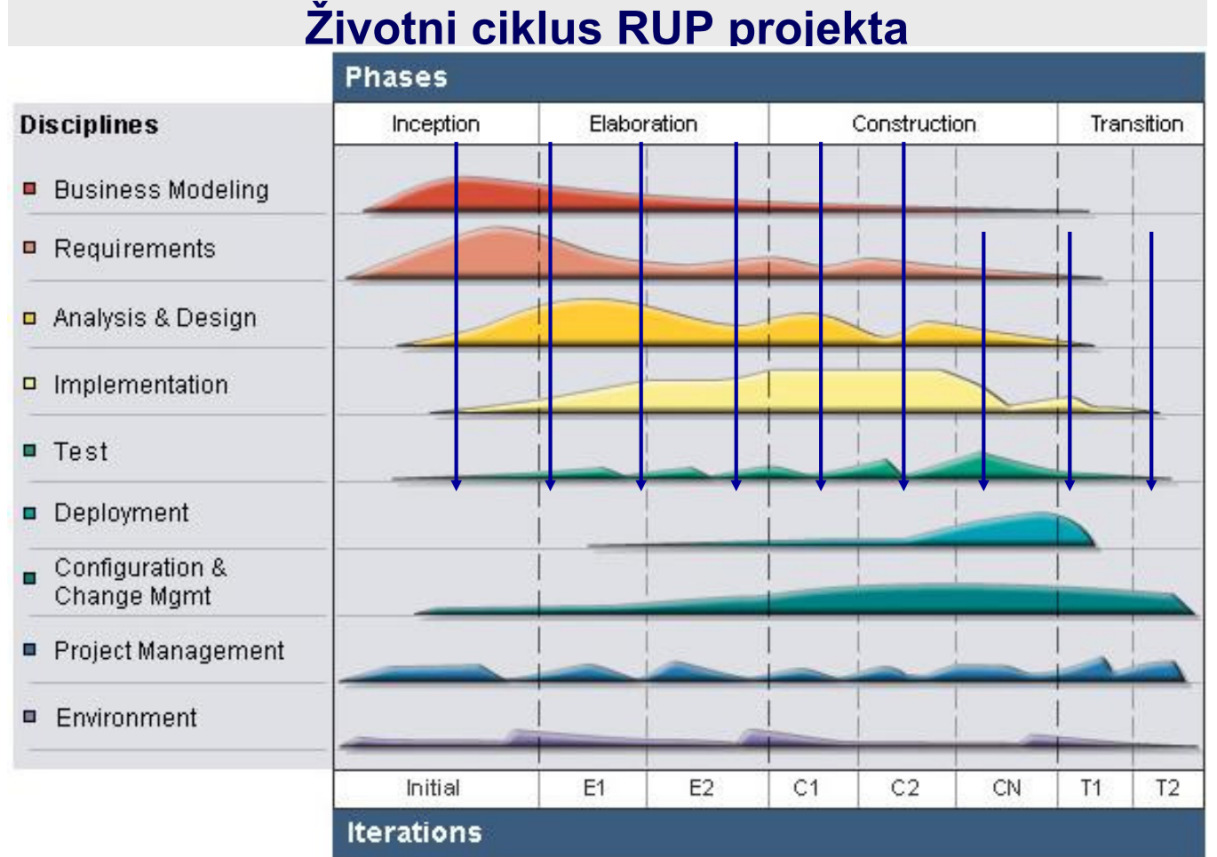
# ZI 2009-10

1.

A ja bi ovih 9 disciplina trebala moć izrecitirat jer zadovoljava pravilo 7+-2?

**Hmm nagađam da moramo znati navesti i ukratko opisati, bar tako piše u ovim ispitima s materijala, praktički su samo maknuli ove unesene faze i discipline**

2. Stupnjevi testiranja:

* Testiranje jedinica (*unit testing*) - najmanja jedinica je razred
  + Funkcionalno (*black-box*) - provjera što cjelina radi
  + Strukturalno (*white-box*) - provjera kako cjelina radi
* Integracijsko testiranje - jedinica je komponenta
  + Testiranje korisničkog sučelja - svaka funkcija sučelja
  + Testiranje slučajeva korištenja - svaki slučaj korištenja
  + Testiranje interakcije - svaki proces korak po korak
  + Testiranje sučelja sustava - razmjena podataka između sustava
* Testiranje sustava - provjera rada sustava kao cjeline
  + Testiranje zahtjeva - zadovoljenost poslovnih zahtjeva
  + Testiranje uporabivosti - prikladnost sustava za korištenje
  + Testiranje sigurnosti - mogućnosti oporavka i neautoriziranog pristupa
  + Testiranje performansi - izvođenje pod velikim opterećenjem
  + Testiranje dokumentacije - ispravnost dokumentacije
* Test prihvatljivosti - dokazivanje da proizvod zadovoljava zahtjeve
  + Alfa-testiranje - verifikacijsko, probna uporaba kod izvođača
  + Beta-testiranje - validacijsko, uporaba kod korisnika bez nadzora
  + Nadzorni test - potvrda da je sustav gotov

3. ZI 2011-12 3.

# ZI 2011-12

1. a) b) c) su isti u ZI 2014-15 1.

d)

**Uvođenje Rizik Trošak Trajanje**

Izravno visok nizak (ako uspije) kratko (ako uspije)

Paralelno nizak visok dugo

Probno nizak srednji srednje

Fazno srednji srednji dugo

Istovremeno srednji srednji kratko

Cijeli sustav visok srednji kratko

Modularno srednji visok dugo

2. Tri karakteristike projekta:

* raspodjeljen sustav (“sjedništa diljem svijeta”)
* heterogena okolina (“sjedišta u svakoj zemlji mogu koristiti drugačiji hardver i softver”)
* nova, nikad prije realizirana ideja

Po čemu ovo agumentira odabir XP? (referenca na...slajd?)

Meni ovo više liči na evolucijsku strategiju kod RUP-a. Al opet nemam pojma, jer za evolucijsku domenu kažu da je neiskusan projektni tim, dok se u zadatku to nigdi ne spominje.

**Ja bih isto rekao da je ovo RUP, mene ovo novo područje vuče na to**

[**http://www.fer2.net/showthread.php?p=1784525**](http://www.fer2.net/showthread.php?p=1784525)

U temi piše da su puno dijelili bodove na ovakvim tipovima zadataka, jer ne postoji “točan” odgovor, odnosno sve tri metodologije (RUP, XP, Scrum) i njihove verzije se mogu ukomponirati uz dobre argumente. Ako vam više paše RUP ovdje, možete ispod nadopisati drugu verziju odgovora koja koristi RUP.

XP (mislim da nije bitno koja):

* Istraživanje - bilježenje korisničkih priča; trajanje: dva tjedna
* Planiranje - prioritiziranje korisničkih priča; trajanje: jedan tjedan
* Iteracije do izdanja - razvoj; trajanje iteracije: jedan tjedan (kratko jer je sustav nov, nepoznata domena itd. pa da se ranije otkriju nejasnoće)
* Produkcija - uvođenje u primjenu; trajanje: jedan tjedan (dugo jer se zbog heterogene okoline očekuju problemi)
* Održavanje
* Smrt :C

3. Mislim da je ovaj dio gradiva izbačen. Na forumu spominju nekakav spasilački tim i tim specijalaca, a toga više nema u slajdovima.

4. ???

Klase sadržane unutar prodajapotrošnja imaju jednaka svojstva pa tu treba izdvojiti generaliziranu klasu? Al očito da su samo mogući postupci refaktoriranja vezani uz premještaj metodi i inline/extract class. Ipak moguće da fali stranica jer gospodarstvo.nije u kodu te bi to onda ispao normalan zadatak.

**Rješenje četvrtog od mene i frenda (okvirno, možda smo nešto zaboravili definirati / refaktorirati, nije nam se dalo previše zezati jer je zadatak užasno zapetljan):**

* Za životinju treba stvoriti apstraktnu klasu npr. Zivotinja koja sadrži osnovne informacije o životinjama koje su zajedniček za svaku životinju. Onda se stvaraju klase za svaku vrstu životinje (krava, svinja, guska, ….) koje nasljeđuju apstraktnu klasu Zivotinja i definiraju bilo koje vlastite moguće parametre.
* Dodatno se definiraju sučelja za životinje koje nesu jaja, daju mlijeko, itd. Te ta sučelja nasljeđuju odgovarajuće vrste životinja te se onda samo za te životinje piše kod koji izvršava odgovarajuću radnju iz nasljeđenog sučelja.
* ProdajaPotrosnja se skroz razdvaja. Za početak se odvaja prodaja izvan te se stvara apstraktna klasa ProdanaHrana koja sadrži dijelove zajedničke za sve vrste hrabe. Nakon toga se stvaraju klase za svaku vrstu hrane (Jaja, Mlijeko, Meso, …) koje nasljeđuju klasu ProdanaHrana i još dodatno definiraju dodatna svojstva (npr. Vrsta mesa kod klase Meso).
* Ovo sad ovisi o tome kako su oni zamislili zadatak, ali mi smo se odlučili na ovakvo rješenje prema opisu zadatka: Stvara se dodatna klasa EvidencijaProdaje koja se kompozicijom povezuje s vrstama hrane te sadrži liste prodanih elemenata po vrsti (npr. Za životinje koje nesu jaja imamo List<Jaja> prodanaJaja).
* KupljenaHrana se također odvaja iz ProdajaPotrosnja pod posebnu klasu istog imena, dok se ProdajaPotrosnja skroz eliminira. Nova klasa KupljenaHrana se referencira pod SkupinaZivotinja preko kompozicije kao lista za pojedinu vrstu kupljene hrane.
* SkupinaZivotinja bi ostala ista pošto je generička klasa i zapravo može biti SkupinaZivotinja<Kokoš>, SkupinaZivotinja<Guska>, itd.
* Metode iz Gospodarstva se sele, dok ostaju reference na metode (tj. Na klasu koja pazi na prodaju): EvidentirajKupljenuHranu ide pod apstraktnu klasu Zivotinja, EvidentirajKupljenuHranuZaSkupinu ide pod SkupinaZivotinja, dok metode za evidenciju prodanih jaja i mesa idu pod EvidencijaProdaje, dok u Gospodarstvu ostaje referenca na tu klasu
* Treba sve klase prilagoditi sukladno promjenama (npr. Promijeniti konstruktore, maknuti Enum svugdje gdje se koristio, itd.)

# ZI 2013-14

1. Prednosti: njet ulazi u naš ZI

* Pouzdani pristup podacima
* Uklanjanje redundancije podataka i operacija
* Brže ispunjavanje zahtjeva
* Smanjenje troškova
* Jednostavna prilagodba
* Unaprijeđena skalabilnost
* Olakšano održavanje
* Globalni doseg
* E-poslovanje

Nedostatci:

* Vremenski dugotrajan proces uvođenja sustava
* Trošak (a prednost je smanjenje troškova, go figure)
* Ovisnost o proizvođaču
* Mnoštvo mogućnosti i složenost softvera
* Potreba za prilagodbom postojećih poslovnih procesa
* Potreba za kvalificiranim osobljem
* Nedostatna uključenost i osposobljenost krajnjih korisnika

2. -

3. Trajanje sprintova je jednako, svaki sprint <30 dana (jedan do tri tjedna u praksi).

Artefakti:

* Product Backlog - preostali posao, lista poželjnih funkcionalnosti
* Product Backlog Item - korisnička priča koja izražava funkcionalnost
* Sprint Backlog - popis zadataka i statusa
* Sprint Task - opis kako napraviti jedan Product Backlog Item

4. -

# ZI 2014-15

1. Konverzija sustava - tehnički proces u kojem novi sustav zamjenjuje stari sustav

Provodi se u tri “dimenzije”:

* Način konverzije
  + Izravno uvođenje - početak rada novog sustava uz istovremeni prestanak rada starog sustava.
  + Paralelno uvođenje - istovremeni rad starog i novog sustava tako dugo dok se ne pokaže da novi sustav ispravno radi i da su se korisnici navikli na novi način rada.
* Lokacija konverzija:
  + Probno uvođenje - uvođenje sustava na jednoj lokaciji; nakon uspješnog pilota, uvodi se na ostalim lokacijama
  + Postupno uvođenje - uvođenje slijedno po grupama lokacija
  + Istovremeno uvođenje - istovremeno uvođenje na svim lokacijama
* Modularnost konverzije
  + Uvođenje cijelog sustava - čitav sustav instalira se odjednom
  + Modularno uvođenje - postupna zamjena starog sustava novim, uvođenjem po dijelovima

2. Scrum:

* “...voditelj projekta želi svaki dan imati izvješće o napravljenom i preostalom poslu.”  
  ⇒ dnevni Scrum sastanci
* “Također, uprava Kombajna želi aktivno sudjelovati u izradi sustava na način da svaka 2 tjedna sudjeluje na radnim sastancima…”   
  ⇒ trajanje sprinta je dva tjedna
* “...i isprobava ono što je dotad napravljeno te označava zadatke dovršenim ili vraća na doradu.”  
  ⇒ likovi iz kombajna imaju ulogu klijenta (određuju je li ono što je do tad napravljeno dovršeno ili treba doraditi)

Napor = 13 000 čovjek \* sat

Trajanje = 16 tjedan

TjedniNaporProjekta = Napor / Trajanje = 13 000 / 16 = 812.5 čovjek \* sat / tjedan

DnevniNaporČovjeka = 5 sat / dan

TjedniNaporČovjeka = DnevniNaporČovjeka \* 5 = 40 sat / tjedan

VeličinaEkipeRazvojnika = TjedniNaporProjekta / TjedniNaporČovjeka = 812.5 / 40 ~= 21 razvojnika

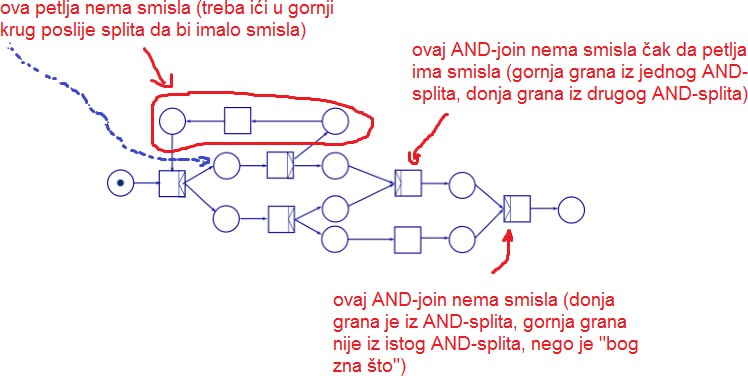
Ako je optimalno [5,9] razvojnika po timu, onih 21 se onda podijeli u 3 ili 4 tima. (npr. 7,7,7 i 5,5,5,6)

Dodatno još *product owner* i *scrum master*.

3.

* Petlja u gornjoj grani prvog AND-Splita završava ponovo u tom AND-Splitu što onemogućava združivanje njegovih paralelnih grana.
* Gornja grana prvog AND-Splita se združuje s gornjom granom drugog AND-Splita, što nema smisla.
* Donja grana drugog AND-Splita se združuje sa bog zna čime i to također nema smisla.

Pojašnjenje ovoga gore:



Zar join ne smije spajat grane iz različitih splitova? Ne znam baš.

Ništa drugo mi ne pada na pamet. Kada bi mogli, onda samo petlja ne bi valjala.

A da je bolja, nebi valjala!

**(by Milašinović)**

**Neispravnosti su u tome da gornja paralelna grana u slučaju ponavljanja može dovesti do toga da se pojave višestruke donje grane te da donja grana može otići u desnu donju čime AND split neće nikad biti izvršen i na kraju da AND join spaja 2 grane, od koji se zapravo samo jedna može izvršiti.**

* **gornja paralelna grana u slučaju ponavljanja može dovesti do toga da se pojave višestruke donje grane**
* **donja grana može otići u desnu donju čime AND split (valjda AND join?, i to u gornjem slučaju) neće nikad biti izvršen**
* **AND join spaja 2 grane, od koji se zapravo samo jedna može izvršiti.**

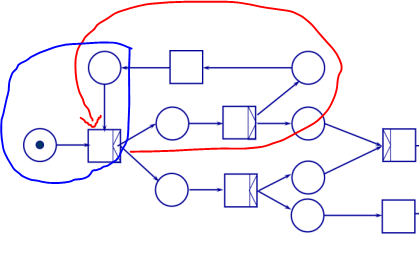
**Meni je svejedno nerazumljivo.**

**Donja grana može otići u x granu… pa AND split ide u obje grane?**

**Koliko sam skužio, ⅔ odgovora su ovo dolje moje šta sam spomenuo, jedino mi ovo treće nije jasno**

**Razbijam glavu i nikako mi nije jasno, tvoja prva 2 su ekvivalentna njegovim 2?**

**Mislim da je u zadnjem napravio lapsus, trebao bi biti AND-split**



**Šta ovo ilustrira? Petlju?**

**Crveno predstavlja ovo prvo Borisovo šta je jednako mojem drugom (petlja koja završava u petlji), dok plavo ilustrira ovo šta ja mislim da je treće Borisovo i moje prvo: AND-split s 2 ulaza**

**Aha, zar nebi boris trebo reći, AND split spaja 2 grane, dok zapravo AND split ne može spajati grane**

**Znači u AND split može ući samo jedna grana, a ako su dvije, onda se samo jedna od njih izvrši, to je bar moja teorija da je mislio pod ovim trećim**

**Aha, a njegovo drugo? Jel i tam napravio lapsus?**

**Sad baš gledam za njegovo drugo**

**Ja mislim da taj drugi znači da ako donja grana ode u jedan donju desnu granu, ona se nikad neće moći spojiti s gornjom granom te se opet neće moći spojiti obe grane u jednu, al nisam 100% siguran .. jer kad se izvrši neki slijed nad granom, to više nije ista grana**

**A glupo mi, ako donje grana ode… zar nije bezuvijetan odlazak u obje**

**Pa ja sam mislio da je, ali očito ima grananja na 2 strane**

**Mislim da sam skužio, donja grana mu je zapravo ona grana koja je stvorena u onoj petlji šta je kod Borisa prvo**

**Poslao Borisu mail za ovo pitanje, napisem kad odgovori (kad je mail poslan? Jer ću i ja poslati mail ako se ovo pisalo jučer ili kad već, zadatak je prezbunjujući)**

(Y), probao sam i ja taman taj riješiti pa sam napisao ispod

**...Bo-RIS**

**Ja isto taman rješavam ovaj, nisam baš siguran za ovo gore, al napišem dok riješim**

**Ja mislim da je prva greška skroz lijevi AND-split: jedan ulaz se u pravilu grana na 2, a ovdje imamo 2 ulaza. Druga je greška prva od gore: gornja petlja gdje AND-split petlja opet završava u njemu što onemogućuje združivanje paralelnih grana. Treća greška bi mogao biti donji AND-split (2. Od lijevo) -> jedna od grana aktivnosti obavlja neki slijed, dok druga grana aktivnosti ne obavlja nikakav slijed, samo spajanje?**

**Nez što želiš reć ovim trećim doduše, al da, ovi and split i join u sredini su redundantni, jedino pravilo koje se spominje u prezzama je to da paralelne grane ne smiju biti povezane - bi li se možda taj dio mogao gledati kao veza među paralelnim granama?**

**Hmm nisam siguran u vezi tog trećeg ni ja :/ mislim, nije obavljanje slijeda neka nužnost u splitu**

4. Mobilni objekt s poslovnom logikom se instancira na serveru koji ga šalje klijentu, pa klijent nešto s njim radi, pa ga šalje nazad serveru da server nešto s njim napravi i tako to. Poanta je da se “isti” objekt može koristiti malo na klijentu, malo na serveru. Ovo je moj opis njihovog primjera, iz predavanja mi ništa nije jasno.

Neda mi se pisat ovdje kompletno objašnjenje, ali da u suštini to je to, RIS07-DataPortal projekt kroz komentare nudi cijelo objašnjenje, pokretanjem instanci client i server…

Može netko potvrdit jesam li dobro shvatio:

To je to, da

**Napomena: Kad DataPortal instancira DataPortalProxy, zapravo se DataPortalProxy instancira kao udaljeni objekt na serveru a DataPortal ima samo remote controlling referencu na taj objekt (takva funkcionalnost se dobije postavkama u config datoteci)**

Ako klijent želi instancirati novi mobilni poslovni objekt, zove metodu DataPortal.Create<T> koja onda stvara posrednika (DataPortalProxy) za komunikaciju s poslužiteljom i poziva metodu DataPortalProxy.Create<T> koja na poslužitelju instancira novi mobilni poslovni objekt. Poslužitelj stvoreni objekt vraća klijentu.

U slučaju da klijent želi na poslužitelju izvršiti neku metodu poslovnog objekta, poziva metodu DataPortal.Execute<T>, koja poslovni objekt prosljeđuje poslužitelju preko DataPortalProxy.Execute<T>. DataPortalProxy.Execute<T> se zapravo izvršava na poslužitelju i na kraju izvršavanja poslužitelj klijentu vraća poslovni objekt.

5.

* GetCoinsValueAndDiscount(out int val, out int discount) - switch dio izdvojiti i prebaciti u Coin; razdvojiti u Value() i Discount(); napraviti hijerahiju novčića sa sučeljem Value() i Discount(); ovo što je ostalo u Payment također razdvojiti na CoinsValue() i CoinsDiscount();
* GetCartItemPrice(CartItem beverage) - prebaciti u CartItem; preimenovati u Price(); switch dio izdvojiti u novu metodu i prebaciti u Beverage te također preimenovati u Price(); napraviti hijerahiju napitaka sa sučeljem Price() i Name();
* GetTotalPrice(out int saldo) - promijeniti u skladu s gornjim promjenama (da koristi item.Price(), a ne GetCartItemPrice(item), isto i sa novčićima); preimenovati u GetTotalPriceAndSaldo;

**Nisam baš siguran je li ovo rješenje dobro (vidit ćemo kad kod sebe riješim do kraja), pogotovo kad sam gledao onaj njihov primjer RIS09-Refaktoriranje i u slajdovima, ali napišem kasnije dok kod sebe složim do kraja.**

**Dobro je ovo riješenje, ali ima još sitnica koje se daju refaktorirat, nez napamet ali ima jedan “replace temp with query”**

**Moguće pitanje:vrste održavanja sustava (nešto je to Fertalj spominjao na predavanjima):**

**Preventivno:**

podrazumijeva zaštitu od mogućih problema

redovita izrada sigurnosnih kopija (backup)

obavlja se periodički (dnevno, tjedno, mjesečno)

**Korektivno:**

podrazumijeva popravak nakon što se problem pojavio

vraćanje podataka iz sigurnosne kopije (restore)

uklanjanje uzroka pogreške (ispravljanje programa)

**Adaptivno:**

prilagodba funkcionalnosti (načina posluživanja)

prilagodba strukture (promjene strukture podataka)

poboljšanje performansi (optimizacija programa)

**Perfektivno:**

nadgradnja sustava da bi se riješili novi problemi

ugradnja novih mogućnosti (features)

Napomena, ak oćete dobre bodove sam naštrebajte RIS10.pdf napamet