SI Vježbe 1 - b

Šta je razvojni proces? Šta je UP?

1. Uvod

Razvoj softvera nije jednostavan zadatak. Između dobre ideje (zahtjeva ili vizije) i funkcionalnog softverskog proizvoda nalazi se mnogo više od pukog programiranja. Između ideje i softvera stoji ***razvojni proces.***

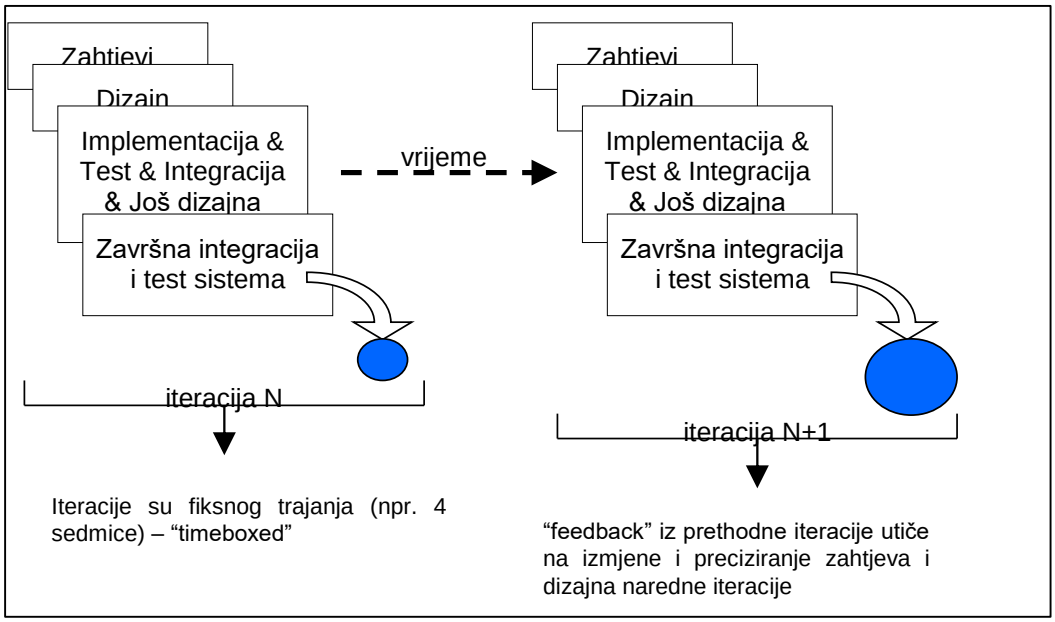
Razvojni proces softvera opisuje razvoj, stavljanje u funkciju i održavanje softvera. Često se o razvojnim procesima govori kao o "heavy" (teškim) ili "light" (lakim) procesima. Teški razvojni procesi imaju slijedeće osobine:

- Obilna dokumentacija koja se kreira u “birokratskoj” atmosferi
- Krutost i kontrola
- Dugoročno detaljno planiranje
- Predstavljaju “predictive”, a ne “adaptive” procese

U okviru “predictive” (predvidljivog) procesa pokušavaju se detaljno isplanirati i predvidjeti aktivnosti i alokacija resursa (ljudi, opreme, materijala...) za relativno duži vremenski period. Ovakvi procesi često imaju sekvencijalni (“waterflow”) životni ciklus:

1. Definisanje zahtjeva
2. Detaljan dizajn
3. Implementacija

Nasuprot tome, u okviru “adaptive” (prilagodljivog) razvojnog procesa, promjene se prihvataju kao neminovnost. Adaptivni procesi obično imaju iterativni životni ciklus. Iterativni razvoj softvera predstavlja proces u kom se softver postepeno razvija, dio po dio, u tzv. iteracijama.



Slika 1.1. Iteracije UP-a

Agilni razvojni proces je "laki" i "adaptivni" proces koji se prilagođava stalnim promjenama zahtjeva. Osnovne ideje agilnog razvoja softvera najbolje se očituju u njegovom manifestu (http://agilemanifesto.org/).

- Pojedinci i interakcije ispred procesa i alata
- Funkcionalan softver ispred sveobuhvatne dokumentacije
- Saradnja sa klijentima ispred ugovaranja i pregovora
- Prilagođavanje promjenama ispred praćenja plana

"The Unified Process" je primjer iterativnog procesa za razvoj softvera namijenjen projektima koji koriste OOA/D. UP je široko prihvaćeni proces razvoja. Osnovna ideja UP-a je postepeni razvoj u iteracijama ("iterative and incremental development"). Za vrijeme razvoja sistem postepeno raste nakon svakog "ciklusa" – iteracije. Ovakav proces predstavlja balans između "rush-to-code" (ishitrenog kodiranja) i dugotrajnog planiranja.

UP je veoma fleksibilan i otvoren razvojni proces, i prihvata dokazane metode i prakse iz drugih iterativnih metoda, kao npr. Extreme Programming (XP), Scrum itd. UP kombinira opšte prihvaćene prakse kao što su iterativni razvojni proces, razvoj vođen smanjenjem rizika itd. u kompaktan i dobro dokumentovan razvojni proces.

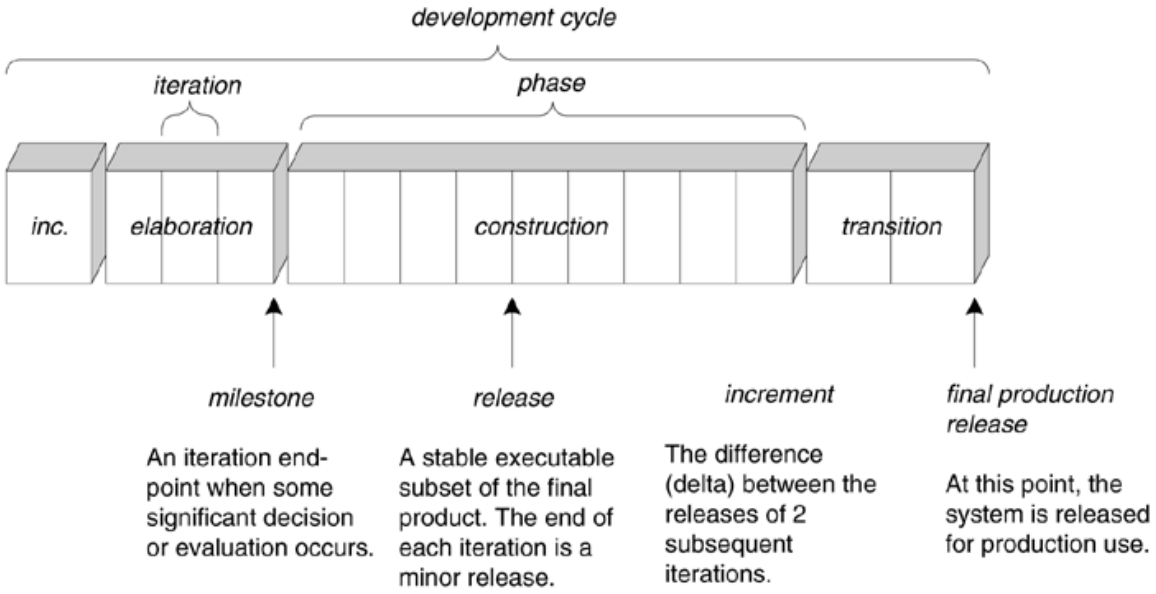
U okviru ovog materijala pokušat ćemo pratiti uputstva i prihvaćene prakse koje su opisane u okviru "Unified Process"-a.

2. Faze UP-a

Rad na UP projektu i iteracije su organizovani u četiri glavne faze:

1. Inception (vizija, definisanje okvira, poslovna opravdanost, grube procjene)
2. Elaboration (dorada vizije, iterativni razvoj osnovne arhitekture, otklanjanje visokog rizika, identifikacija većine zahtjeva sistema, realističnije procjene)
3. Construction (iterativni razvoj preostalog, manje rizičnog, dijela sistema)
4. Transition ("beta" testovi, isporuka)

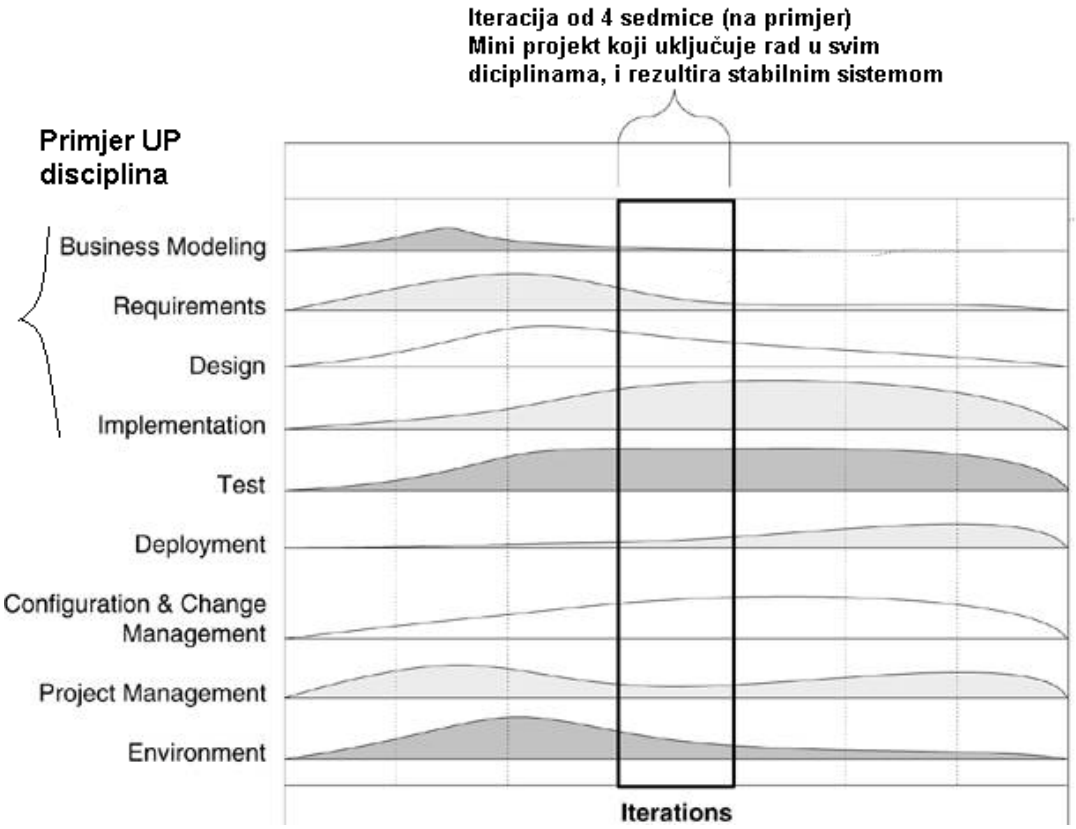
U okviru narednih materijala biće detaljnije objašnjena svaka od faza.



Slika 2.1. Faze UP-a

Treba napomenuti da ove faze ne predstavljaju stari "waterflow" razvojni proces (u kom se prvo definišu svi zahtjevi, pa se nakon toga radi dizajn i implementacija). U svakoj od faza moguće je izvršavanje različitih aktivnosti (ili disciplina) – analize, dizajna ili kodiranja. Pojedine discipline su više ili manje intenzivne u pojedinim fazama.

Slijedeći dijagram prikazuje primjer disciplina UP-a i intenzitet u pojedinoj iteraciji. Dijagram ne treba shvatiti doslovno kao recept, nego samo kao primjer intenziteta pojedinih disciplina u nekom prosječnom projektu.



Slika 2.2. UP discipline

U okviru narednih materijala pristupit ćemo razvoju sistema za studentsku službu, upravo prateći faze UP-a i time kroz primjere pokušati objasniti detalje pojedine faze.

3. Inception

Većina projekata zahtijeva kratak inicijalni korak u kome se istražuju odgovori na slijedeća pitanja:

- Šta je vizija projekta i da li postoji poslovna opravdanost?
- Da li je projekt izvodljiv?
- Kupiti ili izgraditi?
- *Gruba* procjena troškova (10KM, 10.000KM, 100.000KM ili 1.000.0000KM)?
- Nastaviti ili prekinuti projekat?

Upravo je to zadatak "inception" faze. Međutim, treba istaknuti slijedeće:

"Inception" nije faza u kojoj će se definisati svi zahtjevi sistema niti napraviti realistični planovi. To se dešava u "Elaboration" fazi.

Slična faza postoji i kod projekata koji nemaju za cilj razvoj softvera. Na primjer, u poslovima sa naftom, "inception" faza bi:

- odlučila da li postoji dovoljno dokaza čak i za sondažna bušenja
- ako takvi dokazi postoje, izvršila bi se sondažna bušenja i mjerenja
- napravile prve procjene i planovi za nastavak

Inception može biti veoma kratka faza. Ako se radi o manjem projektu to može biti sastanak od par sati na kome bi se odlučilo da li nastaviti sa projektom ili odustati.

Rezultat inception faze može biti niz produkata ("artifacts" u UP terminologiji). Ključna stvar je da će većina dokumenata koja nastane u ovoj fazi biti dorađivana u narednim fazama.

Slijedeći „artifakti“ mogu započeti u "inception" fazi (ključna riječ je "započeti"):

- Vizija (opisuje ciljeve na visokom nivou, ograničenja i sažetak projekta)
- Use-Case Model (opisuje funkcionalne zahtjeve i usko vezane ne-funkcionalne zahtjeve)
- Dodatna specifikacija (opisuje ostale, nefunkcionalne, zahtjeve)
- Rječnik pojmova (pojmovi iz domene problema)
- Plan otklanjanja rizika (opisuje poslovne, tehničke rizike, kao i rizike vezane za resurse i vremenski plan i ideje otklanjanja rizika)
- Prototipi (za pojašnjenje vizije i validaciju tehničkih ideja)
- Plan slijedeće iteracije (šta napraviti u prvoj iteraciji elaboration faze)
- Plan za narednu fazu (neprecizni plan "elaboration" faze)
- "Development Case" (plan UP koraka i artifakta)

Svi navedeni artifakti su opcionalni. Potrebno je odabrati samo one koji će pridonijeti vrijednosti projekta. "Development Case" je dokument u kome se navodi koji od artifakta će se izrađivati u kojoj fazi razvojnog procesa.

Na primjer slijedeća tabela mogla bi predstavljati "development case" za naš projekat razvoja softverskog sistema za studentske službe.

Disciplina	**Artifact** **Iteracija ->**	**Incep.** **I1**	**Elab.** **E1..En**	**Const.** **C1..Cn**	**Trans.** **T1..Tn**
Business Modeling	„Domain“ model		p		
Requirements	Use-case Model	p	r		
	Vizija	p	r		
	Popratna specifikacija	p	r		
	Riječnik pojmova				
Design	Dizajn model		p	r	
	Dokument arhitekture softvera		p		
	Model podataka		p	r	
Implementation	Implementacijski model				
Project Management	Plan razvoja				
Testing	Model za testiranje				
Environment	„Development Case“	p	r		

Tabela 3.1. Artifakti UP-a

Legenda:

p – preliminarna verzija dokumenta

r – detaljna razrada dokumenata

4. Razumijevanje zahtjeva

Zahtjevi su mogućnosti i uslovi koje sistem (ili šire projekat) mora ispunjavati. Osnovne aktivnosti rada sa zahtjevima su pronalazak, opisivanje i zapisivanje stvari koje su zaista potrebne u formi koja je prilagođena i klijentu i članovima razvojnog tima.

UP ne propisuje potpuno definisanje zahtjeva u prvoj fazi. Izmjena zahtjeva je moguća i u kasnim razvojnim fazama, a time se očituje ključna osobina tog razvojnog procesa – prihvaćanje promjena. Ključni termin u prethodnoj definiciji aktivnosti rada na zahtjevima je "pronalazak" jer je zahtjeve sistema potrebno pronaći kroz analiziranje postojećeg stanja, razgovore sa klijentima itd. U tom procesu vrlo je bitna komunikacija sa korisnicima.

U okviru UP-a i drugih iterativnih (evolucijskih) metoda (Scrum, XP, FDD, ...) programiranje i testiranje softvera dovoljno kvalitetnog za rad u stvarnom sistemu (production quality) počinje vrlo rano. Mnogo prije nego su svi zahtjevi sistema definisani, kada su definisani možda 10% ili 20% arhitekturalno bitnih, rizičnih zahtjeva i zahtjeva koji su kritični za poslovanje.

Statistika pokazuje da se, u prosjeku, 25% zahtjeva mijenja u okviru jednog softverskog projekta. Bilo koja metoda koja pokušava "zamrznuti" i potpuno definisati zahtjeve na početku projekta je fundamentalno pogrešna, zasnovana

na netačnim pretpostavkama.

Naravno, sve rečeno ne upućuje na to da treba odmah početi kodiranje i potpuno zaboraviti fazu definisanja zahtjeva. Postoji balans između te dvije krajnosti: iterativna, evolutivna analiza zahtjeva, kombinovana sa ranim, vremenski ograničenim iterativnim razvojem i čestim uključivanjem klijenata u cilju procjene i dobijanja povratnih informacija na osnovu djelomičnih rezultata.

Tipovi zahtijeva - FURPS+

U okviru UP-a zahtjevi sistema su kategorizovani prema FURPS+ modelu. FURPS+ je skraćenica sa slijedećim značenjem:

Functionality (mogućnosti, funkcionalnosti, sigurnost)

Usability (ljudski faktori, "help" sistem, dokumentacija)

Reliability (dozvoljena učestalost prekida rada, oporavke od pogrešaka, predvidivost)

Performance (vrijeme odgovora sistema, propusnost, tačnost, dostupnost, iskorištenost resursa)

Supportability (prilagodljivost, održavanje, internacionalizacija, konfiguracija)

+

Implementation (ograničenja resursa, programski jezici, alati)

Interface (ograničenja vezana za komunikaciju sa vanjskim sistemima)

Operations (upravljanje sistemom u operativnom stanju)

Packaging (način isporuke i instalacije)

Legal (pravna pitanja)

Pored ove kategorizacije, često se o zahtjevima razmišlja kao o funkcionalnim (vezanim za ponašanje sistema) i ne-funkcionalnim (sve ostalo).

Vezano za sistemske zahtjeve UP nudi nekoliko vrsta dokumenata (artifacts):

- **"Use-case" model** - Skup tipičnih scenarija koji upisuju način na koji se sistem koristi. Prvenstveno služi za definisanje funkcionalnih zahtjeva (ponašanje) sistema.
- **Dodatna specifikacija** – Sve ostalo, što nije opisano u use-case modelu navodi se u dodatnoj specifikaciji. Namijenjen je za definisanje nefunkcionanih zahtjeva kao npr. performance, licenciranje itd. Također ovdje se navode i funkcionalni zahtjevi koji se ne mogu upisati na use-case modelu, npr. generisanje izvještaja.
- **Rječnik stručnih pojmova** – U najjednostavnijem obliku definiše pojmove koji su bitni za sistem. Također opisuje posebne zahtjeve vezane za podatke (npr. validacijska pravila, dozovljene vrijednosti itd.). Riječnik može sadržavati opise atributa objekata, parametara operacija, izgled izvještaja i sl.
- **Dokument vizije** – Sažeti prikaz zahtjeva na visokom nivou apstrakcije, koji se razrađuju u domain modelu i dodatnoj specifikaciji. Predstavlja sumarni pregled osnovnih ideja sistema i poslovne opravdanosti.
- **Poslovna pravila** – Obično opisuju pravila poslovanja koja se ne mogu mijenjati i kojim se sistem mora prilagoditi. Najbolji primjer su recimo pravila oporezivanja koja su definisana zakonima. Ova pravila mogu

biti opisana i u dodatnoj specifikaciji, ali pošto nisu vezana za softverski sistem, nekad ih je korisno izdvojiti u poseban dokument ostavlja mogućnost iskorištavanja ovog dokumenta i u druge svrhe.

"Unified process" ne definiše format za ove dokumente. Oni mogu biti u obliku web stranica, tekstualnih dokumenata i sl. Postoje šabloni definisani za ove dokumenta (u okviru "Rational Unified Process" dokumentacije), ali ti šabloni služe samo kao pomoć u kreiranju navedenih dokumenata.

USE-Case model

Slučajevi upotrebe (use-cases) su tekstualne "priče" koje se koriste za otkrivanje i evidentiranje funkcionalnih zahtjeva sistema.

Akter (actor) je nešto učesnik koji komunicira sa sistemom (npr. osoba, drugi kompjuterski sistem, organizacija i sl.).

Scenario je određeni slijed akcija i interakcija između sistema i aktera. Za scenario se često koristi i termin **instanca načina korištenja**. Na primjer, scenario uspješno obavljene kupovine, ili scenario neuspješne kupovine zbog odbijene kreditne kartice.

Use-case (slučaj upotrebe**)** je skup povezanih uspješnih i neuspješnih scenarija koji opisuju *način* na koji akter *koristi* sistem za postizanje nekog cilja. definiše funkcionalnost sistema i okruženje sistema

Use-case model je skup svih zapisanih načina korištenja (model funkcionalnosti sistema i njegovog okruženja).

Slučajevi upotrebe (use-cases) su tekstualni dokumenti, a ne dijagrami. Use-case modeliranje je primarno čin pisanja teksta, a ne crtanja dijagrama.

Slučajevi upotrebe (use-cases) nisu vezani samo za OO programiranje (niti su sami po sebi objektno orijentisani). Međutim, slučajevi upotrebe predstavljaju ključne ulazne informacije za klasičnu OO analizu/dizajn.

Postoje tri tipa načina korištenja (u odnosu na detaljnost opisa):

- **sažeti** (brief) – Obično se jednim paragrafom teksta opisuje samo glavni scenario (uspješno izvršavanje scenarija)
- **neformalni** (casual) **–** Više paragrafa teksta koji opisuje glavni scenario uspješnog izvršenja ali i alternativne scenarije
- **formalni** (fully-dressed) – Svi koraci i varijacije scenarija se detaljno zapisuju. Također, sadrži popratne sekcije (preduslovi, garancije upjeha...).

Akteri mogu biti:

- **primarni akteri** – Primarni akter ima ciljeve koje ispunjava direktno korištenjem sistema koji se razmatra.
- **akteri podrške** – Akteri podrške pružaju neku uslugu sistemu koji se razmatra (na primjer, automatski sistem za validaciju kreditnih kartica, kao podrška plaćanju).
- **vanjski akteri** – Akter koji ima neki interes vezano za izvršavanje "use-case"-a, ali nije primarni niti akter podrške (npr. poreska uprava koja zahtijeva izvještaja o porezima).

5. Kako identifikovati slučajeve upotrebe (use-case)

Osnovna procedura za identifikovanje slučajeva korištenja je slijedeća:

1. Određivanje (odabir) granica sistema (da li posmatramo samo softversku aplikaciju, hardver i aplikaciju kao cjelinu, hardver i softver zajedno sa osobom koja ih koristi ill pak cijelu organizaciju)
2. Identifikacija primarnih aktera (to su oni učesnici koji imaju poslovne ciljeve koje ispunjavaju korištenjem sistema)
3. Za svakog aktera identifikovati ciljeve i to na onom nivou koji zadovoljava kriterij "elementarnog poslovnog procesa" (objašnjenje slijedi)
4. Definisanje slučajeva korištenja (use-cases) koji će ispunjavati identifikovane ciljeve (obično su ciljevi i slučajevi korištenja u 1-na-1 relaciji).

Ako se analizira samo softverska aplikacija (ako je to rezultat 1. koraka) onda se za analizu zahtjeva treba fokusirati se na slučajeve korištenja na nivou **elementarnih poslovnih procesa (EBP – elementary business process).**

EBP predstavlja aktivnost koju izvodi jedna osoba na određenom mjestu u jednom vremenu (odnosno u neprekidnom vremenskom periodu – ne traje danima ili mjesecima). Ta aktivnost je rezultat nekog poslovnog događaja i ima mjerljiv rezultat za poslovanje. EPP je aktivnost koja podatke ostavlja u "konzistentnom" stanju.

U "inception" fazi, samo 10% načina korištenja je opisano u formalnom obliku, dok su ostali opisani u sažetom obliku.