje istih u aktivnosti koje će ih reducirati. U ovoj fazi se preporucuje provođenje SWOT analize rizika. Ishod Analize rizika će rezultirati u evoluciji prototipova.
c) **Faza razvoj i validacija** gdje se analizom prototipa slijedi razvoj programskog rješenja. U ranim spiralama se dobija koncept, međutim, kasnije spirale nose sve detaljnije aktivnosti koje su bazirane na odabiru modela organizacije, kao što su:
    i. projektna organizacija (osoblje organizirano unutar projekta)
    ii. funkcionalna organizacija (osoblje organizirano po funkcionalnim odgovornostima, a pojedina funkcija može podupirati više projekata)
    iii. matrična organizacija (osoblje izmiješano u različitim projektima).
d) **Faza planiranje** gdje se trenutni tok razvoja projekta ispituje i planiraju se faze (ciklusi spirale) od koncepta do produkta.

U svakoj fazi se definisu kontrolne tačke na kojoj dolazi do slaganja između projektnog tima i kupaca ili projektnih deoničara u vezi isporuke i cilja projekta.

U procesu definisanja i stvaranja softverskog prozivoda koriste se iste uloge kao kod kaskadnog modela. (pogledati kaskadni model)

Spiralni model je jako složen za implementaciju i preporučuje se njegova upotreba na velikim projektima koji zahtjevaju mnoštvo različitih resursa, vremena i velikog budžeta. Korištenjem ovog modela će se olakšati rano prepoznavanje i rješavanje mogućih rizika razvoja programskih rješenja, međutim u slučaju da se utvrdi da je rizik razvoja prevelik, projekat se prekida.

3. **Razvoj temeljen na osobinama** – (engl. Feature Driven Development – FDD)

Ova metoda naglašava kvalitet kroz cijeli razvojni proces te uključuje česte i „opipljive“ isporuke programske podrške, kao i precizno nadgledanje napretka projekta, što iziskuje dosta truda i ulaganja u osnovne procese istraživanja, razvoja, testiranja i evaluacije

FDD se sastoji od pet faza:

a) **Faza razvoja modela.** Tokom ovog procesa definiše se nivo zahtjevnosti projekta i ciljanog sistema pri čemu se pažnja ne posvećuje detaljima, nego je bitno shvatiti i razumjeti područje u kojem će se razvijani sistem primjenjivati.

b) **Faza kreiranje liste osobina** na osnovu podataka o nivou zahtjevnosti projekta i ciljanog sistema pristupa se identifikaciji svih „poslovnih aktivnosti - osobina“ zahtjevanih od strane krajnjeg korisnika koje trebaju biti implementirane u programskom rješenju. Sistem se raščlanjuje na manje dijelove koji sadrže određene funkcionalnosti sistema te se svaka pojedinačno realizira.

c) **Faza planiranje na osnovu osobina** gdje se pristupa stvaranju globalnog plana, na način da se liste osobina poredaju po prioriotetima i ovisnostima o ostalim osobinama, te se dodjeljuju glavnim programerima

d) **Faza dizajniranje prema osobinama u** ovoj fazi se definišu funkcionalnosti koje trebaju biti razvijene u određenom vremenskom intervalu prema potvrđenim osobinama. Pri tome se izrađuje detaljni dijagram sekvenci za svaku funkcionalnost.

e) **Faza razvoj prema osobinama** Na osnovu definisanog dizajna započinje se programiranje potrebnih funkcionalnosti. Pri tome nakon osnovnog razvoja slijedi provjera koda i pojedinačno testiranje, te na kraju uključivanje u stabilno izdanje sistema.

U svakoj fazi se definišu kontrolne tačke na kojima dolazi do slaganja između projektnog tima i kupaca ili projektnih dioničara u vezi isporuke i cilja projekta.

U procesu definisanja i stvaranja softverskog proizvoda uključeno je više učesnika koji mogu da imaju sljedeće uloge:

1. Kupac: je organizacija, kompanija ili pojedinac koji naručuje programsko rješenje i finansira njegov razvoj.
2. Korisnik: je osoba ili vise koja će koristit sistem

3. Razvojni tim: čine ga softverski inženjeri specijalizovani za različite aspekte razvoja. Uloge koje učestvuju u procesu razvoja softvera metodologijom zasnovanom na osobinama su:
   - **Projekt menadžer:** administrativni vođa projekta odgovoran za izvještavanje o napretku, upravljanju budžetom i upravljanje resursima uključujući ljude, opremu i prostor.
   - **Glavni arhitekt:** odgovoran za cijelokupni dizajn sistema uključujući tekuće radionice s timom.
   - **Razvojni menadžer:** odgovoran za vođenje svakodnevnih razvojnih aktivnosti uključujući rješavanje konflikata oko resursa.
   - **Glavni programer:** iskusni razvojni programer koji djeluje kao vođa tima, mentor i programer za grupu od tri do šest programera.
   - **Vlasnik klase:** odgovoran za dizajniranje, kodiranje, testiranje i dokumentiranje novih svojstava u klase.
   - **Stručnjak u određenom domenu:** klijenti, sponzori, poslovni analitičari itd. koji imaju duboko znanje o proizvodu koji se razvija.
   - **Tim koji razvija određeno svojstvo:** (eng. Feature team) privremena grupa razvojnih programera koja radi na određenoj osobini. Kada je osobina implementirana, tim se raspušta te formira novi za novu osobinu.

Opisane uloge ne moraju uvijek da pripadaju različitim učesnicima. U nekim manjim projektima, ista osoba ili grupa može da ima dvije ili čak i više uloga.

Ova metoda omogućava česte i vremenske predvidljive isporuke funkcionalnih komponenata sistema, ona se preporučuje za primjenu na velikim i kompleksnim sistemima (npr. Javna uprava). Također može biti primjenjiva i na novim projektima koji tek počinju i na projektima koji poboljšavaju postojeći kod.

4. **Ekstremno programiranje** (engl. Extreme Programming – XP)
   Ova metoda se koristi kod sistema koji se svakih nekoliko mjeseci trebaju mijenjati ili prilagoditi nekoj specifičnoj potrebi. To znači da je cilj isporučivati programe onda kad su potrebni i u obliku u kojem su potrebni. Ona se fokusira na uloge kupca ili naručitelja, upravitelja i razvojnog inženjera te prebacuje ključna prava i odgovornosti onima koji nose te uloge.

   Ova metoda omogućava prilagođavanje promjenama zahtjeva u bilo kojem trenutku u procesu razvoja projekta, te je time realističnija i ima bolji pristup nego metoda koja definira sve zahtjeve na početku projekta, čime se smanjuju troškovi realizacije.

Ekstremno programiranje se sastoji od šest faza:

a) **Faza istraživanja** gdje kupac piše svoje zahtjeve na kartice o onom što bi želio da bude uključeno u prvo izdanje ove metodologije tzv. korisničke priče (engl. user stories). Kupac treba biti uvijek dostupan da bi razjasnio pitanja u vezi zahtjeva i da im dodijeli prioritete.

f) **Faza planiranja** gdje se postavljaju prioriteti na korisničkim zahtjevima i određuju koje će se kartice sa korisničkim zahtjevom koristiti za izradu prvog malog izdanja sistema.

g) **Faza iteracije** do izdavanja u kojoj se uključuje nekoliko iteracija sistema prije prvog izdanja.

h) **Faza produkcije** u kojoj se izvršava dodatno testiranje i provjera performansi sistema prije nego se sistem može isporučiti klijentu.

i) **Faza održavanja** sistema gdje se može zahtjevati novi ljudi unutar projektnog tima kao i promjenu strukture tima.

j) **Faza smrti** se podrazumijeva kada sistem u svakom drugom pogledu zadovoljava klijentove zahtjeve

U svakoj fazi se definišu kontrolne tačke na kojoj dolazi do slaganja između projektnog tima i kupaca ili projektnih dioničara u vezi isporuke i cilja projekta.

U procesu definisanja i stvaranja softverskog proizvoda uključeno je više učesnika koji mogu da imaju (pored ranije definisanog kupca i korisnika) sljedeće uloge:

1. Razvojni tim: čine ga softverski inženjeri specijalizovani za različite aspekte razvoja. Uloge koje učestvuju u procesu razvoja softvera metodologijom zasnovanom na XP, u biti čine male i srednje razvojne timove od 3 do 20 osoba u zavisnosti od obima projektnog zadatka. Uloge definisane XP metodologijom su:
   - **Menadžer:** formira tim te upravlja njime i njegovim problemima
   - **Trener:** uči članove tima o XP metodologiji te prati slijede li se procesi XP-a. Trener je uobičajeno programer, a ne mendžer.
   - **Tragač** (eng. tracker): prikuplja korisničke priče i napredak u slučajevima testova prihvaćanja s ciljem izrade vidljivog zidnog grafa. Tragač je programer.
   - **Programer:** piše testove, dizajnira i kodira, ali takođe identifikuje i procjenjuje zadatke i korisničke priče. Ova osoba takođe može biti tester.
   - **Tester:** pomaže kupcu pisati i razvijati testove.
   - **Kupac:** piše korisničke priče i testove prihvaćanja te odabira korisničke priče koje će se raditi u određenoj iteraciji.

Opisane uloge ne moraju uvijek da pripadaju različitim učesnicima. U nekim manjim projektima, ista osoba ili grupa može da ima dvije ili čak i više uloga.

Metodologija ekstremnog programiranja obuhvata skup tehnika u kojima se naglašava kreativnost timskog rada uz minimizaciju prekomjernog administriranja. Ona se ne preporučuje ukoliko postoji otpor ovakvom načinu programiranja, u velikim timovima i nefleksibilnim okolinama, tamo gde je potrebna opsežna dokumentacija i gdje se povratne informacije dugo čekaju te u okolini gde je prekovremeni rad uobičajen ili ljudi u timu ne žele (ili ne mogu) komunicirati.

5. Scrum

Ova metodologija predstavlja metodu upravljanja projektima koja se oslanja na brze, prilagodlijve i samoorganizirajuće razvojne timove koji rade nove ili poboljšane verzije sistema u okolini (npr. zahtjevi, vremenski okvir, sredstva, tehnologija) koja se neprekidno mijenja.

Scrum se sastoji se od tri faze:

a) **Faza prije igre** sastoji se od:
   I. **Planiranje**: obuhvata definiranje sistema koji će se razvijati. Stvara se lista koja definira sve osobine koje treba sadržavati finalni proizvod, a bazirana je na trenutnim saznanjima Nakon toga, vrši se dodjela prioriteta za zahtjeve te se procjenjuje potreban intenzivni rad za njihovu realizaciju.
   II. **Arhitektura na visokom nivou**: planira se dizajn sistema na osnovu podataka iz kreiranih listi. Ako se radi o poboljšavanju postojećeg sistema, uočavaju se promjene koje je potrebno provesti za implementaciju stavki iz liste, kao i problemi koji mogu iz toga nastati.

b) Faza razvoja (igre): unutar faze razvoja promatraju se i kontroliraju različite varijable okoline i tehničke varijable (vremenski okvir, kvaliteta, zahtjevi, sredstva, tehnologije i alati za implementaciju) koje se mogu mijenjati tokom procesa neprekidno, kako bi se mogla fleksibilno prilagoditi promjenama. U ovoj fazi se sistem razvija iterativno u ciklusima (sprintovima) u kojima se razvija nova funkcionalnost ili poboljšava postojeća.

c) **Faza poslije igre:** u ovoj fazi nema više starih zahtjeva za raditi niti se mogu izmišljati novi zahtjevi. Sistem je spreman za isporuku te se radi priprema za to. Priprema uključuje testiranje, integraciju i dokumentiranje.

U svakoj fazi se definišu kontrolne tačke na kojoj dolazi do slaganja između projektnog tima i kupaca ili projektnih dioničara u vezi isporuke i cilja projekta.

U procesu definisanja i stvaranja softverskog proizvoda uključeno je više učesnika koji mogu da imaju (pored ranije definisanog kupca i korisnika) sljedeće uloge:

1. Razvojni tim: čine ga softverski inženjeri specijalizovani za različite aspekte razvoja. Uloge koje učestvuju u procesu razvoja softvera metodologijom zasnovanom na Scrum, u biti čine male razvojne timove od 5 do 10 osoba u zavisnosti od obima projektnog zadatka. Uloge definisane Scrum metodologijom su:
   - **Scrum Master** je odgovoran za osiguravanje da se projekt izvodi prema pravilima, vrijednostima i fazama Scrum-a te da napreduje bez značajnih prepreka kako je planirano, kako bi se osigurala maksimalna produktivnost. Surađuje sa projektinim timom, kupcem i upraviteljima.
   - **Vlasnik projekta** je službeno odgovoran za projekt, te za upravljanje, kontrolu i vidljivost listi zahtjeva. On donosi konačne odluke o zadacima, učestvuje u procjeni intenzivnog rada za stavke iz listi zahtjeva i pretvara stavke iz listi u osobine programskog rješenja koje je potrebno implementirati. Njega postavlja Scrum Master, kupac i uprava.
   - **Scrum tim** je projektni tim koji ima ovlasti da odlučuje o aktivnostima potrebnim za ostvarivanje ciljeva iz svakog Sprinta, te se može samostalno organizirati. Scrum tim je uključen u procjenu rada, stvaranje i pregledavanje listi i predlaganje uočenih prepreka koje treba izbaciti iz projekta.
   - **Kupac** učestvuje u zadacima vezanim za uspostavu listi zahtjeva za sistem koji se razvija ili poboljšava.
   - **Uprava** je odgovorna za donošenje konačnih odluka kao i za praćenje donešenih standarda i dogovora. Uprava također sudjeluje u postavljanju ciljeva i zahtjeva.

Opisane uloge ne moraju uvijek da pripadaju različitim učesnicima. U nekim manjim projektima, ista osoba ili grupa može da ima dvije ili čak i više uloga.

Scrum pomaže u poboljšanju postojećih inženjerskih postupaka unutar organizacije budući da uključuje upravljačke aktivnosti koje imaju cilj da sistemski uočavaju nedostatke u razvojnom procesu. Propisuje načine upravljanja zahtjevima, iteracijama razvoja, implementacijom i isporukom. Ova metodologija se može primijeniti za upravljanje bilo kojom inžinjerskom praksom, ali preporučuje se kod razvoja financijskih, Internet i medicinskih aplikacija.

B. Vanjski razvoj programskih rješenja

Angažovanje vanjskih saradnika za razvoj programskih rješenja ili njegovih dijelova, podrazumjeva pružanje pomoći u obrazovanju radnika informatičke struke, pomoć pri analizi poslovnog sistema i oblikovanju IS ili obavljanje analize i oblikovanja. Također se podrazumjeva kodiranje (generisanje) cjelovitog programskog sistema, upravljanje

izvođenjem i/ili nadzor izvođenja, kao i konsultativna pomoć prilikom ugradnje složenih poslovnih funkcija. Varijante su slijedeće: ugovoreni razvoj, odnosno ugovara se isporuka gotovog proizvoda ili dugoročna saradnja sa isporučiocem, uz izdvajanje vlastitog informatičkog odjela u glavnog izvođača. Moguća varijanta je i nalaženje strateškog partnera na duži vremenski period, npr. softverske kuće. Prednosti su višestruke. Kompletna programska rješenja ili njegovi dijelovi izrađuju se po mjeri naručioca, sistem je prilagođen organizaciji/poslovanju, a po mogućnosti treba istovremeno poboljšati organizaciju/poslovanje poslovnog sistema. Ovakav razvoj podrazumijeva dugotrajan postupak i odgovarajuću visoku cijenu. Nedostaci i rizici su takođe prisutni. Dolazi do gubitka povjerljivih informacija, gubitka nadzora nad sadašnjim i/ili budućim razvojem (zavisnost o dobavljaču), kao i gubitak vlastite stručnosti. Bitno je da upravljanje projektom informatizacije na sebe preuzme vlastito kompetentno osoblje koje ima mogućnost odlučivanja.

Vanjski razvoj programskih rješenja može biti realiziran putem:

1. **Nabavke gotovih aplikacija**

   Gotovi aplikativni paketi koje institucije Bosne i Hercegovine treba da nabavljaju prvenstveno se odnose na programski paketi za kancelarijsko poslovanje (npr. Desktop aplikacije), programi za upravljanje dokumentima ili specijalističke aplikacije za određene namjene koje u potpunosti zadovoljavaju poslovne potrebe institucije.

2. **Nabavka izvedbenog programskog koda**

   Prednosti nabavke izvedbenog koda su: izvedbeni kod je jeftiniji, brigu i odgovornost o njegovom održavanju preuzima isporučilac, uz izuzetak nekih opšte primjenjivih komercijalnih programa. Prednost izvedbenog koda je i ta da se ne mora kupiti (skupi) razvojni programski alat u kojem je programski proizvod razvijan. Nedostatci izvedbenog koda, obzirom na korisnika, su: izvedbeni kod podrazumijeva potpunu zavisnost od isporučioca, ne postoji mogućnost prilagođavanja specifičnim vlastitim potrebama, osim putem posebnog dogovora sa isporučiocem. Dodatna prilagođavanja lako mogu postati predmetom ucjene. Takođe, ne postoji mogućnost razvoja programske opreme vlastitim snagama.

   Nabavka izvedbenog programskog koda se preporučuje u sljedećim slučajevima:

   - kada se radi o standardnim, masovno prodavanim aplikacijama
   - kada korisnik nema vlastite informatičke stručnjake
   - kada se radi o visokostručnim aplikacijama koje se neće mnogo mijenjati, a korisnik nema namjeru da se baviti detaljima te struke
   - kada korisnik nema novaca ili želje za vlastiti informatički razvoj.

3. Nabavka izvornog programskog koda

Prednosti nabavke izvornog programskog koda su znatne. Izvorni kod omogućava stalni razvoj i praćenje vlastitih posebnosti. Osim toga, pruža mogućnost prilagođavanja vlastitim potrebama, što daje elastičnost pri promjenama organizacije poslovanja.
Nedostaci narudžbe izvornog koda su, takođe prisutne. Izvorni kod je višestruko skuplji od izvedbenog. Potrebna je razvojna varijanta programskog alata u kojem je IS razvijen. Naručilac se izlaže iskušenju da nekompetentno mijenja nabavljeni izvorni kod, onesposobi aplikaciju za rad i izgubi pravo na održavanje. Održavanje je skuplje ukoliko se radi o programskoj opremi podložnoj promjenama. Sniženje cijene izvornog koda može se postići automatizacijom kodiranja, upotrebom generatora izvornog koda.

Nabavka izvornog programskog koda se preporučuje u sljedećim slučajevima:

- kada programska oprema predstavlja stratešku investiciju
- kada korisnik raspolaže kompetentnim informatičarima ili ima motiva razvijati vlastitu informatičku djelatnost
- kada isporučilac ne može preuzeti obavezu održavanja ili ne može garantovati da će ostati na tržištu
- kada na tržištu ne postoji IS koji odgovara potrebama, ne može se povoljno kupiti sličan, a korisnik raspolaže vlastitim informatičkim snagama dovoljnim za projektovanje novog.

C. Alati i tehnike razvoja programskih rješenja

Tehnike razvoja programskih rješenja, kao posebna disciplina, bavi se ekonomsko-tehnološkim aspektima izrade programskih rješenja. Ona se posmatrati sa dva aspekta:

- Konvencionalna tehnologija (ručno programiranje, definisanje i dokumentovanje problema i održavanje sistema)
- Nekonvencionalna tehnologija (Sistemski inženjering pomoću računara - CASE)

Tehnike i koraci ova dva aspekta su slični, a suštinske razlike ogledaju se u načinu kontrole, izrade i održavanja programskih rješenja.
U cilju što kvalitetnijeg razvoja programskih rješenja, preporučuje se korištenje CASE (Computer-Aided System Engineering) alata koja automatizuje cjelokupnu metodologiju razvoja, ubrzava, automatizira mnoge faze procese razvoja sistema i poboljšava kvalitet izradi programskih rješenja.

Vrsta alata koje se koriste u fazama razvoja programskih rješenja:

- **Alati za planiranje poslovnih sistema:** oni prate informacione tokove između organizacionih jedinica.
- **Alati za upravljanje projektima:** oni prate glavne upravljačke aktivnosti, npr. planiranje, procjena vrijednosti, resursi, rizik, troškovi, kvalitet, standarde mjerenja...

- **Alati podrške:** koriste se za dokumentovanje, podršku sistemskom softveru, obezbjeđenje kvaliteta i upravljanje bazama podataka.
- **Alati za analizu i dizajn:** oni se smatraju najvažniji alati, omogućavaju izradu sistema.
- **Alati za programiranje:** oni podržavaju izradu programskog koda
- **Alati integracije i testiranja:** oni prate prikupljanje testnih podataka, analiza izvornog koda i pomoć u aktivnostima testiranja
- **Alati prototipskog razvoja:** oni služe za izradu prototipa
- **Alati za podršku održavanju:** oni se koriste za reverzibilni inženjering, rekonstrukciju koda i reinženjering

Navedeni alati i tehnologije doprinose planiranju, razvoju i implementaciji programskih rješenja. Ipka, korištenje i upotreba istih zahtjeva visoko obrazovane korisnike koji će koristiti ove tehnologije kako bi povećali produktivnost, skratili vrijeme izrade i unaprijedili preformanse sistema.

2. METODOLOGIJA STRATEŠKOG PLANIRANJA INFORMACIONIH TEHNOLOGIJA

2.1. Svakodnevni menadžment informacionih tehnologija

Svakodnevni menadžement IT-a je zapravo upravljanje životnim ciklusom aplikacija, a to podrazumijeva kontinuirani proces upravljanja različitim fazama u cjelokupnom životu softvera kroz automatizaciju procesa i integraciju informacija svake faze procesa. Životni ciklus razvoj programskog rješenja treba da pokriva cjelokupni razvoj, od početnih ideja i zahtjeva, preko implementacije, isporuka, održavanja pa sve do trenutka kad sistem gubi svoju poslovnu vrijednost te se prestaje koristiti. Zbog svoje kompleksnosti životni ciklus aplikacije treba da bude predmet kontinuiranog upravljanja, kontrole, evaluacije, razvoja i održavanja.

Životni ciklus treba da definiše faze i aktivnosti, koje treba obaviti tokom razvoja, bez obzira na veličinu sistema koji se gradi. Svaka pojedina aktivnost proizvodi skup rezultata. Ciklus osigurava "kontrolne tačke" za praćenje napretka, procjenu postignutih rezultata i donošenje odluka o daljnjim koracima.

Tipične faze u životnom ciklusu aplikacije kroz koje prolazi su:

a) **Faza upravljanje zahtjevima:** je faza u kojoj se identificiraju, dokumentiraju, prioritiziraju i prate korisničke potrebe. Tu se precizno definiraju karakteristike koje program mora zadovoljavati. Sve zahtjeve je neophodno iskomunicirati i usuglasiti sa relevantinim dionicima.

b) **Faza dizajna i arhitekture:** su faze u kojima se zahtjevi mapiraju u arhitekturu, u smislu organiziranja sistema u strukturne elemenate, njihova interfejsa i ponašanja na višoj razini (arhitektura) i na nižoj detaljnoj razini (dizajn).
c) **Faza razvoj i testiranje:** su faze tokom koje se moduli i komponente kodiraju, dokumentiraju, testiraju a identificirana odstupanja se ispravljaju, tj. u ovom dijelu softverski inženjeri pokušavaju uočiti i ukloniti greške koje su nastale kod implementacije, te općenito optimiziraju softverski proizvod koji je krajnja isporuka ove faze.
d) **Faza implementacija:** je faza u kojoj se planiraju, izvode i kontroliraju aktivnosti koje imaju za cilj integraciju software-a u postojeću IT okolinu tako da postane dostupan za korištenje. Tipične aktivnosti u implementaciji su instalacija, migracija, verzioniranje, adaptacija i ažuriranje.
e) **Faza održavanje aplikacije:** je životna faza u kojoj je aplikacija ili program u toku praktične svakodnevne primjene. Ovo uključuje održavanje sistemskog okruženja, baze podataka, portale i aplikacijske verzije i međuverzije. Također, obuhvata nadgledanje aplikacije i rješavanje problema kad se pojave. U toj svrsi dokumentacija koja se piše prilikom planiranja i razvoja sistema mora biti pisana cjelovito, jasno i precizno kako bi se maksimalno olakšao posao održavanja softvera drugim inžinjerima koji će za to biti odgovorni.
f) **Faza optimizacije:** je faza aplikacije koja uključuje poboljšanje, dodavanje komponenti i elemenata sistemskog okruženja, potrebne korekcije i podešavanje performansi.

3. METODOLOGIJA RAZVOJA I EKSPLOATACIJE INFORMACIONH SISTEMA

Metodologija razvoja informacionih sistema obezbjeđuje potpunu kontrolu i koordinaciju svih aktivnosti koje treba sprovesti da bi se proizveo željeni softver i ispunili ciljevi projekta.

3.1. Eksploatacija i razvoj programskih rješenja

Metodologija se može definisati kao metoda + idejni pristup. Sadrži u sebi kolekciju procedura, tehnika, alata i dokumentacionih pomagala, potkrijepljenih filozofijom, koji potpomažu izgradnju programskih rješenja. Metodologija predstavlja skup načela izrade programskih rješenja, koji se u određenoj situaciji svodi na metodu jedinstveno prikladnu toj situaciji.

Komponente metodologije se mogu razvrstati u slijedećih pet tačaka:

1. Etape projekta;
2. Zadaci za svaku pojedinu etapu;
3. Izlazi (projektna dokumentacija);
4. Preporuke (vodič) upotrebe odabranih tehnika i pomagala;
5. Način upravljanja projektom i nadzor projekta.

Cilj metodologije je da omogući sistemski postupak razvoja kojim će se moći pratiti napredak, uspostavi komunikaciju između učesnika uključenih u izgradnju IS (poslovodstvo, korisnici, analitičari, programeri), osigura skup tehnika koje će omogućiti da se zadaci izvršavaju na standardne i provjerene načine, osigura efikasan nadzor sa ciljem uočavanja grešaka u ranim fazama. Osim navedenog, cilj metodologije je da omogući elastične promjene poslovanja i tehnologije (npr. odvajanjem analize i oblikovanja), uokviri razvojnu strategiju kojom će se ukloniti ad hoc rješavanje problema, odredi ili ukaže kada i u kojoj mjeri je potrebno uključivanje korisnika, te potiče i omogućava uključivanje korisnika kada se za to ukaže potreba. Metodologija omogućava da se dovoljno pažnje posveti analizi poslovanja, čime će se osigurati izrada sistema koji odgovara poslovanju i zahtjevima korisnika.

Ekonomičnije je otkriti i popraviti grešku u ranim fazama, jer je lakše popraviti dokumentaciju nego mijenjati programski kod. Izmjene u kasnijim fazama zahtjevaju promjene rezultata prethodnih faza. Lakše je pronaći grešku tokom faze u kojoj je nastala, nego tražiti je i popravljati na osnovu posljedičnih učinaka primijećenih u kasnijim fazama.

4. METODOLOGIJA UPRAVLJANJA PROJEKATOM

4.1. Upravljanje promjenama tokom razvoja i imlementacije programskih rješenja

Poslovni procesi unutar svake organizacije, pa tako i javne uprave, mijenjaju se s vremenom. Uobičajno je da organizacija mijenja strukturu i zadaće, dolazi do promjena modela upravljanja, razvijaju se modeli suradnje s drugim tijelima, a također i s vanjskim partnerima. Informacioni sistem treba pratiti sve te promjene, a koje zajedno ulaze u kategoriju nadogradnji i poboljšanja.

Ovi procesi su u osnovi slični procesima kod razvoja novog informacionog sistema. Naime, nadogradnje i poboljšanja obično uključuju izradu novih elemenata informacionog sistema, različitog obima, ovisno o potrebama konkretne nadogradnje ili poboljšanja.

Procesi nadogradnji i poboljšanja dijele se u pet faza:

4.1.1. Faza analiza utjecaja

Analiza utjecaja je iznimno bitan proces kod nadogradnji i poboljšanja informacionog sistema. Potrebno je pažljivo pripremiti novu verziju i ispuniti sve preduslove kako bi se nadogradnje i poboljšanja sistema realizirali bez dodatnih poteškoća u predviđenom vremenu i unutar predviđenih troškova.

Prije rada na modifikacijama postojećeg informacionog sistema potrebno je precizno definirati:

- Modifikacije koje će biti uključene u novu verziju,
- Dijelovi informacionog sistema koji će biti zahvaćeni i postojanje alternative nadogradnje,
- Rješenja koja će se za koristiti za nadogradnju,
- Vrijeme kada će biti modifikacija spremna.

Ovi podaci će se koristiti za planiranje potrebnih kapaciteta, vremena i načina implementacije.

4.1.2. Faza dizajn

Svrha faze dizajna je specificirati potrebne promjene aplikacija, kako bi informacioni sistem mogao funkcionirati u skladu sa definiranim zahtjevima.

Dobar dizajn aplikacija ispunjava nekoliko zahtjeva:

a) pruža nedvosmislenu definiciju o informacionom sistemu, odnosno dijelu informacionog sistema koji će se mijenjati, kako bi se tačno znao obim projekta;

b) uređuje komunikaciju s naručiteljem (nadležnima unutar tijela javne uprave), kako bi bili upoznati koje modifikacije ili funkcionalnosti će biti obuhvaćene;

c) priprema dokumente potrebne za prihvatanje i puštanje u rad unaprijeđenog sistema, kako bi se izbjegli kasniji nesporazumi, ovo uključuje i preciziranje pravnog okvira završetka posla.

Ova faza treba da osigurava interoperabilnosti dizajna unutar i izvan granica sistema.

4.1.3 . Faza realizacije

Realizacija može uključivati modifikaciju postojećih aplikacija, odnosno razvoj novih. Uz modifikaciju, odnosno izradu novih aplikacija potrebno je po potrebi prilagoditi i okoliš u kojem će se aplikacija koristiti (od tehničkih uvjeta do obuke službenika). Također, potrebno je prilagoditi i dokumentaciju, backup procedure te pravila informacijske sigurnosti.

Potrebno je nadzirati kvalitetu rada pri realizaciji. Također, potrebno je pripremiti aplikaciju za testiranje, u skladu sa nekom od priznatih metodologija testiranja softvera.

4.1.4. Faza testiranja

Upravljanje i razvoj aplikacija predstavlja radno intenzivan i kompleksan rad. Stoga nije za očekivati da će informacioni sistem, pa i njegova nadogradnja bilo kojeg tipa, biti razvijeni potpuno bez pogrešaka „bug-ova“. Jedan jedini bug može značiti da aplikacijski softver, pa time i cijeli informacioni sistem neće pravilno ili uopće funkcionirati, što može imati ozbiljne posljedice. Stoga isporučeni aplikacijski sistem ili bilo koja njegova nadogradnja moraju biti testirani. Ova faza je obvezna kod bilo kojih nadogradnji i poboljšanja informacionog sistema.
Faza testiranja osigurava da isporučeni softver i definicije podataka odgovaraju onome što je projektnom dokumentacijom trebalo biti isporučeno, u skladu sa specifikacijama koje su zadali korisnik i dizajneri informacionog sistema.
Proces testiranja mora biti strukturiran i precizno definiran. Potrebno je koristiti prihvaćene metodologije testiranja, alate i pristupe koji će jamčiti da će programski bug-ovi biti identificirani i riješeni. Stoga je potrebno fazu testiranja odgovarajuće ugovoriti i nadzirati. Testiranje mora biti dokumentirano i ta dokumentacija provjerena i prihvaćena od strane naručitelja.

Faza testiranja se dijeli u pet podfaza:

1. programsko testiranje – testiranje novog ili promijenjenog softvera;
2. testiranje tehničkog sistema – testiranje cjelokupnog informacionog sistema uključujući hardver, mreže, napajanje električnom energijom i slično;
3. funkcionalne testove – testiranje da li nove verzije podržavaju tražene nove funkcionalnosti;
4. operacionalne testove – određivanje da li tehnička infrastruktura može podržati nove verzije i funkcionalnosti;
5. test prihvatanja – u ovoj fazi imenovana osoba naručitelja odgovorna za procese koji su predmet rada aplikacija ili nadogradnji provjerava da li je proizvod u potpunosti isporučen u skladu sa svim specifikacijama.

4.1. 5 Faza uvođenja u rad

Uvođenje u rad projektovanog informacionog sistema uključuje instalisanje opreme, završni prenos podataka, te prelazak na novi način rada.

Aktivnosti i preduslovi koji se trebaju primjeniti:

a) Testiranje sistema;
b) Izrada plana konverzije (migracije) za uspješan prelazak na novi sistem. Definisati način uvođenja, poslove, odgovornosti, resurse i redoslijed izvedbe, izradu plana testa prihvatljivosti ukoliko nije urađen ranije;

c) Instalacija opreme, aplikacija i baze (baza) podataka novog sistema, inicijalni unos podataka, prenos postojećih podataka uz konverziju, uspostava sistema zaštite i održavanja;
d) Edukacija tehničkog osoblja i krajnjih korisnika, koja može biti verbalna ili putem raspodjele dokumentacije;
e) Konverzija sistema, prelazak na novi način rada, evaluacija projekta i sistema.

Preporučene metodologije koje institucije treba da koriste prilikom uvođenja programskog rješenja mogu biti:

- **Neposredno uvođenje** podrazumjeva početak rada novog sistema uz istovremeni prestanak rada starog sistema. Provodi se na određeni dan, uobičajeno nakon završetka poslovnog razdoblja, po mogućnosti na kraju sedmice. Mogući problemi su pojava grešaka koje nisu bile uočene tokom testiranja, nepredviđeno preopterećenje opreme u punom pogonu. Nedostatak je neposredna izloženost korisnika greškama sistema.

- **Paralelno uvođenje** podrazumjeva istovremeni rad starog i novog sistema tako dugo dok se ne pokaže da novi sistem ispravno radi i da su se korisnici navikli na novi način rada. Bitno je manje rizičan postupak u odnosu na neposredno uvođenje. Nedostatak je potreba za dvostrukom obradom istih podataka, u starom i u novom sistemu, što stvara otpor korisnika.

Korisnici mogu biti raspoređeni na različitim lokacijama. Probno uvođenje je neposredno/paralelno uvođenje sistema na jednoj lokaciji, a zatim i na ostalim lokacijama, nakon što se utvrdi da sistem ispravno radi. Postupno uvođenje je uvođenje grupa lokacija, dok istovremeno uvođenje predstavlja samo jedno uvođenje na svim lokacijama. Modularno uvođenje je postupna zamjena starog sistema novim, uvođenjem po dijelovima.