

Osnovni pojmovi

2012/13.01

Podatak, informacija i znanje

□ Podatak

- podatak je sirova činjenica koja predstavlja neku istinu iz stvarnog svijeta
- pojedinačni podaci sami za sebe znače malo ili nemaju neko značenje

□ Informacija

- obavijest, predana novost ili znanje
- interpretacija podatka - pročišćen, organiziran i obrađen podatak u smislenom kontekstu
 - informacija je subjektivnog značenja, u kontekstu primatelja

□ Znanje

- gradi se vezivanjem novih informacija na postojeće znanje
- isti podaci mogu biti različito interpretirani ovisno o znanju primatelja
- primjer:

-130000	???
-130000 HRK	?
na računu	!
broj 1313 (tj. Mojem)	!!!

□ Informacijska znanost → informatika (informatics, information theory).

Sustav

- **Sustav = uređeni poredak međuzavisnih komponenti povezanih za postizanje zajedničke svrhe ili cilja.**
- **Fizički sustav – materijalne naravi**
 - skup jedinica, organizacijski ujedinjenih u cjelinu (npr. poduzeće)
 - skup dijelova, povezanih općom funkcijom (npr. živčani)
- **Apstraktни sustav – konceptualni, proizvod ljudske mašte**
 - oblik društvene organizacije (npr. državni)
 - oblik, način ustrojstva, organizacija nečega (npr. izborni)
 - uvjetovan planskim, pravilnim rasporedom dijelova (npr. sistem rada)

Elementi sustava

□ Komponente - podsustavi

- nerazdvojivi dijelovi ili agregacije dijelova koje pripadaju sustavu
 - npr. Nabava, Proizvodnja, Prodaja
 - npr. Studentska služba, Zavodi, Financijska služba
- mogu biti zamijenjeni bez potrebe za promjenom čitavog sustava (!?)
- organizacija, interakcija, međuzavisnost, integriranost

□ Granica - definira sadržaj i domašaj sustava

□ Okolina - sve što je izvan granica sustava, ali ga se tiče

- npr. MZOŠ i studentske udruge kao okolina visokog učilišta

□ Ulazi i izlazi - sve što ulazi u sustav iz okoline ili ga napušta

- sve ?

□ Sučelja - veze/dirališta između sustava i njegove okoline

- u kom smislu ?

□ Ograničenja – čimbenici koji određuju funkcioniranje sustava

- npr. ograničenja prostora ili ljudstva, pravilnici, zakonske odredbe

Informacijski sustav

- **Informacijski sustav (Information System) = skup međusobno povezanih komponenti koji prikuplja, obrađuje, pohranjuje i pruža informacije potrebne za obavljanje poslovne zadaće**
 - obavijesni sustav, informacijski sistem
 - **sustav za upravljanje sustavom ljudskih aktivnosti [Checkland, 1980]**
- **Svrha**
 - prikupljanje i pružanje informacija korisnicima u jednoj ili više organizacija → organizacijski IS (poslovni IS)
- **Podrazumijeva**
 - sustave podržane računalom → računalni (“kompjuterizirani”, “kompjuterski”)
 - sustave koji se ne oslanjaju na računala, ali obrađuju informacije
- **Korisnici**
 - poslovodstvo, djelatnici (zaposlenici, osoblje), klijenti ...

Poslovni i informacijski sustav

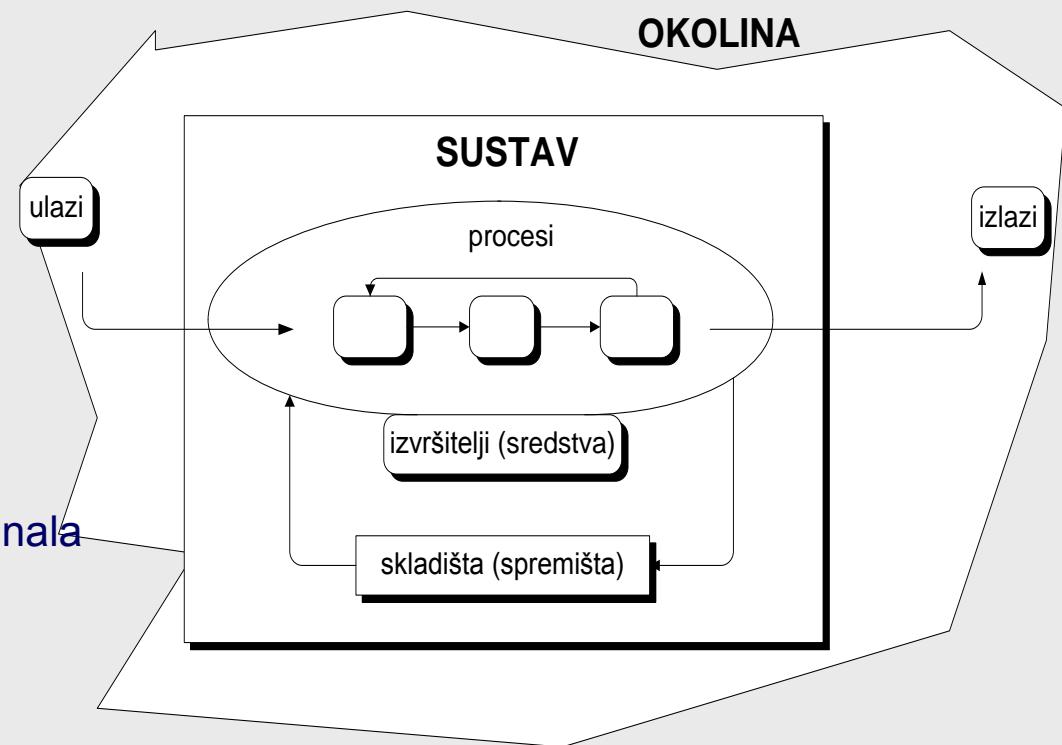
□ Poslovni sustav

- materijalni ulazi i izlazi (sirovine, energija, proizvodi) i informacijski tokovi (poruke, dokumenti, ...)
- procesi: obrada, prerada, proizvodnja
- izvršitelji: osobe, strojevi, alati
- skladišta: spremišta materijala

□ Informacijski sustav

- podsustav poslovnog sustava
- ulazi i izlazi: ulazne informacije, obrađene informacije
- procesi: obrada informacija (podataka) o stanjima stvarnog sustava
- izvršitelji: osobe, programi, računala
- skladišta: spremišta informacija (podataka)

- I(P)S: nabave, prodaje, proizvodnje, financija, ljudskih resursa i plaća, osnovnih sredstava
- IS Subvencionirane Prehrane (ISSP)
- Integrirani PIS Visokih Učilišta (IPISVU)
- IS Personalnog Upravljanja MORH



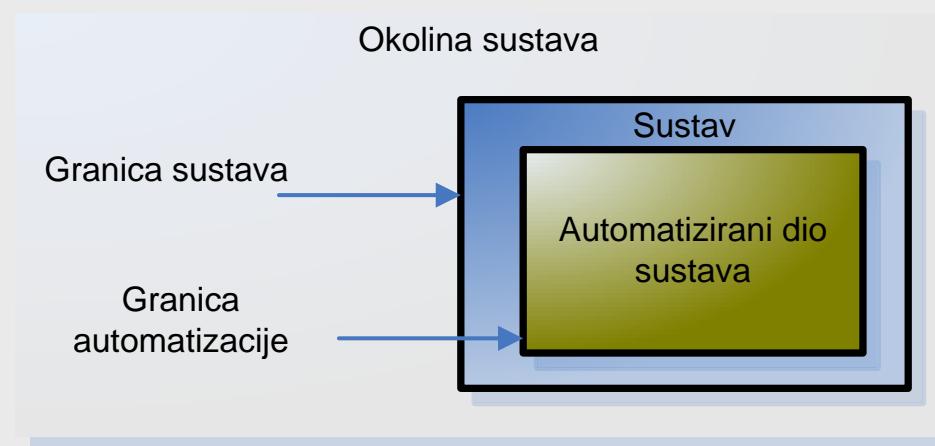
Informacijski sustav podržan računalom

□ Informacijski sustav podržan računalom

- Uređenje ljudi, podataka, procesa, sučelja, poveznica (mreža) i tehnologije koji djeluju međusobno sa svrhom podrške i poboljšanja svakodnevnih poslovnih aktivnosti (obrada podataka), kao i podrške rješavanju problema i donošenju odluka (informacijski servisi). [Whitten et. al, 2000]

□ Komponente IS

- programska oprema (software) – sistemska, namjenska
- računalna oprema (hardware) – računala i periferije
- osobe (lifeware) - korisnici, informatičari
- mrežna infrastruktura (netware) - mrežna oprema
- organizacija (orgware) – postupci povezivanja pojedinih dijelova u cjelinu
- ...



Slojevi poslovnih informacijskih sustava

- **Transaction Processing System (TPS), Data Processing System**
 - evidencija i obrada podataka o poslovnim transakcijama
 - pr. uplate i isplate po računima, upis predmeta, prodaja, narudžba robe
- **Management Information System (MIS) - upravljački**
 - obrada transakcijskih podataka
 - izrada izvješća potrebnih za upravljanje i nadzor poslovanja
- **Decision Support System (DSS) – za potporu odlučivanju**
 - odlučivanje na temelju podataka iz različitih izvora – transakcijskih sustava ali i povijesnih podataka
 - interaktivno okruženje za donošenje odluka (što-ako, analiza hijerarhijski strukturiranih podataka – drill down, grafički modeli poslovnih procesa)
- **Executive Information System (EIS) – (pod)varijanta DSS za izvršne rukovoditelje**
 - koristi se za donošenje strateških odluka
 - informacije dolaze iz IS organizacije ali i vanjskih izvora, npr. novosti o konkurenciji, ekonomska predviđanja, izvješća s burza dionica

Upravljanje IS

□ Upravljanje i razine IS

Sloj IS	Informacije	Učestalost	Korisnici	Upravljanje
transakcijski	obrada podataka	dnevno	niže poslovodstvo	operativno
upravljački	zbirne	periodički	srednje poslovodstvo	taktičko (trendovi)
za potporu odlučivanju	sintetizirane	“ad hoc”	više poslovodstvo, uprava	strategijsko (“što ako”)

Ostale vrste IS

□ Expert System (ES) – ekspertni sustav

- sustav pravila i mehanizam zaključivanja
- ugrađena stručna znanja za korištenje od strane manje

□ Office Automation System (OAS), Office support systems (OSS)

- potpora uredskom poslovanju
- upravljanje elektroničkom poštom, kolanje dokumenata, dijeljenje kalendara

□ Group Support System (GSS), Groupware

- potpora grupnom radu
- kolaboracija, obavješćivanje, zaključavanje dokumenata

□ Supply Chain Management (SCM)

- upravljanje lancem nabave
- integrira razvoj proizvoda, nabavu proizvoda, proizvodnju i upravljanje zalihamama.

□ Customer relationship management (CRM)

- upravljanje odnosom s kupcima
- podržava marketing, prodaju i servis, uključujući interakciju s korisnicima

Primjer poslovnog IS

□ Financijsko računovodstvo

- Upravljanje glavnom knjigom.
- Bilanca i Račun dobiti i gubitka (financijski izvještaji).
- Analitika kupaca.
- Analitika dobavljača.
- Računovodstvo osnovnih sredstava.
- Računovodstvo zaliha.
- Porezno računovodstvo

□ Upravljačko računovodstvo (Kontroling)

- Računovodstvo troškovnih centara.
- Računovodstvo profitnih centara.
- Računovodstvo internih naloga.
- Računovodstvo profitabilnosti.

□ Prodaja i distribucija

- Upravljanje nalozima za prodaju.
- Upravljanje ugovorima.
- Fakturiranje.

□ Nabava, skladištenje i logistika

- Zahtjevnice.
- Obrada narudžbenica.
- Evidencija primki materijala.
- Ovjera faktura (likvidacija).
- Upravljanje ugovorima.
- Upravljanje zalihami i skladištem, uključujući inventuru i fizički promet robe.

□ Upravljanje ljudskim potencijalom

- Kadrovska evidencija.
- Obračun plaća.

□ Proizvodnja

- Planiranje proizvodnje.
- Izvršenje proizvodnje.
- Proizvodnja po narudžbi.
- Procesna proizvodnja.

□ Upravljanje kvalitetom

- Ispitivanje kvalitete.

□ Izvještaji

Primjeri IS (nastavak)

□ IS biljnih vrsta

- Taksonomija
- Herbar
- Nalazišta
- Narodna imena
- Sinonimi
- Literatura
- Ekonomска upotrebljivost
- Zemljopisna rasprostranjenost
- Ugroženost
- ...

□ IS personalnog upravljanja

- Osnovni podaci o osobama (osobine, civilna i vojna povijest)
- Ustroj (postrojbe, dužnosti, ustrojna mjesta)
- Tijek službe (prijam, služba, posebna stanja, izlaz)
- Dodjela činova i redova, ocjenjivanje
- Dodjela odličja i medalja
- Školovanje (povijest, izobrazba u HV)
- Evidencija prekršaja, mjere
- Ostalo (naknade, doplatci, ...)

Značajke informacijskih sustava

□ Značajke informacijskih sustava

- složena okolina, koju je teško u potpunosti definirati
- složeno sučelje prema okolini, koje uključuje različite ulaze i izlaze
- složene veze između ulaza i izlaza (struktурно, algoritmički)
- obujam i složenost podataka

→ Problem projektiranja, izrade i održavanja

□ Važnost IS

- Informacija je postala upravljački resurs jednake važnosti kao što su vlasništvo (osnovna sredstva), ljudski resursi i kapital.
- IS sadrži/predstavlja znanje (know-how) organizacije koju podržava.
- IS i aplikacije pokazuju se prijeko potrebnim za održanje konkurentnosti ili postizanje kompetitivnog prestiža organizacije.

□ Organizacije su visoko ovisne o stalnoj raspoloživosti (vlasitog) IS!

Projektiranje i izgradnja IS

(ne)Uspješnost informacijskih sustava

□ The CHAOS Report , 1994

<http://www.standishgroup.com>

□ Prosječni trošak projekta

- velike kompanije: 2,32 M\$
- srednje kompanije 1.33 M\$
- male kompanije: 434 K\$

□ Prosječno prekoračenje

- troškova: 189%
- rokova: 222%

□ Uspješnost

- 16.2% uspješnih projekata
- 52.7% prekoračenje roka/troška ili reducirana funkcionalnost:
- 31.1% prekinutih projekata

□ CHAOS, 2002:

- 34% uspješnih
- 49% neuspješnih
- 17% propalih

□ CHAOS, 2009:

- 32% successful
- 44% challenged
- 24% failed

□ Čuveni neuspjesi

- London Ambulance System (1992)
- Taurus (1993)
- Denver Airport (1994)

Uzroci neuspjeha

- **Informacije o neuspješnim projektima u RH se rjeđe objavljaju.
Među najčešćim uzrocima su [Kalpić, Fertalj & Mornar, 2001]**
 - Loša organizacija i vođenje projekata
 - oslonac na vanjske voditelje i savjetnike, delegirano upravljanje projektima, nerealno planiranje i pretjerani optimizam, formalno izvještavanje o napretku, formalni nadzor nad projektom, podcijenjena uloga vlastitih stručnjaka
 - Loša izvedba projekata
 - neodgovarajuća analiza sustava, pogreške u dizajnu i kontroli kvalitete, neodgovarajuća CASE pomagala i krivo korištenje, pa čak i svojevrsna zloupotreba CASE pomagala
- **Mnogi sustavi su propali ili su bili odbačeni jer su se izvođači trudili napraviti lijepa programska rješenja a nisu razumjeli suštinu organizacije i poslovanja.**

Poboljšanje uspješnosti IS

□ Poboljšanje uspješnosti IS

- Projektiranje IS
- Planiranje, upravljanje provedbom, praćenje napretka
- Uključivanje korisnika
 - Korisnik poznaje(?) poslovni proces i zna(?) odrediti potrebe.
 - Informatičar upoznaje(?) poslovanje i zna(?) kako izraditi IS.
 - Aktivna uloga rukovodstva

□ Sažetak načela razvoja informacijskih sustava

- Korisnici i vlasnici sustava moraju biti aktivno uključeni
- Treba koristiti pristup koji vodi k rješavanju problema
- Uspostaviti faze i aktivnosti
- Uspostaviti standarde za konzistentan razvoj i dokumentiranje
- Opravdati izgradnju sustava kao kapitalnu investiciju
- Ne okljevati ako treba revidirati doseg ili otkazati projekt
- Projektirati za rast i promjene

Programsko inženjerstvo

- **Programska potpora, oprema = softver (software)**
 - programi + podaci + dokumentacija
- **Primijenjena programska potpora = aplikacija (application)**
 - namjenski program, primjenska programska potpora
 - računalom podržano rješenje jednog ili više poslovnih problema ili potreba
- **Programsko inženjerstvo (software engineering)**
 - sistematičan, discipliniran i mjerljiv pristup razvoju, primjeni i održavanju softvera = primjena inženjerskog pristupa na softver [IEEE Std 610.12-1990, 1994]
- **Informacijski sustav = sustav aplikacija za upravljanje ljudskim aktivnostima**

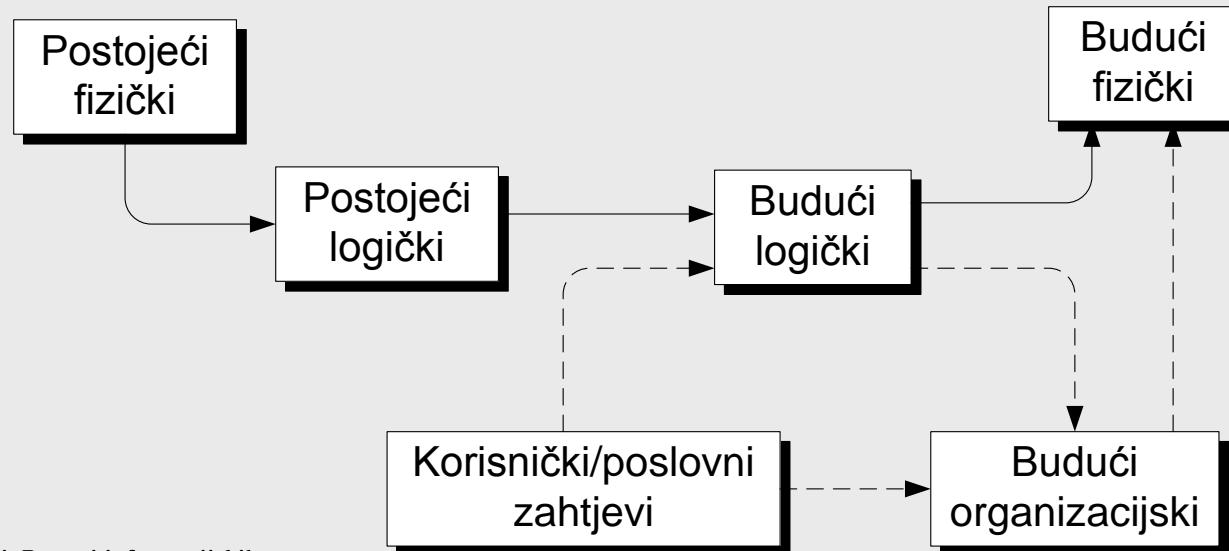
Informacijsko inženjerstvo

- **Informacijsko inženjerstvo (information engineering, IE)**
 - Inženjerski pristup izgradnji IS i upravljanju informacijama
 - IS mora biti projektiran, kao što se to čini s drugim proizvodima (!?)
 - razvoj IS kao strukturiran i planiran proces podržan računalom
 - Sustavna primjena prikladnog skupa metoda, tehnika i alata
- **Metoda (method), metodologija (methodology)**
 - smislen i organiziran skup procedura izrade (modela, softvera), uz korištenje dobro definirane notacije
 - definira primjenu tehnika i njihovo povezivanje (pr. ERD-DFD)
- **Tehnika (technique)**
 - definira način provođenja postupka i dokumentiranja rezultata tog postupka
 - npr. tehnika intervjuiranja, CSF, Delphi, ...
- **Pomagalo, alat (tool)**
 - instrument, sredstvo (artefact) koje se koristi u razvoju IS
 - npr. aplikacija za upravljanje projektima, CASE alat, generator koda

Projektiranje IS

❑ slijed izrade fizičkog i logičkog modela postojećeg i budućeg IS

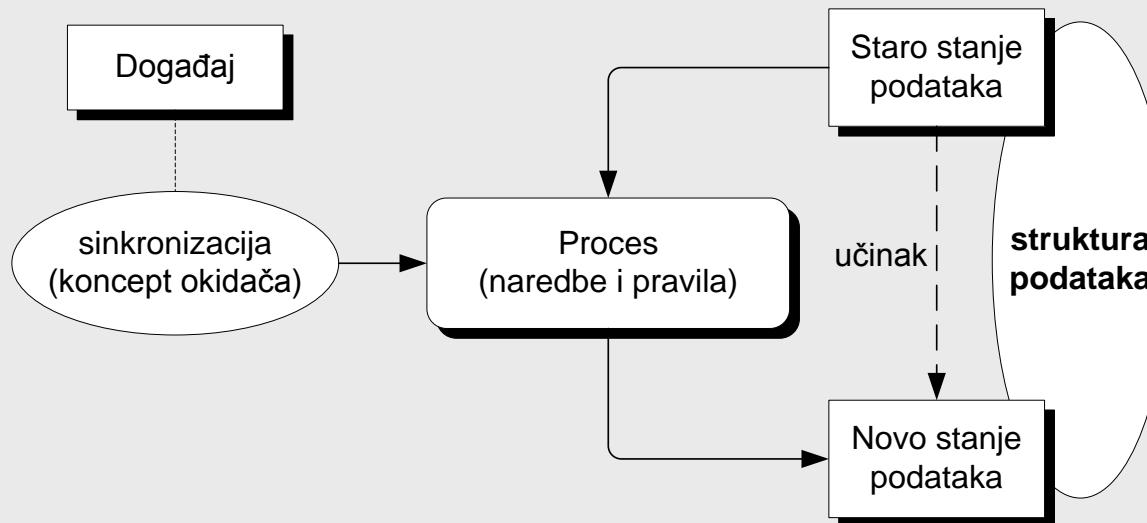
- fizički model (ugradbeni, implementacijski, tehnički)
 - kako je sustav fizički i tehnički ugrađen, te tko, gdje i kada nešto radi
- logički model (esencijalni, konceptualni, poslovni)
 - što je sustav, što radi, što su podaci
- operativni (budući fizički) sustav
 - što, tko i kada, ali ne i gdje
- opcionalno, može se razmatrati organizacijska razina
 - međuzavisnost razvoja organizacije i razvoja sustava



Modeliranje IS

□ Tehnike oblikovanja dijagramima

- podaci (što) – konceptualni (ERD=DEV), logički (relacijski)
- funkcije, procesi (kako) – funkcionalna dekompozicija, dijagram toka podataka
- događaji (kada) – dijagram (promjene) stanja
- objekti, programi, resursi, ... – UML, strukturne karte, mape resursa



□ Izrada modela koji odgovaraju dijelovima stvarnog sustava

- što kada sustav otprije ne postoji ?
- "surogat", preskakanje fizičkog oblikovanja

Ključne aktivnosti

- **Sistemska analiza, Analiza sustava (Systems analysis)**
 - proučavanje poslovanja s ciljem preporuke poboljšanja i specificiranja zahtjeva na rješenje
- **Sistemski dizajn, Dizajn sustava (Systems design)**
 - projektiranje sustava !
 - specifikacija ili konstrukcija računalom podržanog rješenja identificiranih poslovnih zahtjeva
- **SA + SD = informacijsko inženjerstvo**

Ključni sudionici

- **Dionici, interesni sudionici (stakeholders)**
- **Korisnik, naručitelj, klijent (user, customer, client) – osoba ili grupa**
 - Korisnik sustava – krajnji korisnik
 - Vlasnik sustava – naručitelj (stvarni vlasnik ili upravitelj organizacije)
- **Projektant, dizajner sustava (designer)**
 - prevodi poslovne zahtjeve i ograničenja u tehničko rješenje, modeliranjem
- **Graditelj sustava (builder, developer) - razvojnik**
 - izgrađuje, provjerava ispravnost, isporučuje

Ključni sudionici (nastavak)

□ Sistem analitičar (system analyst) [Whitten et. al, 2000]

- proučava problem i poslovne potrebe radi određivanja kako ljudi, podaci, procesi i informacijska tehnologija mogu unaprijediti poslovanje
- "moderni rješavatelj poslovnih problema"
- razumije i poslovanje i tehnologiju
- analitičar/programer, programer/analitičar

□ Poslovni analitičar (business analyst)

- analiza poslovanja, identifikacija koristi, oblikovanje procesa i procedura

[drugi analitičari, npr. infrastrukturni, upravljanja promjenama, ...]

□ Upravitelj projekta

- planiranje, upravljanje ekipom (analitičara, razvojnika, korisnika), nadzor, ...

□ Analitičar ?= konzultant

Životni ciklus razvoja sustava

Systems Development Life-Cycle (SDLC)

Životni ciklus razvoja

□ Općeniti proces razvoja sustava

- Ustanovljavanje problema
- Proučavanje i razumijevanje problema
- Identifikacija zahtjeva na rješenje ili očekivanja
- Procjena alternativnih rješenja i odabir "najboljeg" smjera
- Oblikovanje odabranog rješenja
- Ugradnja odabranog rješenja
- Vrednovanje rezultata. Ukoliko problem nije riješen povratak na korak 1 ili 2

□ SDLC = software/systems development life-cycle

- konzistentan i standardizirani način razvoja IS
- model razvojnog procesa - unaprijed propisan proces razvoja
- definira faze i zadatke (aktivnosti) koje treba obaviti tijekom razvoja
- ciklus osigurava “kontrolne točke” za praćenje napretka, procjenu postignutih rezultata i donošenje odluka o dalnjim koracima

Faze životnog ciklusa



Planiranje sustava

- Zašto graditi sustav ?

□ Inicijacija projekta

- Zahtijevanje sustava (system request), Inicialna strategija (initial strategy)
 - određivanje poslovnih ciljeva, problema i ideja njihovog rješavanja
 - snimka stanja, sažetak poslovnih potreba
- Studija izvedivosti (feasibility study)
 - analiza problemskog područja i određivanje (granica) projekata
 - procjena ključnih aspekata predloženog projekta
 - organizacija, tehnologija, financije, rokovi
- povjerenstvo (steering committee, approval committee) odobrava projekte

□ Upravljanje projektom

- po odobrenju projekta
- Izrada plana rada, kadroviranje projekta, upravljanje i nadzor projekta

→ Poslovni cilj i projekti, plan sustava / informatizacije (glavni projekt)

Faza analize

- Tko će koristiti sustav, što će sustav raditi te gdje i kada će biti korišten.

□ Strategija analize (analysis strategy)

- vodič kroz analizu postojećeg (as-is) i projektiranje novog (to-be) sustava

□ Analiza zahtjeva (requirements analysis)

- detaljna analiza postojećeg sustava (problema i poslovnih zahtjeva)
- modeliranje (postojećeg) sustava
- preciziranje dosega projekta i poslovnih zahtjeva

□ Specifikacija zahtjeva (requirements specification)

- modeliranje (budućeg) sustava
- definiranje zahtjeva na budući IS

→ Poslovni model sustava, prijedlog sustava

Oblikovanje sustava

- Kako će sustav funkcionirati

□ Strategija projekta (design strategy)

- graditi vlastitim snagama (insourcing), naručiti razvoj po mjeri (outsourcing) ili kupiti gotovo rješenje ?

□ Dizajn sustava (system design)

- modeliranje cjelokupnog sustava (pogled projektanta)
- odabir tehničke arhitekture sustava
- konceptualno modeliranje podataka
- opći dizajn sučelja

□ Detaljni dizajn (detailed design)

- razrada rješenja, izrada tehnološkog modela IS (pogled izvođača)
- dizajn sučelja, pohrane podataka i programa

→ Tehnička specifikacija sustava

- kolekcija specifikacija arhitekture, sučelja, pohrane podataka i programa

□ Po potrebi se revidira studija izvedivosti i plan projekta

Izrada sustava

- Ugradnja ili nabava oblikovanih rješenja

□ Izrada, implementacija (implementation), konstrukcija

- ugradnja baze podataka,
- kodiranje procesa (funkcija) novog IS

□ Testiranje (testing)

- provjera komponenti

□ Integracija i provjera sustava (system integration, evaluation)

- udruživanje dijelova i provjera cjeline,
- dokazivanje da sustav radi (da je ispravno napravljen)
- te da radi što je zahtijevano (da je napravljen pravi sustav)

→ Funkcionalni sustav i tehnoški opis sustava

Primjena sustava

□ Uvođenje u primjenu i održavanje

- Uvođenje u primjenu, primjena (operation, production)
 - prijenos (konverzija) podataka, konverzija načina rada
- Održavanje (maintenance)
 - izmjene radi poboljšanja, dorade ili prilagodbe
- Potpora (support)
 - dobavljača opreme, tehničkog osoblja korisnicima

→ Informacijski sustav u primjeni, plan održavanja

□ Pregled (review)

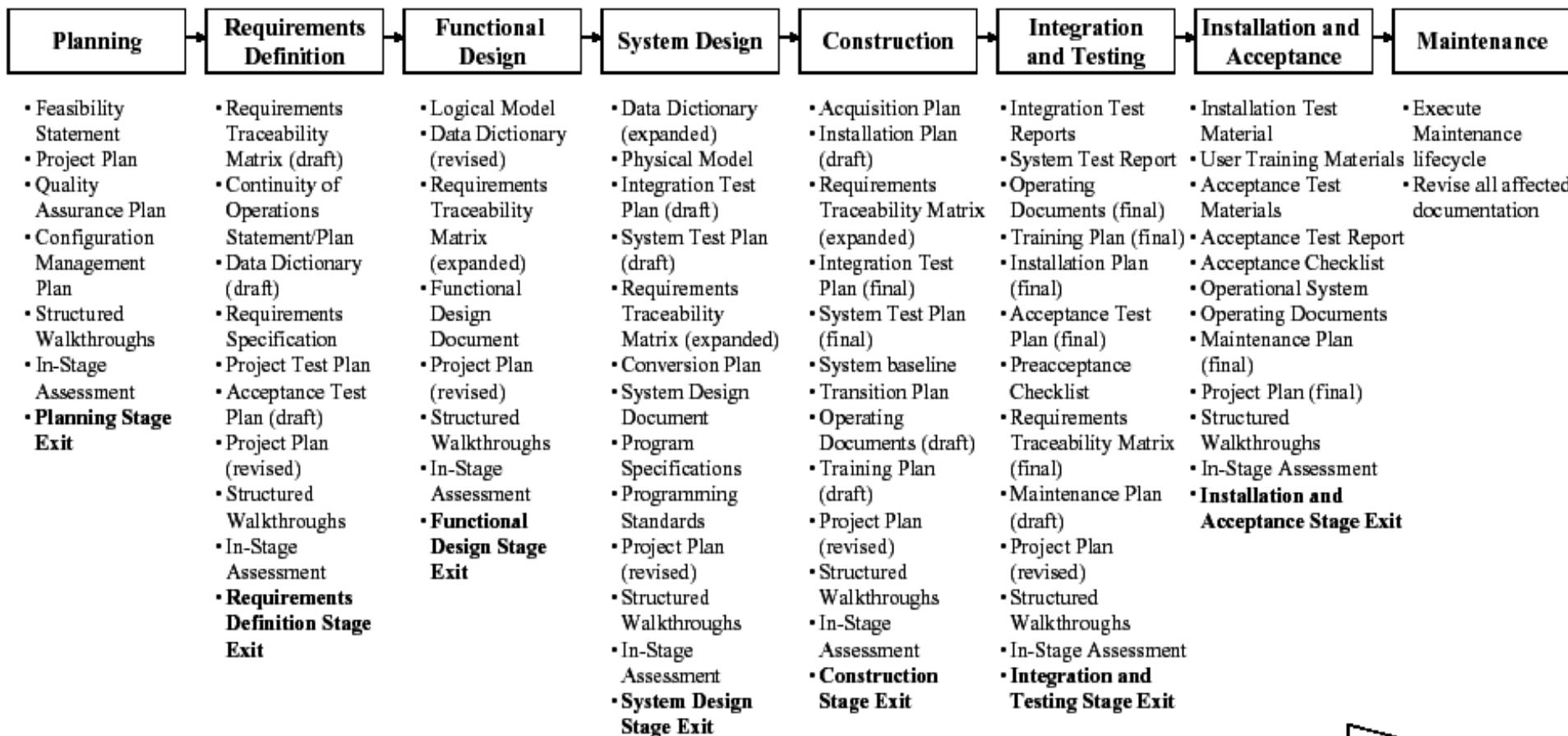
- revizija, preispitivanje čitavog sustava kada su potrebne veće izmjene uslijed promjena u poslovanju ili promjena poslovnih ciljeva

→ Novi projekti, novi razvojni ciklus

□ Navedene su tipične faze životnog ciklusa, bez implikacije da se radi o diskretnim i/ili slijednim aktivnostima!

Primjer: IS LC Stages and Deliverables, DOE Software Development Methodology

Enterprise Information Architecture



Project Management

Configuration Management

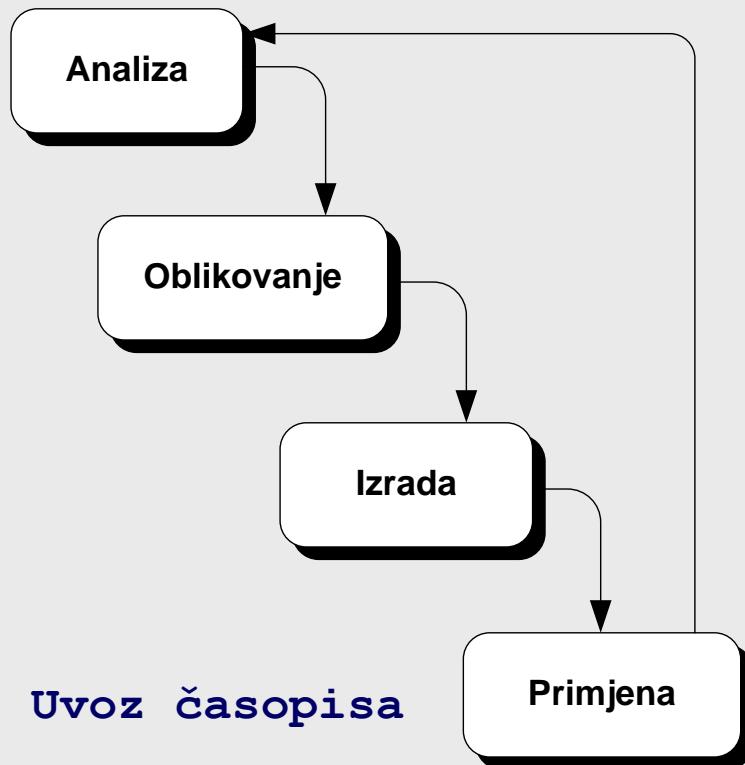
Quality Assurance/Quality Control

Modeli razvoja

Vodopadni (waterfall) model

□ Klasični vodopadni model

- slijedno napredovanje iz faze u fazu
- nisu dozvoljene naknadne promjene rezultata prethodnih faza
- kvaliteta važnija od troškova i rokova



□ Uvoz časopisa

□ Primjenjivost

- dobro definirano i stabilno okruženje, uhodane ručne obrade ili računalski sustav koji treba unaprijediti
- veliki projekti (investicije)
- zaposlenici neiskusni ili posjeduju ograničeno tehničko znanje

□ Nedostaci

- uvođenje prema gore (bottom up): moduli, podsustavi, sustav
- sustav nepotrebljiv dok ne bude potpuno dovršen
- predodžba o proizvodu na temelju pisane specifikacije
- korisnici prekasno uoče nedostatke
- problem u slučaju pogrešaka ili novih/promijenjenih zahtjeva

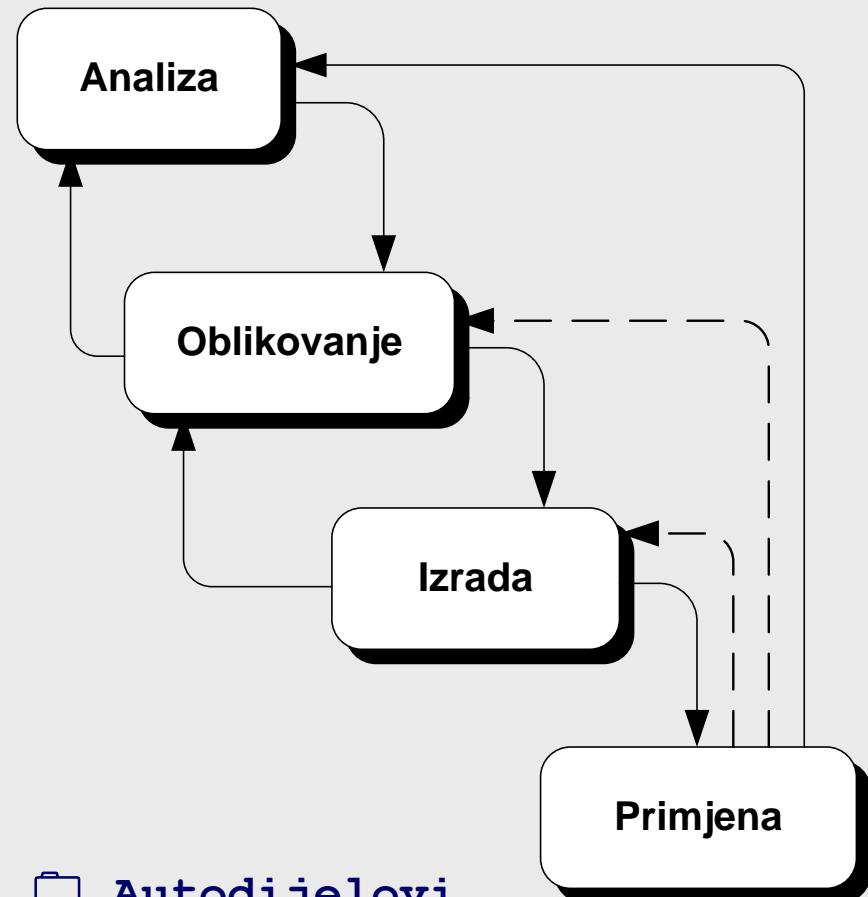
Pseudostruktturni i radikalni vodopadni model

□ Pseudostruktturni model

- povratna veza i mogućnost promjene prethodnih rezultata
- strukturirano programiranje
- uvođenje prema dolje: moduli na višim, pa na nižim razinama

□ Struktturni (radikalni)

- istovremenost različitih faza
- 4GL i generatori aplikacija
- prikladan kada se unaprijed ne zna konačni izgled sustava
- u konačnici mora nastati (papirnati) model sustava

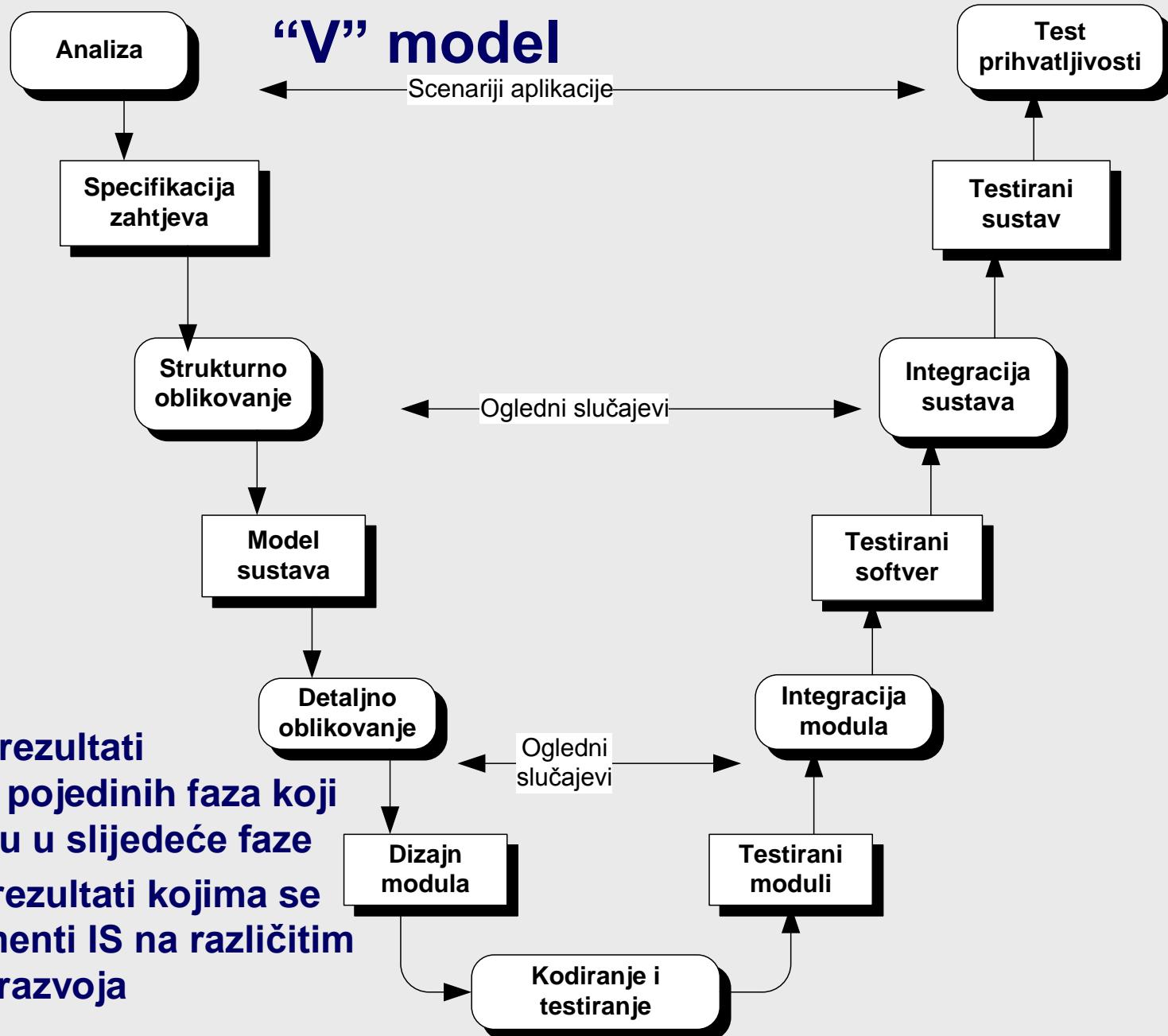


□ Autodijelovi

Validacija

Verifikacija

- definiraju se rezultati ("proizvodi") pojedinih faza koji se prosljeđuju u slijedeće faze**
- određuju se rezultati kojima se testiraju elementi IS na različitim stupnjevima razvoja**



Paralelni razvoj po podprojektima

podprojekti koji se oblikuju i izrađuju paralelno

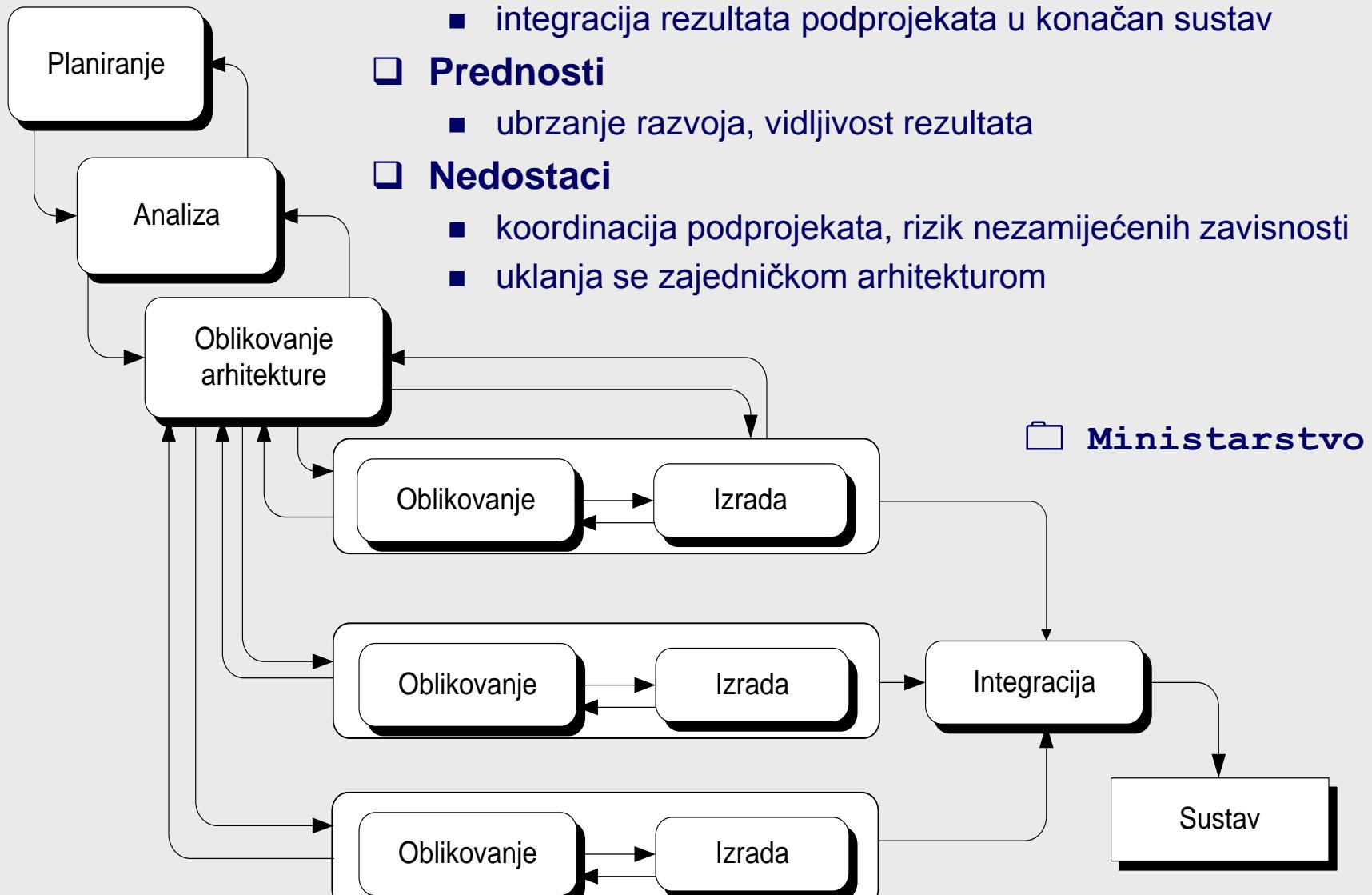
- integracija rezultata podprojekata u konačan sustav

Prednosti

- ubrzanje razvoja, vidljivost rezultata

Nedostaci

- koordinacija podprojekata, rizik nezamijećenih zavisnosti
- uklanja se zajedničkom arhitekturom



Prototipski model razvoja

- **Prototip = model koji se radi da bi se isprobale neke mogućnosti**
- **Vrste prototipova**
 - Model oponašanja (mock-up, model u naravnoj veličini)
 - jednoekranski ili višeekranski model kojim se prikazuje kako će izgledati dio sustava (npr. sučelje)
 - Istraživački model (research model)
 - istraživanje dijelova sustava kako bi se provjerile neke ključne postavke (npr. provjera performansi određenog modula)
 - Ugradbeni model (implementation model)
 - traženje načina na koje se sustav može izraditi (npr. koji sustav za upravljanje BP, programski jezik, računala...)

Brzo prototipiranje (rapid prototyping)

□ Evolucijsko prototipiranje (sinonim)

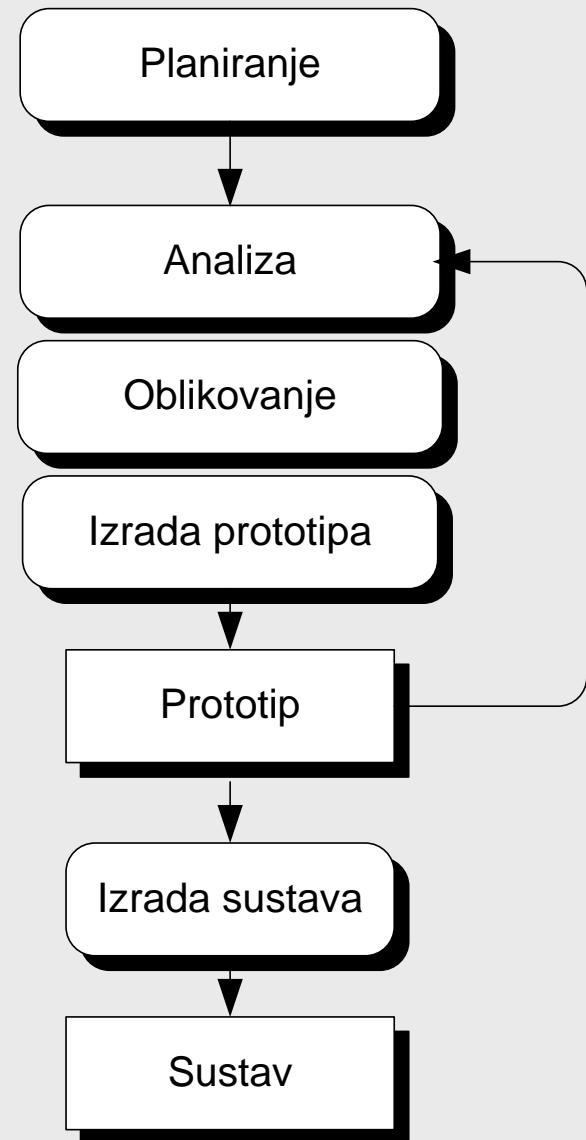
- funkcionalni prototip
- povratna informacija korisnika
- inkrementalna dorada (stepwise refinement)
- na kraju samostalna izrada (bez korisnika)
- mali (*one-man*) projekti s promjenjivim zahtjevima

□ Prednosti

- pokretljivost, prilagodljivost ☺
- bolje određivanje zahtjeva
- vidljivost rezultata

□ Nedostaci

- "zaboravljanje" da prototip nije pravi sustav
- "vječni razvoj"
- izostanak specifikacije, pa druge dokumentacije
- nemogućnost ispravne procjene i planiranja resursa



Ograničeno prototipiranje (constrained prototyping)

□ Nefunkcionalni prototip

- samo prikaz izgleda

□ Prototipiranje - sredstvo određivanja zahtjeva

- smanjuje rizik neispunjениh očekivanja
- po određivanju zahtjeva prototip se odbacuje (throwaway prototyping)
- slijedi faza oblikovanja

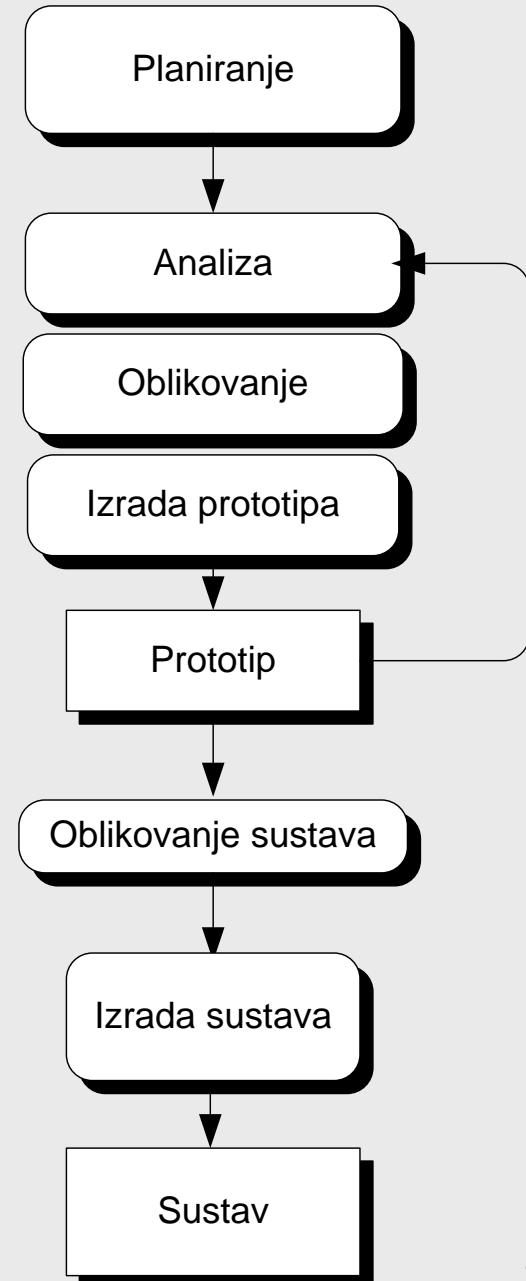
□ Prednosti

- postoji dokumentacija
- moguće je planiranje

□ Nedostatak

- "bacanje" posla

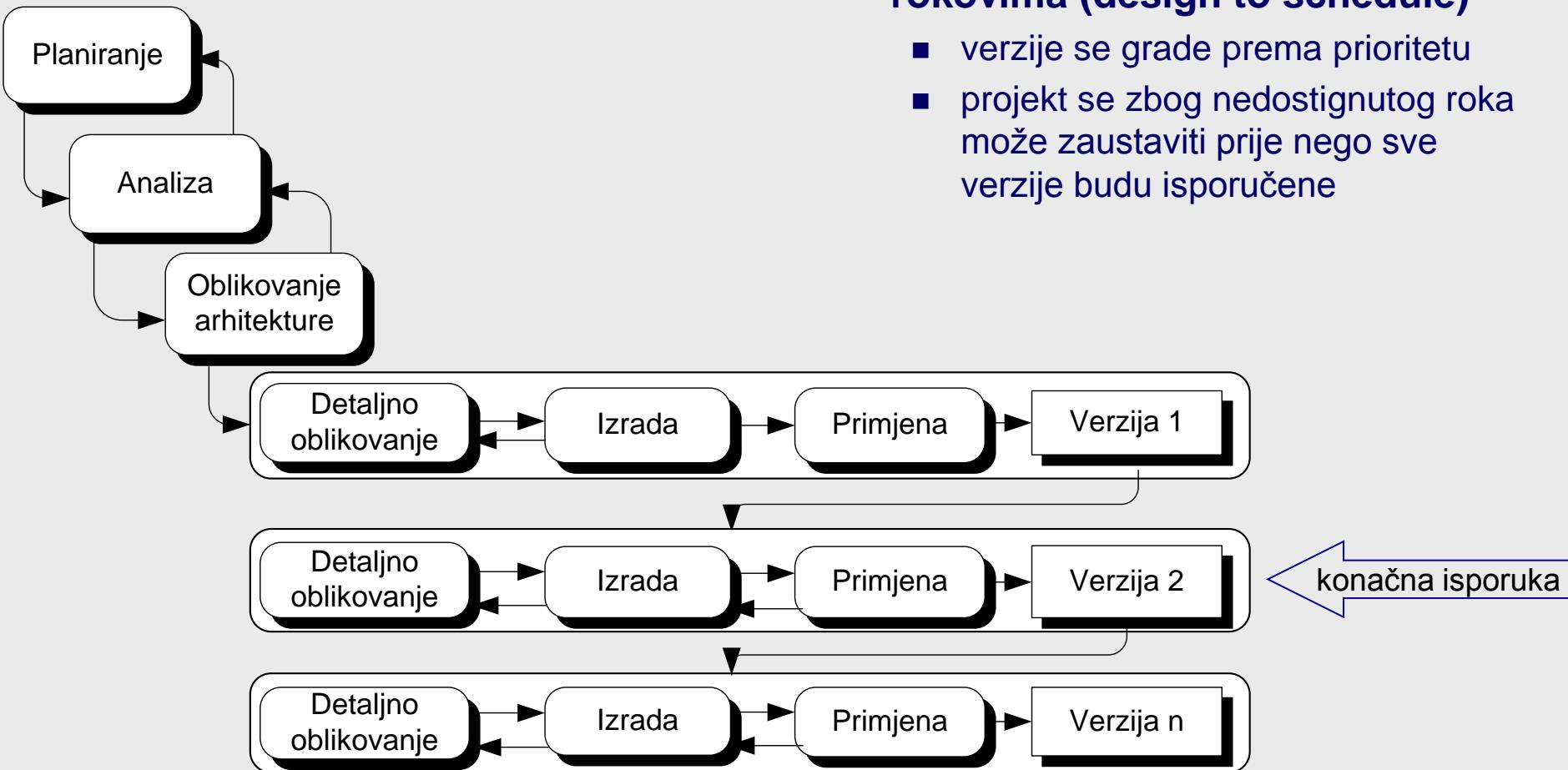
❑ Izdavačka kuća



Fazni razvoj, model postupne isporuke

- **Fazni razvoj, model postupne isporuke (Staged Delivery), evolucijski razvoj, evolucijsko prototipiranje, inkrementalni razvoj**
 - predstavnik brzog razvoja aplikacija, RAD (Rapid Application Development)
 - slijed verzija koje se isporučuju i generiraju nove zahtjeve
- **Prednosti**
 - isporuka dijelova prije konačnog završetka projekta
 - brža uporaba - ranije vidljiva korist sustava
 - korisnici brže identificiraju dodatne zahtjeve
- **Nedostaci**
 - korisnici upotrebljavaju sustav koji je “namjerno” nedovršen
 - potreba za pažljivim planiranjem
 - raspodjela resursa, ovisnosti između pojedinih faza isporuke
 - ključne i nužne mogućnosti ugraditi prve!

Evolucijski model, model postupne isporuke

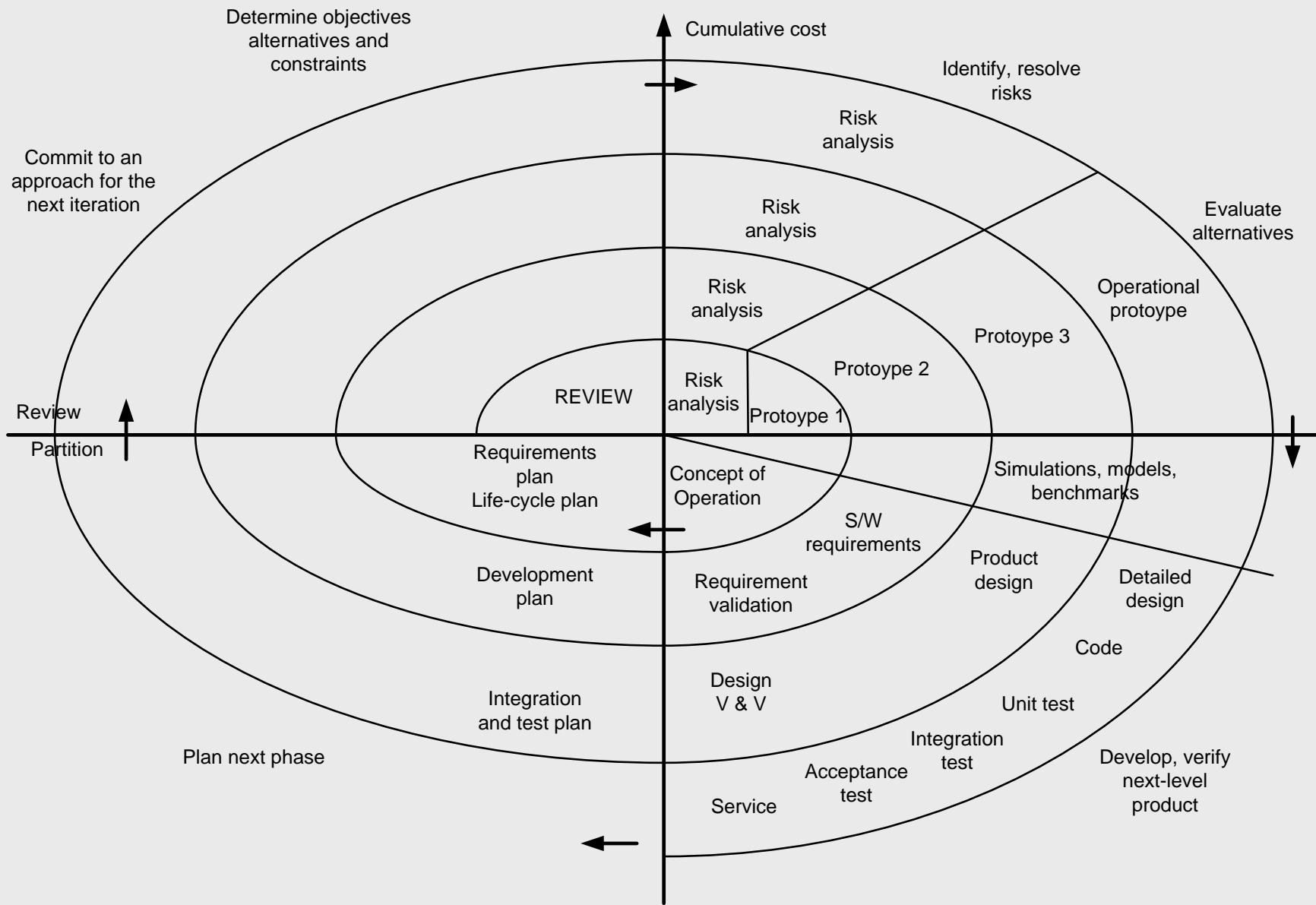


□ Evidencija biljnih vrsta

Spiralni model

- **Iterativno se analizira rizik i planira naredna iteracija od 6 koraka:**
 - Određivanje ciljeva, alternativnih rješenja i ograničenja,
 - Određivanje i pronalaženje rješenja za moguće rizike,
 - Procjenjivanje alternativnih rješenja,
 - Razvoj izlaznih proizvoda iteracije i potvrđivanje njihove točnosti,
 - Planiranje sljedeće iteracije,
 - Pokretanje sljedeće iteracije (ako se odluči krenuti u novu iteraciju).
- **Svaka iteracija dovodi projekt bliže kraju.**
 - U slučaju da je rizik prevelik, projekt se obustavlja ili prekida.
 - Radijalna koordinata predstavlja kumulativni trošak
 - Zadnja iteracija predstavlja klasični ciklus
- **Primjena**
 - veliki (skupi) sustavi ili je analiza rizika relativno mali trošak
 - najčešće na interne projekte (naručitelj i izvođač iz iste organizacije)
- **Prednosti**
 - praćenje napretka, sigurnost
- **Nedostaci**
 - složenost, trošak

Spiralni model



Osnovni objektno orijentirani model razvoja

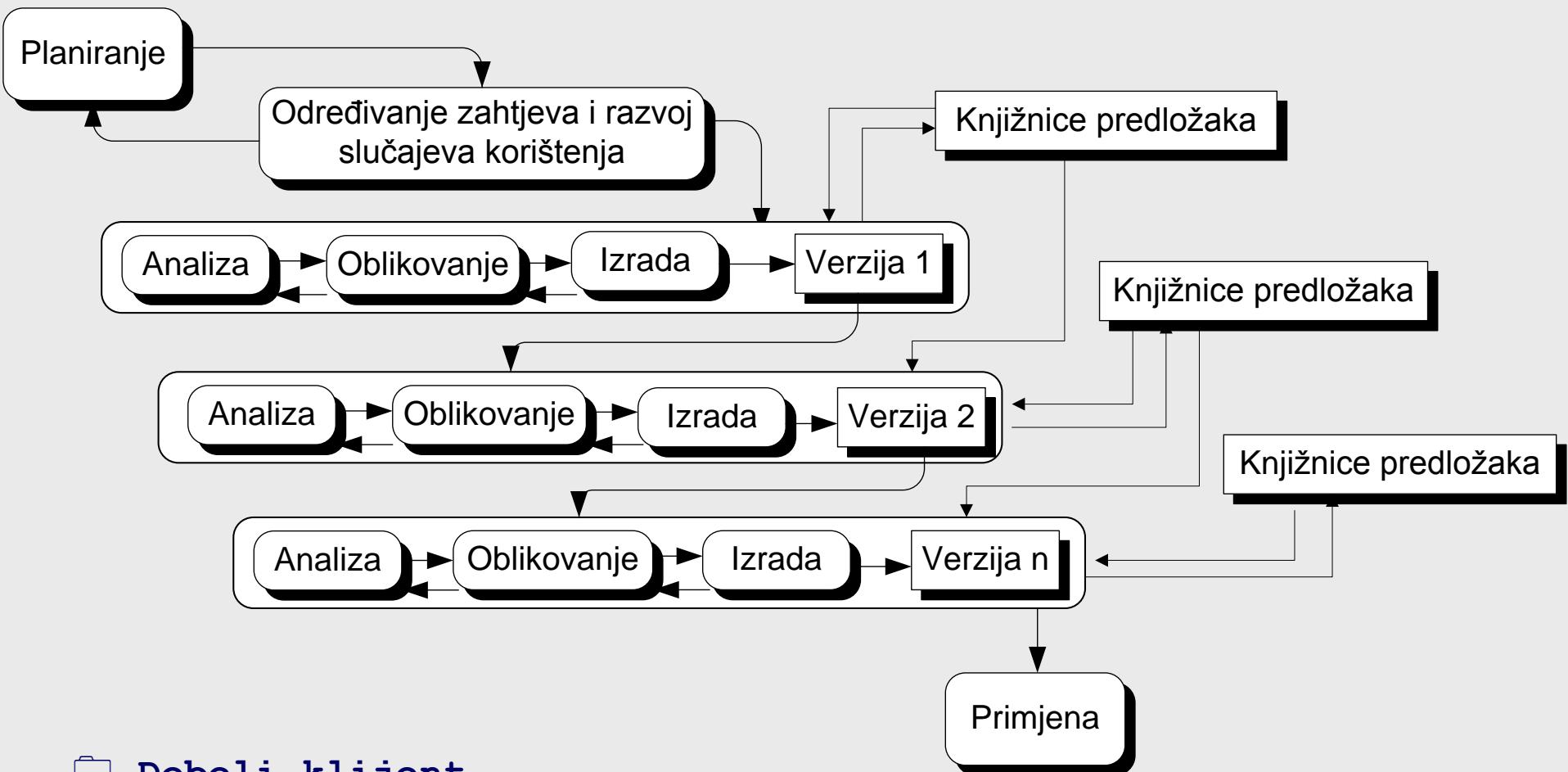
□ Minimalist Object-Oriented Systems Analysis & Design (MOOSAD)

- generički pristup OOSAD, temeljem ujedinjenog procesa (Unified Process) i ekstremnog programiranja (eXtreme Programming)
- po uzoru na fazni razvoj, ključna razlika je u pristupu dekompoziciji
 - tradicionalno, dekompozicija je usmjerena procesima (process-centric) ili podacima (data-centric), OO dekompozicija na objekte (podaci i procesi) + UML

□ U osnovi inkrementalni model razvoja s verzijama-gradnjama

- gradnja (build) – inačica
- gradnje se temelje na slučajevima korištenja po prioritetima
- (svaka) gradnja predstavlja inkrementalni napredak završetku
- gradnja obuhvaća analizu, oblikovanje i izradu
- gradnja isporučuje funkcionalni sustav i predloške rješenja za knjižnice
- gradnja vraća povratnu informaciju o zahtjevima
- gradnje omogućuju upravljanje čvrstim rokovima (timeboxing)
 - u intervalima do dva tjedna ili mjeseca ovisno o veličini i složenosti projekta

MOOSAD pristup



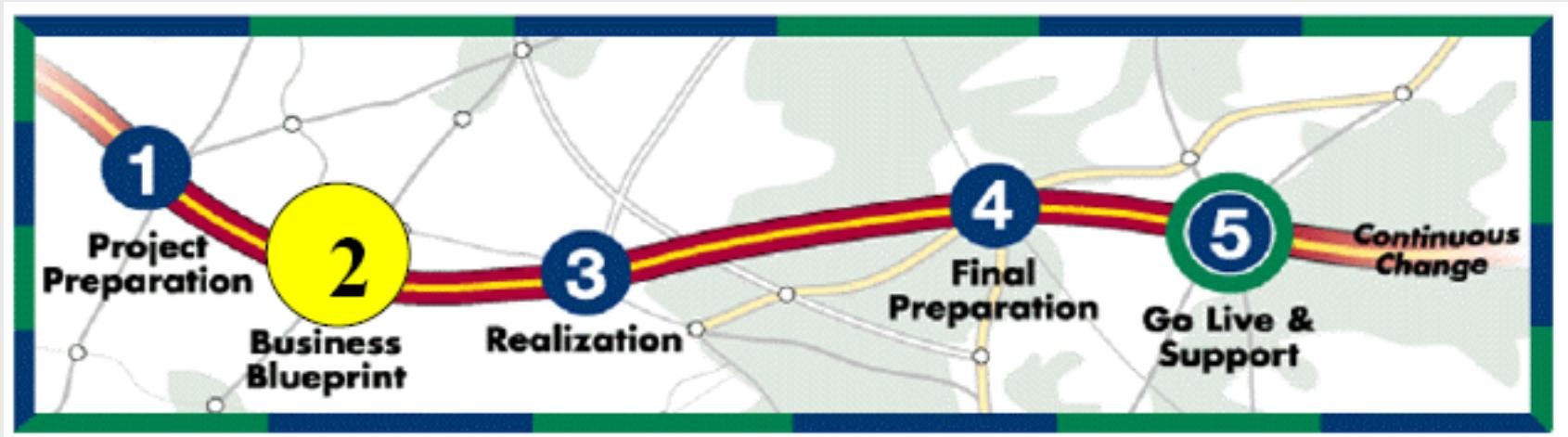
Debeli klijent

Višeslojni poslovni sustav

Nabava gotovih proizvoda

- **Alternativa razvoju je kupnja postojećeg proizvoda.**
 - Commercial-Off-The-Shelf (COTS), Shrink-wrap – općenito
 - Enterprise Resource Planning (ERP) – sustav za podršku poslovanju
- **Proizvodi rijetko imaju svu potrebnu funkcionalnost, ali imaju i prednosti**
 - Proizvod je vidljiv i može se ocijeniti (!?)
 - Proizvod je "odmah" dostupan po nabavi (!?)
 - U slučaju ERP, treba obaviti prilagodbu koja može potrajati mjesecima
 - SAP: 20k tablica i zaslona, ABAP (Advanced Business Application Programming)
- **Životni ciklus ERP proizvoda**
 - Analiza (Project Planning, Project Preparation)
 - Specifikacija zahtjeva (Business Blueprint)
 - Implementacija i testiranje (Realization)
 - Kontrola kvalitete i poduka (Final Preparation)
 - Uvođenje u primjenu i održavanje (Going Live i podrška)

Životni ciklus ERP proizvoda





Diskusija, sport i glazba

Statistika razvoja softvera (Mynatt)

1. Tipični projekt razvoja softvera traje
a) 1 - 5 b) 6 - 11 c) 12 - 23 d) 24 - 48 mjeseca.

2. Za sustav srednje veličine, proizvede se tokom čitavog ciklusa
a) manje od 10 b) 10 - 20 c) 21 - 30 d) više od 30 redaka/osoba/dan

3. Približan broj pogrešaka koje se pronađu u svakih 1000 programskih redaka tijekom razvoja jest:
a) manje od 30 b) 30 - 40 c) 40 - 50 d) 50 - 60

4. Približan broj pogrešaka koje se pronađu u svakih 1000 programskih redaka u isporučenom sustavu jest:
a) manje od 4 b) 4 - 8 c) 8 - 12 d) više od 12

5. Koji je približan postotak programskih sustava čiji je razvoj započeo, a da budu i konačno dovršeni?
a) 90 - 100% b) 80 - 90% c) 70 - 80% d) 60 - 70%

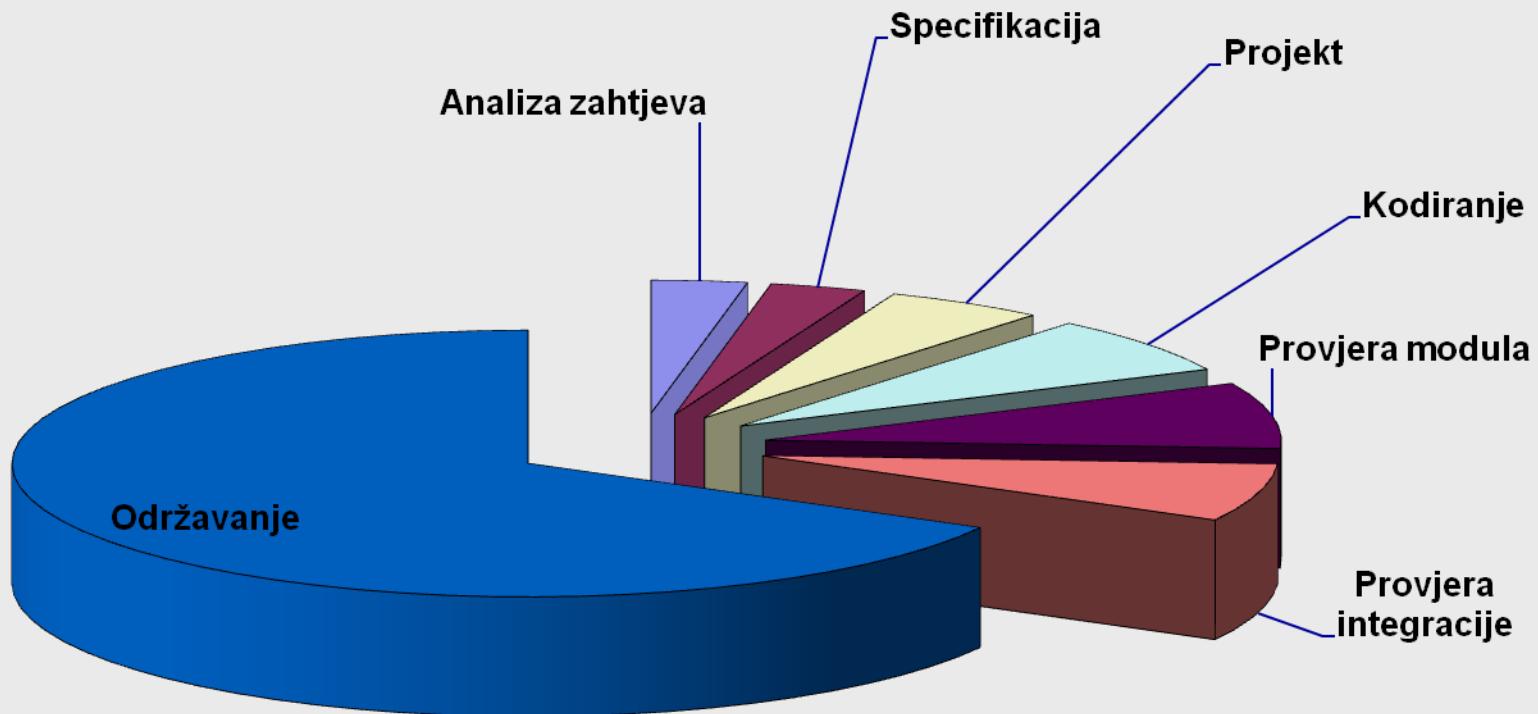
Statistika u svezi razvoja softvera (Mynatt)

- 6. Trošak posjedovanja i održavanja softvera je tipično**
 - a) za polovicu b) jednako c) dvostruko

puta veći nego što je bio trošak razvoj softvera.
- 7. Većina pogrešaka koje korisnici pronađu u softveru su rezultat**
 - a) programerove pogreške
 - b) problema u oblikovanju zadatka ili njegova razumijevanja
 - c) administrativnih pogrešaka
 - d) pogrešaka u projektu
- 8. Koliki je udio programiranja u razvoju softvera?**
 - a) 99% b) 70% c) 50% d) 20% e) 10%
- 9. Sustav se smatra velikim ako sadrži barem**
 - a) 2.000 b) 10.000 c) 100.000 d) 1.000.000

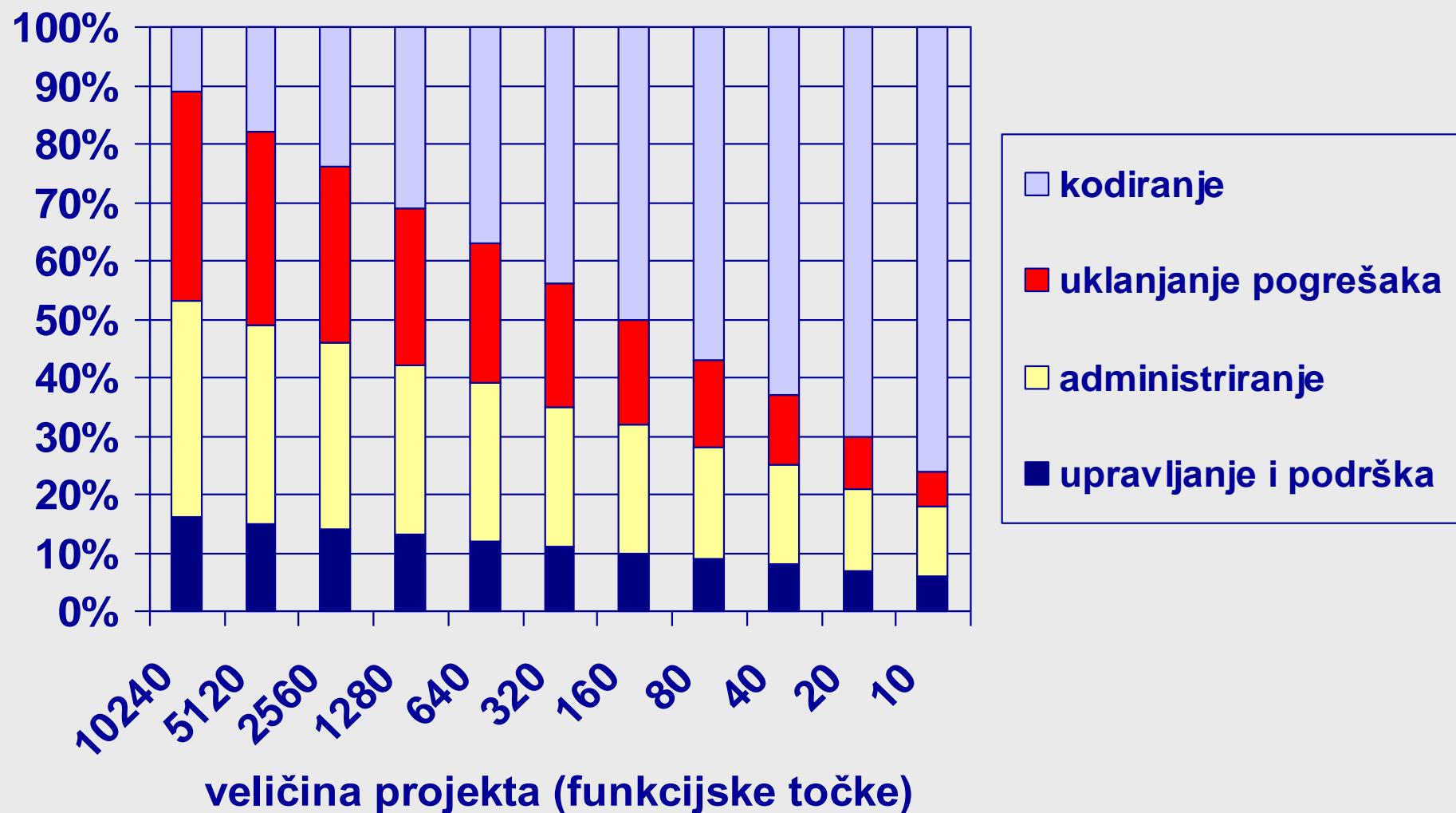
redaka izvornog koda

Razdioba troškova u životnom ciklusu softvera



- Analiza:** 3%
- Specifikacija:** 3%
- Projekt:** 5%
- Kodiranje:** 7%
- Provjera modula:** 8%
- Provjera integracije:** 7%
- Održavanje:** 67%

Što se vidi ?



Buzzwords

- ❑ Refaktoriranje je ...
- ❑ TDD ?
- ❑ DDD vs. MDD ?
- ❑ Component Diagram vs. Package Diagram – kojeg kada ?
- ❑ Sequence Diagram ili Collaboration Diagram ili oba ?
- ❑ DSL ?

Reference

□ Resursi

- Association for Information Systems, <http://home.aisnet.org/>
- Kontrolne liste i predlošci projektne dokumentacije <http://www.construx.com>
- Prilagodljivi razvojni proces (Adaptable Process Model) <http://www.rspa.com>

□ Literatura

- Dennis A., Wixom B.H., Tegarden D. Systems Analysis and Design with UML Version 2.0: An Object-Oriented Approach, 2nd Edition, Wiley, 2005
- Shelly G.B., Rosenblatt, H.J. Systems Analysis and Design, Course Technology, 2011.

Strategija organizacije

Identifikacija i selekcija projekata

Pokretanje i planiranje projekata

2012/13.02

Strateško planiranje poslovanja (1)

- Izrada poslovne strategije, dugoročno planiranje resursa i akcija
- Uprava definira viziju i misiju organizacije, tj. strateške ciljeve
 - **misija** – svrha (postojanja) organizacije
 - postojeće stanje i ključni procesi te poželjne performanse
 - **vizija** – namjeravano, očekivano stanje u daljoj budućnosti
 - određuje kriterije za donošenje odluka
- Slijede poslovni ciljevi, procesi i zadaci za ispunjenje
 - **što** se želi postići (vrijednosti): prepoznatljivost, kvaliteta, prihodi, ...
 - **kako** to postići: promjenom organizacije, poboljšanjem administracije, ...

Strateško planiranje poslovanja (2)

□ Čimbenici koji utječu na postavljanje ciljeva

- ograničenja (organizacijska, finansijska, zakonska, ...)
- potrebe i želje uprave, poslovodstva, zaposlenika (ugled, utjecaj, ...)

□ Vremenski okviri, razdoblje [Awad, 1985]:

- kratkoročno, obično manje od 2 godine
- srednjeročno, 2-5 godina
- dugoročno, više od 5 godina

□ Primjer:

- <http://www.fer.unizg.hr/strategija>

Planiranje informacijskog sustava

□ Traženje odgovora na pitanja:

- Čime se organizacija bavi (grana, proizvodi, tržište, konkurencija)?
- Koji su problemi, zadaće i ciljevi poslovnog sustava?
- Koja je željena uloga IS u postizanju postavljenih ciljeva?
- Postojeće aplikacije, aplikacije koje se razvijaju, potrebne aplikacije?
- Koji su raspoloživi resursi (osoblje, tehnička sredstva, tehnologija, ...)?

□ Razlozi zbog kojih treba planirati IS

- otoci informatizacije
- redundancija, nepotpunost, nepovezanost
- tehnološka različitost

Strateško planiranje IS

□ Tradicionalno planiranje IS

- provodi se odvojeno od poslovnog planiranja ili
- provodi se kao reakcija na promjene u poslovnoj politici

□ Strateško planiranje IS

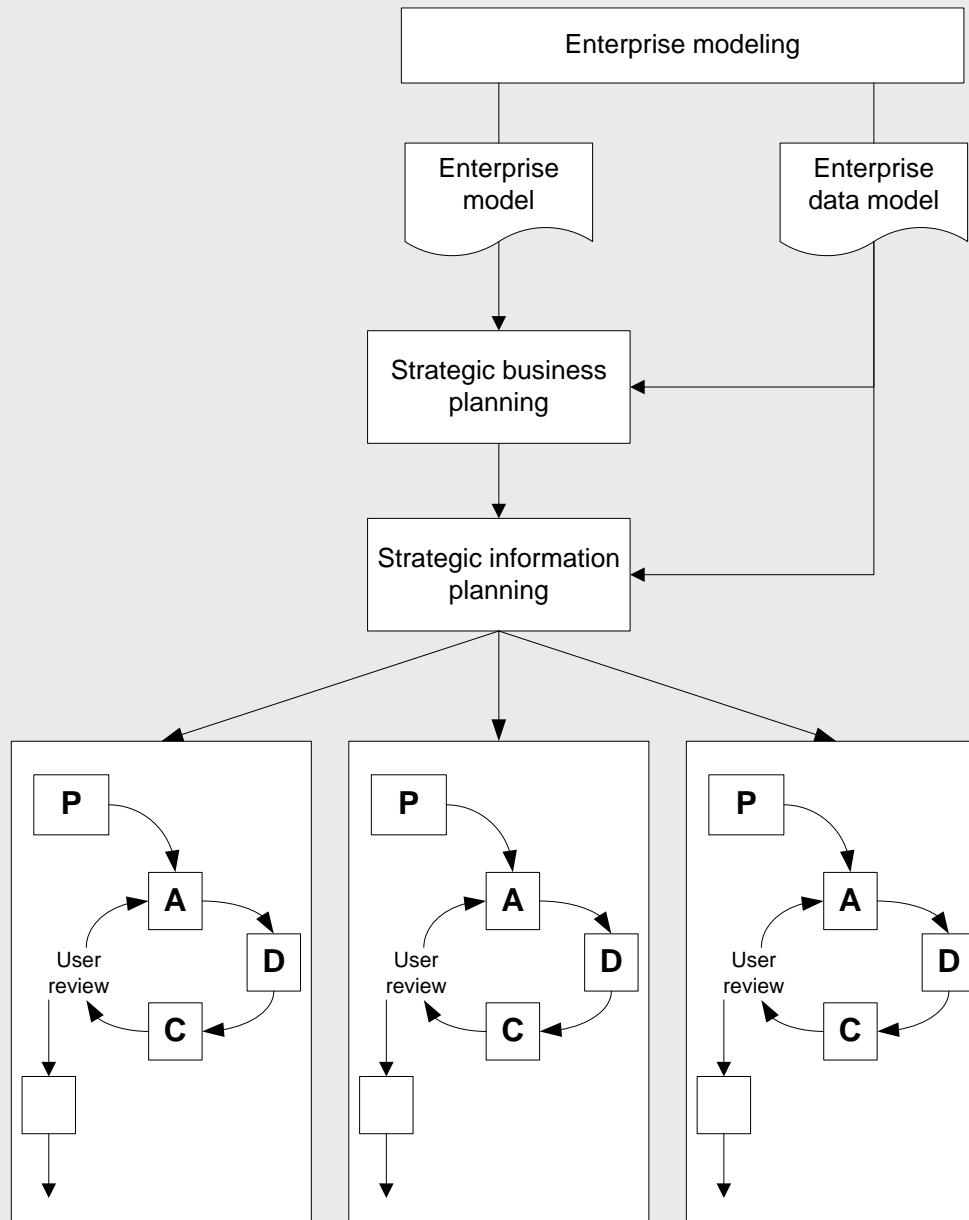
- planiranje IS sukladno strategiji razvoja organizacije
 - informatizacija kao potpora promjeni organizacije i poslovnih procesa
- *istraživanje poslovnog sustava s ciljem definiranja općeg (sveobuhvatnog) plana i arhitekture IS čiji razvoj slijedi*

□ U praksi ...

- organizacija se "dovodi u red" tijekom informatizacije i s pomoću nje
- analiza sustava – evidencija problema i slabosti poslovnih procesa
- dizajn sustava – predlaganje ili nametanje rješenja (uf)

□ Primjer: (pilot) projekt informatizacije visokih učilišta

Okosnica plana razvoja sustava



Primjer strategije: Društvo

□ Poslovna strategija – potpora IT ne može se kupiti

- Ključne vrijednosti: tradicija, ugled, povjerenje, iskustvo, ..., IS
- Orijentacija ka klijentu (CRM) – personalizacija, ..., mobilna prodaja
- Razvoj poslovne inteligencije (BI) – skrivene zakonitosti, ..., konkurentnost, ..
- Usvajanje regulativa (Basel II, Solvency II)

□ Informacijska strategija

- Sigurnost i zaštita IS, ..., kontinuitet poslovanja
- Integriranje aplikacija (SOA, ..., ESB)
- Promjena arhitekture (dvoslojna u višeslojne)
- ...

□ Upravljanje IS – reorganizacija

- Ured za upravljanje projektima
- Centar kompetencije za poslovnu inteligenciju
- Služba za poslovne procese
- ...

Primjer strategije: Poduzeće

□ Prosinac 2005.

- Tridesetak aplikacija, informacijski otoci
- Različite tehnologije, zastarjela tehnologija
- Raspršeno osoblje, zastarjela podjela posla

□ Strategija

- Smjernice razvoja i organizacije informatičke podrške
 - sistematizacija radnih mesta u informatici i reorganizacija službe
 - poduka informatičkih djelatnika
- Smjernice razvoja i arhitekture računalnog sustava
 - integracija sustava nad zajedničkim modelom podataka
 - razvoj aplikacija
- Smjernice upravljanja IS-om
- Smjernice upravljanja sigurnosti IS-a

□ Taktički plan provedbe projekta izgradnje IS

Primjer strategije: Poduzeće (2)

□ Plan za razdoblje 2006-2009.

ID	Task Name	Duration	Start	Finish	5	2	H2	H
1	Održavanje aplikacija	900 days	Mon 21.11.05	Fri 15.09.09				
2	Faza pripreme	274 days	Mon 16.1.06	Thu 1.2.07				
51	Vladanje informacijskim sustavom	188 days	Mon 13.2.06	Wed 1.11.06				
126	Izgradnja imeničke infrastrukture	53 days	Wed 5.4.06	Fri 16.6.06				
49	Sistemski poslovi	800 days	Mon 17.4.06	Fri 8.5.09				
94	Osiguranje IT sustava na razini cjelokupne infrastrukture	17 days	Mon 19.6.06	Tue 11.7.06				
108	Izgradnja sigurnosnih politika/procedura/uputa uskladjeni	40 days	Wed 12.7.06	Tue 5.9.06				
146	Implementacija sustava za upravljanje IT infrastrukturom	34 days	Wed 6.9.06	Mon 23.10.06				
16	Aplikacije osnovne djelatnosti	334 days	Fri 2.2.07	Wed 14.5.08				
28	Poslovne i računovodstvene aplikacije	476 days	Fri 2.2.07	Fri 28.11.08				
50	Skladište podataka	6 mons	Mon 1.12.08	Fri 15.5.09				

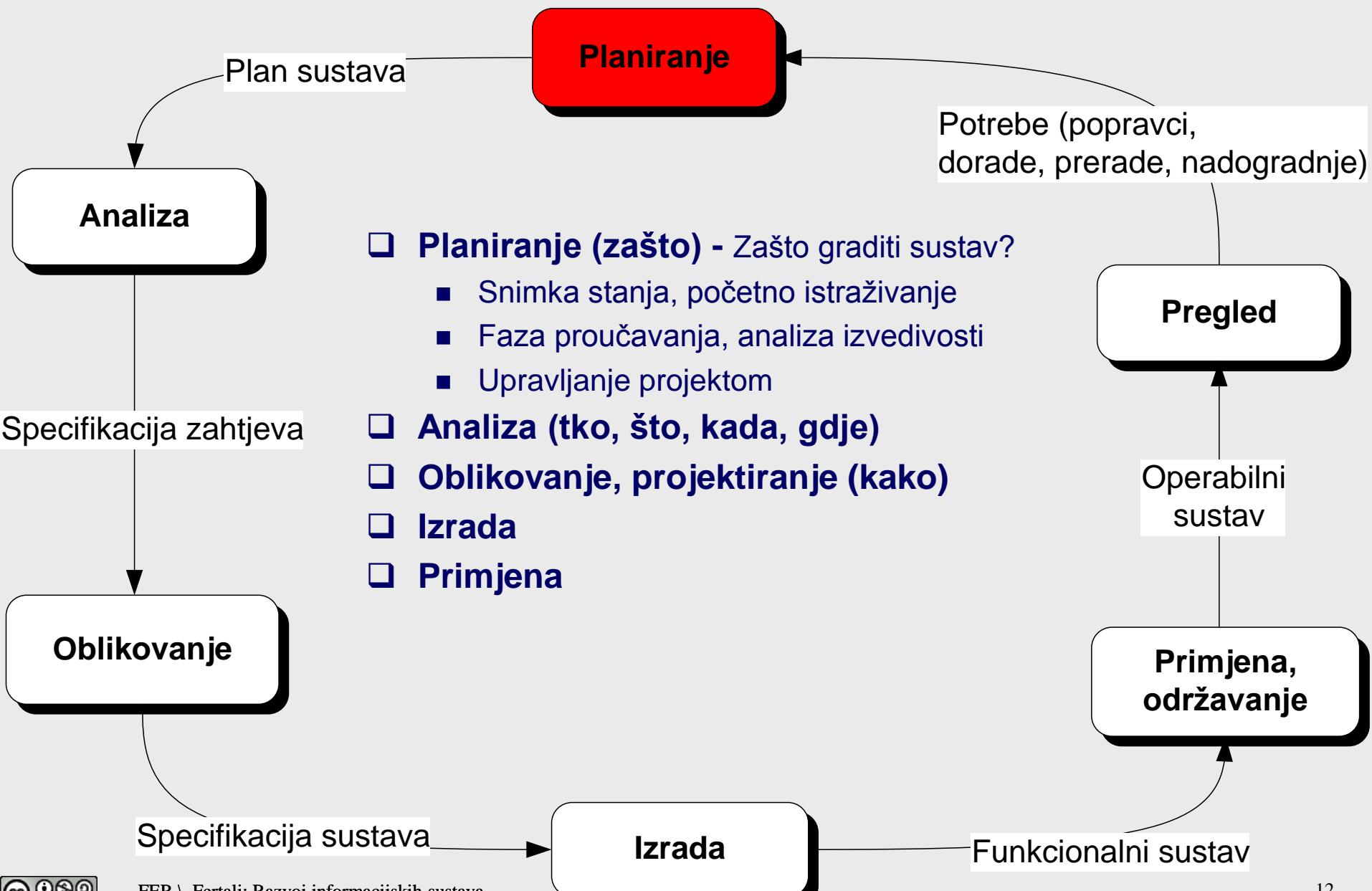
Primjer strategije: Poduzeće (3)

□ Ažurirani plan pripreme 2009.

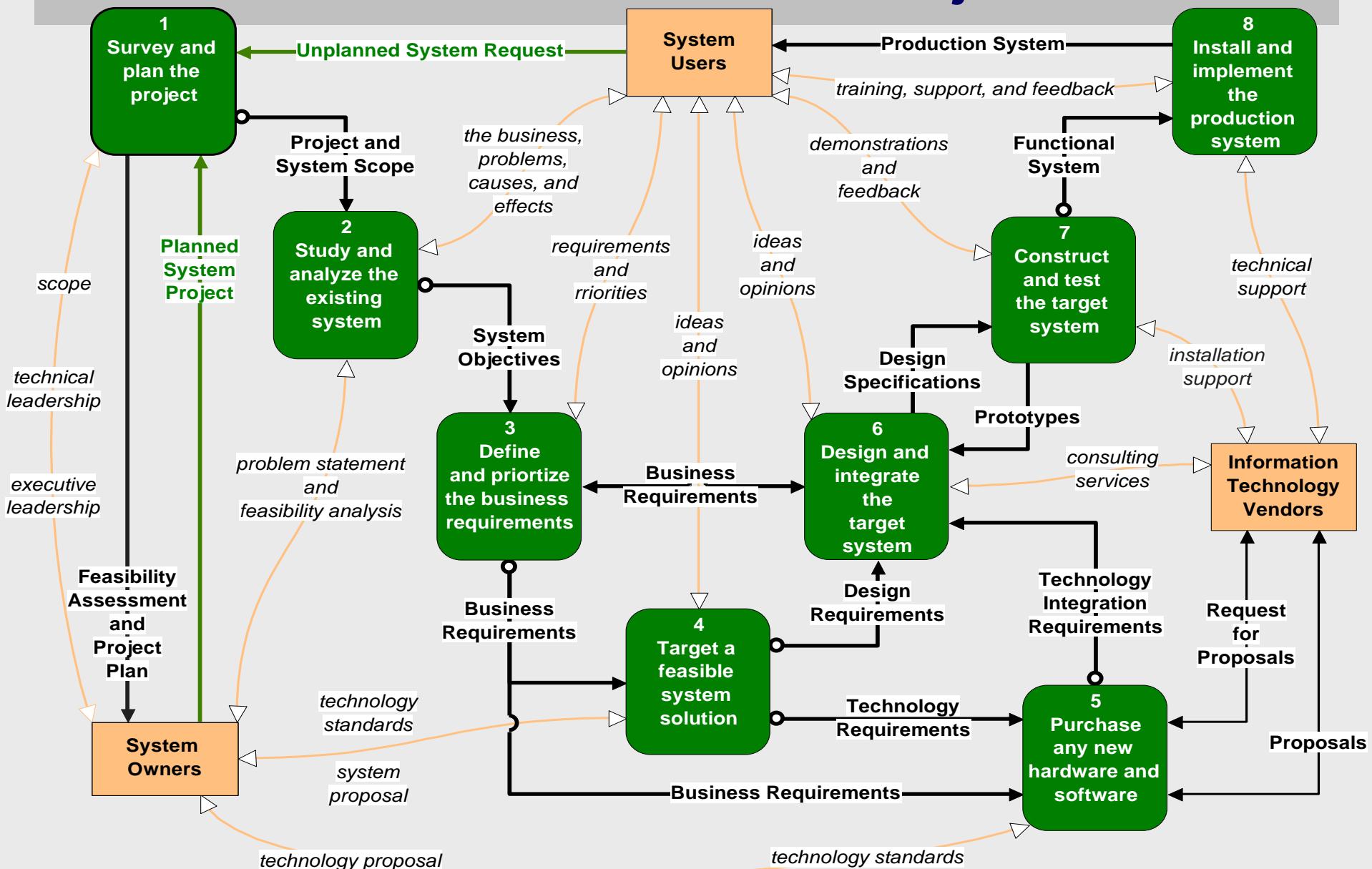
ID	Task Name	Duration	Start	Finish	Qtr 1, Jan
1	Faza pripreme	80 days	Mon 26.1.09	Fri 22.5.09	
2	Ustavljanje infrastrukture projekta (TFS, SQL)	5 days	Mon 26.1.09	Fri 30.1.09	
3	Školovanje za modeliranje podataka	5 days	Mon 2.2.09	Fri 6.2.09	
4	Školovanje za SQL	5 days	Mon 16.2.09	Fri 20.2.09	
5	Školovanje za razvojno okruženje	15 days	Mon 2.3.09	Fri 20.3.09	
6	Formiranje ekipa	30 days	Mon 23.3.09	Fri 8.5.09	
7	Metodologija razvoja	30 days	Mon 26.1.09	Fri 6.3.09	
8	Razvoj programske okosnice	60 days	Mon 26.1.09	Fri 24.4.09	
9	Poduka primjene metodologije i okosnice	10 days	Mon 11.5.09	Fri 22.5.09	

Odabir i pokretanje projekata

Planiranje i pokretanje projekta



Kontekst informatizacije



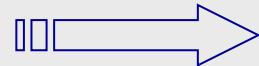
Identifikacija projekta

- **Sponzor projekta = osoba ili grupa koja ustanovi poslovne potrebe**
 - najčešće netko iz poslovanja (Financije, Studentska, Proizvodnja)
- **Pokretači promjena**
 - korisnici - nezadovoljstvo aplikacijama i/ili podacima
 - nepouzdanost, nedostupnost, manjkavost
 - "informacijska glad"
 - reorganizacija, utjecaji okoline
 - promjene organizacijske strukture, poslovnih procesa, ... (pr. Bolonja)
 - pokazatelji poslovanja
 - npr. pad prodaje, uska grla proizvodnje, neplanirano povećanje troškova
 - zastarjela tehnologija
 - npr. razvojni alati, arhitekture, ...

Predlaganje i odabir projekata

□ Predlaganje projekata

- temeljem (prethodno napravljenog) plana informatizacije
 - planu prethodi analiza i prijedlog strategije
- temeljem zahtjeva na sustav unutar organizacije
 - sponzor projekta interno dostavlja zahtjev na sustav (system request)
 - provodi se snimka stanja i izrada prijedloga - povelje projekta
- temeljem prijava na natječaj organizacije koja traži rješenje
 - javlja se više ponuditelja od kojih se odabire najbolji (najjeftiniji !?)
 - primjer: <https://eojn.nn.hr/Oglasnik/>
- temeljem prijava na natječaj fondova
 - predlagatelj navodi potencijalno tržište te očekivanu korist
 - primjer : <http://www.bicro.hr>



□ Odabir projekta (project selection)

- povjerenstvo/odbor za odabir (steering committee, approval comitee)

Zahtjev na sustav

- **sažetak projekta (sponzor, naziv, cilj, svrha)**
- **poslovne potrebe – poslovni razlozi za pokretanje**
 - npr. povećanje prodaje, povećanje udjela na tržištu, poboljšanje usluge ...
- **poslovni zahtjevi – poboljšane mogućnosti poslovanja**
 - npr. pristup putem interneta, izrada upravljačkih izvješća, ...
- **očekivana korist – poslovna vrijednost (!?)**
 - npr. povećanje prodaje 3%, udjela na tržištu 1%, uštede u održavanju
- **posebnosti i ograničenja - argumenti relevantni za odobrenje**
 - npr. rok, kompatibilnost s politikom sigurnosti i zaštite

Snimka stanja

- Početno istraživanje (survey phase, initial study, preliminary investigation)
 - "Je li projekt vrijedan pažnje?"
 - **problem:** sprječava ispunjenje svrhe, postizanje ciljeva, obavljanje zadaća
 - **prilika:** mogućnost pozitivne promjene, čak i kada ne postoji problem
 - **direktiva:** poslovno ograničenje (pr. pravilnik) ili vanjski utjecaj (pr. zakon)
 - aktivnosti
 - pregled poslovnih planova, inventura aplikacija, intervjuiranje
 - procjena dorade, nadgradnje i izgradnje
- Rezultat:
 - povjedna projekta (project charter),
 - početni doseg i plan projekta
- Povjerenstvo odobrava, odbacuje ili odgađa prijedlog
- Primjeri:  \Planiranje\PISFER-Snimka-*
 - jedan krug intervjuja i prikupljanja dokumentacije – 30 SA, 1 mjesec

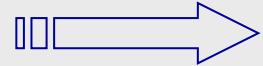
Primjer: postojeći problemi, prijedlozi rješenja

Kratko obrazloženje problema, mogućnosti ili direktive	Hitnost	Vidljivost	Korist	Prioritet	Predloženo rjesenje
1. Vrijeme odgovora na narudžbu mjereno od vremena zaprimanja narudžbe do isporuke klijentu se povećalo na prosječno 15 dana	HITNO	Visoka	175,000	2	Novi razvoj
2. Nedavno preuzimanje kompanija: <i>Privatna predstava i Filmsko platno</i> nametnulo je povećanje zahtjeva za protokol informacija i dokumenata.	6 mjeseci	Srednja	75,000	2	Novi razvoj
3. Trenutno 3 različita sustava za unos narudžbi servisiraju odjeli za audio, video i video igre. Svaki sustav ima vlastito sučelje prema različitom skladišnom sustavu, pa treba objediniti skladišnu evidenciju.	6 mjeseci	Srednja	515,000	2	Novi razvoj
4. Postoji nedostatak pristupa informacijama nužnim za upravljanje i donošenje odluka. Ovo će se još pogoršati preuzimanjem dva dodatna sustava za obradu narudžbi (<i>iz Privatna predstava i Filmsko platno</i>).	12 mjeseci	Niska	15,000	3	Po razvitku novog sustava, pružiti korisnicima lako svladive alate za pisanje izvještaja.
5. Izražena je nedosljednost (nekonzistentnost) između podataka u evedencijama članova i narudžbi.	3 mjeseca	Visoka	35,000	1	Brza ispravka, a zatim novi razvoj
6. Sustavi datoteka u <i>Privatna predstava i Filmsko platno</i> nisu kompatibilni s onim u <i>Zvučna pozornica</i> . Problemi s podacima obuhvaćaju nedosljednosti u podacima i nedostatak upravljanja ulazom i izmjenama.	6 mjeseci	Srednja	nepoznato	2	Novi razvoj, dodatna ocjena koristi može povećati žurnost
7. Postoji mogućnost uvođenja sustava naručivanja putem Interneta, ali su sigurnost i kontrola pristupa problematični.	12 mjeseci	Niska	nepoznato	4	Buduće verzije tek razvijenog sustava
8. Postojeći sustav unosa narudžbi nije kompatibilan s planiranim sustavom za automatsku identifikaciju (štapićasti kod) koji se razvija za skladište.	3 mjeseca	Visoka	65,000	1	Brza ispravka, a zatim novi razvoj

•Vidljivost: U kolikoj mjeri će rješenje ili novi sustav biti dostupni korisnicima.

•Korist: Paušalna procjena koliko bi rješenje povećalo dobit ili smanjilo trošak u jednoj godini.

Proučavanje problema, analiza izvedivosti



□ Proučavanje problema (study phase)

- "Jesu li problemi vrijedni rješavanja?"
- produbljenje snimke
- uočavanje problema, uzroka i mogućih posljedica
- preciziranje ciljeva, prijedlozi rješenja, procjena izvedivosti
- preciziranje dosega projekta

□ povjerenstvo odobrava, odbacuje ili reducira projekt

□ Analiza izvedivosti (feasibility analysis)

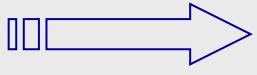
- procjena može li se projekt provesti s obzirom na uvjete ?
- mali projekti - dio pokretanja projekta
- veliki projekti – zasebno, čak i više puta, ili kao zaseban projekt !

Pokretanje projekta i početno planiranje

- Može teći usporedno i temeljem snimke stanja i dublje analize
- Uspostava projekta
 - plan rada, ustroj ekipe, uspostava nadzora i kontrole
 - važno je osigurati predanost dionika zajedničkom cilju (commitment) !
- Određivanje svrhe i ciljeva projekta (goal, objectives)
 - **Svrha** – nespecifična, krajnji cilj djelovanja i krajnji predmet želje
 - **Cilj** – konkretan, može ih biti više, stanje koje projekt nastoji postići
- Doseg (scope)
 - **System boundary** – Granice sustava ? Što će biti napravljeno ?
 - **Constraints** – Ograničenja (teh, org, fin) ? Što neće biti napravljeno ?
 - **Objectives** – Što će se postići ? Čemu će služiti ?
 - **Permissions** – Tko će i pod kojim uvjetima koristiti ?
 - **End products** – Kako se mjeri dovršenost i uspjeh (neuspjeh) ?
- Početno planiranje
 - strukturiranje životnog ciklusa
 - okvirni vremenski plan po fazama

Primjeri strategije, izvedivosti i plana

- Strategija srednjoročnog razvoja IS Croatia osiguranja d.d., 2008.
- Specifikacija sustava Cro-Fauna, Državni zavod za zaštitu prirode, 2007.
- Strategija razvoja informacijskog sustava Hrvatskih šuma d.o.o., 2005.
- ...
- Idejno rješenje informacijskog sustava Personalne uprave MORH, 1996.
- Globalno idejno rješenje informatizacije za Sljeme d.d., Sljeme, Zagreb, 1995.



Proučavanje problema, analiza izvedivosti

(detaljnije)

Analiza problema

□ Produbljivanje analize za projekte koji prođu početnu selekciju

- "Jesu li problemi vrijedni rješavanja?", "Je li gradnja isplativa?"
- Detaljnija analiza problema, njihovih uzroka i posljedica
 - **"Ne pokušavaj popraviti prije nego shvatiš kako radi!"**
- Analiza poslovnih procesa
 - Koji su najveći problemi?
 - Koja su moguća rješenja problema?
 - Kako informatizacija može pomoći?
- Grubo modeliranje sustava

□ Mogu se koristiti različite metode # primjeri slijede

- Analiza kritičnih faktora uspjeha (CSF - Critical Success Factors)
 - čimbenici kojima poslovodstvo posvećuje posebnu pažnju
 - čimbenici koji razmjerno brzo i lako doprinose ostvarivanju ciljeva
- Procjena troškova-koristi

Istraživanje problema, uzroka i posljedica

□ Primjer: Istraživanje uzroka

- Problem: pad prodaje
- Vidljivi znak (manifestacija): povećani opoziv (storno) narudžbi
- Razlog (posljedica): nezadovoljstvo kupaca
- Uzrok: spor sustav za naručivanje

□ Treba razdvojiti uzroke i posljedice problema !

- sporost sustava nije izoliran problem nego može biti posljedica
 - pomanjkanja osoblja, "pretjerane" obrade, ručne obrade, ili ... ?
- primjer: pisarnica FER-a
 - (diskusija: što učiniti)

Postavljanje ciljeva

□ Primjeri poslovnih ciljeva (stvarnih projekata)

- Potpomoći reorganizaciju u tržišno orijentirano poduzeće prema EU normama.
- Osigurati informacije o izvorima, razlozima i mjestu nastanka svakog troška u sustavu.
- Uskladiti hijerarhiju odlučivanja s hijerarhijom u poduzeću.
- Racionalizirati utrošak novca za ...

□ Primjer ciljeva IS

- Problem: Predugo vrijeme unosa nečega
- Cilj: Ubrzanje unosa
- Kriterij: performanse, izvedba – mora biti mjerljiv
 - Apsolutni iznos ili relativni iznos konkretnе vrijednosti
 - Može biti binaran – "postoji ili ne postoji", npr. online pretraživanje
- Loš primjer: "Povećati broj unosa u 24 sata za 5%"
- Bolji primjer: "Unijeti 98% nečega unutar 24 sata"

Ograničenja

□ Osoblje

- Popunjenošć odjela informatike je 60%
- Moguće je zaposliti najviše $X << 100\%$ zaposlenika

□ Materijalni trošak

- Nabavka uredskog i potrošnog materijala ne smije premašiti XX HRK

□ Računalna oprema

- Projekt se mora obaviti bez nabavke novog hardvera – gospodarstvo ?
- Trošak opreme poželjno predstavlja barem 33% budžeta – znanost !

□ Financijska sredstva

- Ukupni budžet projekta je XX HRK (uvijek manji od traženog!)
- Naknade izvođačima ne smiju premašiti XX% ukupnog iznosa (teži 0!)

□ Tehnička ograničenja (OS, IDE, ...) – trivijalna !

Ključni čimbenici uspjeha

□ Primjer: Urudžbeni zapisnik

- *Izrada adekvatne programske podrške*
- *Unos svih podataka iz knjiga urudžbenog zapisnika u bazu podataka*
- *Povezivanje sa službama na fakultetu koje koriste iste dokumente (napraviti dijeljene dokumente)*
- *Ospozobiti djelatnike za rad na novom sustavu*

□ Primjer: CIP

- *odobrenje samostalnog budžeta, te dodatno investiranje u softver, hardver i sustav rezervnog napajanja*
- *integracija sustava*
- *investicije u fizičku zaštitu (protupožarni i protuprovalni sustav)*
- *zapošljavanje dodatnih djelatnika sa SSS*

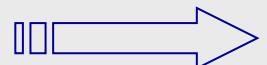
□ Rješenja, kao i problemi, najčešće nisu informatički !!!

Analiza izvedivosti

□ Analiza/Studija izvedivosti (feasibility analysis, feasibility study)

- procjena korisnosti, praktičnosti i isplativosti projekta
- izvješće (feasibility report) - relevantnim dionicima ili odboru za odabir
 - eventualni povratak u studiju izvedivosti → revidirano izvješće

□ Kakva izvedivost ?



- organizacijska izvedivost
- tehničko-tehnološka izvedivost
- vremenska izvedivost
- ekonomska izvedivost - analiza troškova koristi

□ Kada provesti ?

- uobičajeno tijekom planiranja
- po promjeni dosega – početno izvediv projekt može postati neizvediv
- između ključnih faza projekta – uz izvedivost, dokaz održivosti

Organizacijska izvedivost (1)

» Ima li smisla graditi ? Hoće li se koristiti?

□ Operativna izvedivost

- procjena hitnosti rješavanja problema (planiranje)
- procjena prihvatljivosti rješenja (kasnije faze)

□ Vrijedi li rješavati problem? Rješava li predloženo rješenje problem?

» PIECES koncept

- **P** [performance]
 - protočnost i odziv sustava
- **I** [information]
 - informacije (podaci): pravodobne, prikladne, ažurne, korisne
- **E** [economics]
 - kontrola troškova ili povećanje profita
- **C** [control]
 - poboljšanje kontrole ili sigurnosti, npr. točnost, sigurnost i zaštita podataka
- **E** [efficiency]
 - efikasnost, maksimizacija uporabe resursa (ljudi, sredstva, novac - vrijeme)
- **S** [service]
 - usluga korisnicima (zaposlenici, partneri, ...), elastičnost i prilagodljivost

Organizacijska izvedivost (2)

□ Socio-psihološka izvedivost

- Da li će rješenje zadovoljiti korisnikove potrebe?
- Do kojeg stupnja?
- Kako će rješenje promijeniti korisnikovu radnu okolinu?

□ Koji je stav korisnika prema rješenju? Da li će se sustav koristiti?

- Podrška uprave
- Promjena radnog okruženja, procedura – prilagodba promjenama
- Prihvatanje krajnjih korisnika – otpori budućoj ulozi ili tehničkom rješenju
 - Rizici : informatička nepismenost, strah od neznanja, novi način rada, gubitak posla, kontrole i utjecaja

□ Procjena upotrebljivosti (najčešće prototipom u kasnijim fazama)

- Krivulja učenja – vrijeme osposobljavanja potrebno za postizanje pune primjene
- Lakoća korištenja – jednostavno sučelje za početnike, složenije za iskusne
- Zadovoljstvo – ponuđenom rješenju korisnik daje prednost pred postojećim

Tehničko-tehnološka izvedivost

» Može li se sagraditi? Može li se kupiti?

□ Procjena mogućih tehničkih rješenja i alternativa

- postojeća rješenja na tržištu ili u drugim organizacijama

□ Primjenjivost rješenja, tehnologije

- Najsuvremenija tehnologija - naprednost, zanimljivost, rizik
- Zrela i dokazana tehnologija - sigurnost, bolja podrška

□ Raspoloživost tehnologije

- Može li se nabaviti?
- Ima potrebne karakteristike?
- Treba prilagoditi ili doraditi?

□ Stručnost

- Postoji li potrebna stručnost i vještina za primjenu tehnologije?

□ Primjer: Uprava-IzborAlata

Vremenska izvedivost

» Kada će biti gotovo? Može li biti gotovo na vrijeme?

□ Prihvatljivost vremenskog rasporeda

- opravdanost rokova s obzirom na raspoloživu stručnost

□ Poželjni rokovi

- bolje isporučiti potrebno nešto kasnije, nego beskorisno na vrijeme !!!
- uvažavanje promjena posmiče završetak projekta
- iterativno se ažurira i predlaže alternativni vremenski plan

□ Čvrsti rokovi (timeboxing)

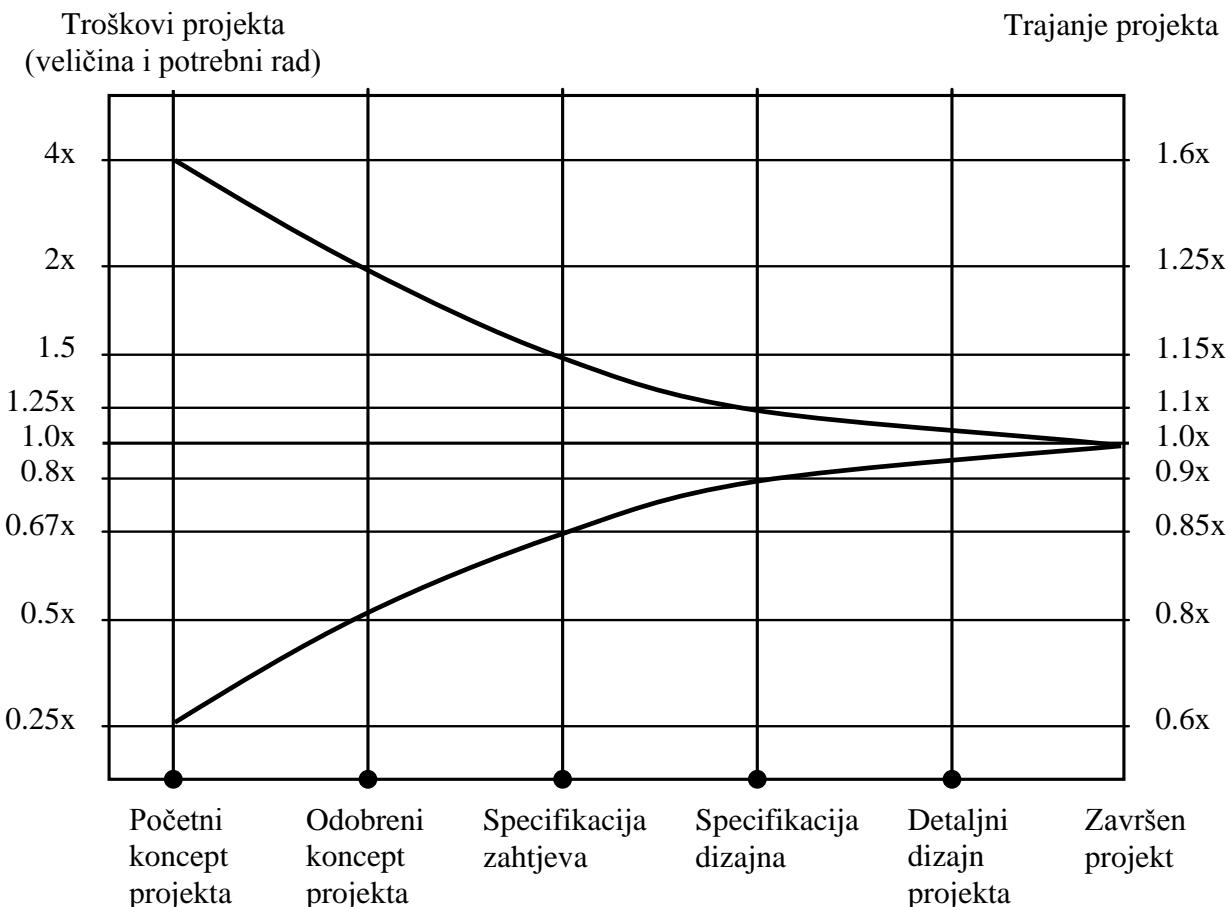
1. postavlja se čvrsti rok isporuke
2. određuju prioriteti funkcionalnosti koju treba ugraditi
3. gradi se jezgra sustava (nužne funkcionalnosti)
4. odgađa se ugradnja funkcionalnosti koja ne može biti dovršena na vrijeme
5. u roku se isporučuje sustav s dosegnutom funkcionalnošću
6. ponavljaju se koraci 3. do 5. s nadodanom funkcionalnošću

□ Koliko je razuman predloženi plan?

- iskustvo na sličnim projektima (najbolje)
- modeli koji su predviđeni za tip industrije i tip razvoja (npr. COCOMO)
- opći modeli razvojnog procesa (npr. Rapid Development)

Problem (ne)preciznosti procjene

Faza projekta	Potrebni rad i veličina projekta		Trajanje projekta	
	Optimistično	Pesimistično	Optimistično	Pesimistično
Početni koncept projekta	0.25	4.0	0.60	1.6
Odobreni koncept projekta	0.50	2.0	0.80	1.25
Specifikacija zahtjeva	0.67	1.5	0.85	1.15
Specifikacija dizajna	0.80	1.25	0.90	1.10
detaljni dizajn projekta	0.90	1.10	0.95	1.05



Postupak procjenjivanja

□ Osnovni koraci

1. Procjena veličine projekta (metrika projekta)

- broj funkcijskih točaka (function point – FP) ili programskih redaka (LOC)
- broj objektnih točaka
- analogija s prethodnim projektima

2. Procjena potrebnog napora u čovjek-mjesecima (čm)

- iz procjene veličine projekta i produktivnosti
- tablice produktivnosti, algoritmički modeli, povijesni podaci

3. Procjena trajanja projekta (vremenski plan)

- trivijalno iz napora i veličine ekipe u kalendarskim mjesecima ili godinama
- npr. procijenjeni rad 36 čm / 6 članova ekipe = 6 kalendarskih mjeseci

□ Meta korak kojim se rješava problem nepreciznosti procjene

- Procjene treba izražavati u rasponima i, kako projekt napreduje, postupno smanjivati raspone i povećavati preciznost procjene.

Procjena veličine projekta i potrebnog napora

□ Funkcijske točke (FP)

- procjena broja ulaza, izlaza i datoteka i njihove složenosti * utjecaj * faktor prilagodbe
- primjer: 30 LOC / FP za C++
- procijenjenih 6000 FP zahtjeva izradu 180000 LOC
- produktivnost za poslovni sustav: 500 LOC/čm → ukupni napor = 360 čm

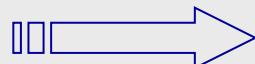
□ Objektne točke (aplikacijske točke, ne programski razredi!)

- procjena broja zaslonskih maski, izvješća i modula za pristup podacima
- produktivnost 4 do 50 OT / mjesec, ovisno o alatu i vještini (statistike uglavnom nema)
- primjer: 500 OT uz produktivnost 25 OT/mj. zahtjeva 20 čm
- primjer2: 2 h / šifrarnik, 4h / osnovna tablica, 8h / master-detail, 40 h / izuzetak *

□ Funkcijske i objektne točke

- temeljem dijagrama dekompozicije funkcija, dijagrama toka podataka i ER dijagrama, odnosno UC dijagrama i dijagrama razreda
- ... koji slijede kao rezultat dublje analize, a češće oblikovanja !

□ Procjena analogijom



Procjena analogijom

□ Skaliranje poznavanjem ciklusa i provedene aktivnosti

- npr. poznata je razdioba opterećenja za poslovne projekte: Planiranje (15%), Analiza (20%), Dizajn (35%), Izrada (30%)
- ako je na sličnom projektu na planiranje utrošeno 4čm, u ovom se može očekivati Analiza ($4/15\% * 20\% = 5.33\text{cm}$), Dizajn (9.33čm), Izrada (8čm)

□ Skaliranje analizom povijesnih podataka (paušalno - rule of thumb)

- struktura proizvoda, vrste projekata, evidencija utrošenog vremena, ...

Project	DBMS	Entities	Relationships	Generated	Manually coded	Coding
EOM	Zim	56	27	75%	sophisticated functions	84%
ISPAP*	Zim	106	310	55%	sophisticated functions	31%
ISSA**	Informix	78	90	20%	sophisticated functions	73%
ISTM**	Informix	85	116	10%	sophisticated functions	64%
ROS	Informix	39	44	90%	sophisticated reports	58%
SEE	Zim	47	83	80%	sophisticated functions	71%

Model	Domains	Attributes	Entities	Primary Keys	Foreign Keys
Initial Model of the central system	285	285	103	99*	0
The model at the end of integration	553	636	146	151*	243
The model at the end of design	338	459	131	135	228

Program element	No. of elements	No. of source code lines	Source code size (KB)
Program module	400	250.000	8.200
Form	170	10.000	300
TOTAL	570	260.000	8.500

Procjena trajanja projekta

- **Iskustveno pravilo za optimalnu procjenu trajanja temeljem npora:**
 - optimalno trajanje u mjesecima = $3.0 \cdot (\text{čovjek-mjeseci})^{1/3}$
- **Za procjenu projekta koji zahtijeva napor 65 čm,**
 - optimalno trajanje je $3.0 \cdot 65^{1/3}$, tj 12 mjeseci
- **To dalje znači da je optimalna veličina tima**
 - $65/12$, tj. okvirno 5-6 članova
- **Što se događa kada projekt treba dovršiti brže ili sporije?**
 - prema različitim izvorima faktor 3.0 može varirati od 4.0 do 2.5
 - više članova komplicira komunikaciju pa će biti manje produktivni
 - možemo angažirati manje članova, ali će nas dulje koštati
- **I zato ... procjena ekonomske izvedivosti**

Ekonomска извештавост

» Isplati li se graditi? Kada će se isplatiti?

□ Analiza troškova-koristi (Cost-Benefit Analysis - CBA)

□ Fiksni troškovi – neovisni o poslovnim aktivnostima

- absolutni iznos, početna procjena, ažuriranje tijekom projekta
 - osoblje: plaće, izobrazba (tečajevi)
 - oprema – nabava nakon odabira tehničkog rješenja

□ Varijabilni troškovi – proporcionalni poslovnim aktivnostima

- relativan iznos, ovisan o uporabi, npr:
 - režije (struja, telefon, internet)
 - putni troškovi
 - materijalni troškovi i troškovi održavanja (ljudski rad)

Kategorije troškova i koristi

□ Mjerljivi (tangible – opipljiv, određen, shvatljiv)

- zna se točan iznos ili iznos može biti procijenjen
- mjerljivi troškovi -
 - plaće, režije, licence, ...
- mjerljive koristi
 - izražene kao godišnja ušteda ili ušteda po proizvedenom predmetu

□ Nemjerljivi (intangible)

- postojanje ili "vrijednost" ne može se egzaktno mjeriti ili dokazati
- nemjerljivi troškovi
 - pad morala, (nemjerljivi) pad produktivnosti ili gubitak tržišta
- nemjerljive koristi
 - mogu pomoći ili odmoći korisnosti proizvoda,
 - npr. poboljšano zadovoljstvo kupca, zadovoljstvo zaposlenika, ...

Primjeri troškova i koristi

□ Vrijednost novog kupca

- Vrijednost 300 novih kupaca godišnje koji prosječno potroše \$500 po proizvodu koji nakon troškova donosi 12% dobiti
- godišnja dobit iznosi $300 * \$500 * 12\% = \18.000

□ Vrijednost postojećih kupaca

- gubitak 100 kupaca od kojih svaki troši \$2500 godišnje uz dobit 12%, a za čije nadomještanje/reklamiranje treba uložiti \$50.000, gubitak iznosi:
- $100 * \$2500 * .12 + \$50.000 = \$80.000$

□ Smanjenje cijene rada ili ušteda smanjenjem posla

- smanjimo trajanje zadatka s 5 min na 30 sec, a radi osoba za \$50 na sat
- ušteda je $(5 - 30/60)/60$ sati po zadatku * \$50 na sat = \$3,75 po zadatku

□ Nemjerljive koristi nastojimo izraziti paušalnom procjenom iznosa

- nezadovoljni kupci naručuju manje i rjeđe – postotak gubitka u prihodu

Primjer: troškovi razvoja

Osoblje:			
Vrsta	Količina	Cijena	
Analitičar sustava	900h * 45kn/h	40,500 kn	
Programer	1375h * 36kn/h	49,500 kn	
Stručnjak za komunikacije	60h * 40kn/h	2,400 kn	
Administrator baza podataka	30h * 42kn/h	1,260 kn	
Pisac dokumentacije	240h * 25kn/h	6000 kn	
Tajnica	160h * 15kn/h	2,400 kn	
Unos podataka	80h * 12kn/h	960 kn	

Edukacija:			
Vrsta	Količina	Cijena	
„in-house“ poduke za programere	3 dana	7,000 kn	
„in-house“ poduka za korisnike	3 dana	10,000 kn	

Materijal:			
Vrsta	Količina	Cijena	
Kopiranje		500 kn	
Diskovi, trake, papir		650 kn	

Sklopoljje i programska podrška:			
Vrsta	Količina	Cijena	
Windows licence		1,000 kn	
Memorija za 20 klijenata		8,000 kn	
Periferni uređaji za 20 klijenata		2,500 kn	
Mrežni programi		15,000 kn	
Office alati		20,000 kn	

Primjer: godišnji troškovi rada

Osoblje:		
Vrsta	Količina	Cijena
Programer održavanja/analitičar	250h/god*42kn/h	10,500 kn
Mrežni administrator	300h/god*50kn/h	15,000 kn

Nadogradnja sklopoljia i programske podrške:		
Vrsta	Količina	Cijena
Sklopolje		5,000 kn
Programska podrška		6,000 kn
Ostali troškovi		3,500 kn

Sadašnja vrijednost troškova i koristi

□ Sadašnja vrijednost (Present value - PV)

» \$ označava novčanu jedinicu u bilo kojoj valuti

- Današnja vrijednost onoga što će postati \$1.00 nakon 'n' godina u budućnosti, ako uzmemo u obzir kamate 'I' iznosi:

$$PV = 1/(1 + I)^n = (1 + I)^{-n}$$

- Razlika predstavlja kamatu koja se može zaraditi tim novcem

□ Primjeri:

- troškovi razvoja od \$100.000 imaju trenutnu vrijednost od \$100.000
- oričenje tih sredstava na 3 godine uz kamatu od 8% donijelo bi 25.97% dobiti od kamata, tj. $(1 + 0.08)^3 / 100$
- obratno, korist \$100.000 postignuta u 3. godini sada vrijedi \$79.380, tj. $\$100.000 / (1 + 0.08)^3$

□ Primjer: \Planiranje\NPV.xls

- Pitanje: Što je realna "kamata" na budući trošak / prihod projekta ?

Neto sadašnja vrijednost

□ Neto sadašnja vrijednost - Net Present Value (NPV)

- budući trošak i korist s obzirom na gubitak vrijednosti sredstava
- razlika PV budućih priljeva i PV budućih odljeva
- $NPV = (\text{ukupna korist} - \text{ukupni troškovi})$ preračunati na današnji dan

□ Primjer: Koji je projekt isplativiji ?

$$NPV = \sum_{i=1}^n \frac{\text{values}_i}{(1 + rate)^i}$$

Kamata	10%					
Projekt 1	1	2	3	4	5	Ukupno
Trošak	-5,000	-1,000	-1,000	-1,000	-1,000	-9,000
Korist	0	2,000	3,000	4,000	5,000	14,000
Cash flow	-5,000	1,000	2,000	3,000	4,000	5,000
Projekt 2						
Trošak	-2,000	-2,000	-2,000	-2,000	-2,000	-10,000
Korist	1,000	2,000	4,000	4,000	4,000	15,000
Cash flow	-1,000	0	2,000	2,000	2,000	5,000

NPV

2,316

3,201

Povrat investicije

□ Povrat investicije

- Ulaganja donose korist koja s vremenom postaje sve veća
- U jednom trenutku prihod dosegne rashod

□ Indeks profitabilnosti (Cost Benefit Ratio)

- omjer sadašnje vrijednosti koristi i sadašnje vrijednosti troškova (PVB / PVC)
- za prethodni primjer $9743/7427 \sim 1.31$, odnosno $10783/7582 \sim 1.42$
- favorizira brzi povrat investicije a ne dugoročnu dobit

□ Vrijeme povrata investicije (Payback period)

- Vrijeme povrata ukupnog troška
- Razdoblje potrebno da prihod dosegne rashod

□ Točka povrata investicije (Break-even point)

- Trenutak u kojem prihod dosegne rashod

Primjer: Vrijeme povrata investicije

Kamata	12.00%					
Trošak / Korist	Godina 0	Godina 1	Godina 2	Godina 3	Godina 4	Godina 5
Trošak razvoja	(418,040)					
Operativni troškovi		(15,045)	(16,000)	(17,000)	(18,000)	(19,000)
Faktor za kamatu	1.00	0.893	0.797	0.712	0.636	0.567
Sadašnja vrijednost	(418,040)	(13,435)	(12,752)	(12,104)	(11,448)	(10,773)
Kumulativni trošak	(418,040)	(431,475)	(444,227)	(456,331)	(467,779)	(478,552)
Korist od novog IS		150,000	170,000	190,000	210,000	230,000
Faktor za kamatu	1.00	0.893	0.797	0.712	0.636	0.567
Sadašnja vrijednost	0	133,950	135,490	135,280	133,560	130,410
Kumulativna korist	0	133,950	269,440	404,720	538,280	668,690
	0	1	2	3	4	5
Ukupno NPV	(418,040)	(297,525)	(174,787)	(51,611)	70,501	190,138

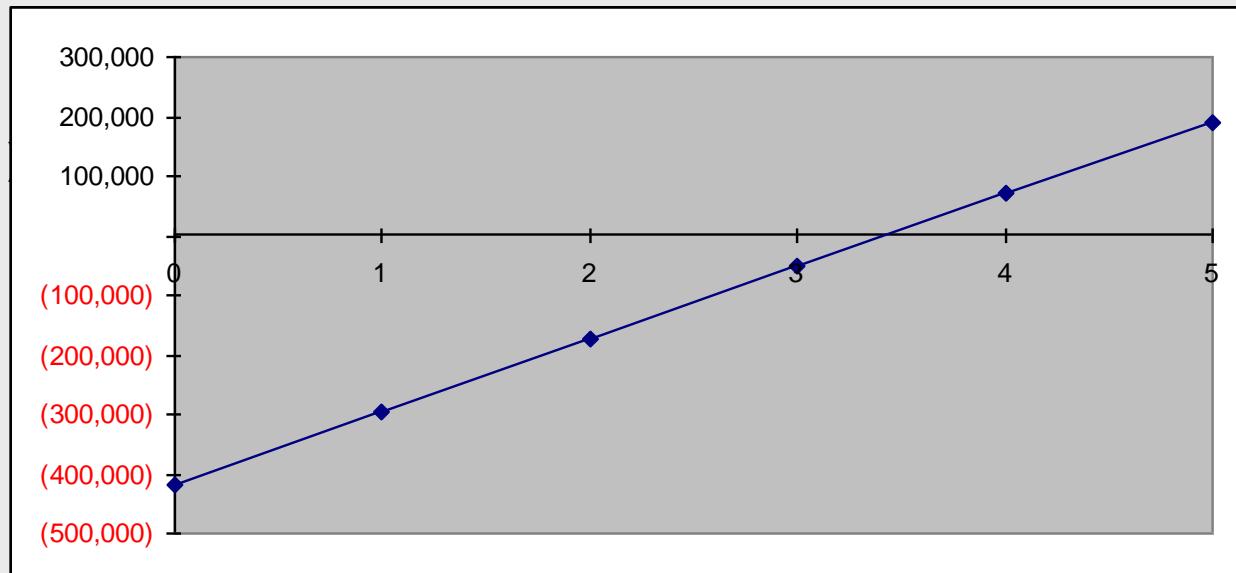
Vrijeme povrata

$$= 3 + 51611 / (70501 + 51611)$$

$$= 3.42 \text{ godine}$$

Primjer: [\Planiranje\ PovratInvesticije.xls](#)

Komentirati 0. i 1. godinu



Primjer: Analiza povrata investicije

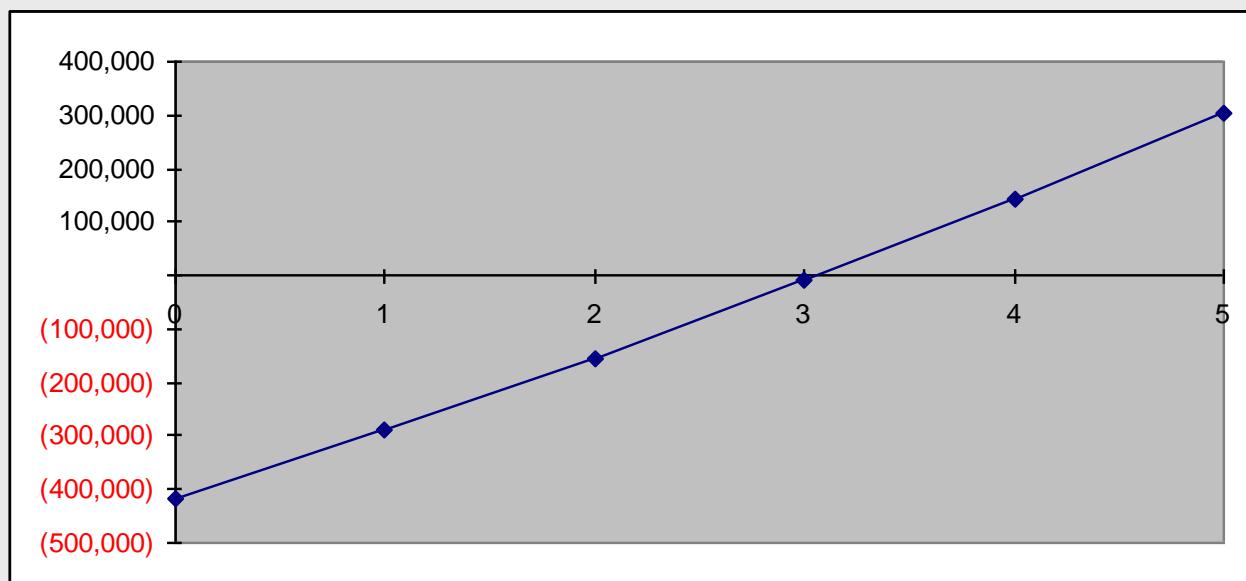
Kamata

6.00%

Trošak / Korist	Godina 0	Godina 1	Godina 2	Godina 3	Godina 4	Godina 5
Trošak razvoja	(418,040)					
Operativni troškovi		(15,045)	(16,000)	(17,000)	(18,000)	(19,000)
Faktor za kamatu	1.00	0.943	0.890	0.840	0.792	0.747
Sadašnja vrijednost	(418,040)	(14,187)	(14,240)	(14,280)	(14,256)	(14,193)
Kumulativni trošak	(418,040)	(432,227)	(446,467)	(460,747)	(475,003)	(489,196)
Korist od novog IS		150,000	170,000	190,000	210,000	230,000
Faktor za kamatu	1.00	0.943	0.890	0.840	0.792	0.747
Sadašnja vrijednost	0	141,450	151,300	159,600	166,320	171,810
Kumulativna korist	0	141,450	292,750	452,350	618,670	790,480
	0	1	2	3	4	5
Ukupno NPV	(418,040)	(290,777)	(153,717)	(8,397)	143,667	301,284

Za kamatu 6%

- NPV = 300k
- VP = 3.05 godina



Povrat investicije

□ Postotak povrata investicije

- ROI - postotak relativne koristi projekta u odnosu na trošak
- $ROI = (\text{ukupna korist} - \text{ukupan trošak}) / (\text{ukupan trošak})$
- preračunato u sadašnju vrijednost, $ROI = NPV / (\text{ukupan sadašnji trošak})$
- Za primjer s kamatom 12%, $ROI = 190378 / 478552 = 0.3973 = 39.73\%$
- Za primjer s kamatom 6%, $ROI = 301284 / 489196 = 0.6159 = 61.59\%$

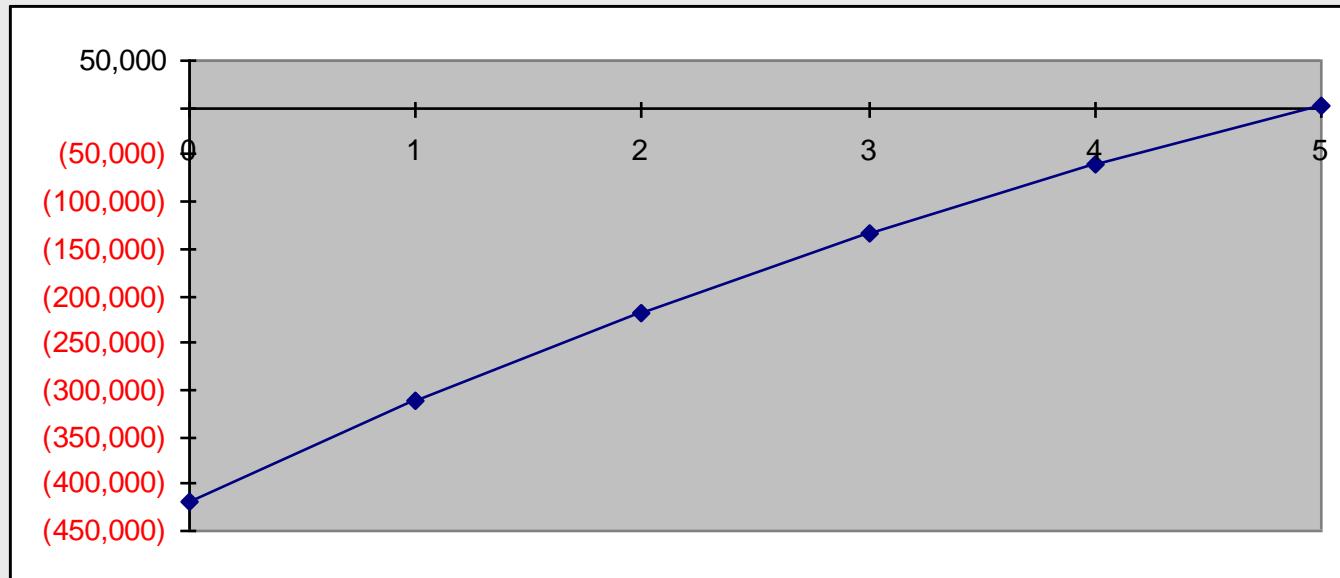
□ Faktor obnavljanja kapitala

- ROI se obično dijeli s dužinom projekta kako bi se dobio godišnji ROI → faktor obnavljanja kapitala
- Nizak ROI (~ manji od 10% godišnje) može pokazivati da je korist preniska da bi bila isplativa
- Za primjer s kamatom 12%, $ROI\% = 7.95\%$ godišnje
- Za primjer s kamatom 6%, $ROI\% = 12.32\%$ godišnje

Interna stopa rentabilnosti

□ Interna stopa rentabilnosti - Internal Rate of Return (IRR)

- Interna stopa povrata investicije
- Izražava potrebnii postotak povrata potreban da bi se trošak i korist izjednačili u nekom vremenskom razdoblju
- Kamatna stopa pri kojoj je $NPV = 0$



Modaliteti izgradnje sustava

Dio analize izvedivosti ili međukorak analize i dizajna

Vlastiti razvoj (insourcing)

□ Varijante

- Razvoj vlastitim snagama (**in-house**), može sadržavati osposobljavanje i angažiranje netehničkog osoblja
 - Primjeri: Ministarstvo, Društvo, Poduzeće
- Najam osoblja povremeno/dugoročno (**buy-in, preferred supplier**)
 - Primjer: Banka

□ Prednosti

- povećanje stručnosti naručitelja, samostalnost

□ Nedostaci

- trajni trošak kućne informatike
- problem paralelnog razvoja i održavanja, gomilanje posla

□ Argumenti za vlastiti razvoj

- informatizacija osnovnog posla (core business), kada ne postoji gotova rješenja na tržištu ili takva da se postigne komparativnu prednost
- posebni razlozi - povećana tajnost podataka i procesa ili sigurnost IS

Vanjski razvoj (outsourcing)

□ Najam usluge razvoja IS ili njegovih dijelova

- poduka djelatnika informatičke struke
- savjetništvo pri provedbi pojedinih faza ciklusa
- izrada rješenja po specifikaciji naručitelja
- ...

□ Varijante

- Ugovoren razvoj isporuke gotovog proizvoda (**contract out**)
- Dugoročna suradnja s isporučiteljem (strateškim partnerom) ili izdvajanje odjela informatike u preferiranog izvođača (**preferred contractor**)
- Multisourcing – optimalni skup različitih dobavljača, uz vlastiti razvoj

□ Ključna karakteristika

- zavisnost o dobavljaču

Izrada po mjeri ili nabava gotovog rješenja ?

□ Razvoj od strane vanjskih dobavljača

- Prednost: rješenje po mjeri naručitelja
- Nedostatak: visoka cijena unikata
- Primjer: Turistička Agencija

□ Nabava gotovog rješenja

- aplikacije za standardno poslovanje
 - financijsko poslovanje (accounting)
 - robno-materijalno poslovanje (material management/distribution)
 - upravljanje ljudskim resursima i plaće (HR management, payroll)
 - proizvodnja (manufacturing)
- niža cijena (!?) i kraće vrijeme "konfekcije" (!?)
- također zahtijeva prilagodbu (parametrizaciju) i konverziju podataka !

Graditi ili kupiti gotovo (**build.vs.buy**) ?

□ Određivanje mogućih rješenja

- Identifikacija rješenja na temelju analize poslovnih zahtjeva
- Ulazi: specifikacija HW i SW te odabrana tehnočka arhitektura
- Izlazi: moguća rješenja novog sustava i njihove karakteristike

□ Analiza izvedivosti alternativnih rješenja

- Procjena alternativa s obzirom na njihovu izvedivost
- Ulazi: karakteristike rješenja, cijene, reference i uvjeti dobavljača
- Izlazi: analiza izvedivosti za svako moguće rješenje

□ Prijedlog rješenja sustava koje će se oblikovati i ugraditi

- Odabir onog rješenja koje ima najbolju ukupnu kombinaciju izvedivosti
- Ulazi: analiza izvedivosti, plan projekta, procjena veličine projekta
- Izlazi: prijedlog sustava

□ Kriteriji za odabir

- Primjeri:  \Planiranje\IEEE-KriterijiNabave

Primjer: moguća rješenja i njihove karakteristike

Karakteristike	SuperVideo	Video Boss	Video	ZZ Video
Operacijski sustav	Windows	Windows	Dos	Linux
Baza podataka	Access 2	Paradox 8	dBase III	MySQL
Brzina pretraživanja i dohvata podatka	velika	velika	mala	srednja
Programski jezik	Visual Basic	C++	Cliper	Clarion
Raspoloživ izvorni kod	ne	ne	da	ne
Korisničko sučelje	grafičko	grafičko	tekstovno	tekstovno
Integrirani sustav pomoći (on-line help)	da	ne	ne	ne
Dokumentacija (papirnata)	dobra	ne	ne	dobra
Mogućnosti aplikacije	velike	vrlo male	male	velike
Integracija s drugim aplikacijama	dobra	srednja	ne	ne
Brzina ispisa računa	srednja	srednja	velika	velika
Rad s različitim pisačima	da	da	ne	ne
Rad u mreži	da	ne	ne	ne
Krivulja učenja	1-2 dana / 2 tjedna	1 dan / 1 mjesec	1dan / 2 tjedna	1-2 dana / 1 tjedan
Arhiviranje podataka	da	ne	da	da
Upotreba konfiguracije za druge poslove	velika	velika	ne	vrlo mala
Min. potrebno računalo
Preporučeno računalo				
Broj instaliranih paketa	27	10	152	87
Datum prve instalacije	9/95	7/96	4/93	10/94
Cijena paketa	1500 kn	100 kn	500 kn	2000 kn
Cijena min. potrebnog računala (plus monitor i pisač)	3500 kn	4000 kn	2200 kn	3500 kn
Cijena preporučenog računala (plus monitor, pisač i modem)	8000 kn	8000 kn	-	
Cijena operacijskog sustava i licenci				

Vrednovanje mogućih rješenja

- **Svojstva treba kvantificirati da bi se mogla usporediti**
 - Koristi se sustav bodovanja da bi se usporedio značaj različitih kriterija.
- **Model ponderiranog vrednovanja (Weighted Scoring Model)**
 - Odredi se težinski faktor za svaki kriterij (npr. 0-3).
 - Pojedinačnom kriteriju svakog od rješenja dodjeljuje se ocjena iz dogovorenog raspona (npr. 0-5), pomnožena s odgovarajućom težinom.
 - Dobiveni pojedinačni rezultati sumiraju se za svako od rješenja.

gdje su

$$S_i = \sum_{j=1}^n s_{ij} w_j$$

S_i = ukupna vrijednost i -tог rješenja

s_{ij} = vrijednost j -tог kriterija za i -to rješenje

w_j = važnost ili težina j -tog kriterija

- **Što učiniti kada su sustavi (pod)jednako bodovani ?**
 - Primjer: Uprava-IzborAlata
- **Što učiniti ako pojedino svojstvo ima više podsvojstava ?**
 - <http://www.zpr.fer.hr/zpr/Projekti/IPIS/>

Primjer: analiza izvedivosti za moguća rješenja

Karakteristike:	Težinski faktor	SuperVideo		Video Boss		Video		ZZ Video	
		Ocjena	Bodovi	Ocjena	Bodovi	Ocjena	Bodovi	Ocjena	bodovi
Operacijski sustav	2	4	8	4	8	1	2	3	6
Baza podataka	1	4	4	4	4	2	2	1	1
Brzina pretraživanja i dohvata podatka	4	5	20	4	16	1	4	4	16
Programski jezik	1	4	4	5	5	2	2	2	2
Raspoloživ izvorni kod	1	0	0	0	0	5	5	0	0
Korisničko sučelje	2	5	10	5	10	3	6	3	6
Integrirani sustav pomoći (on-line help)	2	5	10	0	0	0	0	0	0
Dokumentacija (papirnata)	2	4	8	0	0	0	0	4	8
Mogućnosti aplikacije	4	5	20	1	4	2	8	5	20
Integracija s drugim aplikacijama	3	4	12	3	9	0	0	0	0
Brzina ispisa računa	4	2	8	3	12	5	20	5	20
Rad s različitim pisačima	3	5	15	5	15	0	0	0	0
Rad u mreži	1	5	5	0	0	0	0	0	0
Vrijeme obuke korisnika	1	3	3	5	5	5	5	3	3
Arhiviranje podataka	2	5	10	0	0	5	10	5	10
Upotreba konfiguracije za druge poslove	3	5	15	5	15	0	0	3	9
Broj instaliranih paketa	1	3	3	2	2	5	5	5	5
Datum prve instalacije	1	3	3	3	3	5	5	5	5
Cijena paketa	2	2	4	5	10	4	8	2	4
Cijena računala i sistemskog softvera	3	3	9	2	6	5	15	3	9
Ukupno bodova:			171		124		97		124

Primjer:
 \Planiranje\ **PISFER-Studija izvedivosti**

		Komponente poslovno – upravljačkog IS-a		Skladište i materijalno		Opća služba		Urudžbeni zapisnik	
Nadogradnja	Operativna	3	3	3	3	1	1	1	0
	Tehnička	3	3	3	3	1	1	1	0
	Vremenska	3	3	3	2	1	1	1	0
	Ekonomска	3	3	3	3	1	1	1	0
	Ocjena alternative	3	3	3	2.75	1	1	1	1.75
Izrada vlastitog	Operativna	1	1	1	2	3	3	2	2
	Tehnička	1	1	1	2	2	2	1	2
	Vremenska	1	1	1	1	2	1	1	1
	Ekonomска	1	1	1	1	2	1	1	1
Nabava gotovog	Ocjena alternative	1	1	1	1.5	2.25	1.75	1.75	1.25
	Operativna	2	2	1	1	2	1	1	2
	Tehnička	3	2	1	1	2	1	2	2
	Vremenska	3	2	1	1	3	1	2	3
	Ekonomска	1	2	1	1	1	1	1	2
Konačan prijedlog alternative	Ocjena alternative	2.25	2	1	1	2	1	1.5	2
	Nadogradnja	x	x	x	x			x	
	Izrada vlastitog					x	x	x	
	Nabava gotovog								x



Je li izvediv projekt održiv ?

□ Klizanje dosega (creeping scope)

- polagano, ali značajno povećanje obujma uslijed pogrešne procjene, različitog tumačenja zahtjeva ili promjene zahtjeva
- granice projekta moraju biti definirane što je moguće preciznije
- klizanje možda neće biti uklonjeno ali će se barem moći nadzirati

□ Opcionalno se planira i provodi

- Izrada prototipa ili oglednog (pilot) projekta i procjena njegove uspješnosti
 - projekt koji se može uspješno i "brzo" realizirati (npr. 3-9 mjeseci)
 - Primjer: Projekt informatizacije zdravstva
 - Primjer: Pilot projekt Integriranog PIS visokoobrazovnih učilišta
- Procjena održivosti projekta
 - SurvivalTest [McConnell, 1998], <http://www.construx.com/survivalguide/>
 - Primjer:  Planiranje\OdrzivostProjekta, SurvivalTest



Primjeri

□ Natječaj : Simulator za izračun isplativosti uzgoja biljnih vrsta

- Cilj: izračunati prihod po hektaru površine
- Opseg:
 - prinos, cijene otkupa, potpore,
 - općine, gnojivo, sadnja, mehanizacija, priprema, navodnjavanje, energenti, ljudski rad ...
- **Prijava:** idejno rješenje i vrijednost ponude (rok je 6 mjeseci) ?

□ Koji je bolji u razdoblju od 5 godina

- stari projekt:
 - stara tehnologija i održavanje 10 kkn / mjesечно
- novi projekt:
 - nakon 3 godine besplatnog razvoja i prilagodbe
 - nabava licenci za 300 kkn uz održavanje 20% / god
 - dodatni trošak održavanja 200 kkn/god

Studijski slučaj

- Firma živi od aplikacije zasnovane na staroj tehnologiji (Clipper)
- Umorni vlasnik održava 30ak klijenata i 10x toliko krajnjih korisnika
- Ustrojena je ekipa od 11 programera koji rade na izradi nove verzije (korisnik-poslužitelj, objektni jezik i RBP sa 100 tablica)
- Modul koji čini pola aplikacije je 90% dovršen i na njemu radi 7 ljudi
- Preostala 2 modula nisu ni započeta, a može ih preuzeti 4 zaposlenika
- Po završetku slijedi primjena u više različitih poduzeća.

Strategija ?
 Model razvoja ?

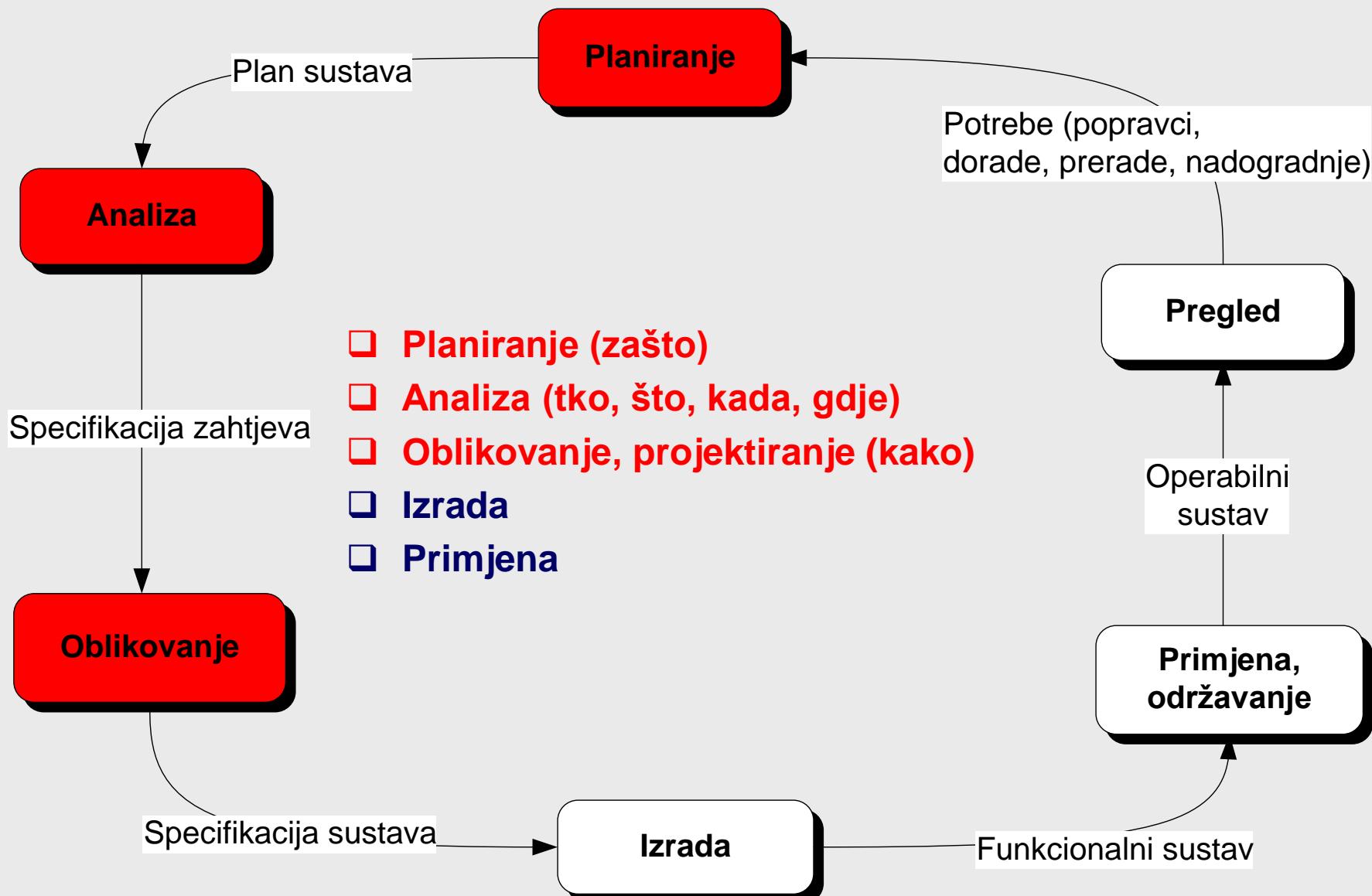
Reference

- **International Functional Point Users Group (IPUG)**
 - <http://www.ifpug.org/>
- **Modern Cost Estimation Tools**
 - <http://www.dacs.dtic.mil/techs/estimation/comparison.shtml>
- **T.T.Wilkens: Earned Value, Clear and Simple**
 - <http://handle.dtic.mil/100.2/ADA402619>
- **COCOMO**
 - <http://sunset.usc.edu/research/COCOMOII/>
 - <http://www1.jsc.nasa.gov/bu2/COCOMO.html>
- **Parametric Cost Estimating Handbook**
 - <http://cost.jsc.nasa.gov/pcehg.html>

Prikupljanje informacija i utvrđivanje zahtjeva

2012/13.03

Kontekst prikupljanja informacija



Postupci prikupljanja informacija

- **Intervjui s ključnim korisnicima i radne sjednice**
 - Lokalne informatičare svakako uključiti u analizu – sučeljavanje, rješavanje proturječja
- **Upitnici i ankete**
 - Nadomjestak za intervju ili prikupljanje informacija o resursima.
- **Analiza dokumentacije**
 - Treba prikupiti sve dokumente značajne za poslovanje (pravila, strukture podataka)
- **Promatranje**
 - Neposredni uvid u poslovne procese promatranjem radnih sredina
- **Ostale tehnike**
 - Prosudba postojećih aplikacija ili evidencija na računalu
 - analiza funkcionalnosti, strukture podataka te podataka koje treba “spasiti”
 - Prototipiranje – kada nema uzora ili korisnik ne zna što hoće
- **Postupak analize mora biti prilagođen korisniku !**

Postupak intervjuiranja

□ Intervju

- neusiljen i elastičan razgovor s ispitanikom, slijed pitanja i odgovora
 - ispitanik se pojavljuje u ulozi pasivnog sugovornika (!?)
 - sugovornici: rukovoditelji, krajnji korisnici, ostali sudionici
- standardno uključuje dva sugovornika ali može i više (korisnika, ispitivača):
 - **individualni** intervju – jedan korisnik ili suradnici koji rade isti posao
 - intervjuiranje **grupe** – više korisnika iz različitih područja

□ Organizacija razgovora

- niz pojedinačnih razgovora
- prema unaprijed dogovorenom planu i rasporedu → koordinator
 - Primjeri:  \Analiza\PlanRazgovora
- sporo i neučinkovito - organizacija svakog pojedinog razgovora
- neke razgovore (ponekad i seriju) treba ponoviti – nadopuna, proturječja
- broj razgovora ne može se unaprijed točno odrediti - prilagoditi situaciji

Tehnika intervjuiranja (2)

□ Priprema za razgovor

- utvrđivanje informacija koje treba saznati
- proučavanje postojeće dokumentacije i prethodnih nalaza
- određivanje dokumentacije koju bi trebalo prikupiti
- priprema pitanja koja će biti postavljena tijekom razgovora
 - izrada jezgre tema, to jest standardnih pitanja
 - izrada sveobuhvatnog podsjetnika i izdvajanje prikladnih pitanja

□ Primjeri: \Analiza\

- Podsjetnik-teme
- Podsjetnik-priprema
- Podsjetnik-upitnik

Tehnika intervjuiranja (3)

□ Planiranje i obavljanje razgovora

- okvirno trajanje prvog razgovora je 2 sata (općenito 1,5 – 2,5 sata)
- Početak razgovora
 - predstavljanje sudionika
 - upoznavanje sa svrhom razgovora (prikljicanje informacija o ...)
- Vođenje razgovora
 - postavljanje pitanja i bilježenje odgovora
 - prikupljanje dokumentacije
- Završetak razgovora
 - približno 5-10 minuta prije isteka planiranog vremena
 - provjera potpunosti rečenog
 - dogovara se nastavak razgovora (dopunski razgovor) kada:
 - nisu odgovorena sva pitanja ili je razgovor otvorio nova
 - treba proširiti krug sugovornika
 - zahvala na razgovoru

Tehnika intervjuiranja (4)

- **Preispitivanje i pojašnjenje sadržaja (follow-up)**
 - provjera zabilježenih navoda radi upotpunjavanja bilješki
 - telefonom, elektroničkom poštom, novim sastankom
- **Dokumentiranje razgovora**
 - Pisanje zapisnika
 - samostalno pisati zapisnike ("Tko ne razumije, ne može bilježiti.")
 - više analitičara – voditelj razgovora bude zapisničar, a ostali učestvanti primjedbe
 - Sadržaj zapisnika
 - Vrijeme i mjesto, sudionici, zapisničar
 - Sadržaj razgovora (tekst zapisnika)
 - Popis prikupljene dokumentacije
 - zapisnik mora sadržavati ono što je rečeno i slijediti tok razgovora
 - zapisnik ne smije nametati zaključke, jer su oni rezultat analize
- **Autorizacija (ovjera) zapisnika**
 - zapisnik se šalje sugovorniku, koji potvrđuje zapisano (!)
 - po potrebi sugovornik korigira tekst
- **Primjeri:**  \Analiza\
 - Zapisnik-*
 - TragRazgovora

Preporuke za vođenje intervjeta

□ Treba pitati o svemu što se smatra važnim

- ništa nije samo po sebi razumljivo i svima jasno
- ne prepostavljati da se unaprijed zna o čemu se radi

□ Repertoar i vrste pitanja

- Pitanja zatvorenog tipa:
 - Koliko ... obrađujete (u nekom razdoblju)?
 - Na koji način obrađujete ... ?
- Pitanja otvorenog tipa:
 - Što mislite o ... ?
 - Koji su najveći problemi ... ?
- "Probna" pitanja:
 - Zašto ?
 - Možete li navesti primjer za takvu situaciju?
 - Molim detaljnije objašnjenje za ...

Preporuke za vođenje intervjeta (2)

□ Analiza odgovora

- razlučivanje činjenica od mišljenja
- utvrđivanje da li pojedine činjenice odgovaraju drugima
- analiza činjenica koje se ne poklapaju
- provjera odgovora različitih sugovornika na isto pitanje

□ Preporučljivo ponašanje

- iskrenost i nepristranost
 - cilj je naći za korisnika najbolje rješenje, a ne podupirati neki proizvod
- pažnja i jezgrovitost
 - “brzo misli, jasno govori”
- izbjegavanje sugestivnih pitanja
- nemametanje zaključaka
- praćenje reakcija !

□ Grupno intervjuiranje

- izbjegavati i po potrebi nadomjestiti radnim sjednicama
- iznimno provesti, kada se želi utvrditi zajednički interes ili proturječje

Radne sjednice (združeni razvoj)

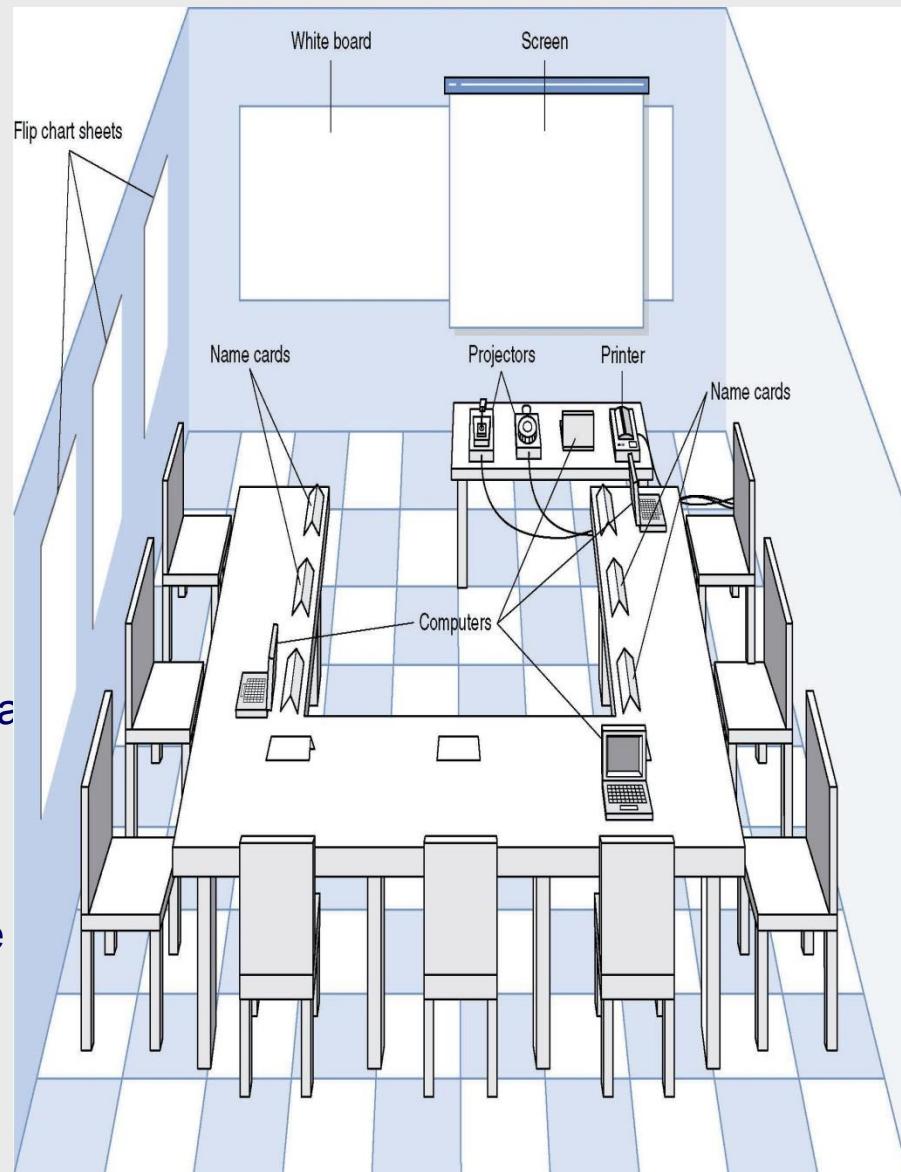
Slika: A.Dennis & B. Haley Wixom,
Systems Analysis and Design,
John Wiley & Sons, 2000

□ Radni sastanci, sjednice (workshop)

- analitičari i korisnici provode SA
- cilj - (zajedničko) pronalaženje rješenja
- poseban prostor i izolacija
- moderator, dnevni red i zapisnici

□ Genijalnost grupe (tzv. brainstorming)

- prikupljanje ideja i info. potreba
- aktivno i kreativno sudjelovanje
- svakog sudionika se stimulira da definira po njemu idealno rješenje
- svaki sudionik iznosi sve čega se sjeti, a odnosi se na problem
- od predloženih rješenja odabire se najbolje prema realnoj izvodljivosti
- postupak je učinkovit tamo gdje postoje korisnici koji dobro poznaju sustav ali teško prihvataju nove ideje



Radne sjednice (2)

□ Prednosti

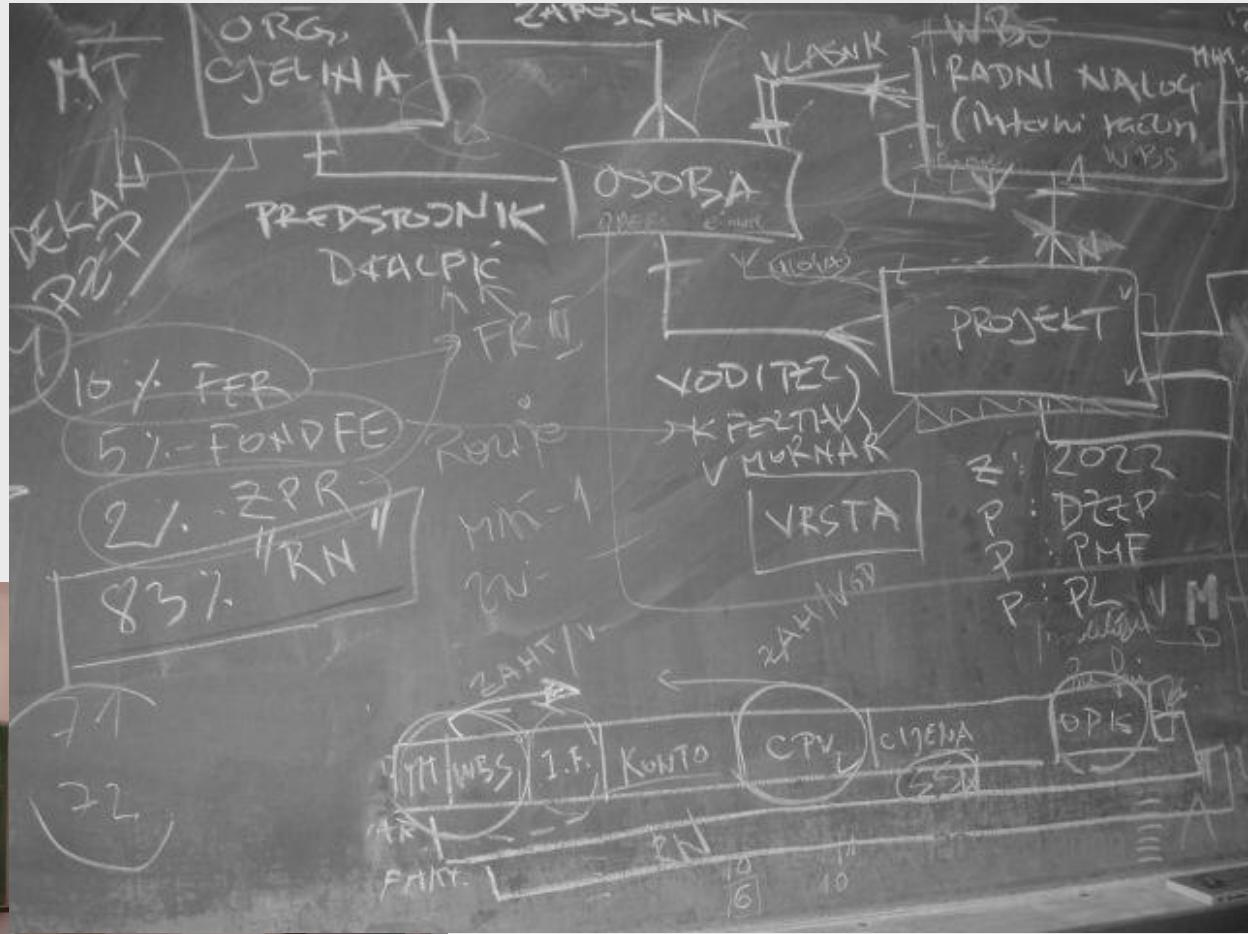
- rješavanje problema važnih za čitavu organizaciju ili veći dio poslovanja
- izbjegavanje specifičnih (egzotičnih) i nejasnih zahtjeva
- preciznije ustanovljavanje dosega projekta
- bolje uočavanje proturječnih zahtjeva

□ Nedostaci

- sadržaj i dinamika
 - pasivnost sudionika
 - “usitnjavanje” razgovora
 - udaljavanje od tema
- trajanje
 - više dana (5-10) u nekoliko tjedana
 - u praksi je teško udovoljiti zbog obveza sudionika
- otpor sjednici i pratećoj izolaciji
 - razmjeran je razini položaja pojedinog sudionika
 - izraženiji među korisnicima koji smatraju da je to informatički posao

□ Primjeri: Plastika, VisokoUčilište

IPIS visokih učilišta



Gigi ? Toni ? Vrt ?

Upitnici i ankete

□ Upitnik (pismeni intervju)

- sadrži pitanja koja se postavljaju tijekom razgovora (okvirno 20)
- može se dostaviti korisniku prije, umjesto ili nakon intervjeta
- nedostaci:
 - ispitanik može prilagoditi (kontrolirati) svoje odgovore
 - teško je procijeniti iskrenost (spontanost) odgovora
 - može obeshrabriti ispitanika
- Primjer:  \Analiza\Podsjetnik-Upitnik

□ Anketa (inventar)

- može se obuhvatiti više ispitanika
- pitanja zatvorenog tipa – standardizacija obrade
- prikladna za popis resursa
- primjer:  \Analiza\Anketa-resursi

□ Intervjuiranje je elastičnije!

Analiza dokumentacije, proučavanje dokumenata

□ Prikuplja se sve do čega se može doći

- Analiza procesa
 - Tipični dokumenti: opis radnih procedura !?, pravilnici i zakoni
- Analiza podataka
 - Tipični dokumenti: obrasci (formulari), izvješća

□ Vrijednost informacija dobivenih analizom (samo) dokumenata je niska

- praksa može odudarati od pravilnika i obrazaca
- treba shvatiti zašto i kada dokumenti nastaju, kako se popunjavaju, koliko su potrebni
- poželjno je da budu reprezentativni, tj. popunjeni na tipičan način, ali ...
 - reprezentativni dokumenti ne ukazuju na izuzetke - rjeđe bilježene podatke
 - stalno bilježenje nekih podataka ne mora značiti da su ti podaci stvarno potrebni
- treba prikupiti više uzoraka iste vrste dokumenta (uzorkovanje) !

□ Primjeri: \Analiza\PrimjeriDokumenata\

Promatranje poslovnog sustava

- **Uvid u poslovne procese promatranjem radnih sredina**
 - promatranje lokacija, kretanja ljudi, koljanja dokumenata te tijek posla
- **Prednosti**
 - analitičar je u stanju realno sagledati poslovni proces
 - učinkovitost, ako se dobro provede
 - pouzdanost prikupljenih informacija
- **Nedostaci**
 - utrošak vremena
 - ometanje i neugoda promatranih osoba
 - mogućnost manipulacije promatrača
 - npr. prikrivanjem uobičajenog kršenja radnih procedura
 - problem određivanja dužine i učestalosti promatranja
 - informacije iz malog broja kratkotrajnih promatranja mogu biti nepouzdane
- **Promatranje je najteža metoda za pronalaženja činjenica.**
- **Primjer: Ministarstvo**

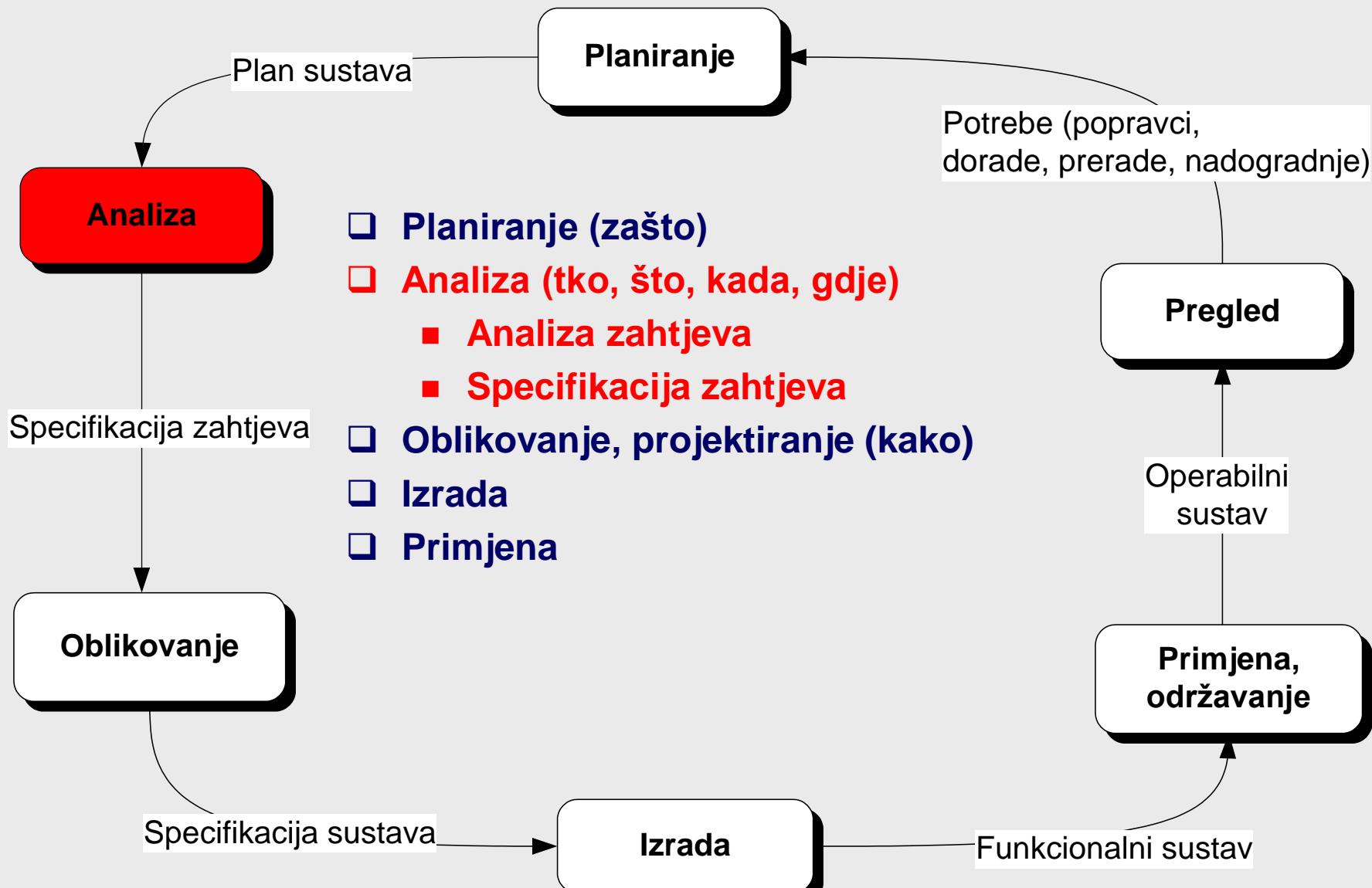
Izbor prikladne tehnike prikupljanja zahtjeva

- Tip informacije – područje primjene
- Dubina informacije – stupanj detaljizacije, mogućnost razumijevanja razloga
- Raspon informacije – širina, raznolikost
- Integracija informacije – mogućnost povezivanja ili provjere informacija dobivenih iz različitih izvora
- Sudjelovanje korisnika – stupanj uključenosti, angažiranost
- Trošak – organizacije, provedbe

	Intervju	Ankete	Analiza dokumenata	Udruženi razvoj	Promatranje
Tip informacije	as-is, pooboljšanja, to-be	as-is, pooboljšanja	as-is	as-is, pooboljšanja, to-be	as-is
Dubina informacije	veliki	srednji	mali	veliki	mali
Raspon informacije	mali	veliki	veliki	srednji	mali
Integracija informacije	mali	mali	mali	veliki	mali
Sudjelovanje korisnika	srednje	mali	mali	veliki	mali
Trošak	srednje	mali	mali	srednji do velik	mali do srednjeg

Analiza sustava

Kontekst analize sustava



Analiza sustava

□ Analiza sustava, sustavska analiza

- Analiza sustava je (1) razmatranje i planiranje sustava i projekta,
(2) proučavanje i analiza postojećeg poslovnog i informacijskog sustava te
(3) definiranje poslovnih zahtjeva i prioriteta novog ili poboljšanog
postojećeg sustava. [Whitten et. al, 2000]

□ Aktivnosti analize

- detaljna analiza postojećeg sustava te utvrđivanje potreba i zahtjeva
 - Kako radi postojeći sustav?
 - Na koji način ljudi koriste sustav da bi obavili svoj posao?
 - Koji su problemi pri korištenju aplikacija ili uslijed nedostatka aplikacija?
- detaljna specifikacija zahtjeva na IS
 - Koji su podaci potrebni? Tko ih treba? Kada? Od koga? Tko ih stvara?
 - Koji su izlazni podaci? Kakav im je oblik? Koji su izvori podataka?
 - Na koji način se podaci prikupljaju i objedinjuju?
- daljnja razrada granica projekta

□ Primjeri: \Analiza\ ProtokDokumenata, RazmjenaPodataka

Analiza sustava (2)

□ Pozadinska analiza

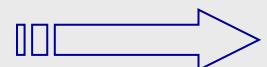
- Važno je shvatiti strukturu organizacije, tko u njoj radi, tko je kome podčinjen, kako surađuju različiti odjeli, itd.

□ Modeliranje sustava

- pozadinska analiza - shema organizacijske strukture osobe i grupe koje obavljaju neki dio posla → Modeliranje funkcija
- detalji poslovanja → Modeliranje procesa, modeliranje podataka

□ Svrha, cilj i dubina analize

- automatizacija poslovnih procesa,
- poboljšanje poslovnih procesa i
- preoblikovanje poslovnih procesa



Automatizacija poslovnih procesa

❑ Business Process Automation (BPA)

- Povećanje učinkovitosti korisnika primjenom računalne tehnologije

❑ Tehnike

- Analiza problema (problem analysis)
 - Otkrivanje problema sustava i predlaganje rješenja uz pomoć korisnika
 - Poboljšanja su mala i inkrementalna – učinkovitost, lakoća korištenja
- Analiza uzroka (root cause analysis)
 - Usmjereno na probleme (simptome), a ne na rješenja
 - Problemima se dodjeljuju prioriteti te traže svi mogući uzroci
 - Uzroci se analiziraju sve dok se ne utvrdi pravi uzrok
 - Prednost imaju uzroci koji izazivaju više problema

Primjer: automatizacija poslovnog procesa

□ Primjer: Izrada putnih naloga

- izrađuje se 2700 putnih naloga godišnje
 - nalog za putovanje odobravaju voditelj, predstojnik te dekan
 - urudžbiranje (2 djelatnice) i knjiženje te isplata predujma (1 djelatnica)
 - po povratku putnik popuni zapisnik, nalog se obračunava (1 djelatnica) te se na blagajni (1 djelatnica) isplaćuje razlika / plaća povrat
- u pojedinim razdobljima broj putovanja značajno je povećan
 - gomilanje naloga u pisarnici
 - česte gužve na blagajni

□ Rješenja:

- kreiranje i ažuriranje naloga na mjestu nastanka (podnositelj)
- automatizacija postupka odobravanja
- isplata predujmova putem tekućeg računa

Poboljšanje poslovnih procesa

❑ Business Process Improvement (BPI)

- manje promjene radi boljeg iskorištenja tehnologije i učinkovitosti/djelotvornosti

❑ Tehnike

- Analiza trajanja (duration analysis)
 - Analiza vremena po svim koracima procesa
 - Integracija i paralelizacija procesa koje obavlja više ljudi
 - Integracija – smanjenje glavnog procesa da manje ljudi radi na ulazima
 - Paralelizacija – povećanje istovremenosti koraka
- Koštanje poslovnih procesa (activity-based costing)
 - Analiza troškova po svim koracima procesa
 - Određivanje direktnog troška rada i resursa za svaki ulaz
- Baždarenje i usporedba (Informal benchmarking)
 - Proučavanje poslovanja drugih organizacija
 - Predstavljanje novih ideja koje mogu imati dodatnu vrijednost
 - Uobičajeno za procese kod kojih se obavlja interakcija s korisnicima

Primjer poboljšanja: integracija poslovnog procesa

□ Primjer: obračun nastave na poslijediplomskom studiju

- Stanje: (velik broj sitnih transakcija ugovaranja i isplate, uz prijepis)
 - po obavljenoj nastavi Studentska dostavlja evidenciju predavanja
 - izvođači zasebno potpisuju autorski ugovor u Pravnoj službi
 - nezavisni ugovori za vođenje (mentorstvo) i obrane doktorata
 - sve isplate temeljem ugovora provodi Računovodstvo
- Poboljšanje:
 - jednom godišnje (za nastupajuću ak. godinu) izrađuje se autorski ugovor s općim uvjetima - uključeni svi predvidivi izvođači
 - primjerak potписанog ugovora dostavlja se u Studentsku službu
 - naknadno angažirani izvođači potpisuju dodatak ugovoru
 - Studentska služba po obavljenom poslu evidentira obavljeno te generira nalog za isplatu s pozivom na broj autorskog ugovora

Primjer poboljšanja: paralelizacija poslovnog procesa

□ Primjer: poboljšanje obrade putnih naloga

- izvješće s putovanja temeljem predloška putnog naloga s funkcijom za urudžbiranje – urudžbiranje se obavlja web servisom koji uz attribute pohranjuje i dokument (BLOB)
- potpisani primjerak dostavlja se u pisarnicu na skeniranje
- nalog se može paralelno knjižiti jer je u međuvremenu urudžbiran

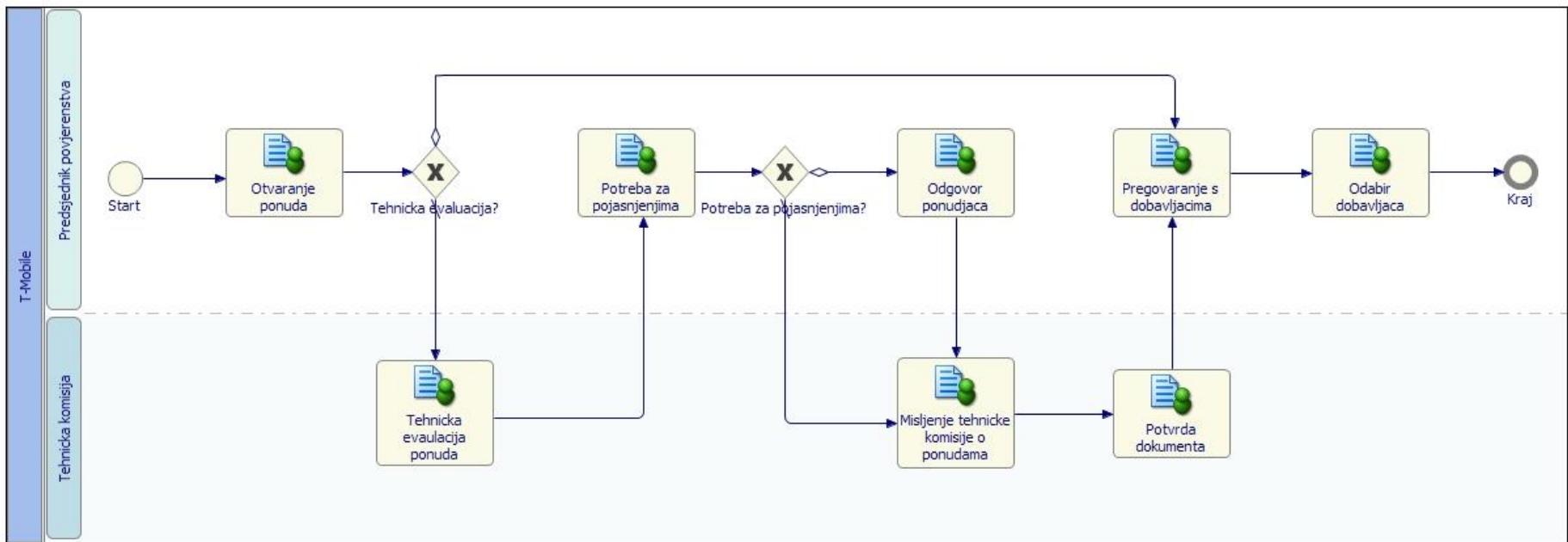
Preustroj, preoblikovanje poslovnih procesa

- **Business Process Reengineering (BPR)**
- **Business Process Redesign (BPR)**
 - radikalni redizajn analizom posljedica, procjenom alternativnih tehnologija, ukidanjem ili zamjenom aktivnosti, analizom troškova-koristi i analizom rizika
- **Tehnike**
 - Analiza posljedica (outcome analysis)
 - Razumijevanje rezultata koji predstavljaju vrijednost za korisnika
 - procjena što bi se moglo ponuditi krajnjim korisnicima
 - Analiza tehnologije (technology analysis)
 - Analitičari i voditelji razvijaju listu važnih i zanimljivih tehnologija
 - donose odluke o koristi i primjeni u poslovnim procesima
 - Uklanjanje aktivnosti (activity elimination)
 - Analitičari i voditelji donose odluke za svaku aktivnost unutar poslovnog procesa o mogućnosti njenog uklanjanja i posljedicama

Primjer: heuristički preustroj procesa

❑ Kreće se od postojećeg stanja poslovnog procesa

- npr. potproces unutar poslovnog procesa nabave
- proces se sastoji od otvaranja ponuda, tehničke evaluacije, slanja pitanja ponuđačima, definiranja mišljenja tehničke komisije, ... i odabira dobavljača.
- predsjednik povjerenstva nabave (15 ljudi) i tehnička komisija (400 ljudi).



Primjer: heuristički preustroj procesa (2)

- simulira se izvođenje postojećeg procesa,
- vrši se preoblikovanje
 - npr. prepostaviti će se da nije potrebno tražiti dodatna pojašnjenja od ponuđača, jer su zahtjevi dobro definirani
- te se simulira izvođenje preoblikovanog procesa.



Stanje poslovnog procesa	Ukupno prosječno vrijeme trajanja procesa	Ukupan prosječni broj resursa bez posla
Prije preoblikovanja	213.849,88 min	45,980225
Nakon preoblikovanja	198.256,66 min	46,1718

Izbor prikladne tehnike analize

	BPA	BPI	BPR
Poslovna vrijednost	mala do umjerena	umjerena	velika
Trošak projekta	mali	mali do umjeren	veliki
Doseg analize	ograničen	ograničen do umjeren	vrlo širok
Rizik	mali do umjeren	mali do umjeren	vrlo veliki

Potencijalna poslovna vrijednosti (potential business value)

- BPA ne mijenja procese, poboljšava učinkovitost
- BPI poboljšava i učinkovitost i djelotvornost
- BPR radikalno poboljšava prirodu poslovanja

Trošak projekta (project cost)

- BPA ima ograničen doseg
- BPI ovisi o dosegu projekta
- BPR skup zbog utroška vremena i napora preoblikovanja

Doseg analize (breadth of analysis)

- BPA obično za pojedinačni proces
- BPI nekoliko procesa
- BPR vrlo širok doseg

Rizik pogreške (risk of failure)

- BPA mali do umjeren rizik jer je sustav jasan
- BPI mali do umjeren rizik jer je sustav jasan
- BPR slabo predvidiv i podložan riziku te se provodi uz odobrenje



Preporuke za analizu

Slika: Awad, 1985;
preveo Anonymous
(WWW)

□ Ispravni pristup

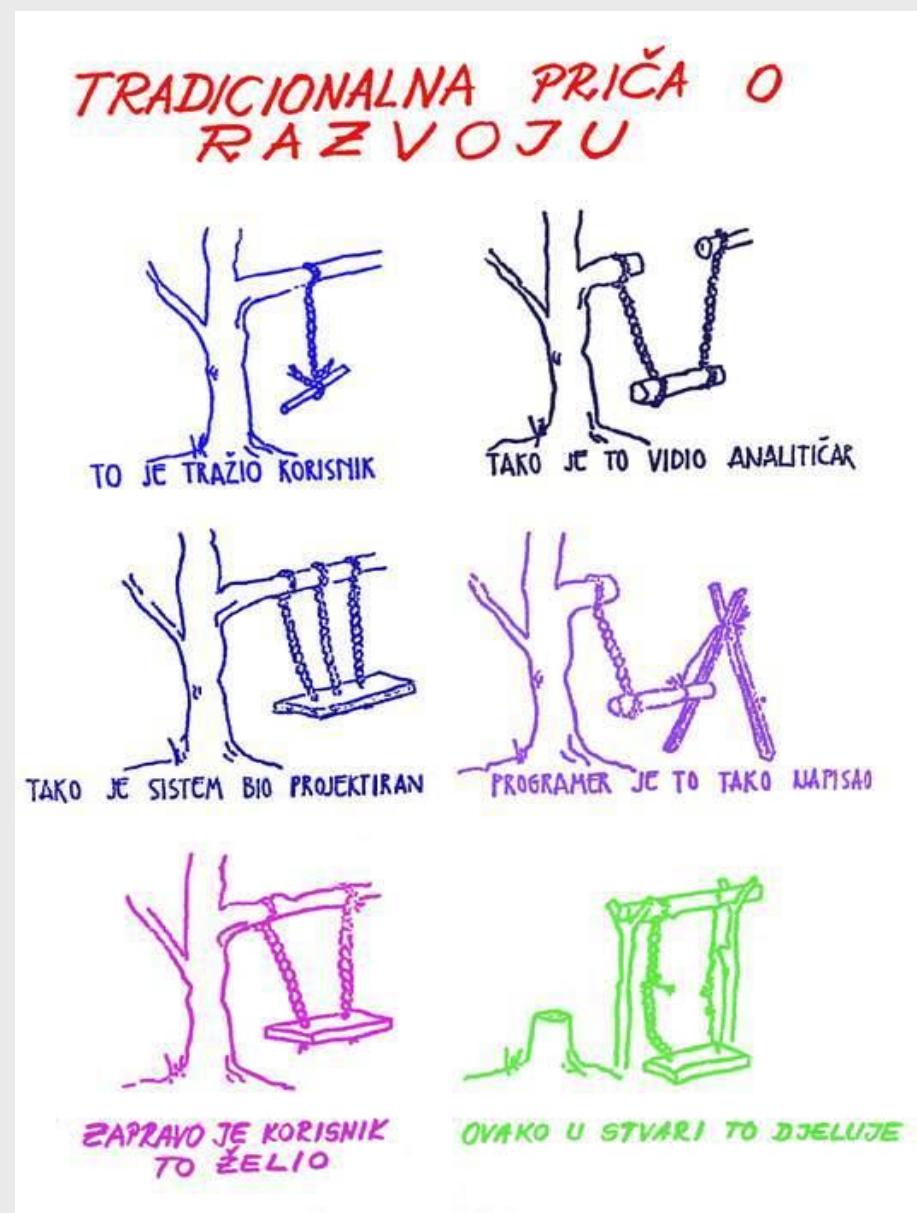
- analiza mora biti prilagođena korisniku
- ključno je razumjeti suštinu organizacije i način poslovanja
- Primjer: \Analiza\PreporukeZaAnalizu

□ korisnik želi <> korisnik treba

- umješnost je uvjeriti ga da je „sve sam smislio“
- da se priča o razvoju ne bi ponavljala - modeliranje

□ Paraliza analize

- analysis paralysis, paralysis of analysis
- vječito modeliranje „dok ne bude dobro“
- ne pretjerivati s modeliranjem postojećeg sustava tj. nečeg što će biti izmijenjeno ili zamijenjeno



Inženjerstvo zahtjeva

Requirements engineering

Zahtjevi

- **IEEE standard definira zahtjeve kao [IEEE Std 610.12-1990]**
 - (1) Uvjet ili sposobnost koje korisnik treba da bi riješio problem ili ostvario cilj.
 - (2) Uvjet ili sposobnost koji mora posjedovati sustav ili komponenta sustava da bi zadovoljila ugovor, standard, specifikacije ili neki drugi ugovoreni dokument.
 - (3) Dokumentiranu reprezentaciju uvjeta ili mogućnosti definiranih pod (1) ili (2).
- **Zahtjevi ne smiju sadržavati detalje dizajna ili implementacije**
 - ali ako su nepotpuni onemogućuju planiranje projekta i praćenje promjena
- **Ključni problemi [Leffingwell, 1997]**
 - 40-60% pogrešaka projekta uzrokovano je pogreškama postavljanja zahtjeva
 - posljedica - "neispunjena očekivanja" (expectation gap)
 - Naknadne prepravke mogu predstavljati do 40% troškova razvoja,
 - od čega je 70-85% pripisivo pogrešnim zahtjevima

Vrste zahtjeva

□ Poslovni zahtjevi (zašto)

- ciljevi organizacije ili zahtjevi rukovodstva, poslovni problemi ili potrebe
- opisani u viziji i dosegu projekta
 - primjer: poslovna potreba, „Pridobivanje novih kupaca“
 - primjer: poslovni zahtjev, "Omogućiti Internet prodaju"

□ Korisnički zahtjevi (zahtjevi krajnjih korisnika)

- opisuju zadatke koje korisnik mora moći obaviti služeći se aplikacijama
- sadržani u opisima slučajeva korištenja, tj. opisima scenarija rada

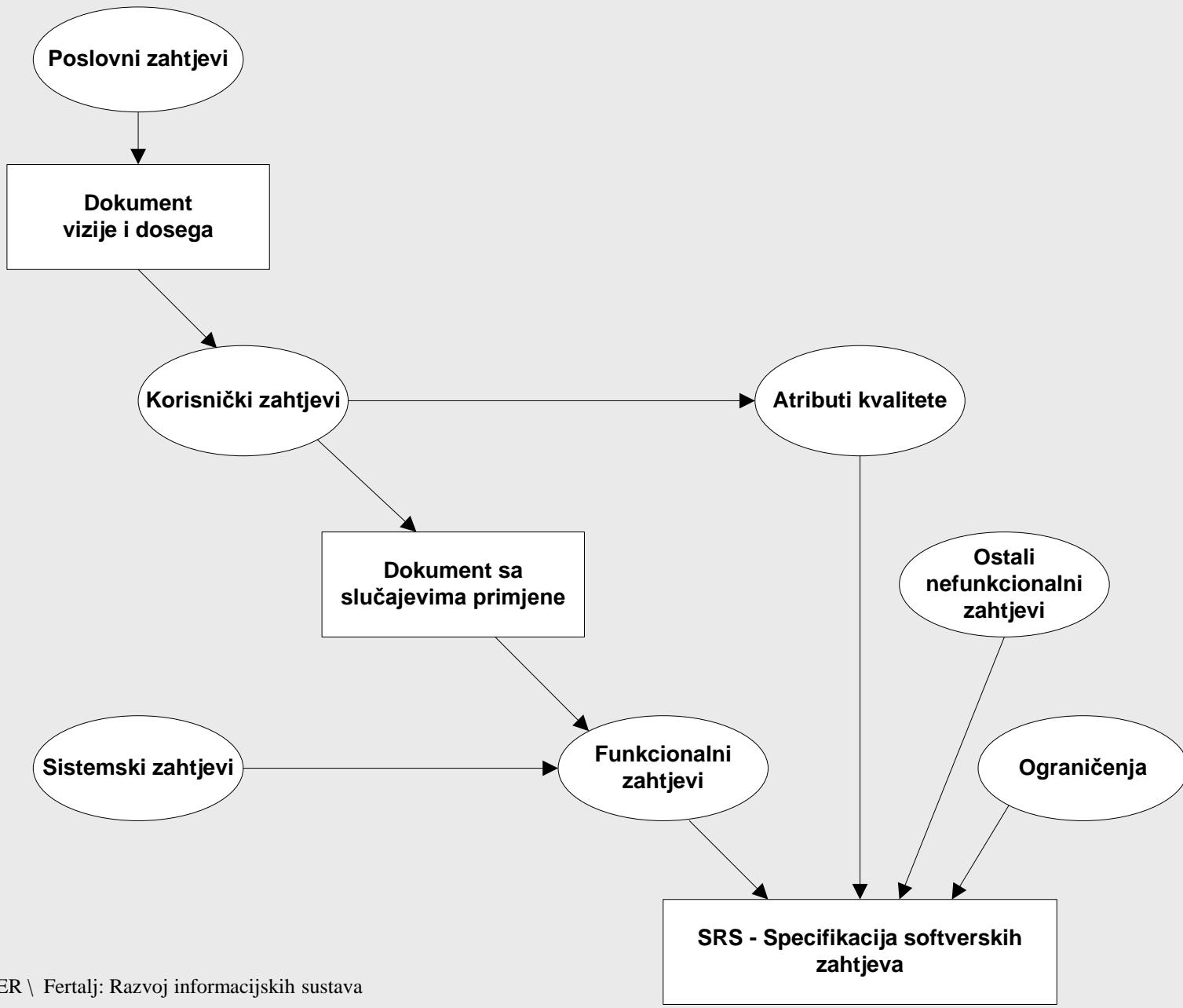
□ Funkcionalni zahtjevi (što)

- očekivana softverska funkcionalnost potpore korisničkih zadataka
- mogućnost programa (feature) – skup funkcionalnih zahtjeva za ispunjenje poslovnih

□ Nefunkcionalni zahtjevi (kako ili kako dobro)

- standardi, pravila i ugovori, opisi vanjskih sučelja, zahtjevi na performanse, ograničenja na dizajn i implementaciju, te svojstva kvalitete

Kontekst postavljanja zahtjeva



Primjeri poslovnih zahtjeva

□ Očekivana novčana ušteda

- prava na subvencioniranu prehranu može koristiti samo student koji ih je stekao te ih može koristiti samo u svrhu prehrane

□ Sustav mora onemogućiti:

- korištenje subvencije od strane osoba koje nemaju na to pravo
- zaradu ilegalnih posrednika
- korištenje subvencije za druge svrhe osim prehrane
- naplatu usluga koje nisu pružene

□ Idealni slučaj - zahtjevi naručitelja odgovaraju poslovnim ciljevima!

Primjeri zahtjeva krajnjih korisnika

- Prehrana kod bilo kojeg pružatelja usluga**
 - Novi sustav mora omogućiti da student ostvaruje svoje pravo kod bilo kojeg pružatelja usluge subvencionirane prehrane. Dosadašnja praksa je bila da svaki pružatelj usluga izdaje svoje bonove ...
- Ukinuti plaćanje unaprijed**
 - Treba izbjegići bilo kakvo plaćanje od strane studenata za potrebe ostvarivanja prava, a posebice unaprijed.
- Ukloniti nepotrebne postupke za ostvarivanje prava**
 - Kad se studentu jednom zavedu prava na maticnoj ustanovi, nisu potrebna nikakva daljnja dokazivanja za ostvarivanje prava kod pružatelja usluga.
- Smanjiti rizik gubitka ostvarenih prava**
 - Sustav mora onemogućiti zloporabu stečenih prava.
- Lakše ostvarivanje ostalih prava iz studentskog standarda**
 - Npr. javni prijevoz po povlaštenoj cijeni, kazališta, kina, smještaj u studentskim domovima, student-servis, itd.

Primjer: sistemski zahtjevi

□ Primjer: PISFER-Podprojekt

- Sustav treba biti integriran, tako da se sve informacije unose na mjestu nastanka
- Mogućnost da sve organizacijske jedinice imaju pravo pregleda, ažuriranja i unosa podataka uz određene dozvole
- Omogućiti kontinuirani pristup podacima (ne samo krajem mjeseca)
- Svi podaci bi trebali biti u digitalnom obliku
- Jednostavno i intuitivno korištenje sustava
- Korištenje jedinstvene baze podataka za sve službe

Primjer: funkcionalni zahtjevi

□ Primjer: storno dokumenta

- Radi se storno ulaznih dokumenata, unosom odgovarajućeg dokumenta. Dokument storna „nasljeđuje“ konto dokumenta koji bude storniran.
 - Stavka storna može se obrisati
 - Storno storna nije moguć
 - Storno nužno ne poništava cijeli iznos, nego ga korigira za iznos storna, ...
- Ne obavlja se storno uplate.
- Uz storno omogućiti i ispravak dokumenta (koji ne utječe na proknjižene stavke).
 - Ako je dokument uplata iznos uplate uvijek mora odgovarati sumi vezanih iznosa po računima pa i nakon ispravka, s tim da je u fazi ispravka moguće izmijeniti specifikaciju računa i pripadajućih iznosa.

□ Primjer: izvješće otvorenih stavaka (referencira primjere izvješća)

- IOSi za tekuću poslovnu godinu (IOSI.doc)
- Sumarni IOSi (SumarniOSI.doc)
 - Kartica dobavljača (KarticaTekuca.doc)
 - Kartica dobavljača s uključenom arhivom i kontinuiranim nastavkom u tekuću poslovnu godinu (KarticaArhiva.doc)

Primjer: nefunkcionalni zahtjevi

□ Viševalutni rad

- Devizni izvod poslovne banke sadrži iznose po tečaju poslovne banke.
- Za devizne uplate evidentirat će se uz kunski iznos i devizni iznos te valuta.
- Sva izvješća osim Kartica moraju sadržavati mogućnost konverzije primarne valute u zadanu valutu po srednjem tečaju NBH na datum knjiženja.

□ Praćenje promjena nad podacima

- svaki zapis sadrži identifikator korisnika i datum unosa/promjene

□ Izvoz podataka

- Omogućiti izvoz izvješća u Excel formatu

□ Primjeri: \Analiza\SistemskaZahtjevi.xls

- Operativni, performanse, sigurnost, kulturni i politički

Zahtjevi na kvalitetu programske podrške

- **Jedna podjela može biti kao što je prikazano tablicom:**
 - Odabire se podskup kvalitativnih svojstva važnih za konkretni projekt.
- **Pojedini nefunkcionalni zahtjevi su međusobno proturječni.**
 - Proturječnost se razrješava postavljanjem prioriteta

Svojstva važnija korisnicima	Svojstva važnija razvojnicima
Dostupnost	Lakoća održavanja
Učinkovitost	Prenosivost
Prilagodljivost	Ponovna upotrebljivost
Integritet	Podložnost testiranju
Interoperabilnost	
Pouzdanost	
Robustnost	
Upotrebljivost	

Zahtjevi na kvalitetu programske podrške (1)

Dostupnost (availability)

- postotak predviđenog vremena tijekom kojeg sustav treba biti funkcionalan
- omjer MTTF / (MTTF+MTTR), to jest srednjeg vremena do kvara i sume srednjeg vremena do kvara i srednjeg vremena do oporavka od kvara
- Primjer: "Sustav će biti barem 99.5% dostupan radnim danima od 6 do 24 i barem 99.95 posto dostupan između 16 i 18 sati."

Učinkovitost (efficiency)

- Stupanj iskorištenja resursa (procesor, disk, memorija)
- Primjer: „70% procesorskog kapaciteta i 85% memorije sustava će biti raspoloživo za vrijeme maksimalnog opterećenja sustava.“
- Potrebno je razmotriti minimalne hardverske konfiguracije

Prilagodljivost (adaptability)

- Lakoća dodavanja novih mogućnosti
- Važno za proizvode koji se rade inkrementalno

Zahtjevi na kvalitetu programske podrške (2)

□ Integritet (integrity)

- Integritet (ili sigurnost) – neovlašteni pristup, virusi, tajnost podataka, ...
- Primjeri: role-based security, single sign-on (SSO), ...

□ Interoperabilnost (interoperability)

- razmjena podataka ili usluga s drugim sustavima.
- koje druge aplikacije se namjerava koristiti i koji podaci će se razmjenjivati.
- Primjeri: „uvoz iz Access, izvoz u Excel“

□ Pouzdanost (reliability)

- vjerojatnost da će softver u nekom razdoblju raditi bez pogreške
- postotak ispravno izvedenih operacija, trajanje rada bez pogrešaka ili učestalost pogrešaka.
- proizvod koji zadovolji zahtjeve za pouzdanošću je isporučiv unatoč defektima
- Primjer: „Maksimalni očekivani mjesecni broj pogrešaka nakon zakrpe je 32“

Zahtjevi na kvalitetu programske podrške (3)

□ Robustnost (robustness)

- stupanj do kojeg sustav nastavlja ispravno funkcionirati u slučaju pogrešnih podataka, defekata u komponentama ili nepredviđenim operativnim uvjetima
- primjer: „pri učitavanju iz datoteke sustav će za redak s pogreškom prijaviti pogrešku i broj retka te pitati korisnika da li da nastavi od sljedećeg retka ili da otkaže učitavanje“

□ Upotrebljivost (user friendliness)

- mjera napora za pripremu, obradu ili tumačenje podataka, to jest lakoću korištenja
- krivulja učenja, korištenje početnika / iskusnih korisnika
- sukladnost standardima ili uzoru: "Sve kratice na u izborniku *File* će biti iste kao ..."

□ Lakoća održavanja (maintainability)

- mjera lakoće popravka pogreške ili izmjene
- prosječno vrijeme da se riješi problem i postotak obavljenih popravaka
- primjer: "Izvješće će biti izmijenjeno u roku tjedan dana od zaprimanja obrazaca"

Zahtjevi na kvalitetu programske podrške (4)

□ Prenosivost (portabilnost)

- napor prijenosa iz jedne operativne okoline u drugu
- projektantski principi slični iskoristivosti
- dijelovi koji moraju biti prenosivi te ciljane okoline

□ Ponovna upotrebljivost (reusability)

- mjera upotrebe u drugim aplikacijama.
- razvoj ponovno iskoristive komponente znatno skuplji
- softver mora biti modularan, dokumentiran, donekle općenit

□ Podložnost testiranju

- lakoća provjere softverske komponente
- ključna kod proizvoda sa složenim logikom obrade
- važno kod proizvoda koji će se često mijenjati - regresijsko testiranje

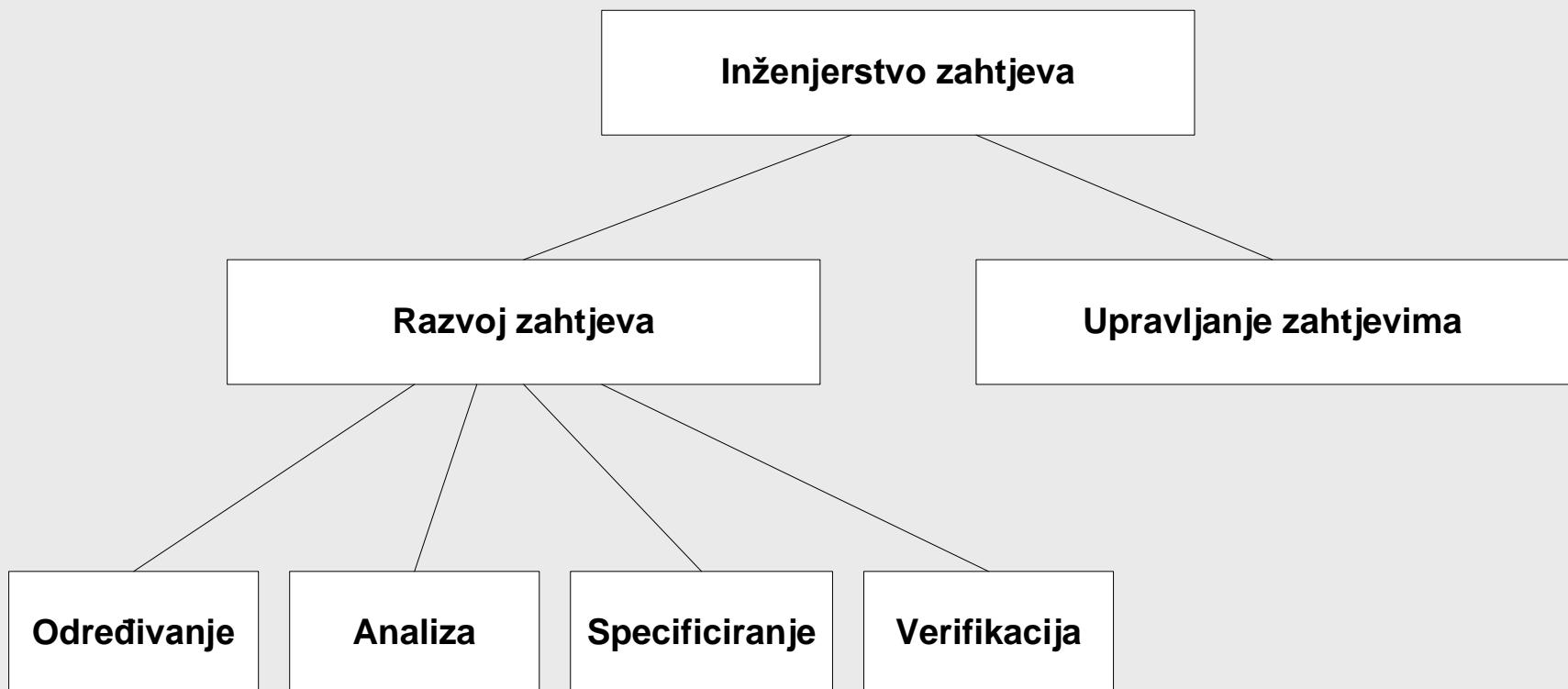
Među zavisnost zahtjeva

	Dostupnost	Učinkovitost	Prilagodljivost	Integritet	Interoperabilnost	Lakoća održavanja	Prenosivost	Pouzdanost	Ponovna isupotrebljivost	Robustnost	Podložnost testiranju	Upotrebljivost
Dostupnost												
Učinkovitost			-		-			-		-		-
Prilagodljivost		-		-		+		+				
Integritet		-			-				-			-
Interoperabilnost		-	+	-			+					
Lakoća održavanja	+	-	+				+	+				
Prenosivost		-	+		+	-			+			-
Pouzdanost	+	-	+			+				+		+
Ponovna isupotrebljivost		-	+	-	+	+	+	-				+
Robustnost	+	-						+				+
Podložnost testiranju	+	-	+			+		+				+
Upotrebljivost		-							+	-		

Inženjerstvo zahtjeva

□ Inženjerstvo zahtjeva (requirements engineering)

- razvoj zahtjeva + upravljanje zahtjevima.
- razvoj zahtjeva: određivanje (iznalaženje) zahtjeva (elicitation), analiza, specificiranje (dokumentiranje) i verifikacija



Razvoj zahtjeva i upravljanje zahtjevima

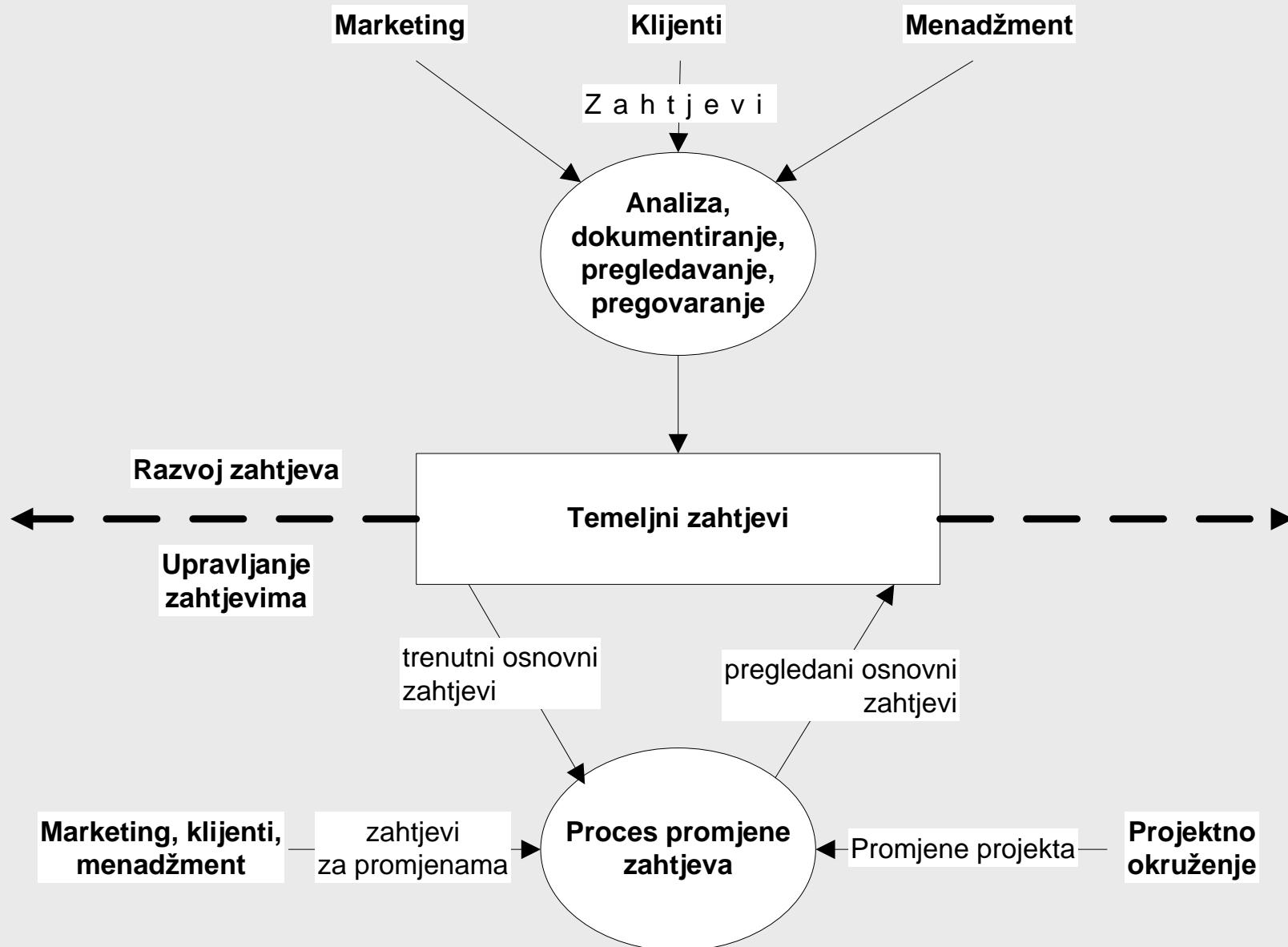
□ Razvoj zahtjeva

- Razredi korisnika ("najbolji", ključni)
- Poslovne potrebe, zadaci i ciljevi
- Analiza informacija, kategorizacija
- Razdioba s razine sustava na podsustave i udio u komponentama
- Razumijevanje relativne važnosti (težine) svojstava kvalitete
- Određivanje prioriteta ugradnje
- Dokumentiranje, modeliranje zaht.
- Provjera specifikacije zahtjeva

□ Upravljanje zahtjevima

- Rokovi prihvatanja zahtjeva
- Pregled promjena zahtjeva, procjena utjecaja, odluka o (ne)prihvatanju
- Kontrolirana ugradnja promjena
- Ažuriranje plana projekta
- Pregovori o rokovima isporuke
- Praćenje i povezivanje zahtjeva s izvornim kodom i primjerima za test
- Praćenje statusa aktivnosti i promjena zahtjeva tijekom projekta

Razvoj zahtjeva naspram upravljanja zahtjevima



Razvoj zahtjeva

Određivanje – analiza – specifikacija – verifikacija

* **specificiranje (dokumentiranje) je izdvojeno u zasebnu cjelinu
i opisano malo kasnije**

Određivanje zahtjeva (kako razlučiti)

□ Poslovni zahtjevi

- Sve što opisuje financijski ili drugi poslovni probitak za korisnike ili organizaciju, je najvjerojatnije poslovni zahtjev

□ Slučajevi korištenja ili scenariji

- Općenite izjave o poslovnim zadacima korisnika – slučajevi korištenja
- Specifični opisi zadataka - korisnički scenariji
- Specifične zadatke treba generalizirati u općenite slučajeve korištenja

□ Poslovna pravila

- izjava da neku aktivnost mogu obavljati samo pojedine osobe ili uloge, ili da se aktivnost obavlja pod određenim uvjetima
- operativni principi poslovnih procesa - nisu funkcionalni zahtjevi

Određivanje zahtjeva (2)

❑ Funkcionalni zahtjevi

- Izjava „Korisnik mora moći ...” ili „Sustav treba ... neko ponašanje”
- opisuju vidljivo ponašanje sustava i definiraju što će sustav raditi.
- **funkcionalni zahtjevi ponekad predstavljaju zastarjele/neučinkovite procese!**
 - treba biti jasno zašto sustav nešto „mora”

❑ Atributi kvalitete

- vrsta nefunkcionalnih zahtjeva
- Izjave kako dobro sustav obavlja funkciju
- karakteristike: „brzinu, jednostavnost, intuitivnost, ...” treba razjasniti s korisnicima

❑ Zahtjevi vanjskih sučelja

- veze između sustava i vanjskog svijeta
- mehanizmi komunikacije za korisnike, hardver i druge sustave

Određivanje zahtjeva (3)

□ Ograničenja

- Uvjeti koji ograničavaju izbor rješenja
- Spadaju u nefunkcionalne zahtjeve
- Neka ograničenja mogu pomoći u zadovoljavaju atributa kvalitete
 - Primjer: prenosivost korištenjem samo standardnih naredbi

□ Definicije podataka

- opis formata, dozvoljenih vrijednosti, pretpostavljenih vrijednosti ili struktura

□ Ideje o rješenju

- Ako korisnik opisuje način interakcije
 - npr. „Kad kliknem, otvor se menu, ... a tu mi ugradi NN” (T.Nikolić)
- Prijedlog rješenja nije zahtjev!
- Usmjeriti se na ono što je potrebno obaviti, a ne na način realizacije
- Istražiti zašto korisnik predlaže određenu ugradnju – razumijevanje potrebe



Najčešći problemi pri određivanju zahtjeva

□ Taktika kuhinjskog sudopera

- korisnik navodi (preko)brojne potrebe: permutacije izvještaja, formi, sortiranja, ...
- pristup uzrokovani pomanjkanjem iskustva korisnika koji ne zna razlučiti bitno
- pristup: analiza, "normalizacija", prilagodljiv GUI / generika (dizajn)

□ Taktika dimne zavjese

- korisnik traži više mogućnosti nego što treba
- skriveni interes: postizanje bolje nagodbe ("neka se ima")
- pristup: redukcija na realne i izvedive zahtjeve, postavljanje prioriteta

□ Taktika "Treba mi isto ali bolje"

- korisniku nedostaje volja ili znanje, a ponekad oboje
- pristup: ustrajnost, iznošenje primjera, prototipiranje

□ Korisnik je sklon prešutjeti izuzetke, koji su bitni (nužni !!!) za uspješnu realizaciju, a obično zahtijevaju i najviše napora tijekom ugradnje.

- "To je naša jedina procedura ... (osim kada ...)" - pravilo 80/20

□ Ne izjednačavati "tako se uvijek radi" s "tako treba raditi"!

- softversko "betoniranje" loših navika – "politika kuće" (više riječi kasnije)

Analiza zahtjeva

□ pročišćavanje, proučavanje i preispitivanje zahtjeva

- da budemo sigurni da svi sudionici razumiju što zahtjevi znače
- ujedno služi provjeri zahtjeva

□ Izrada dijagrama konteksta sustava

- jednostavni model – granice i sučelja + tok informacija i materijala

□ Izrada prototipova korisničkog sučelja

- model oponašanja kao sredstvo razjašnjenja zahtjeva

□ Analiza ostvarivosti zahtjeva

- procjena troškova, performansi, rizika, konflikata prema ostalim zahtjevima

□ Postavljanje prioriteta

- relativni prioritet ugradnje za slučaj korištenja, svojstvo proizvoda ili drugi zahtjev

□ Modeliranje zahtjeva

- izrada modela strukture, ponašanja i stanja

□ Izrada rječnika podataka

- središnji repozitorij za definiranje struktura podataka i zajedničku terminologiju

Analiza i modeliranje sustava

- **Na temelju izjava korisnika i prikupljene dokumentacije modeliraju se pojedine komponente sustava (procesi, podaci, događaji)**
 - preslikavanja imenica i glagola u elemente modela (n puta u nastavku)

Tip riječi	Primjer	Komponente analitičkog modela
Imenica	•Ljudi, organizacije, softverski sustavi, podatkovne jedinice, ili postojeći objekti	•Terminatori ili spremnici podataka (DFD) •Entiteti ili njihovi atributi (ERD) •Razredi ili njihovi atributi (dijagram razreda)
Glagol	•Akcije, ono što korisnik može poduzeti, ili događaji koji se mogu dogoditi	•Procesi(DFD) •Odnosi (ERD) •Prijelazi (iz stanja u stanje) (STD) •Metode razreda (dijagram razreda)

"Kemičar ili član osoblja kemijskog laboratorija može podnijeti zahtjev za jednom ili više kemikalija. Zahtjev može biti udovoljen ili dostavom pakiranja kemikalije koja se već nalazila na zalihi kemijskog laboratorija ili upućivanjem narudžbe za novim pakiranjem kemikalije od vanjskog dobavljača. Osoba koja upućuje zahtjev mora imati mogućnost pretraživanja kataloga kemikalija vanjskog dobavljača dok sastavlja narudžbu. Sustav mora pratiti status svakog zahtjeva za kemikalijama od trenutka kad je ispunjen do trenutka kad je udovoljen ili otkazan. Također, mora pratiti povijest svakog pakiranja kemikalija od trena kad stigne u kompaniju do trenutka kad je potpuno upotrijebljen ili odbačen."

Postavljanje prioriteta

□ Nužno svojstvo (must have) - Ključno imati ?

- Sustav koji nema definirane nužne zahtjeve ne može ispuniti svoju svrhu
- Postoji tendencija da se previše zahtjeva proglaši nužnim!
- Testirati svaki zahtjev i nastojati ga rangirati
 - Ako se zahtjev može rangirati onda nije obvezan!

□ Poželjno svojstvo (should have) – Ispuniti do kraja projekta ?

- Međuverzije sustava mogu biti funkcionalne bez tih zahtjeva
- Poželjni zahtjevi mogu i trebaju biti rangirani.

□ Neobvezno svojstvo (could have) – Moglo bi se ispuniti

- "bilo bi lijepo, ali se može bez"
- nisu pravi zahtjevi, ali mogu biti rangirani i odrediti kvalitetu

□ Nepotrebna svojstva (won't have)

Primjeri rangiranja prioriteta

oznaka	značenje	referenca
visok	<i>mission-critical</i> zahtjevi, potrebni za sljedeće izdanje (<i>release</i>)	Wiegers 99
srednji	podrška sistemskim operacijama; zahtijevano, ali može se odgoditi za sljedeće izdanje	
nizak	poboljšanje funkcionalnosti ili kvalitete	
neophodan	proizvod neprihvatljiv ukoliko ovi zahtjevi nisu ispunjeni	IEEE 98
kondicionalan	može poboljšati proizvod, ali proizvod neće biti neprihvatljiv bez toga	
opcionalan	klasa funkcija koje bi mogle biti vrijedne uključivanja, ali možda i ne	
3	mora biti savršeno implementirano	Kovitz 99
2	treba raditi, no ne spektakularno dobro	
1	može sadržavati pogreške (<i>bugove</i>)	

Fertalj: 0, 1, 2, 3, 5, 9

Verifikacija zahtjeva

- ovjera da su zahtjevi precizni, potpuni i da zadovoljavaju traženo

□ Provjera dokumentacije sa zahtjevima

- jedna od najkorisnijih softverskih tehnika
- mikro ekipa (analitičari, projektanti, ... , korisnici) ispituje specifikacije i modele

□ Pisanje testova, probnih slučajeva

- prototipiranje, izvođenje scenarija korištenja – provjera očekivanog ponašanja
- slučajeve povezati s funkcionalnim zahtjevima da se osigura potpunost

□ Pisanje korisničkog priručnika

- skica korisničkog priručnika - kao specifikacija ili dio analize
- dobre upute opisuju svu vidljivu funkcionalnost – ostalo je u SRS

□ Definiranje kriterija prihvatljivosti

- definiranje (s korisnicima) uvjeta pod kojima će proizvod zadovoljiti zahtjeve
- na osnovu slučajeva/scenarija korištenja

Zaključivanje zahtjeva

- **Prikupljanje zahtjeva završava postizanjem (potpisivanjem) "sporazuma" o zahtjevima.**
- **Važno je da obje strane znaju što potpisuju.**
 - U ranoj fazi projekta nije moguće odmah i detaljno znati sve zahtjeve.
 - Nedvojbeno je da će se neki zahtjevi tokom vremena promijeniti.
- **Razvojnici moraju uvažiti da se korisnik može predomisliti**
- **Korisnici moraju uvažiti da to ne može biti unedogled**
 - Specifikacija zahtjeva sadrži osnovicu zahtjeva
 - kasnije promjene zahtjeva ulaze u proces upravljanja promjenama
- **Primjer:  \Analiza\DOO-Specifikacija**
 - dvije iteracije teksta, prototip, nekoliko? promjena pred isporuku

Specifikacija zahtjeva

Dokumentiranje zahtjeva

□ Definicija zahtjeva (Requirements Definition)

- izjava o stanju i ograničenjima sustava te potrebama
- narativni dokument namijenjen korisniku ili ga piše korisnik
 - poslovni i korisnički zahtjevi te njihovi prioriteti
 - uočeni problemi, ključne pretpostavke i preporuke rješenja

□ Specifikacija zahtjeva (Requirements Specification)

- često se naziva i funkcionalnom specifikacijom
- strukturirani dokument s detaljnim opisom očekivanog ponašanja sustava
- namijenjen ugovarateljima i izvoditeljima razvoja
- ugradbeno nezavisno pogled na sustav
 - funkcionalni i nefunkcionalni zahtjevi te njihovi prioriteti
 - model organizacijske strukture (strukturalni dijagrami)
 - opis protoka dokumenata (dijagrami toka rada)
 - model procesa (dijagram toka podataka, dijagrami slučajeva korištenja)
 - konceptualni model podataka (dijagram entiteti-veze)

□ Primjeri:

-  \Analiza\DOO-Specifikacija
-  \Predlošci\SpecifikacijaZahtjeva

Predložak dokumenta specifikacije zahtjeva

□ Predložak nastao na temelju standarda IEEE 830

- Stavke koje nisu primjenjive u konkretnom projektu se izostavljaju.
- Treba uključiti i sadržaj i povijest ispravki dokumenta

1. Uvod

- 1.1 Namjena
- 1.2 Konvencije dokumenta
- 1.3 Tko treba čitati dokument i savjeti za čitanje dokumenta
- 1.4 Opseg proizvoda
- 1.5 Reference

2. Sveobuhvatni pregled

- 2.1 Kontekst proizvoda
- 2.2 Funkcije proizvoda
- 2.3 Kategorije korisnika i svojstva
- 2.4 Okružje u kojem se izvodi proizvod
- 2.5 Ograničenja dizajna i ugradnje
- 2.6 Prepostavke i ovisnosti

3. Zahtjevi za sučeljem

- 3.1 Korisničko sučelje
- 3.2 Hardversko sučelje
- 3.3 Softversko sučelje
- 3.4 Komunikacijsko sučelje

4. Svojstva sustava

- 4.x Svojstvo X
- 4.x.1 Opis i prioriteti
- 4.x.2 Nizovi pobuda/odziv
- 4.x.3 Funkcijski zahtjevi

5. Ostali nefunkcionalni zahtjevi

- 5.1 Zahtjevi za performansama sustava
- 5.2 Zahtjevi za sigurnošću korisnika
- 5.3 Zahtjevi za sigurnošću podataka
- 5.4 Kvaliteta programske podrške
- 5.5 Poslovna pravila
- 5.6 Korisnička dokumentacija

6. Ostali zahtjevi

- Dodatak A: Rječnik
- Dodatak B: Modeli i dijagrami
- Dodatak C: Lista nedovršenih/neodređenih zahtjeva

Označavanje zahtjeva

Uzastopni brojevi

- Novi zahtjev dobiva raspoloživi serijski broj (npr. Z01, Z02, ... ili FZ01,...).
- Broj obrisanog zahtjeva ne koristi se ponovno
- Nepregledno, teško za pratiti veći broj zahtjeva

Hjerarhijsko numeriranje (X.Y.Z)

- Jednostavna i najčešće korištena tehnika
- Problem brisanja zahtjeva koje uzrokuje posmak slijednih oznaka
- Znamenke ne govore ništa o namjeni zahtjeva
 - Poboljšanje: tekstovne oznake unutar brojčanih hjerarhija
 - Primjer: "3.2. - Funkcije editora", sa zahtjevima "ED-01", "ED-02", itd.

Označavanje zahtjeva (nastavak)

□ Hijerarhijske tekstovne oznake

- Primjer : "Sustav će upozoriti korisnika da potvrdi ispis više od 10 kopija".
 - Ovaj zahtjev može se označiti s: ISPIS.KOPIJE.POTVRDA.
- prednost - oznake su strukturirane i smislene, ažuriranje jednostavno
- nedostatak - nezgrapnije u odnosu na hijerarhijsko numeriranje

□ Potpunost

- Nijedan zahtjev ili informacija ne smiju nedostajati.
- Zahtjeve koje nedostaju teško je uočiti.
- Kontrola „praćenjem“ poslovnog procesa
- Nepotpune informacije posebno označiti (npr. TBD - "to be determined"), te razriješiti prije ugradnje

Upravljanje zahtjevima

Upravljanje zahtjevima

- **Definiranje postupka za promjenu zahtjeva**
 - postupak kojim se novi zahtjev ili promjena postojećeg analizira i prihvata
 - predložene promjene moraju slijediti unaprijed definiranu proceduru
- **Uspostava odbora za promjene (CBB)**
 - Ključni članovi projekta
 - odlučuje o usvajanju zahtjeva te postavlja prioritete i rokove
- **Analiza utjecaja promjena zahtjeva**
 - procjenjuje se utjecaj promjene na organizaciju, raspored ili drugo
 - služi donošenju dobrih (ispravnih, mogućih) odluka o (ne)prihvaćanju promjena
- **Praćenje promjena zahtjeva na svim proizvodima**
 - za prihvaćenu promjenu se kroz matricu praćenja zahtjeva pronađe sve ovisne komponente (izvorni kod, testni slučajevi, neki drugi zahtjev,...).

Upravljanje zahtjevima (2)

□ Uspostava vremenske osnove (baseline) i kontrole verzija

- definiranje slike dogovorenih zahtjeva u određenom trenutku
- nakon osnove, zahtjevi prolaze postupak promjena
- verzioniranje specifikacije – uklanjanje neodređenosti

□ Praćenje povijesti promjena zahtjeva

- zapis o vremenu, vrsti i sadržaju, razlozima, verziji i autoru promjene

□ Praćenje statusa zahtjeva

- status (predložen, odobren, ugrađen, provjerен) + statistika

□ Mjerenje stabilnosti zahtjeva

- brojanje promjena – mjera da je problem shvaćen, opseg definiran

□ Korištenje alata za upravljanje zahtjevima

- evidencija zahtjeva, međusobne povezanosti i statusa
- verzioniranje zahtjeva i dokumentacije

Reference

- Maciaszek L. Requirements Analysis and System Design: Developing Information Systems with UML, 3/ed, Addison Wesley Higher Education, 2007.
- Managing Software Requirements: A Use Case Approach (2nd Edition), Addison-Wesley Professional, 2003.
- http://www.jiludwig.com/Requirements_Management_Tools.html



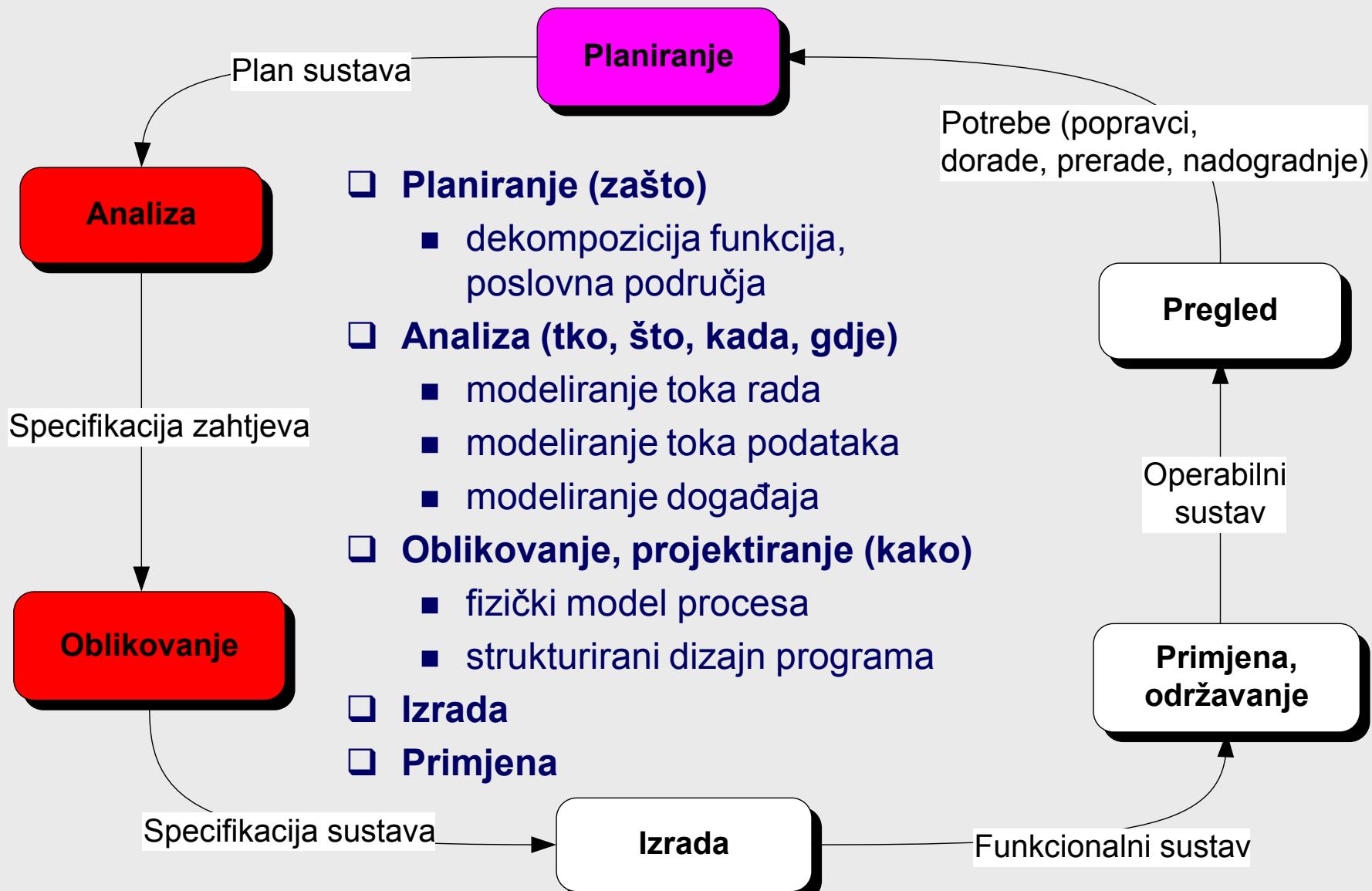
Strukturirana analiza i dizajn

2012/13.04

Sadržaj

- **Strukturirana analiza**
- **Dekompozicija procesa i funkcija**
 - Logički procesi
 - Poslovna pravila i poslovna politika
- **Modeliranje funkcija**
 - Dijagram dekompozicije funkcija
 - Hiperarhijski prikaz funkcija/procesa
- **Izrada dijagrama dekompozicije**
 - Dijagram organizacije
 - Razrada poslovnih procesa
- **Oblikovanje toka rada**
- **Modeliranje toka podataka**
 - Dijagram toka podataka
 - Pravila i ograničenja
 - Preporuke i metode
 - Elementarni procesi
 - Opisivanje podataka
 - Primjena modela procesa
- **Modeliranje događaja**
 - Događaji i vrste događaja
 - Modeliranje vođeno događajima
 - Matrični prikaz modela događaja
 - Dijagram prijelaza stanja
 - Mape dijaloga

Strukturirana analiza i strukturirani dizajn



Strukturirana analiza

□ Moderna strukturirana analiza = Logički dizajn (česti sinonim)

- strukturirani proces i rezultati analize
- tehnička modeliranja poslovnih zahtjeva na sustav
 - usmjereni procesima, ali se razvila tako da obuhvaća i podatke
- logički modeli – određivanje ŠTO je sustav i ŠTO mora raditi (ne KAKO)
 - dijagrami toka podataka za prikaz poslovnih zahtjeva nezavisno od tehničkih rješenja → **logički dizajn**
 - izražavaju suštinu sustava → esencijalni, konceptualni, poslovni modeli
- uključuje određivanje prioriteta zahtjeva

□ Postupak dekompozicije - strukturno raščlanjivanje

- podijeli pa s/vladaj (lat. divide et impera, eng. divide and conquer)
- hijerarhijski modeli, iterativno, s vrha prema dolje
 - funkcije i procesi, organizacija, podaci, softver

□ Aplikacijski model procesa = logički model procesa sustava ili aplikacije koji se radi u fazi analize

Logički procesi

□ Funkcija (djelatnost, posao)

- skup logički povezanih trajnih poslovnih aktivnosti i zadataka
 - obavlja se stalno (nema određeni početak i kraj)
 - obavljaju osobe, grupe ili organizacijske cjeline
- primjeri: prodaja, proizvodnja, računovodstvo
- može se sastojati od desetina pa i stotina diskretnih procesa
 - hijerarhijsko razlaganje do razine procesa koji obavljaju neki zadatak

□ Događaj (poslovni događaj)

- logički dio posla koji se obavlja kao nedjeljiva cjelina → *transakcija*
 - pokretan diskretnim ulazom
 - završava nakon što proces odgovori odgovarajućim izlazom
- može se predstaviti procesom kojim sustav reagira na taj događaj
 - logički događaj dalje se razlaže do elementarnih procesa

□ Proces (elementarni, primitivni proces)

- postupak, način rada, dosljedna izmjena stanja
- diskretna odluka, aktivnost ili zadatak kojima se obavlja neki posao
- obavlja se uvijek na jednak način (za određeni ulaz daje isti izlaz)
- trajanje je konačno i odredivo (poznati početak, završetak, ponavljanje)
- za obavljanje se koriste sredstva, npr. ljudska, materijalna (strojevi), financijska

Poslovna pravila i poslovna politika

- **Poslovno pravilo - instrukcije i logika procedure obavljanja procesa**
 - ugrađuje se u računalni program
 - npr. preduvjeti izlaska na ispit, broj polaganja ispita, uvjeti upisa
- **Poslovna politika – skup poslovnih pravila**
 - u većini organizacija podloga za donošenje odluka
- **Primjer: subvencioniranje studentske prehrane**
 - *prvog u mjesecu dodjeljuju se prava ...*
 - *mjesечni iznos raspoloživ za subvenciju jednak je umnošku broja dana u mjesecu i dnevног subvencioniranog iznosa za razinu prava*
 - *Npr. za razinu 1 (domicilni) dnevni iznos je 16.70 Kn (cijena menija), pa student te razine ima raspoloživo u svibnju $16.70 * 31 = 517.7$ Kn.*
 - *subvencija se akumulira tijekom ak. godine - neiskorištena prava iz jednog mjeseca prebacuju se na slijedeći (!?)*
 - ...

Primjer poslovne politike

- ❑ Start with a cage containing five monkeys. In the cage, hang a banana on a string and put a set of stairs under it. Before long, a monkey will go to the stairs and start to climb towards the Banana. As soon as he touches the stairs, spray all of the monkeys with cold water. After a while, another monkey makes an attempt with the same result - all the monkeys are sprayed with cold water.

Pretty soon, when another monkey tries to climb the stairs, the other monkeys will try to prevent it. Now, turn off the cold water. Remove one monkey from the cage and replace it with a new one. The New monkey sees the banana and wants to climb the stairs. To his horror, all of the other monkeys attack him. After another attempt and attack, he knows that if he tries to climb the stairs, he will be assaulted.

Next, remove another of the original five monkeys and replace it with a new one. The newcomer goes to the stairs and is attacked. The previous Newcomer takes part in the punishment with enthusiasm. Again, replace a third original monkey with a new one. The new one makes it to the stairs and is attacked as well. Two of the four monkeys that beat him have no idea why they were not permitted to climb the stairs, or why they are participating in the beating of the newest monkey.

After replacing the fourth and fifth original monkeys, all the monkeys which have been sprayed with cold water have been replaced.

- ❑ Nevertheless, no monkey ever again approaches the stairs. Why not?

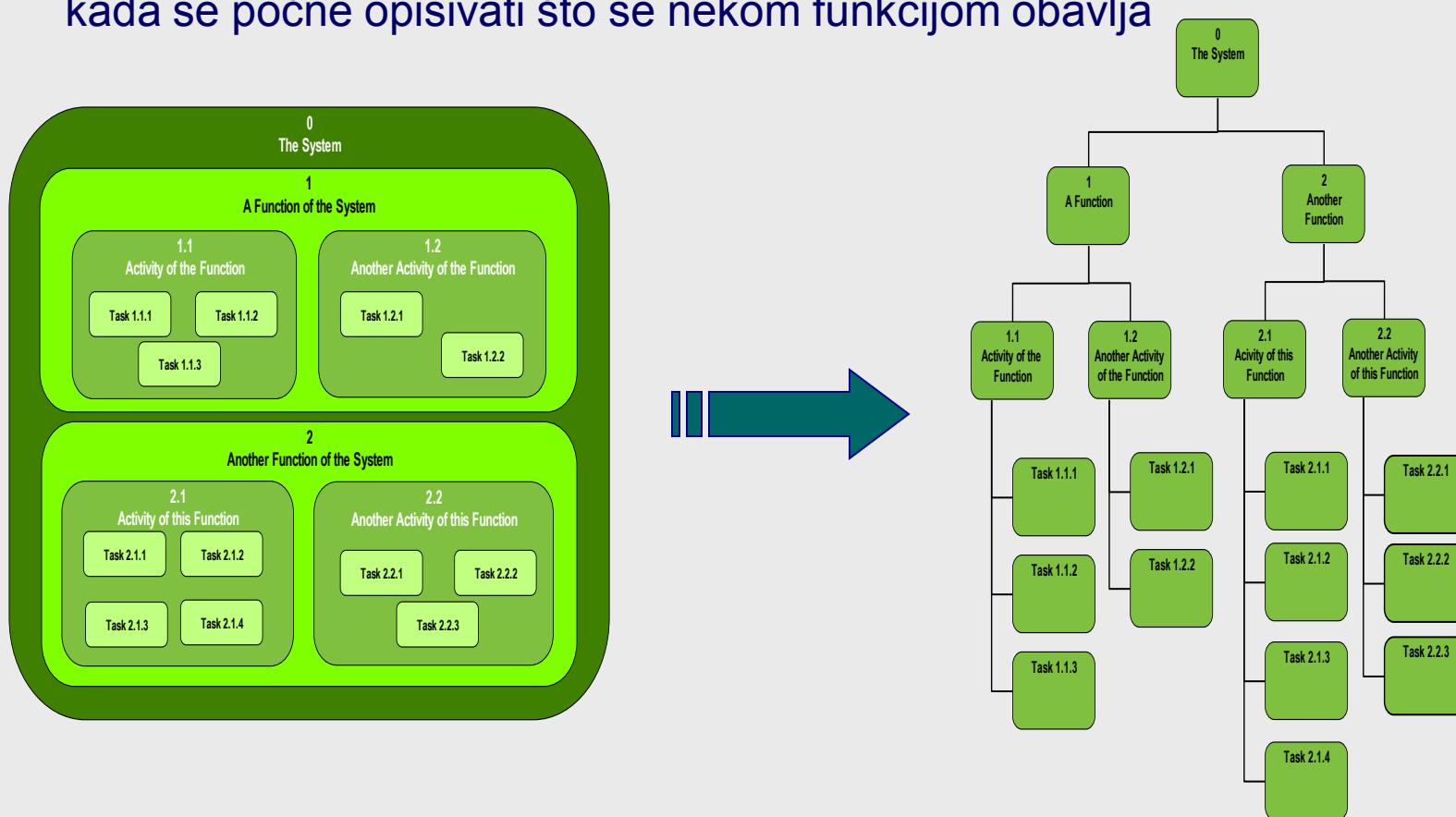
Because that's the way it's always been around here.

And that's how company policy begins.... (Google search: company policy)

Modeliranje funkcija

❑ Funkcionalna dekompozicija, dekompozicija funkcija

- izrada općeg modela funkcija (modela poslovnih funkcija) promatranog sustava u fazi planiranja → strukturirano planiranje
- hijerarhija funkcija iterativno se razlaže do razine procesa, tj. do trenutka kada se počne opisivati što se nekom funkcijom obavlja



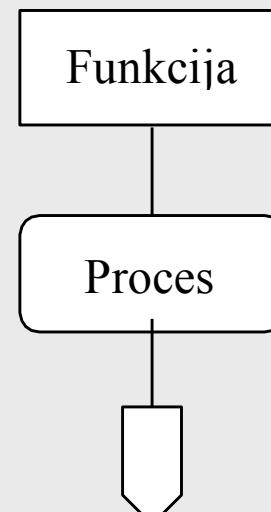
Dijagram dekompozicije funkcija

□ Dijagram funkcionalne dekompozicije

- eng. Functional Decomposition Diagram (FDD)
- ista notacija koristi se za razlaganje bilo koje hijerarhijske strukture pa se često zove samo Dijagram dekompozicije ili Mapa hijerarhije

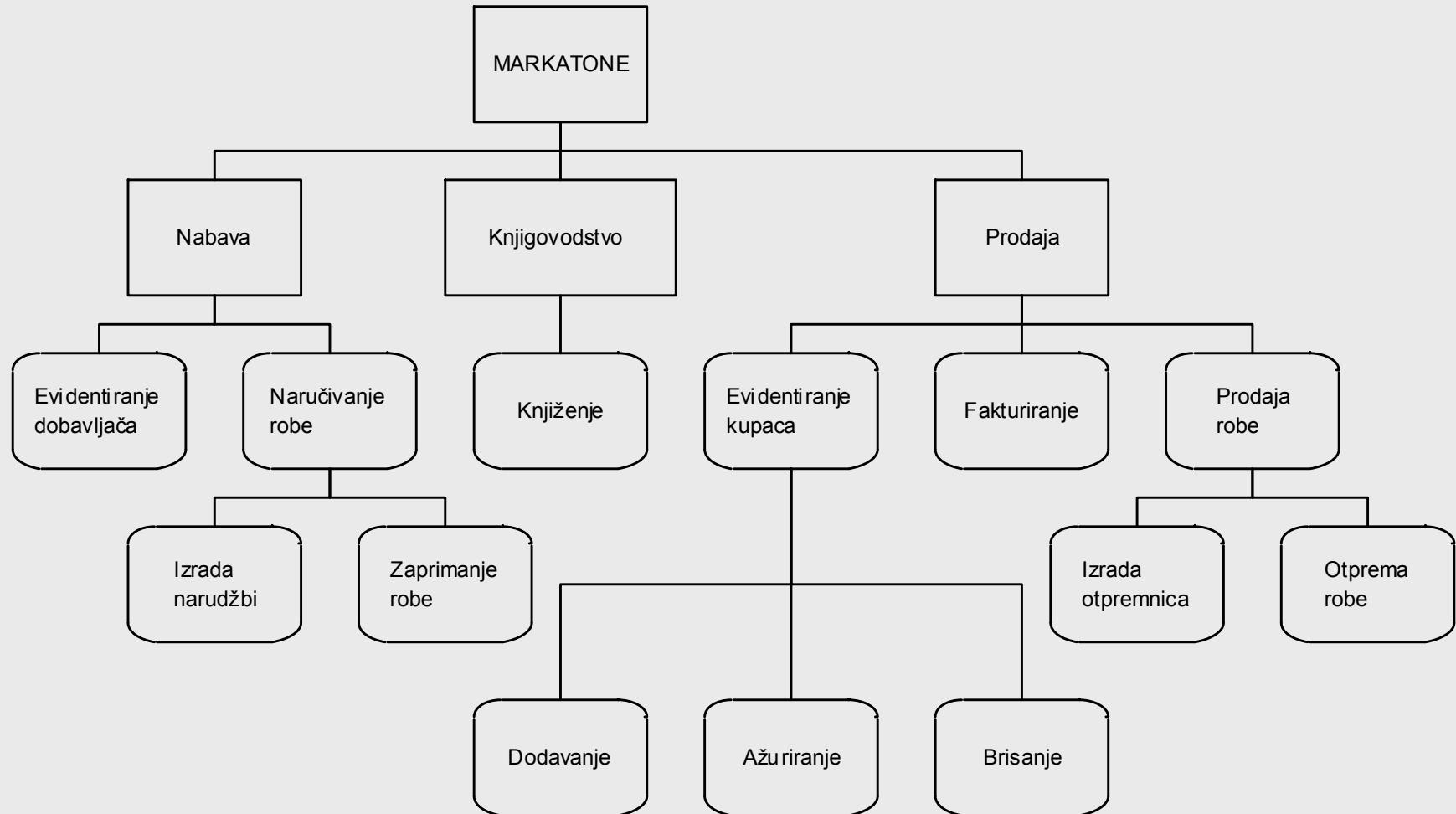
□ Elementi

- funkcije - označavaju se (glagolskom) imenicom, npr. Prodaja, Proizvodnja
- procesi - označavaju se glagolskim izrazom oblika infinitiv+objekt
- spojnice - spojevi između funkcija i procesa (connector)
- vanjski spojevi - s dijelovima na drugim stranicama (off-page connector)



Dijagram dekompozicije funkcija - primjer

- Primjer, dijagram funkcija za jedan sustav/podsustav:



- Primjeri: [\Modeliranje\PutniNaloziBlagajna](#)

Hijerarhijski prikaz funkcija/procesa

- Izrada globalnog modela funkcija može započeti izradom hijerarhijske liste funkcija po pojedinim organizacijskim cjelinama.
- Primjer:
 - NABAVA
 - Evidentiranje dobavljača
 - Nabavka robe
 - Izrada narudžbi
 - Zaprimanje robe
 - UPRAVLJANJE OSOBLJEM
 - Evidentiranje službe
 - Zaprimanje u službu
 - Praćenje službe
 - » redovan rad
 - » prekovremeni rad
 - » bolovanje
 - » godišnji odmori
 - Otpuštanje iz službe
 - Obračun plaća

Izrada dijagrama dekompozicije

□ Postupak

- Korijen = sustav
- Razrada u podsustave i poslovne funkcije
- Daljnja razrada do razine operacionalizacije

□ Pravila

- svaki proces je roditelj ili dijete (mora biti dio hijerarhije)
- roditelj mora imati barem dvoje djece

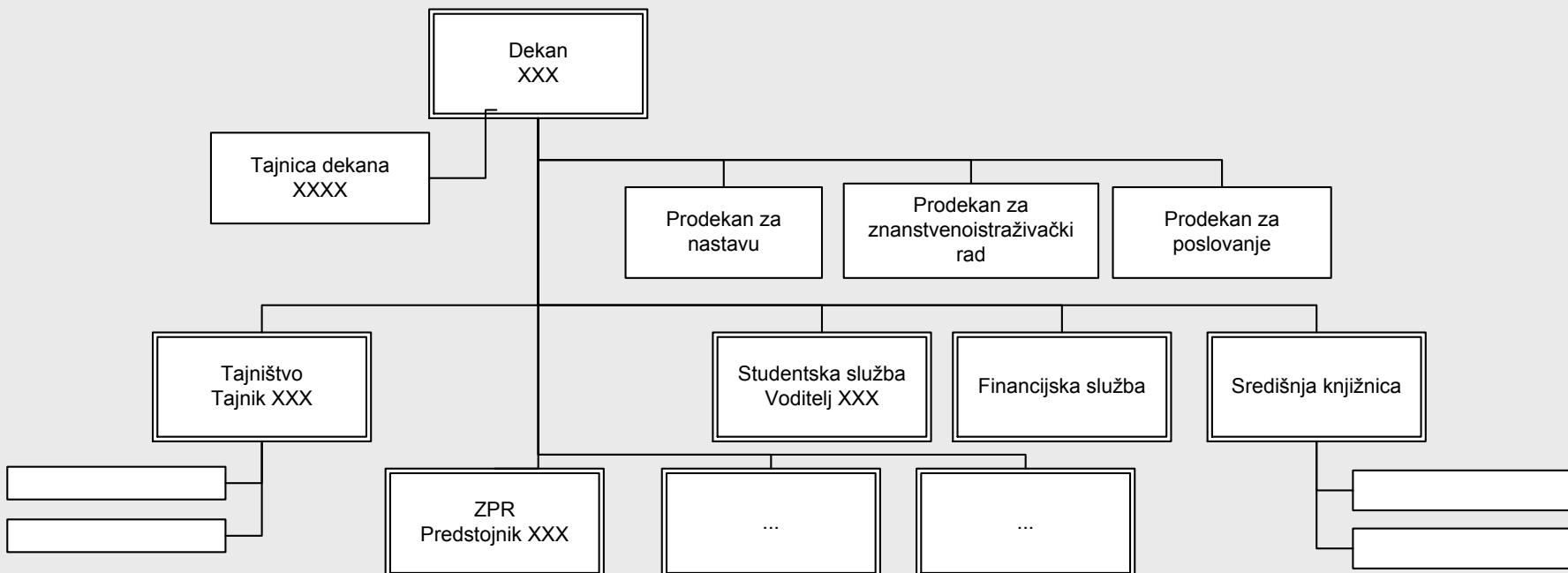
□ Preporuke

- izostaviti procese koji premještaju ili preusmjeravaju podatke, ali ih ne mijenjaju
- pažnju usmjeriti na procese koji
 - nešto računaju (npr. prosjek ocjena)
 - potpomažu odluke (npr. određivanje raspoloživosti robe pri naručivanju)
 - filtriraju ili agregiraju podatke (npr. računi kojima je istekao rok plaćanja)
 - organiziraju podatke u korisne informacije (npr. generiranje izvješća)
 - pokreću druge procese (npr. generiranje subvencije)
 - rukuju podacima (npr. stvaranje, čitanje, ažuriranje, brisanje) - Ne pretjerivati!

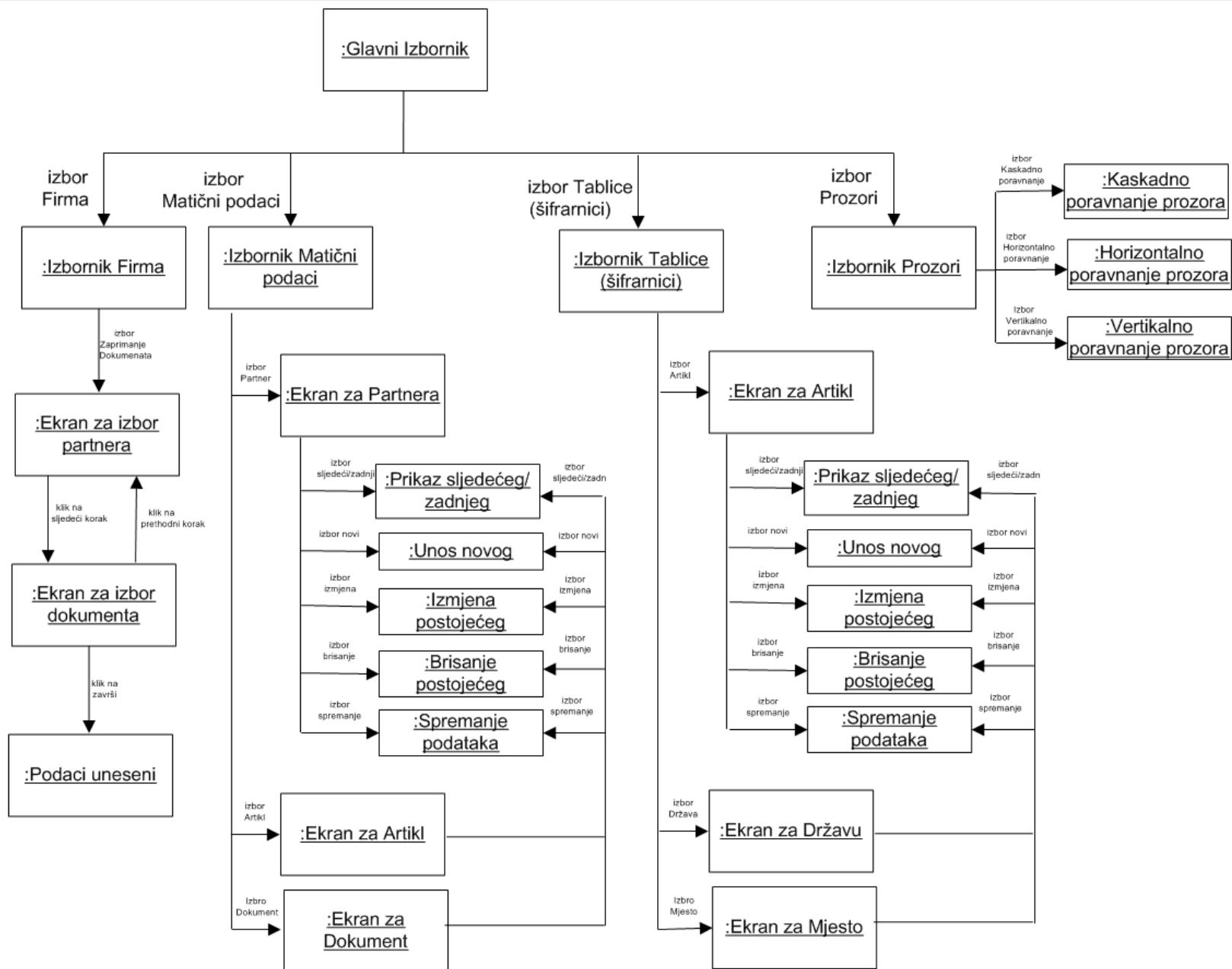
Dijagram organizacije

□ Shema, mapa, karta organizacije (Organization chart)

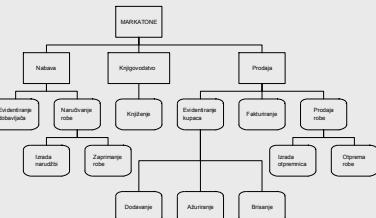
- prikaz strukture organizacije hijerarhijom pravokutnika ("kućica")
- svaki pravokutnik reprezentira određenu ulogu ili odgovornost u organizaciji
 - problem, poistovjećivanje funkcije i organizacije: dekan = ured dekana
 - rješava se pročišćavanjem tijekom analize i odvajanjem uloga korisnika



Primjer: hijerarhija funkcija aplikacije



Razrada poslovnih procesa



□ Tehnike modeliranja

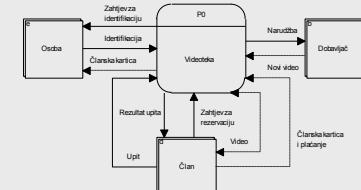
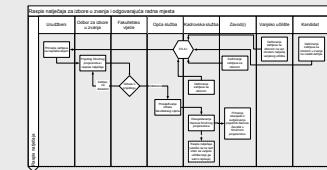
- u širinu - svaki dijagram se detaljizira prije dekomponiranja (breadth-first)
- u dubinu - identificira se hijerarhija, a zatim se detaljizira (depth-first)

□ Razina dekompozicije - Kada stati ?

- do dubine dovoljne za razumijevanje modela (!?)
- napredak do stanja u kojem ulazi i izlazi prevladaju na dijagramu
 - (predzadnji red Markatone)

□ nastavak se može provesti

- modeliranjem toka rada (dijagram aktivnosti, Visio: Cross-Functional FlowChart, IDEF0) ili
- modeliranjem toka podataka (Data Flow Diagram - DFD)



Modeliranje toka rada i toka podataka

Modeliranje toka rada

□ Tok rada (workflow) - radna procedura, poslovna procedura

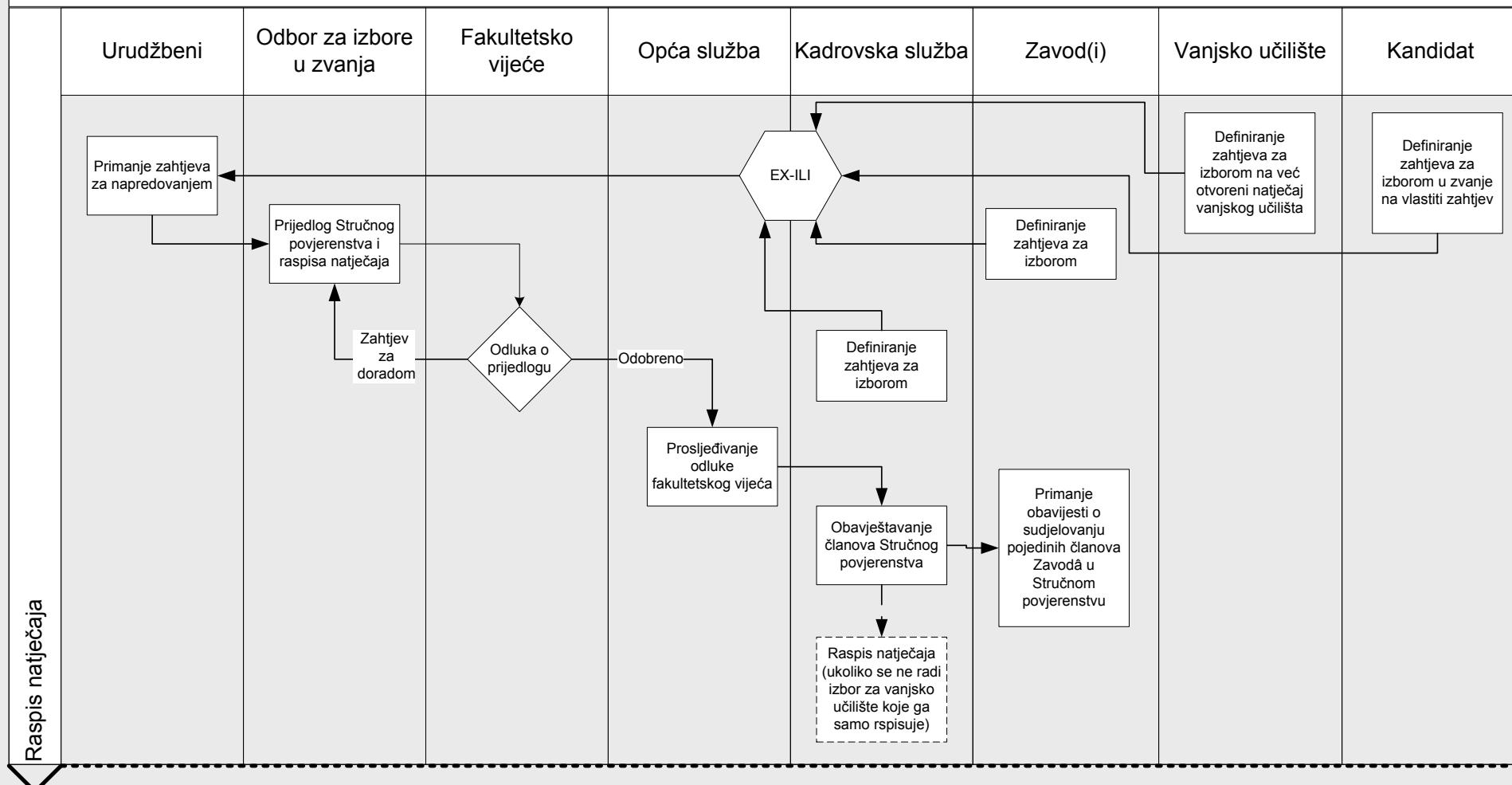
- slijed koraka obrade koji obrađuje jednu poslovnu transakciju
- definira logiku obrade i precizira nositelja
- može imati više varijanti (scenarija)

□ Primjer: Raspis natječaja za izbor u zvanje ili na radno mjesto

- *U urudžbeni zapisnik se predaje zahtjev za raspisom natječaja u zvanje i/ili radno mjesto / na osobni zahtjev ili na zahtjev organizacije. FER provodi postupak i na zahtjev s drugih organizacija koje nisu ovlaštene ...*
- *Zahtjev se proslijeđuje Odboru za izbore u zvanja koje predlaže članove Stručnog povjerenstva za provedbu postupka i definira javni natječaj*
- *Prijedlog razmatra Fakultetsko vijeće te ga ili šalje nazad na doradu Odboru za izbore u zvanja ili samo definira nove članove Stručnog povjerenstva*
- *Opća služba proslijeđuje odluku Fakultetskog vijeća Kadrovskoj službi*
- *Kadrovska služba raspisuje natječaj i obavještava članove Stručnog povjerenstva*
- *Izbor na zahtjev pojedinca ili njegove organizacije*

Primjer: Raspis natječaja

Raspis natječaja za izbore u zvanja i odgovarajuća radna mjesta



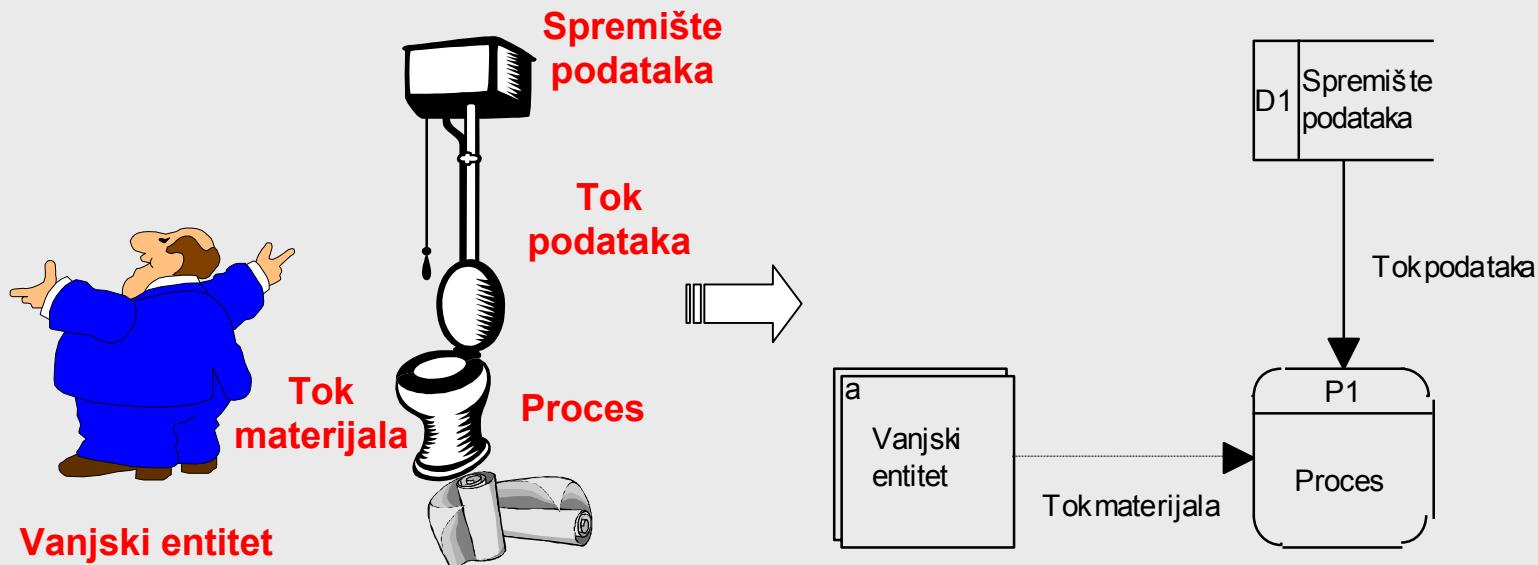
Slika vrijedi 1024 riječi

- može se automatizirati (u drugom ciklusu predavanja)

Modeliranje toka podataka

□ Dijagram toka podataka (DFD - Data Flow Diagram)

- prikaz protoka, strukture i obrade podataka
- dokumentiranje logike, poslovnih pravila i procedura
- sinonimi: transformacijski graf, mjehurasti dijagram (Bubble Chart)



- Ne može se koristiti za opis programske logike, opis promjene stanja, izradu upravljačkih specifikacija ili dizajn korisničkog sučelja!!!

Elementi dijagrama toka podataka

□ Tok podataka (data flow)

- skupovi podataka koji kolaju sustavom
- ulaze u procese (ulazni), mijenjaju (ulazno/izlazni) ili nastaju (izlazni)
- jedinstveni nazivi oblika *imenica* ili *pridjev+imenica*, npr. Potvrđena prijavnica, Izlazni račun

□ Proces

- aktivnost pretvorbe (ulaznog u izlazni tok podataka)
- naziv oblika infinitiv+objekt (npr. Prijaviti ispit) ili glagolske imenice (npr. Prodaja, Prijava ispita)
- izbjegavati općenite nazive (npr. Obavljanje računovodstvenih poslova)
- opis procesa sadrži opis aktivnosti (algoritam) njegovog djelovanja

□ Spremište podataka (data store)

- organizirani i trajni skup podataka
- mjesto pohrane, npr. dokument, registrator, datoteka, tablica u BP (izbjegavati u nazivlju)
- promjena sadržaja (CRUD) procesima
- naziv imenicom (ev. u množini), npr. Prijavnica (Prijavnice)

□ Vanjski entitet (external entity / agent)

- objekt vanjskog svijeta povezan s promatranim sustavom
- određuje granice sustava
- predstavlja izvorišta i odredišta podataka, (source, sink)
- mogu biti osobe, org. jedinice, ustanove, drugi sustavi ...
- nazivaju se imenicama, npr. Student, Kupac, Dobavljač

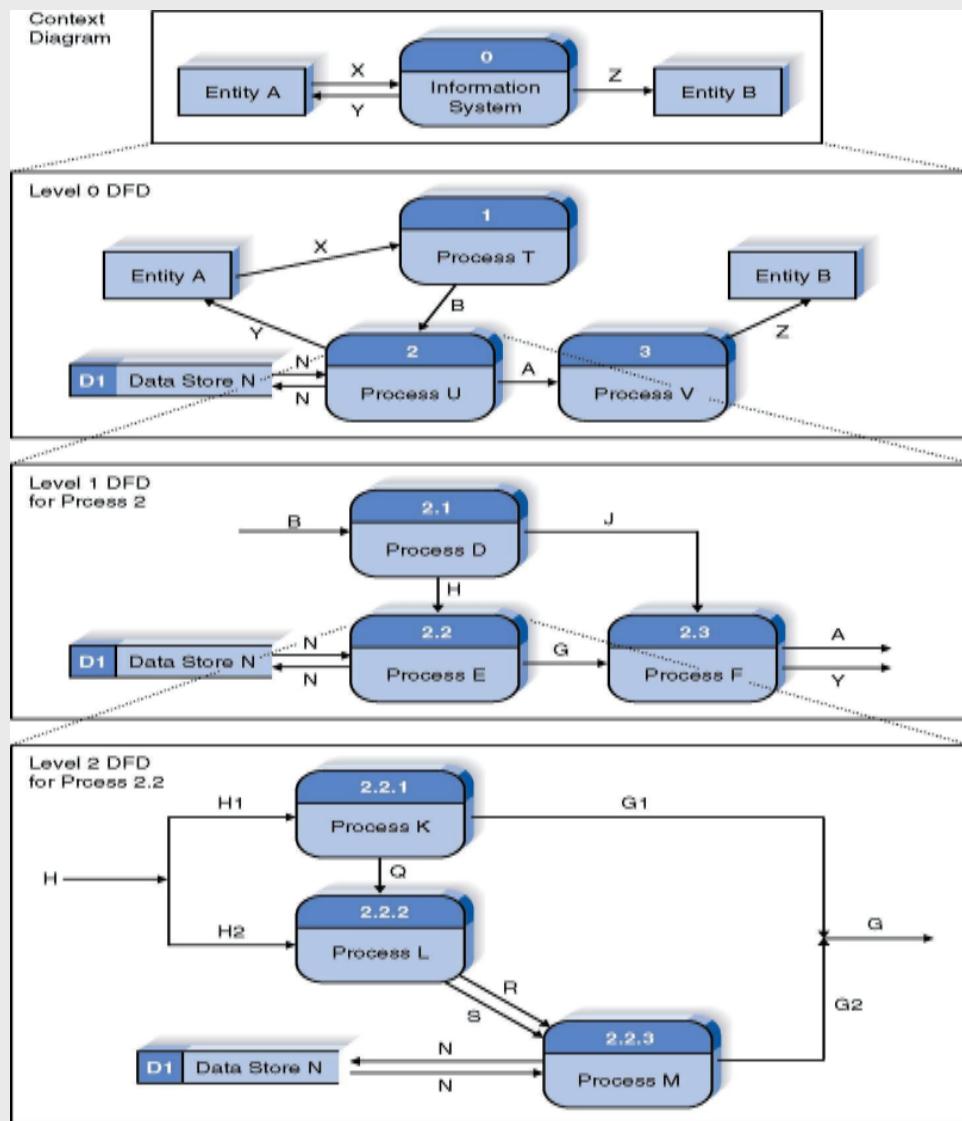
Izrada dijagrama toka podataka

□ Dekompozicija procesa

- polazni, dijagram konteksta (context diagram) hijerarhijski se razlaže na poddijagrame do osnovnih procesa
- niveličacija (leveling) – proces (parent) razrađuje se (explode) dijagramom na nižoj razini (child)
- preporuka: dijagrami s 2 do 9 procesa, a poželjno je slijediti "pravilo 7 ± 2 "
- zaustavljanje kada postane očigledna ugradnja procesa na najnižoj razini

□ Preporuke za označavanje elemenata

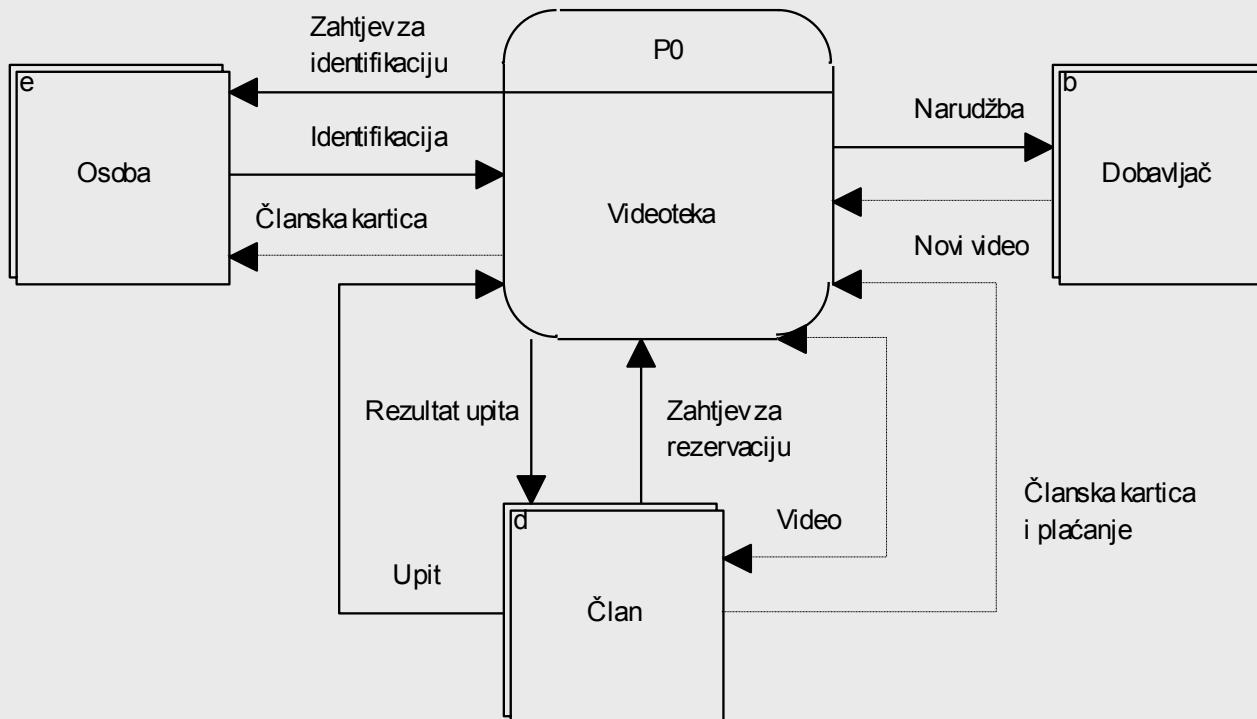
- hijerarhija, razina konteksta = 0
- spremišta, izvori i odredišta – nazivlje velikim slovima, oznake oblika slovo ili slovo+broj (D1 kao spremište prvo)
- procesi i tokovi podataka - malim slovima



Izrada dijagrama toka podataka (2)

□ Dijagram konteksta

- prikazuje sustav na najvišoj razini hijerarhije prikaza (top level diagram)
- definira okruženje sustava i područje analize (environmental model)
- prikazuje jedan proces i vanjske entitete
 - započeti s procesom koji prikazuje sustav u cijelini
 - odrediti vanjske entitete i njihovu povezanost sa sustavom



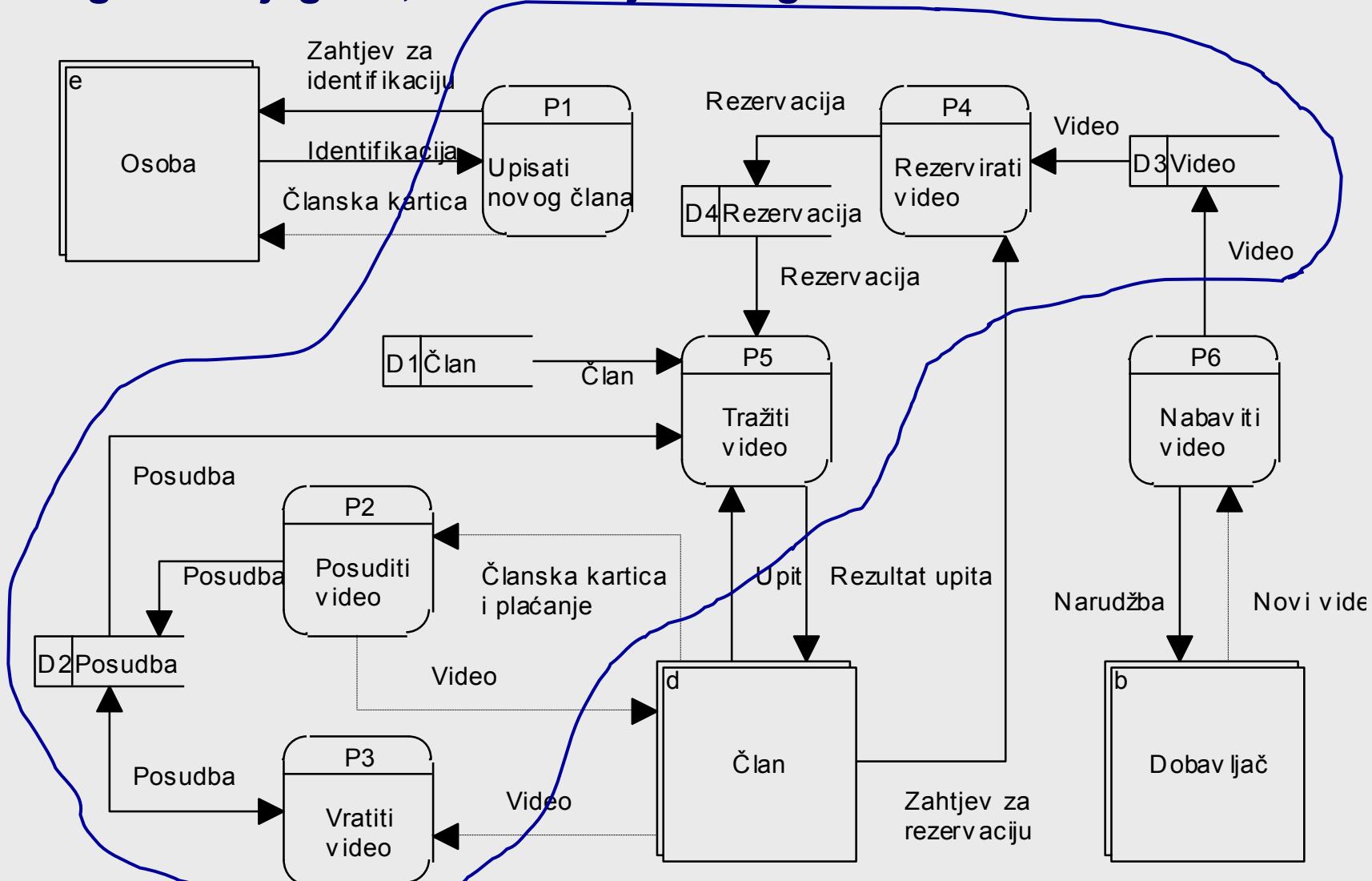
Izrada dijagrama toka podataka (3)

□ Pregledni dijagram (initial diagram)

- uočiti glavne tokove informacija (npr. korišteni dokumenti, potrebni podaci)
- odrediti glavne aktivnosti sustava i prikazati ih odgovarajućim procesima
- uključiti vanjske entitete i tokove podataka s dijagrama konteksta
- složiti se s korisnikom oko granica sustava
- utvrditi procese i spremišta podataka

Izrada dijagrama toka podataka (4)

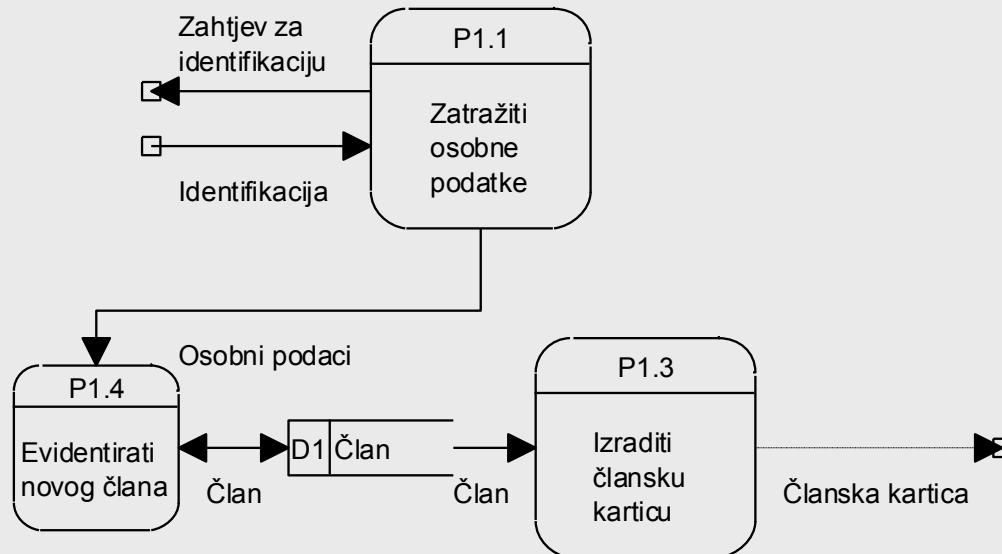
□ Pregledni dijagram, određivanje dosega



Izrada dijagrama toka podataka (5)

Razrada

- za svaki proces s preglednog dijagrama identificirati podaktivnosti
 - na primjer, za proces Upisati novog člana:



- Ponavljati postupak za svaki od procesa na poddijagramu

- uspostaviti razinu detalja slijedeći "pravilo 7 ± 2 "
 - provjeriti potpunost i ispravnost modela

Model obrazložiti korisniku a zatim ga ažurirati po potrebi

- Dubinu i uravnoteženost modela teško je odrediti.
 - U praksi to može značiti doradu u većem broju ponavljanja!

Pravila i ograničenja prilikom izrade DTP

□ Pravilo bilance (očuvanja) tokova (level balance rule)

- količina tokova koji ulaze u proces i izlaze iz procesa mora odgovarati količini tokova podprocesa na nižoj razini hijerarhije
- nije dozvoljeno variranje tokova neke razine na nižim razinama (npr. tok T na nižim razinama prikazivati kao T1, T2)

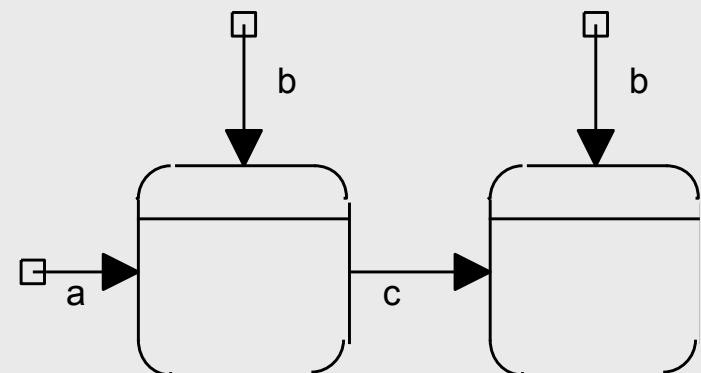
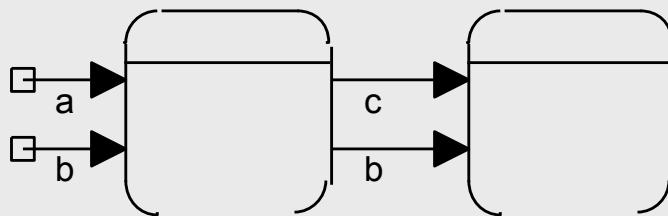
□ Ograničenja i posebni slučajevi

- Svi objekti moraju biti povezani. Nepovezanost ukazuje na nepotpunost :
 - postojanje procesa bez ulaza i/ili izlaza (tzv. čuda i crne rupe)
 - izlaze za koje ne postoji dovoljno ulaza (tzv. sive rupe – najčešće)
 - postojanje nepovezanih spremišta ili vanjskih entiteta
- Ne dozvoljava se neposredna povezanost:
 - vanjskih entiteta
 - spremišta
 - spremišta i vanjskog entiteta
- Nije dozvoljeno:
 - grananje toka u različite tokove, spajanje različitih tokova
 - postojanje “rekurzivnih” procesa

Preporuke za izradu DTP

□ Treba pripaziti na:

- trivijalne tokove – izlazi iz procesa koji ne ulaze u spremišta ili odredišta
 - posebno značenje - prikaz posebnih stanja (npr. dojava pogreške)
- neposredno povezane procese
 - ako postoje - neki čeka na završetak prethodnog
- procese koji ne obavljaju pretvorbu podataka
 - ako je izlazni tok jednak ulaznom
 - treba preimenovati jedan od tokova ili
 - treba obaviti prespajanje tokova
 - primjer:

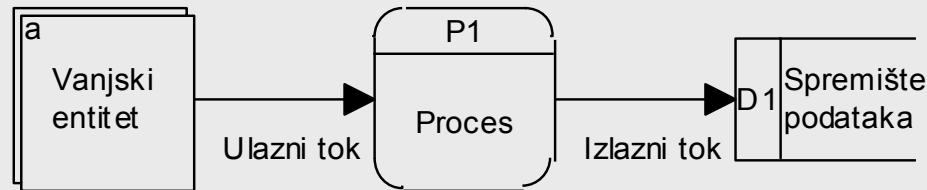


□ Procesi se mogu zbivati istovremeno - DTP se ne smije tumačiti kao dijagram toka (flowchart)!

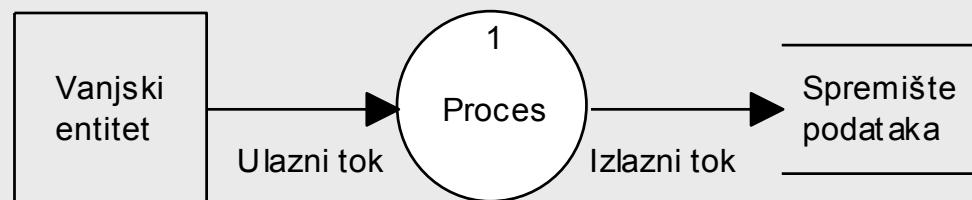
Metode koje koriste DTP

□ Notacije

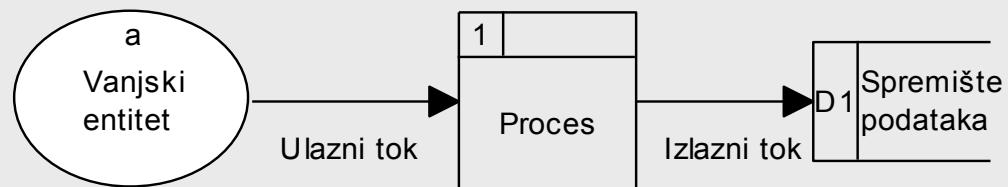
- Gane/Sarson (korištena u primjerima)



- Yourdon/DeMarco



- SSADM



□ Proširenja modela

- okidač (trigger) - prikaz učestalosti procesa (npr. tri puta dnevno)
- posebni simboli za prikaz ponavljanja procesa
- razdvajanje i spajanje tokova (alternativni tokovi)
- posebni simboli za tok resursa, dokumenata ili upravljanja

Primjena modela procesa

□ Strateško planiranje sustava

- definiranje arhitekture sustava, modeliranjem procesa poduzeća (globalni model procesa)
 - identificira poslovna područja i poslovne funkcije
 - najčešće u formi dijagrama dekompozicije funkcija ili nerazrađenog DTP (npr. dijagramom konteksta određuju se doseg i sučelja sustava)

□ Preoblikovanje poslovnih procesa

- analiza i restrukturiranje poslovnih procesa radi poboljšanja učinkovitosti i uklanjanja birokratizma, prije primjene informacijskih tehnologija
- postojeći procesi se analiziraju i dokumentiraju prikladnim modelima procesa
 - dijagrami toka podataka s fizikalnim primjesama, koje uključuju vremensku dimenziju, protočnost podataka i troškove

□ Analiza sustava

- aplikacijski modeli procesa - logički modeli procesa sustava ili aplikacije
 - teorijski, moguće je proizvesti DTP povratnim inženjerstvom postojećih aplikacija, ali će takav dijagram biti preopterećen fizikalnim primjesama

□ Dizajn sustava

- fizički modeli procesa - dodavanjem upravljačkih komponenti i resursa

Elementarni procesi

□ Mini-specifikacije (funkcionalne primitive)

- specifikacije za opisivanje osnovnih procesa u dijagramu toka podataka
- mogu poprimiti različite oblike, ali imaju nekoliko zajedničkih elemenata:
 - naziv i broj procesa
 - listu podataka koji ulaze u proces (ulaznih tokova podataka)
 - listu podataka koji iz procesa izlaze (izlaznih tokova podataka)
 - tijelo opisa procesa

□ Opis procesa

- sadrži algoritam zadatka koji se procesom obavlja, koji može poprimiti različite oblike (→ dizajn programa - opis programske logike)
- na temelju ovog algoritma može se, ovisno o alatu, generirati kôd

Definiranje procesa

□ Primjer definicije procesa (1)

Proces 1:	Provjera raspoloživosti filma
Opis:	Provjera da li u videoteci postoji kopija filma koja se može iznajmiti
Ulazni tokovi:	Upit (Film) Rezervacije Posudbe
Izlazni tokovi:	Rezultat upita (raspoloživ nije raspoloživ ne postoji)
Logika procesa:	<u>izračunaj</u> broj kopija traženog filma u videoteci <u>ako</u> je broj kopija veći od nule <u>tada</u> <u>ako</u> je broj kopija veći od (broj posudbi + broj rezervacija) rezultat = "Raspoloživ" <u>inače</u> rezultat = "Nije raspoloživ" <u>inače</u> rezultat = "Ne postoji"

Definiranje procesa

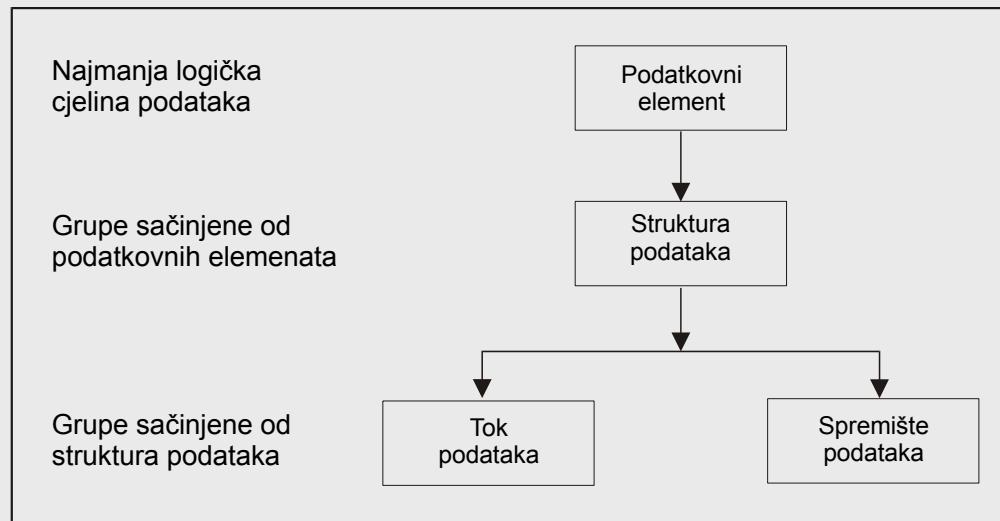
□ Primjer definicije procesa (2)

Proces 1:	Provjera raspoloživosti filma
Opis:	Provjera da li u videoteci postoji kopija filma koja se može iznajmiti
Ulagani tokovi:	Upit (Film), Rezervacije, Posudbe
Izlazni tokovi:	Film nije raspoloživ, Film je raspoloživ
Logika procesa:	<p><u>izračunaj</u> broj kopija traženog filma u videoteci <u>ako</u> je broj kopija veći od nule <u>tada</u> <u>ako</u> je broj kopija veći od (broj posudbi + broj rezervacija) <u>nastavi</u> s P2 (izl. tok Film je raspoloživ) <u>inače</u> <u>ako</u> član želi rezervirati film <u>tada</u> <u>upiši</u> rezervaciju (izl. tok Film nije raspoloživ) <u>inače</u> <u>poruka</u> "Ne postoji"</p>

Opisivanje podataka

□ Rječnik podataka (Data Dictionary)

- mjesto pohrane definicija podatkovnih elemenata i struktura podataka
- strukturirano spremište meta-podataka, to jest podataka o podacima
 - prvotno se pojavio kao proširenje dijagrama toka podataka, za pohranu opisa spremišta podataka i tokova podataka
 - može se koristiti kao alternativna tehnika za prikaz modela podataka
- standardno se upotrebljava(ia) BNF notacija (Backus-Naur Form)



□ Alternativa : ERD koji nastaje preslikavanjem spremišta u entitete

Definiranje podataka BNF notacijom

□ Notacija

=	struktura s lijeva sastoji se od dijelova s desna ("sastavljeno od")
+	agregacija elemenata (logičko <i>I</i>)
()	opcionalnost elemenata u zagradi (0 ili 1)
{ }	ponavljanje (iteracija) elemenata u zagradi, ni jednom do konačni broj puta
[]	alternativa (selekcija) elemenata u zagradi
	odvajanje alternativnih elemenata u [] izrazu
-	početna i završna vrijednost raspona definiranog [] izrazom
**	komentar
@	oznaka ključa

□ Primjer, račun i stavke računa

```
Racun = @BrRac + DatRac + BrKupca  
      + { SifArt, NazArt, Cijena, Kol, Vrijednost }  
      + (IznosRac)
```

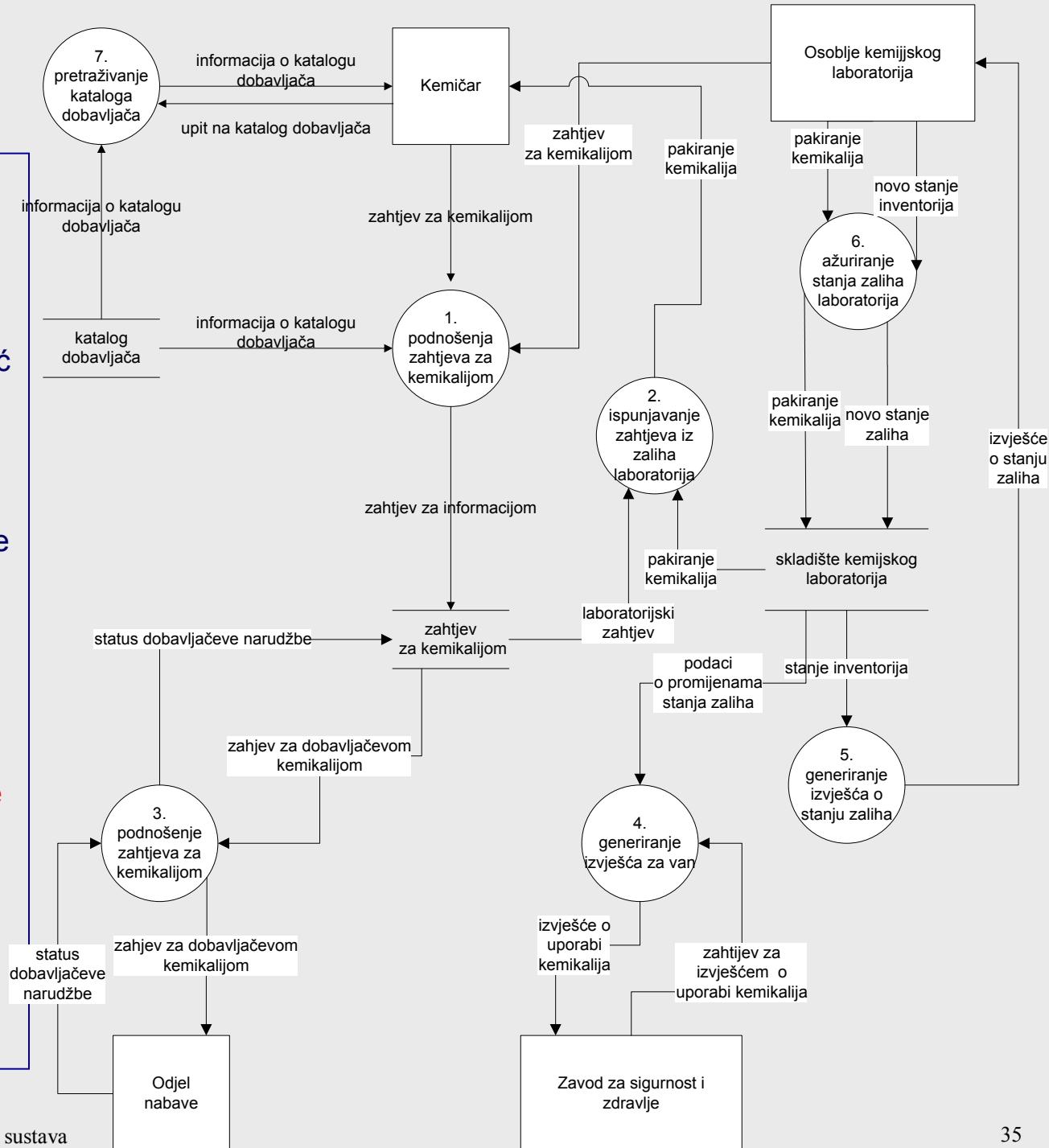
□ Može se napisati i kao:

```
Racun = @BrRac + DatRac + BrKupca  
      + { StavkaRac }  
      + (IznosRac)
```

StavkaRac = @SifArt, NazArt, Cijena, Kol, Vrijednost

DTP analizom izjava korisnika

"Kemičar ili član osoblja kemijskog laboratorija može podnijeti zahtjev za jednom ili više kemikalija. Zahtjev može biti udovoljen ili dostavom pakiranja kemikalije koja se već nalazila na zalihi kemijskog laboratorija ili upućivanjem narudžbe za novim pakiranjem kemikalije od vanjskog dobavljača. Osoba koja upućuje zahtjev mora imati mogućnost pretraživanja kataloga kemikalija vanjskog dobavljača dok sastavlja narudžbu. Sustav mora pratiti status svakog zahtjeva za kemikalijama od trenutka kad je ispunjen do trenutka kad je udovoljen ili otkazan. Također, mora pratiti povijest svakog pakiranja kemikalija od trena kad stigne u kompaniju do trenutka kad je potpuno upotrijebljen ili odbačen." ...



Primjer: Procesi sustava ISSP

- ISSP se može podijeliti na sljedeće funkcionalne cjeline:

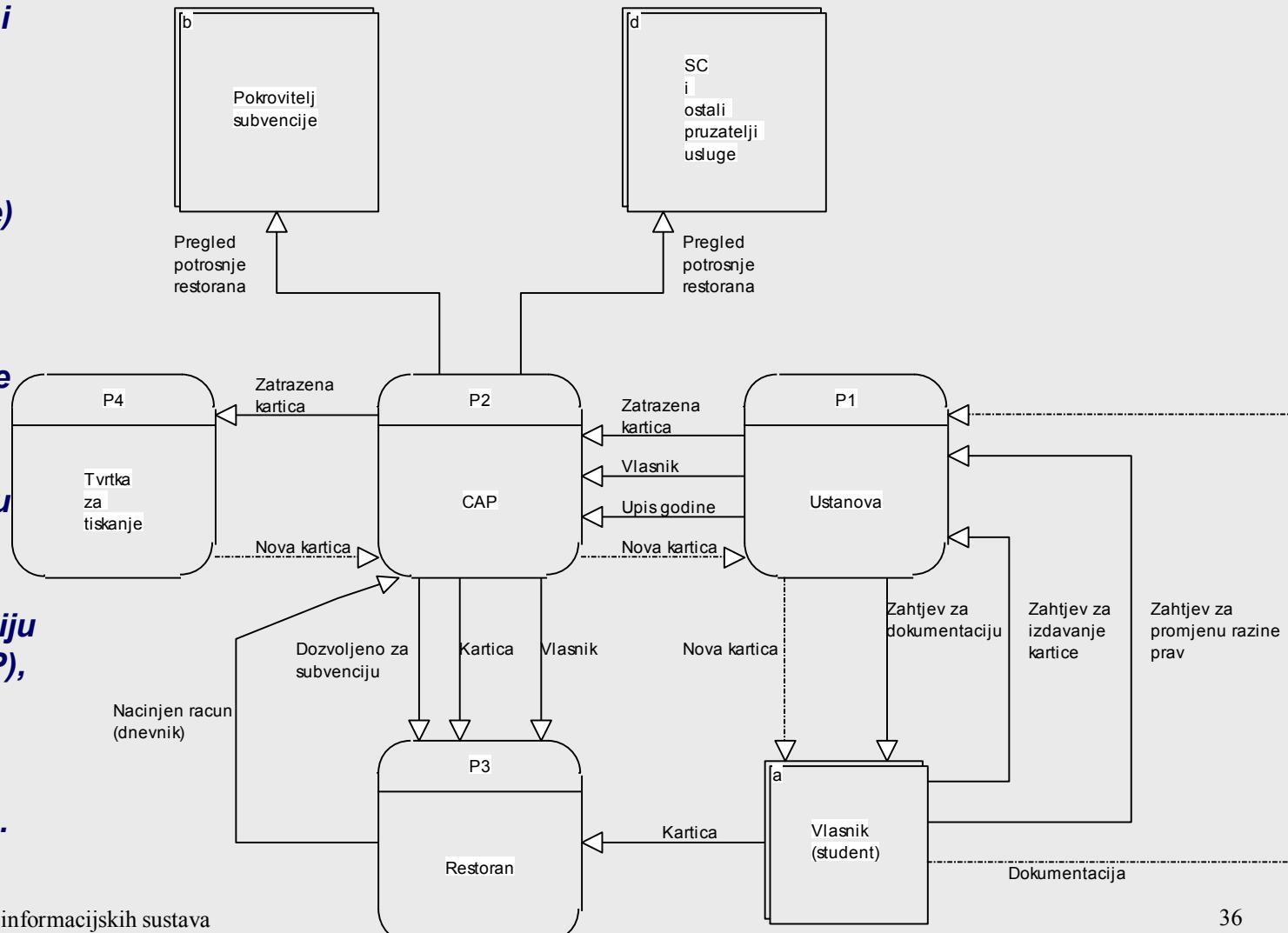
□ Članice sveučilišta i veleučilišta, čiji studenti ostvaruju pravo na subvencioniranu prehranu (ustanove)

□ Restorane koji pružaju uslugu prehrane prehranu članovima ustanove (mesta troška)

□ Tvrta zadužena za održavanje i nabavu opreme te tiskanje iskaznica članova

□ Centar za autorizaciju prava članova (CAP), kao ustanova zadužena za uvođenje i trajno održavanje sustava.

- Unutar svake od cjelina obavlja se više specifičnih procesa koji ...



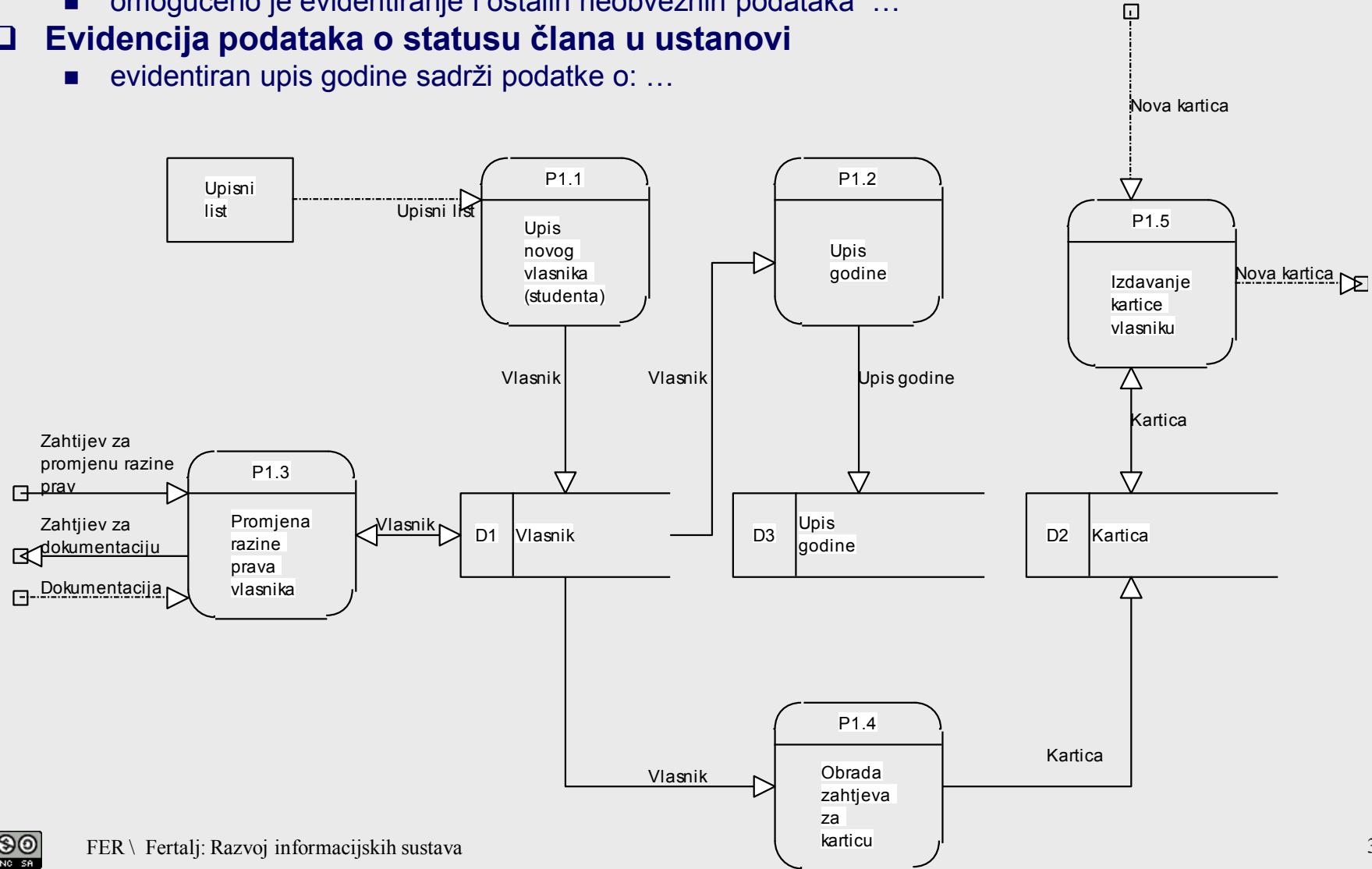
Primjer: razrada procesa ISSP

□ Evidencija matičnih podataka članova

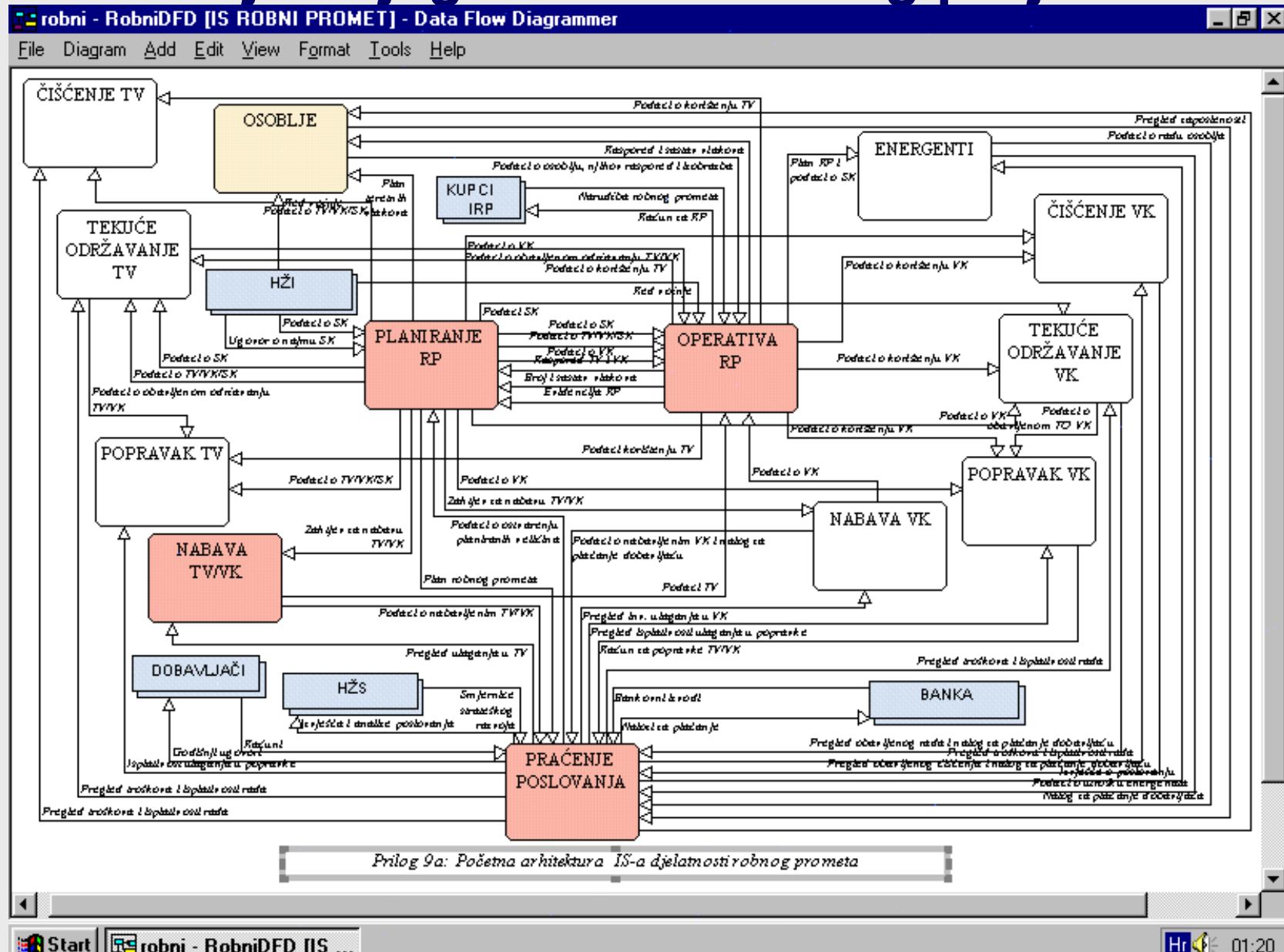
- matični podaci studenta : Ime, Prezime, JMBG, Matični broj (broj indeksa), Razina prava, JMBAG
- omogućeno je evidentiranje i ostalih neobveznih podataka ...

□ Evidencija podataka o statusu člana u ustanovi

- evidentiran upis godine sadrži podatke o: ...



Primjer dijagrama iz stvarnog projekta



Modeliranje događaja

Događaji

□ Događaj

- zgoda, zbivanje u sustavu koja vodi ili pokreće procese sustava
- sâm događaj nije proces, nego okidač procesa koji se njime pokreće

□ primjer: kupac dostavi narudžbu, čime pokreće

- proces provjere da li se radi o narudžbi postojećeg ili novog kupca,
- proces stvaranja podataka o narudžbi i stavkama narudžbe,
- provjeru prethodnih zaduženja kupca,
- provjeru stanja skladišta
- itd.

Vrste događaja

□ vanjski događaji

- potaknuti od strane vanjskih entiteta, koji zahtijevaju informaciju ili ažuriranje podataka → ulazni tokovi podataka
- naziv sadrži naziv vanjskog entiteta
- primjer: zahtjev za upis studenta ili zaprimanje narudžbe kupca

□ vremenski događaji

- vremenski uvjetovani (rok, učestalost) → ulazni upravljački tokovi
- naziv sadrži vremensku oznaku
- primjer: istek roka plaćanja računa, mjesecni obračun plaća, zaključivanje ispitnog roka

□ unutarnji događaji, događaji stanja

- posljedica prijelaza sustava iz jednog stanja u drugo, na način da to zahtjeva obradu → ulazni upravljački tokovi
- primjer: isporuka robe sa skladišta zahtjeva naručivanje nove robe

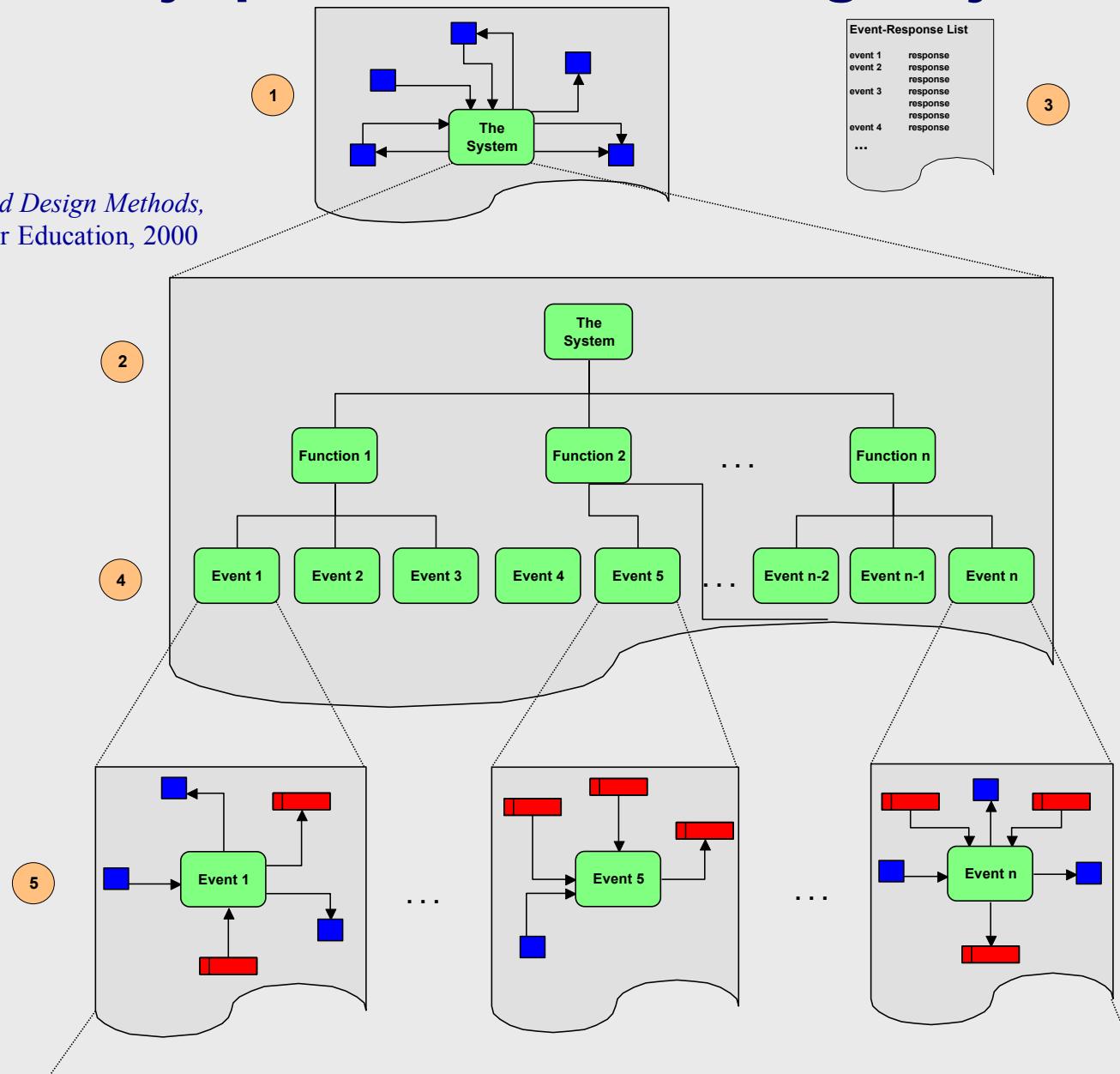
Modeliranje procesa vođeno događajima

□ Raspodjela događaja (event partitioning)

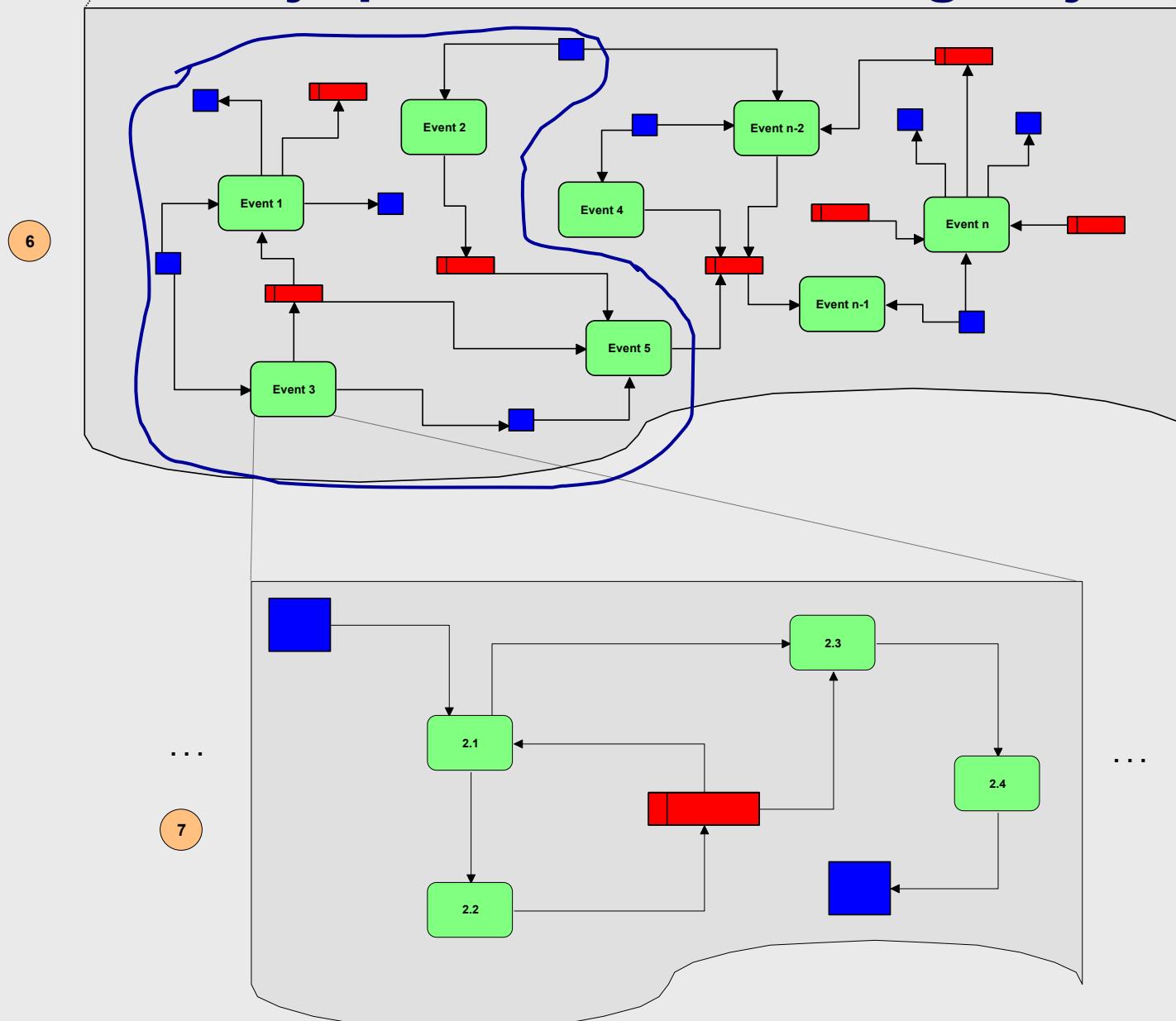
1. Izrada dijagrama konteksta sustava
 - postavljanje početnog dosega projekta
2. Izrada dijagrama funkcionalne dekompozicije
 - podjela sustava u logičke podsustave i/ili funkcije
3. Izrada popisa događaja i odziva
 - utvrđivanje poslovnih događaja na koje sustav mora odgovoriti
 - element popisa = događaj, ulaz, izlaz
4. Izrada dijagrama dekompozicije događaja
 - dodavanje procesa za rukovanje događajima (event handler)
 - dodaje se po jedan proces za svaki utvrđeni događaj
5. Izrada dijagrama događaja
 - razrada procesa za obradu događaja
 - po jedan dijagram za svaki svaki događaj
6. Izrada dijagrama sustava
 - udruživanjem dijagrama događaja
7. Izrada primitivnih dijagrama
 - razrada DTP koji sadrži osnovne procese, spremišta i tokove za svaki pojedini događaj

Modeliranje procesa vodeno događajima (2)

Slika: Whitten et.al,
Systems Analysis and Design Methods,
McGraw-Hill Higher Education, 2000



Modeliranje procesa vodđeno događajima (3)



Tablica događaja u sustavu

□ Događaj

- Događaj zbog kojeg sustav obavlja određene aktivnosti

□ Okidač

- Kako sustav zna da se dogodio događaj?
- Za vanjske događaje, to budu podaci koji ulaze u sustav.
- Za vremenski uvjetovane događaje - trenutak koji aktivira djelovanje sustava

□ Izvor

- Za vanjski događaj - vanjski uzročnik, tj. izvor podataka koji ulaze u sustav

□ Aktivnosti

- Što sustav radi u slučaju događaja?

□ Odgovor

- Koji/kakav izlaz (ako postoji) izlazi iz sustava

□ Odredište

- Kamo ide odgovor?

□ Primjeri: \Modeliranje\StrukturiranaAnalizaDizajn

Primjer: tablica događaja i procesa

Događaj	Okidač	Izvor	Aktivnosti	Odgovor	Odredište
1. Kupac želi provjeriti raspoloživost artikla	Upit o proizvodu	Kupac	Provjeriti raspoloživost	Podaci o raspoloživosti	Kupac
2. Narudžba	Nova narudžba	Kupac	Stvoriti novu narudžbu	Real-time veza	Kreditni ured
				Potvrda narudžbe	Kupac
				Podaci o narudžbi	Otpremništvo
				Transakcija	Banka
...					
4. Vrijeme za zatražiti izvještaje o poslovanju	Kraj tjedna, mjeseca, godine		Napraviti izvještaje	Izvještaji	Uprava
5. Vrijeme za izraditi izvještaje o transakcijama	Kraj dana		Napraviti izvještaje o transakcijama	Izvještaji o transakcijama	Računovodstvo
6. Kupac, ili uprava, želi provjeriti stanje narudžbe	Upit o stanju narudžbe	Uprava ili Kupac	Ispitati stanje narudžbe	Podaci o narudžbi	Uprava ili Kupac

Matrični prikaz modela događaja

□ Matrica Entiteti/Događaji

- Entity/Event Matrix, Event/Enquiry, Entity Access Matrix
- pogled usmjeren događajima (event oriented view)
- elementi koji prikazuju učinak događaja na entitete
 - Stvaranje: C (create)
 - Čitanje: R (read) - u nekim metodama se ne bilježi
 - Ažuriranje: U (update) ili M (modify)
 - Brisanje: D (delete)
- provjera dovršenosti:
 - svaki događaj mora imati učinak na barem jedan entitet
 - svaki entitet mora imati događaj koji čita, stvara, mijenja ili briše

□ Primjer: pojednostavnjena rezervacija sobe u hotelu Proljeće

Događaj/Entitet	korisnik	rezervacija sobe	zaduženje	soba	...
privremena rezervacija	C/M	C			
potvrda rezervacije		C/M			
opoziv rezervacije		M			
dolazak gosta	C/M	C/M		M	
povećanje troškova		M	C		
plaćanje usluga		M	M	M	
arhiviranje rezervacije		D	D		

Vrste matrica

□ U primjeni su različite vrste prikaza, ovisno o metodologiji

- matrica entiteti/događaji (Entity/Event Matrix), npr. SSADM
 - odnos događaja i entiteta: C (create), M (modify), D (delete)
- matrica funkcije/entiteti (Function/Entity Matrix), npr. IEM
 - odnos obrade i podataka: C (create), R (read), U (update), D (delete)
- matrica funkcije/događaji (Function/Event Matrix), npr. SSM
 - odnos obrade i događaja: X (povezanost) ili praznina

□ Slične matrica može se iskoristiti za definiranje prava

- matrica Korisnik / Proces / X
- matrica Korisnik / Entitet / CRUD
 - Korisnik predstavlja grupu korisnika ili ulogu korisnika
 - Korisnik može biti i konkretna osoba

Primjer: matrica entiteti/događaji

□ Primjeri:  Modeliranje\StrukturiranaAnalizaDizajn

	Katalog	Mušterija	Raspoloživ artikl	Narudžba	Naručeni artikl	Transakcija (narudžba)
Provjeriti raspoloživost artikla			R			
Stvoriti novu narudžbu		CRU	RU	C	C	C
Ažurirati narudžbu		RU	RU	RUD	RUD	RUD
Provjeriti status narudžbe		RU		R	R	RUD
Zabilježiti ispunjenje narudžbe					RU	
Zabilježiti istek zaliha					RU	
Stvoriti povrat narudžbe		CRU		RU		C
Dati informacije o katalogu	R		R			
Ažurirati korisnički račun		CRUD				
Distribuirati promo pakete	R	R	R			
Prilagodba naplaćene cijene		RU				CRUD
Ažuriranje kataloga	RU		R			
Kreiranje posebne promocije	R		R			
Kreiranje novog kataloga	C		R			

Određivanje podsustava matricom događaja

□ Primjer, prvi dio matrice procesi/entiteti za jedan stvarni sustav

Klase Proces	K3	K54	K55	K38	K30	K39	K5	K31	K40	K7	K32	K41	K8	K42	K14	K63	K36	K50	K37	K46	K53	K12	K26	K61	K60	K13	K6	K29	K52	K15	K22
P9	CRUD	CRUD	CRUD	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R								
P6	CRUD	R	CRUD																												
P10	CRUD	R	R	R	CRUD	CRUD	CRUD																								
P11	CRUD	R	R	R																											
P12	CRUD	R	R	R																											
P8	CRUD		R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R			
P32																															
P44																															
P37																															
P22																															
P20																															
P1																															
P4																															
P3																															
P36																															
P48																															
P45																															
P18																															
P16	R		R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R			
P7	R		R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R			
P24	R																														
P30	R		R		R		R		R		R		R		R		R		R		R		R		R		R		RU		
P25	R																														
P29	R																														
P28																															
P33	R		R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R			
P19	R		R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R			
P14	R		R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R			
P38	R																														
P50																															
P52	R		R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R			
P51	R		R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RU			
P47	R		R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R			
P46																															
P31																															
P15																															
P49																															
P26																															
P13	R																														
P27																															
P21	RU		RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	RU	
P17	R																														
P2																															
P5																															
P35		R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R		
P34	R		R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R		
P41																															
P42																															
P39																															
P40																															
P43																															
P23																															

Dijagram prijelaza stanja

□ Dijagram prijelaza stanja (State Transition Diagram)

- zasniva se na ideji stroja s konačnim brojem stanja (finite state machine), hipotetičkom mehanizmu koji u nekom trenutku može biti u jednom od konačno mnogo diskretnih stanja
- grafički prikaz promjena stanja, tj vremenski zavisnog ponašanja sustava

□ Elementi prikaza:

- stanje – kumulativni rezultat ponašanja nekog objekta (pravokutnik, krug ili elipsa)
- prijelaz – promjena stanja uzrokovana nekim događajem (usmjereni linija od jednog stanja prema drugom)
- događaj – uvjet promjene stanja i akcija koja se obavlja (opis linije prijelaza oblika događaj/akcija)

□ DPS najčešće opisuje vremenski zavisno ponašanje čitavoq sustava

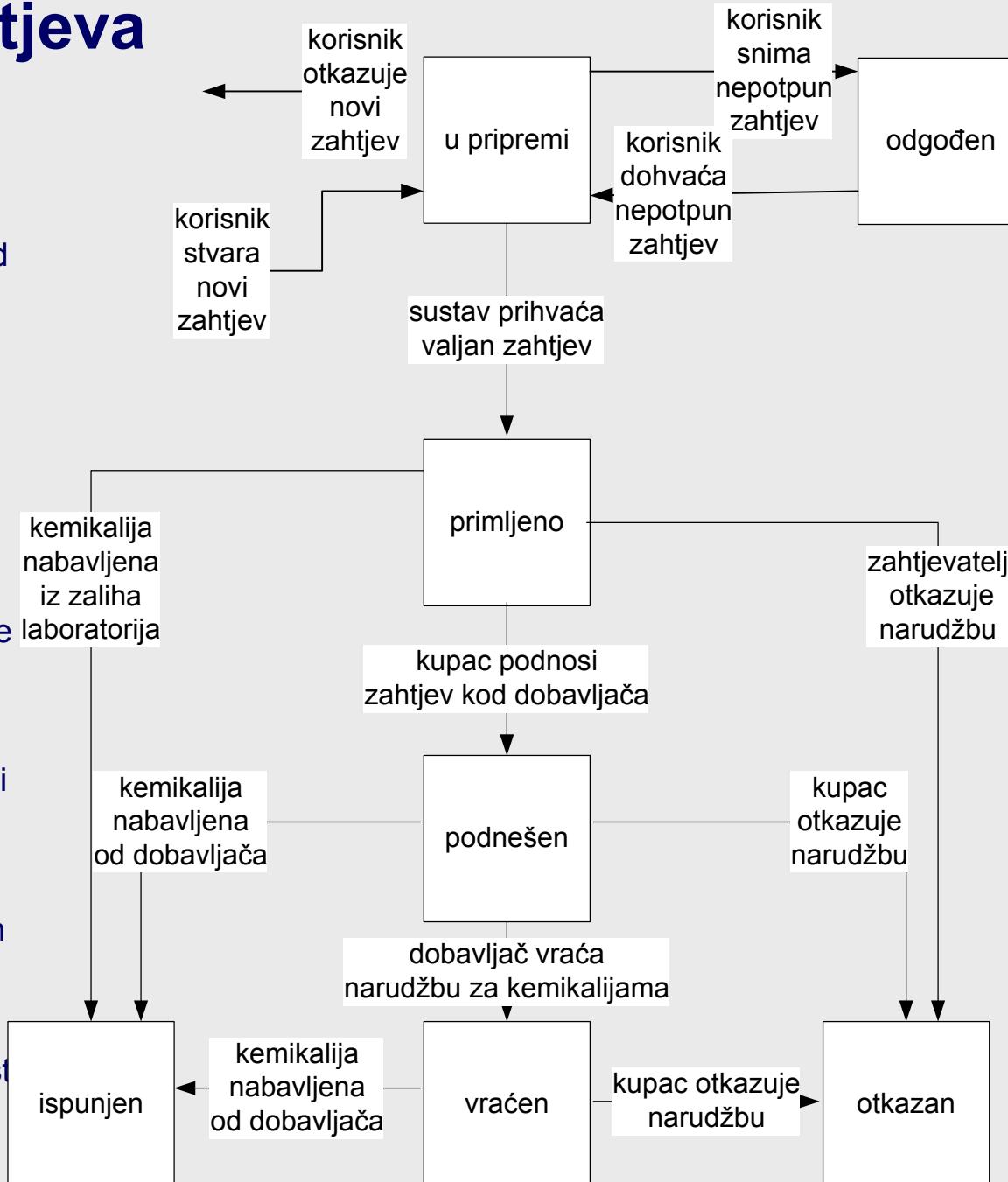
- manji sustavi mogu se prikazati jednim dijagramom
- veći sustavi razlažu se slično dijagramima toka podataka, pri čemu stanje na nekoj razini postaje početno stanje dijagrama na nižoj razini

□ Primjena

- sustavi za rad u stvarnom vremenu (real-time system)
- jezična analiza (parsing)
- dizajn korisničkog sučelja

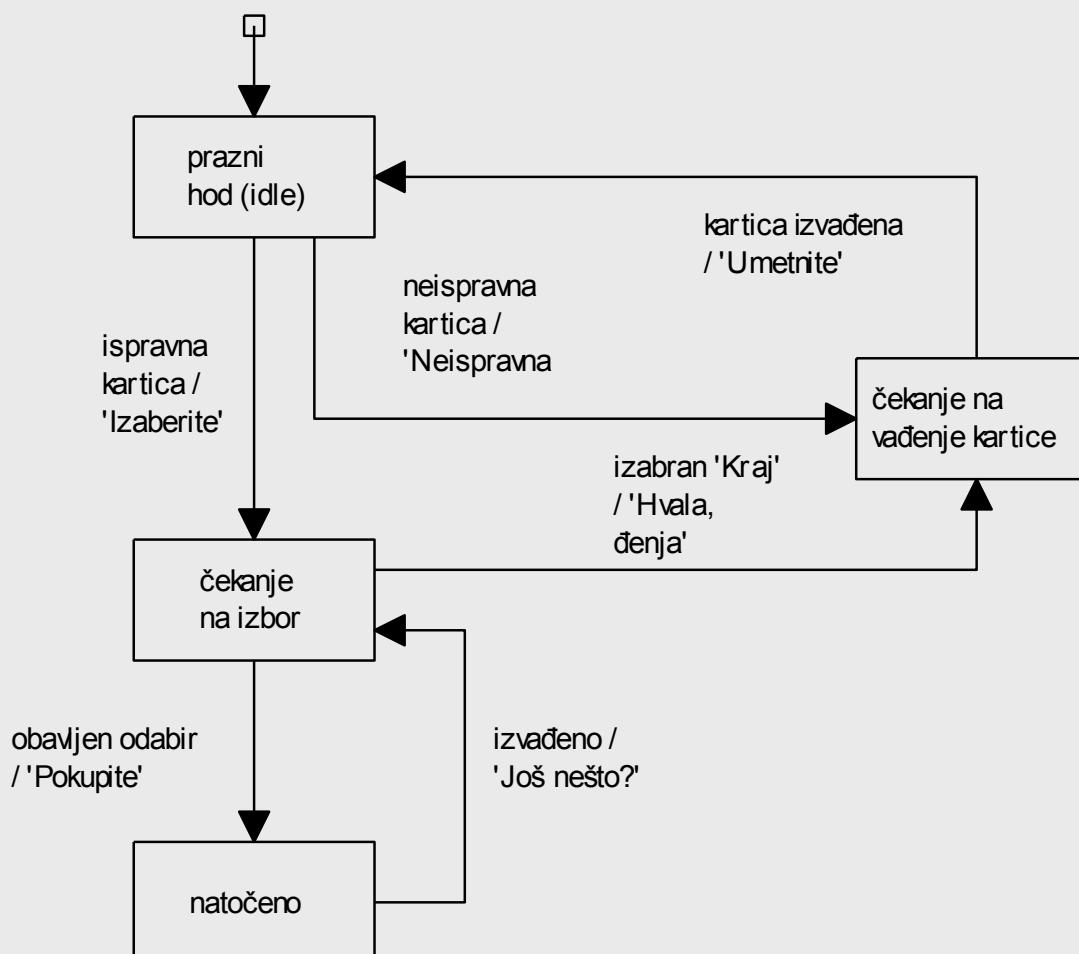
Primjer: Stanja zahtjeva

- U pripremi: Zahtjevatelj stvara novi zahtjev, pokrenuvši tu funkciju iz nekog drugog dijela sustava
- Odgođen: Zahtjevatelj je snimio svoj rad (narudžbu) kako bi ju dovršio kasnije bez da ju je predao ili otkazao
- Zaprimljen: Zahtjevatelj je predao svoj zahtjev (narudžbu) koji je sustav prihvatio kao valjan
- Podnesen: vanjski dobavljač je sposoban isporučiti zahtjev i kupac podnosi zahtjev (narudžbu) kod vanjskog dobavljača
- Ispunjen: zahtjev je zadovoljen bilo da je dostavljen iz zaliha kemijskog laboratorija ili da je dostavljen od strane vanjskog dobavljača
- Vraćen: dobavljač nije mogao zadovoljiti narudžbu i vratio je narudžbu uz mogućnost kasnije dostave
- Otkazan: Zahtjevatelj je otkazao primljen zahtjev prije negoli je isporučen ili je kupac otkazao narudžbu kod vanjskog dobavljača prije nego što je dostavio narudžbu ili nakon što je dobavljač vratio narudžbu uz mogućnost kasnije dostave



Primjer: sustav za rad u stvarnom vremenu

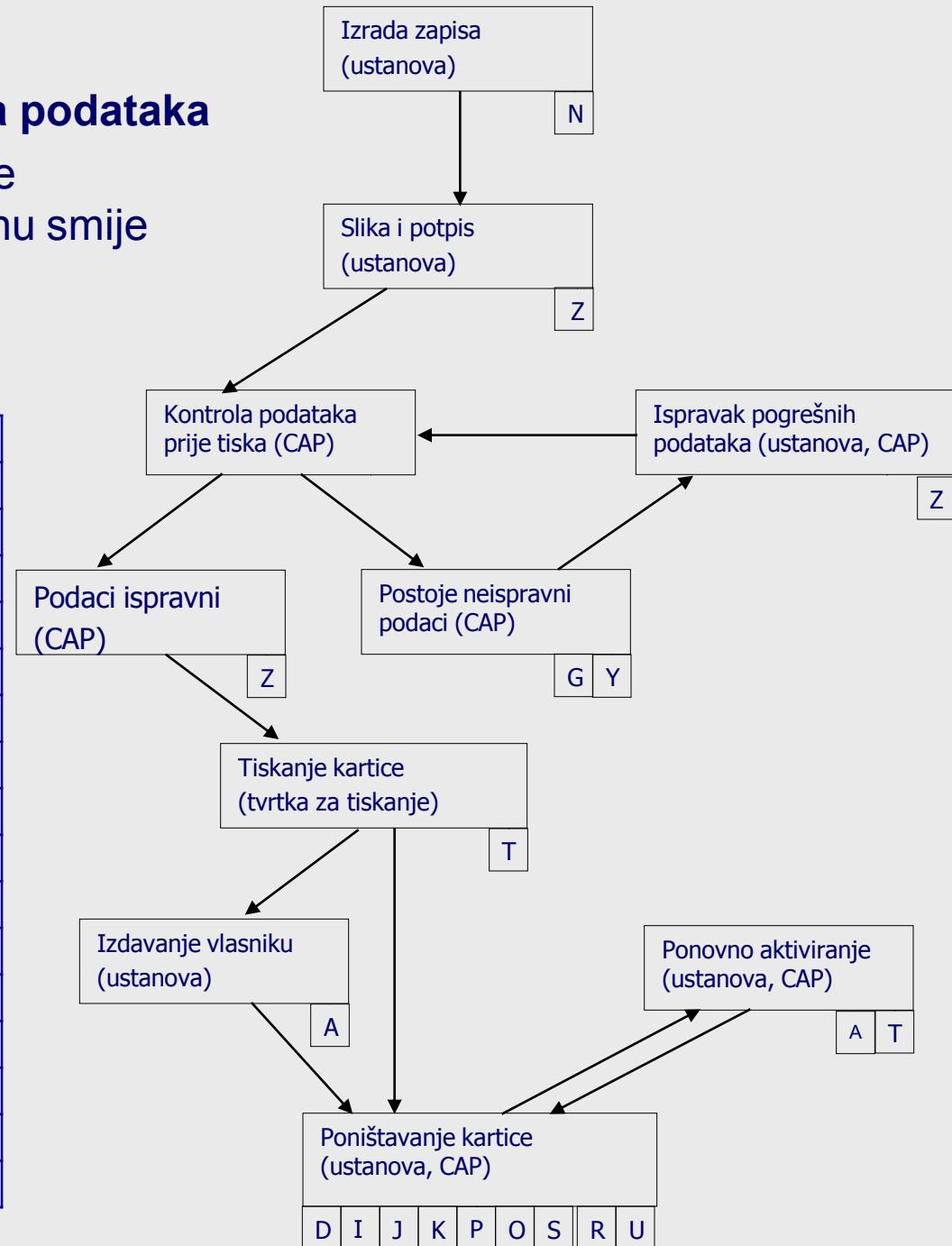
- Dijagrami sustava za rad u stvarnom vremenu razlikuju se po tome što sadrže posebno stanje "besposlen"
 - Npr. Cugomat hotela *Proljeće* ili Bankomat banke *Oprosti nam duge naše*



□ Primjer: prijelazi statusa podataka

- X-ice : u zagradama je navedeno tko promjenu smije načiniti.

A	Izdana
D	Student je diplomirao
G	Pogrešni podaci (koristi samo tvrtka za tiskanje)
I	Izgubljena
J	Poništена zbog promjene JMBG-a
K	Istekao apsolventski rok
N	Nezavršena
O	Oduzeta
P	Poništена
R	Student je upisao prekid
S	Student se ispisao
Š	Oštećena
T	Tiskana
U	Ukradena
X	Nestala
Y	Poništена zbog neispravnih podataka
Z	Zatražena



Mape dijaloga

- **Korisničko sučelje kao konačni automat**
 - Jedan element sučelja (forma, izbornik, dijalog) aktivan u trenutku
 - pristup ograničenom broju drugih elemenata ovisno o akcijama korisnika
 - broj putanji može biti velik, ali je konačan a mogućnosti poznate

- **Mape dijaloga kao analitička tehnika**
 - prikaz dijaloga i navigacije, ali se ne bavi dizajnom zaslona
 - korisnici i razvojnici - usaglašavanje interakcije
 - vizualizacija strukture web sjedišta (eng. site maps)
 - navigacijski linkovi kao prijelazi na mapi dijaloga

Primjena mapa dijaloga

□ Koristi se notacija dijagrama prijelaza stanja

- Uvjet pokretanja navigacije kao tekst na strelici
 - **Korisnička akcija**, npr. pritisak tipke ili klik na link ili gumb dijaloga
 - **Podatkovna vrijednost**, npr. pogrešan unos koji izaziva poruku o pogrešci
 - **Sistemski uvjet**, npr. signal da je pisač ostao bez papira
 - **Kombinacija uvjeta**, npr. tipkanje opcije iz izbornika i pritisak na tipku Enter

□ Izostavljanje općih funkcija radi jednostavnosti

- pritiskanje tipke F1 da bi se dobila pomoć
- standardni navigacijski linkovi (npr. Natrag, Doma)

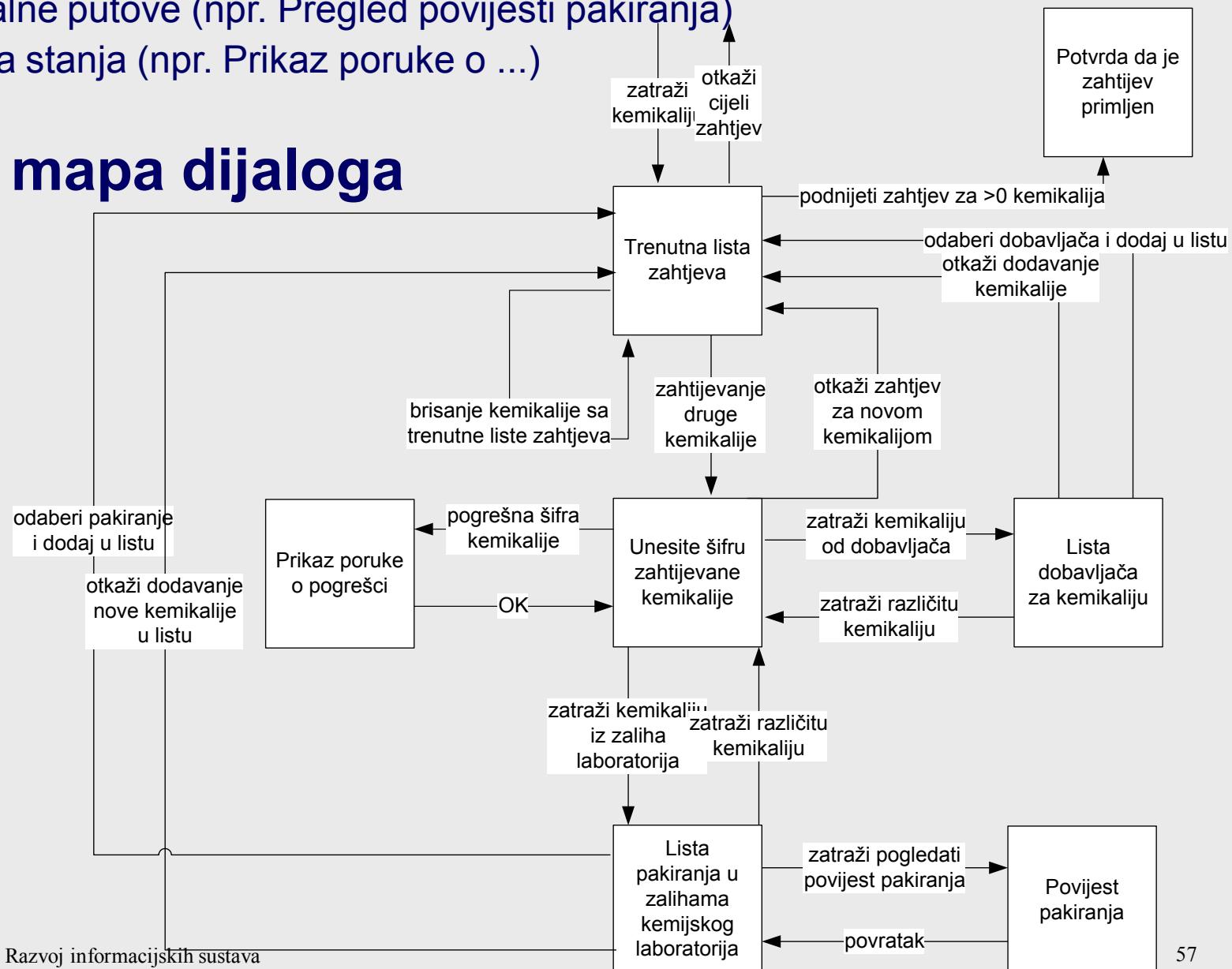
□ Pri analizi mapa predstavlja interakciju na konceptualnoj razini

- Konkretna ugradnja može biti drugačija.
- Primjer:
 - Korisnik inicira slučaj korištenja odabirom opcije "zatraži kemikaliju" iz glavnog izbornika a što se ne vidi na dijagramu.
 - Nije nužno prikazivati detalje (npr. Potvrdu otkazivanja čitavog zahtjeva)
 - Ključno je da korisnik i razvojnik jednako "razumiju" opisanu funkcionalnost.

□ Mape dijaloga mogu prikazati

- Alternativne puteve (npr. naručiti od dobavljača) koji odstupaju od uobičajenog puta (kemikalija iz zaliha).
- Opcionalne puteve (npr. Pregled povijesti pakiranja)
- Posebna stanja (npr. Prikaz poruke o ...)

Primjer: mapa dijaloga





Primjer: debeli klijent

Resursi i primjeri



📁\Modeliranje\RIS-Preporuke

- preporuke za upotrebu različitih notacija izrade dijagrama

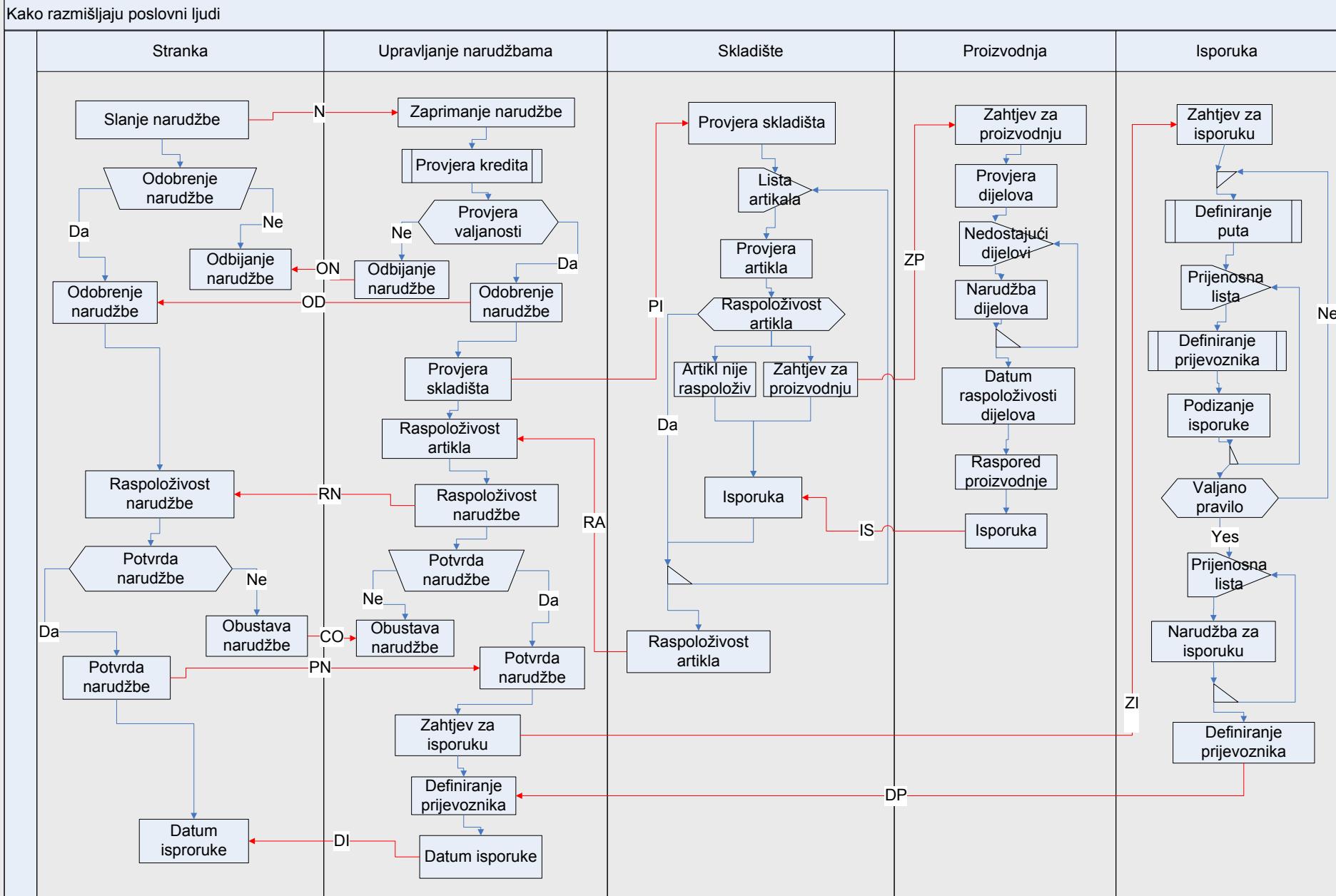
📁\Modeliranje\StrukturiranaAnalizaDizajn

- primjer strukturiranog oblikovanja
- primjer analize izvedivosti
- primjer strukturiranog dizajna programa

📁\Modeliranje\KakoRazmišljajuPoslovniLjudi

- Izvor: Howard Smith - The Third Wave of Business Process Management
- prijevod na sljedećem slajdu

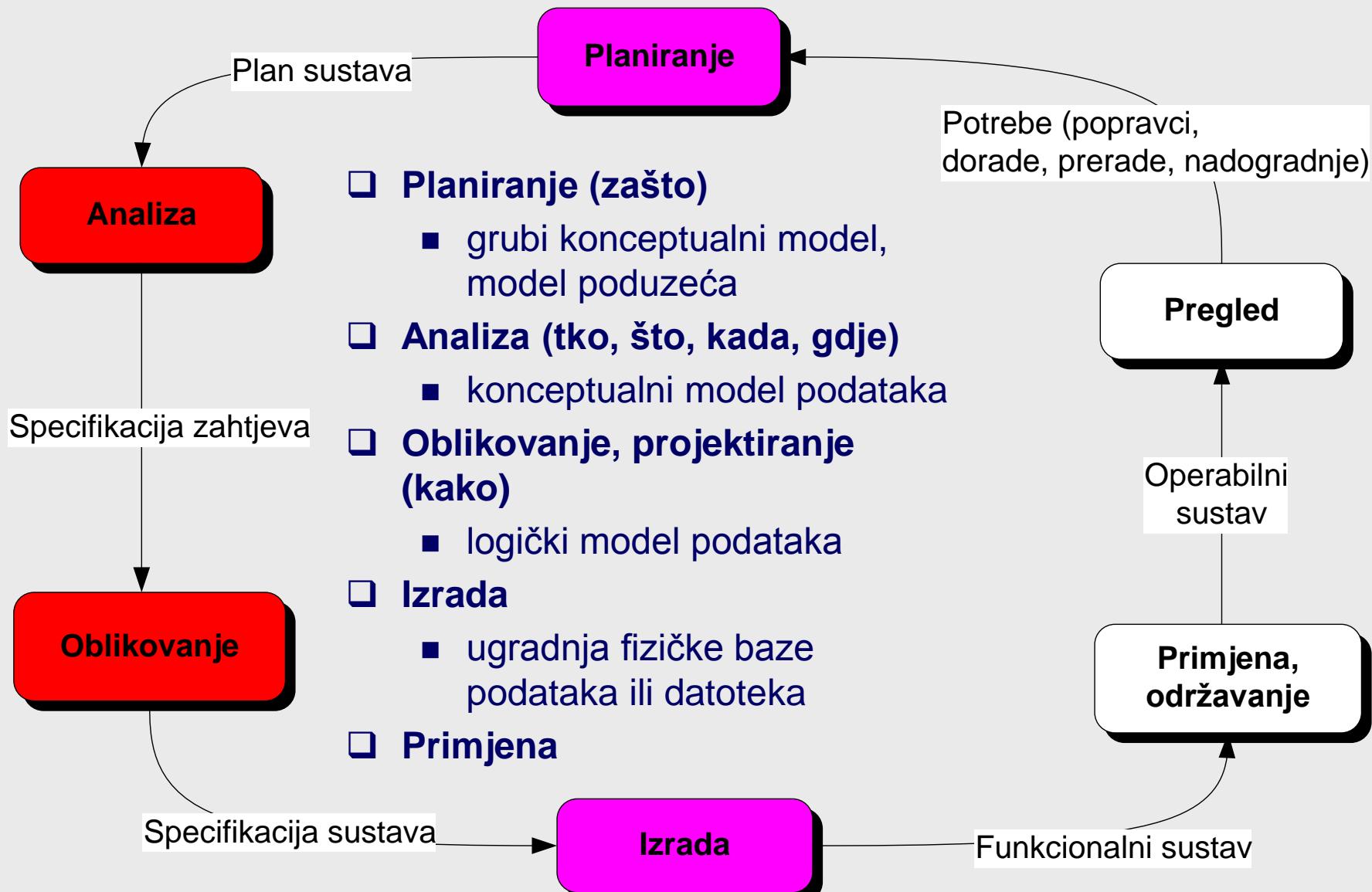
Kako razmišljaju poslovni ljudi





(Konceptualno) Modeliranje podataka

Kontekst modeliranja podataka



Vrste modela podataka

□ Konceptualni

- najčešće model entiteti-veze
- poslovni pogled na podatke
- zanemaruje nekritične detalje
- uglavnom sadrži značajne entitete i veze "više prema više"
- može sadržavati važne attribute i ključeve

□ Logički

- relacijski (najčešće), postrelacijski, objektno-relacijski
- uobičajeno podliježe relacijskoj teoriji
- potpuno definirani attribute i domene (logički tipovi vrijednosti)
- potpuno definirani ključevi
- potpuno normalizirani entiteti (samo veze 1:N)

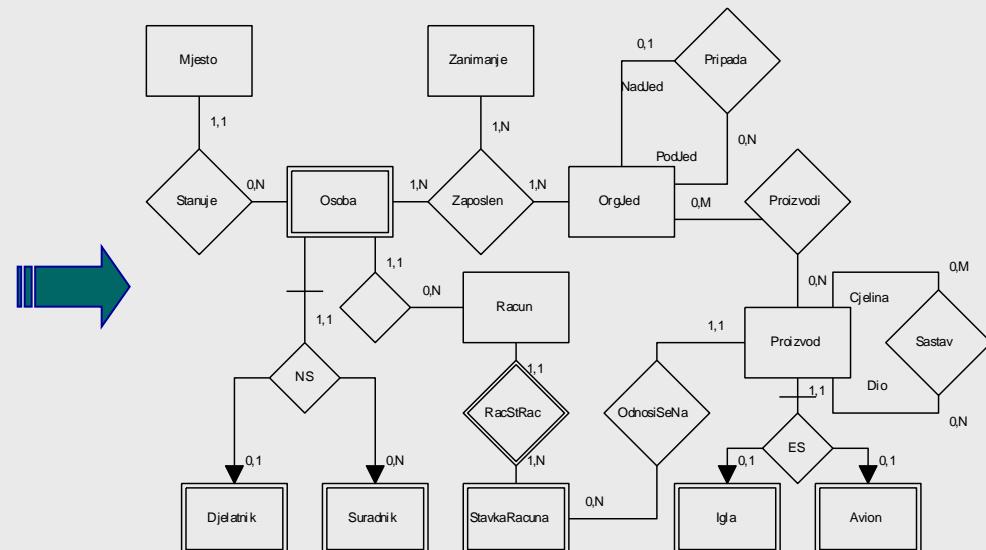
□ Fizički

- logički model preveden u model podataka za odabrani SUBP
- stvarni tipovi, indeksi, pogledi, procedure, ... te fizički parametri

Model entiteta-veze

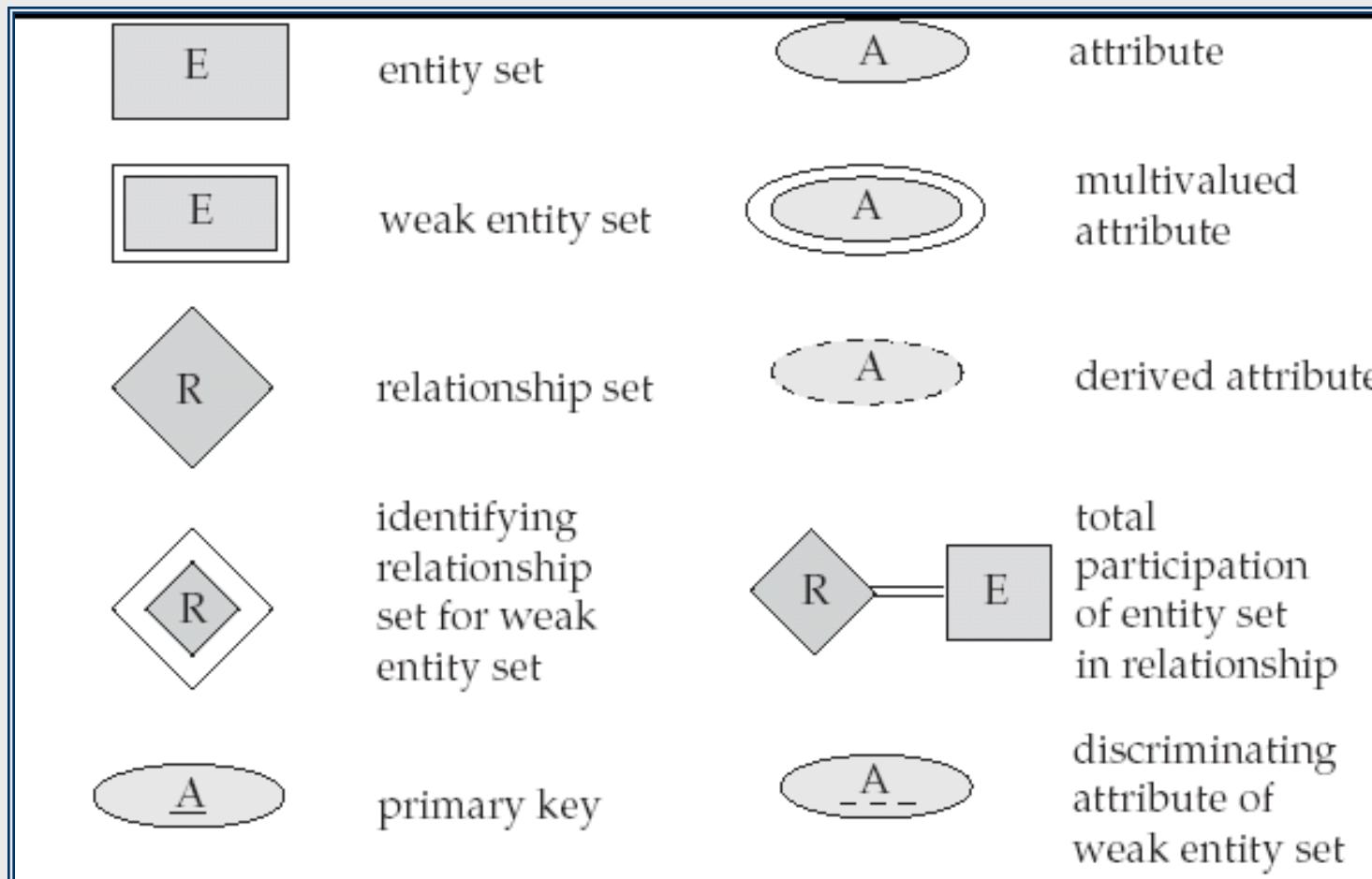
□ Model entiteta-veze

- model podataka koji opisuje entitete (pojavnosti), njihove atributе i veze.
 - "The entity-relationship model: toward a unified view of data", P.P. Chen, ACM Transactions on Database Systems 1:1 pp 9-36, 1976.
- za prikaz se koristi dijagram entiteta-veze (Entity-Relationship Diagram, ERD)
 - naziva se još i dijagram objekti-veze (izbjegavati!)
 - postoje različite notacije, npr. Chen, Martin, ...
- ne postoje jednoznačni standardi postupka izrade



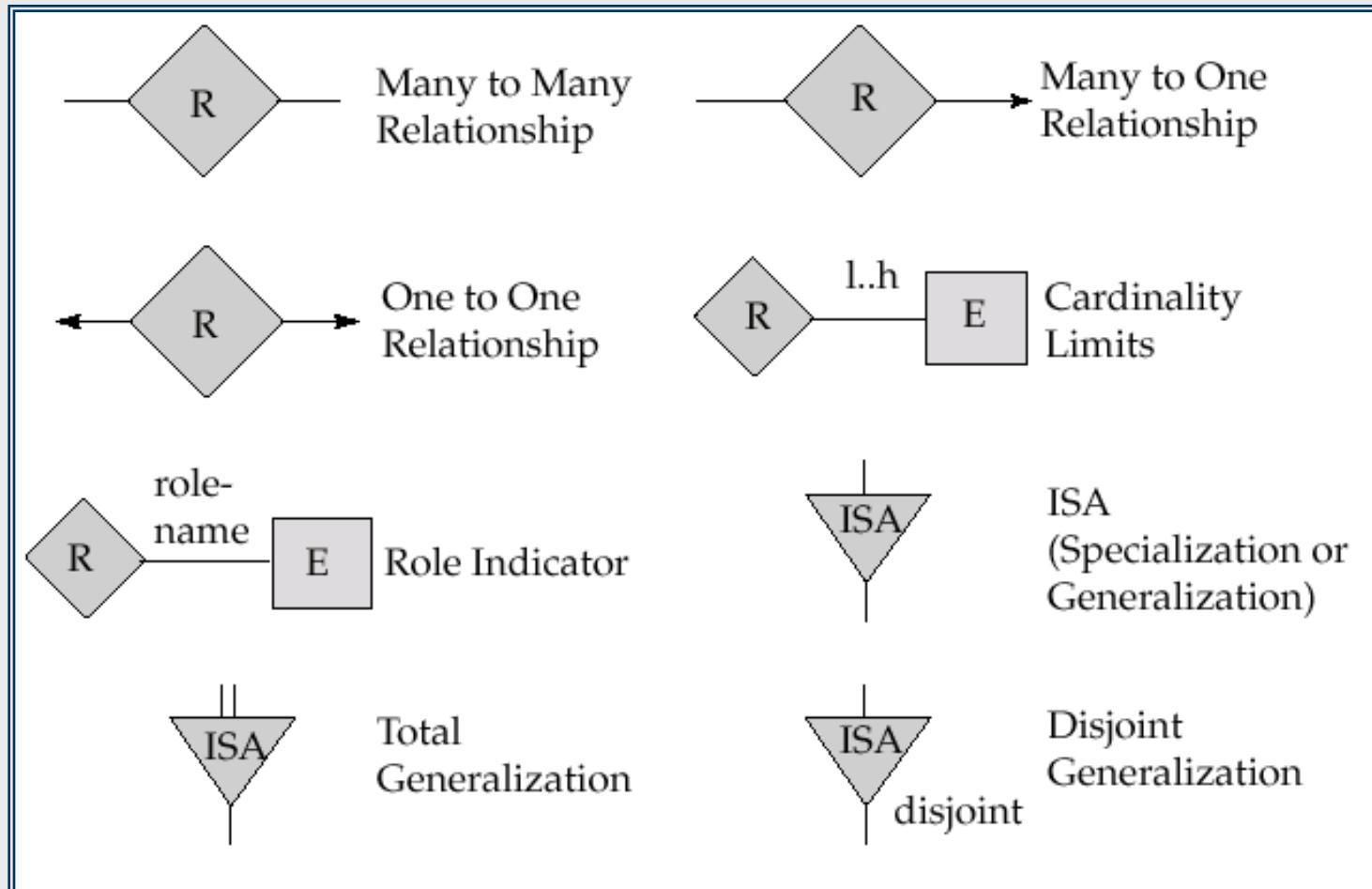
Proširena Chenova notacija (1)

- Potpuna participacija – svaki entitet mora sudjelovati u barem jednoj vezi



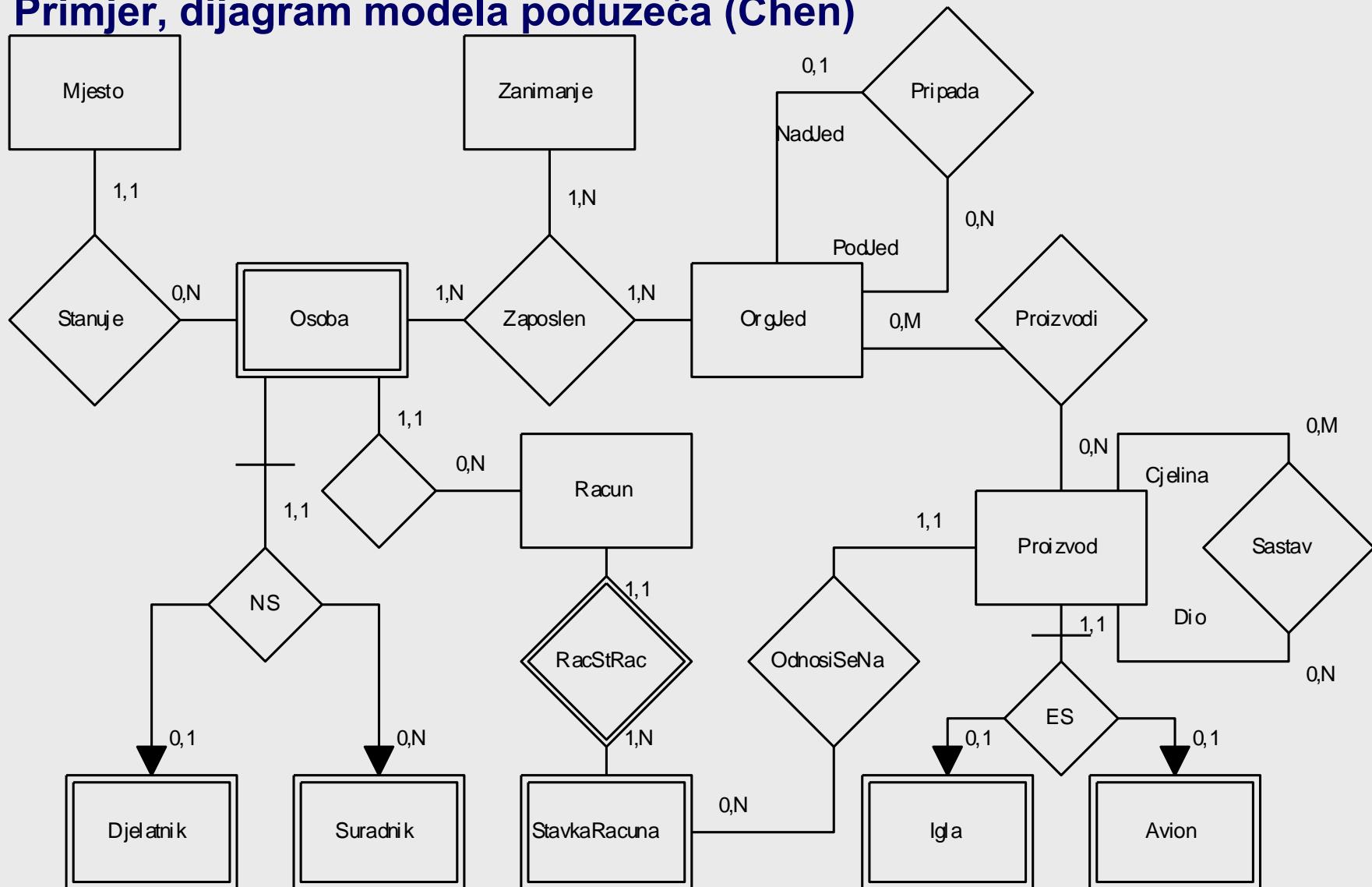
Proširena Chenova notacija (2)

- Potpuna generalizacija – entitet se mora specijalizirati



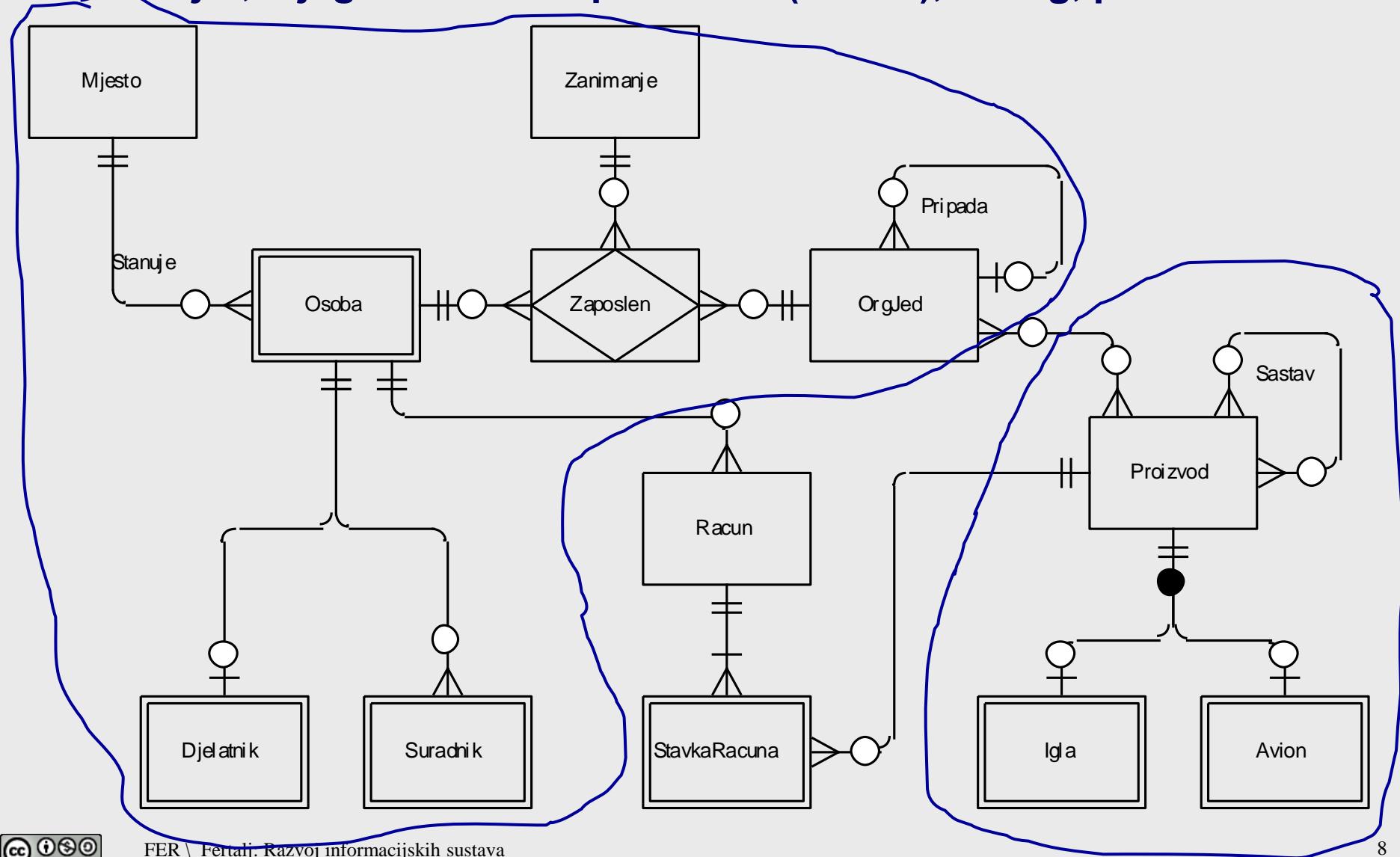
ERD notacija - Chen

□ Primjer, dijagram modela poduzeća (Chen)



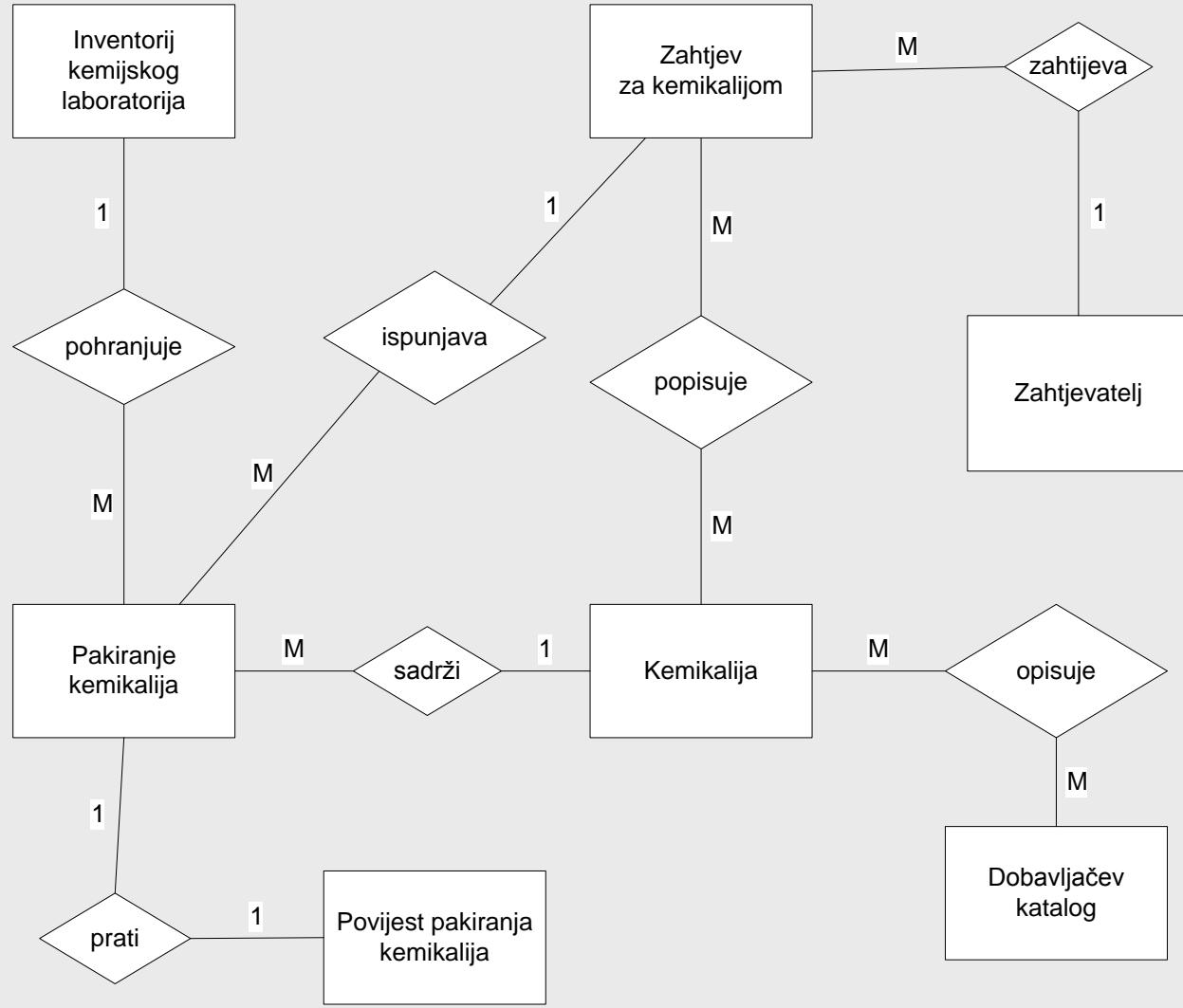
ERD notacija - Martin

□ Primjer, dijagram modela poduzeća (Martin), doseg, podsustavi



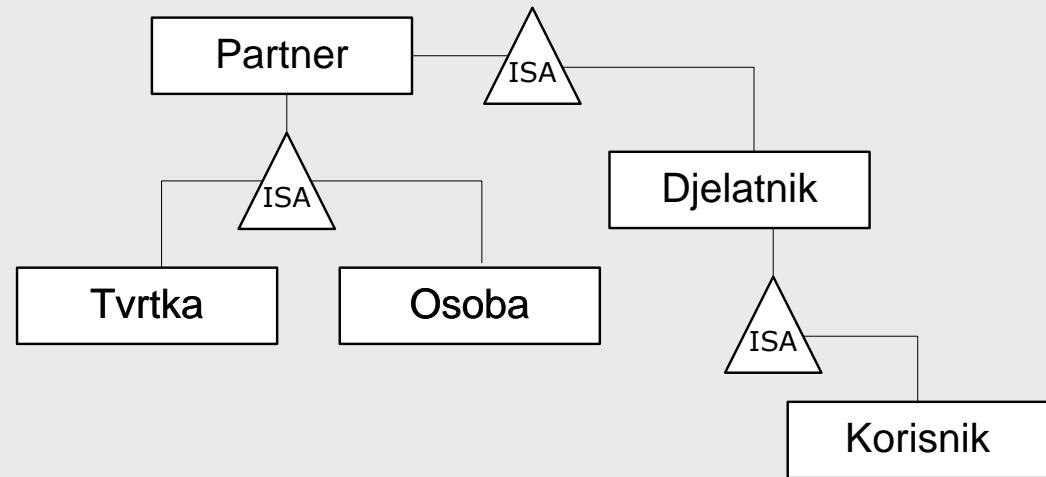
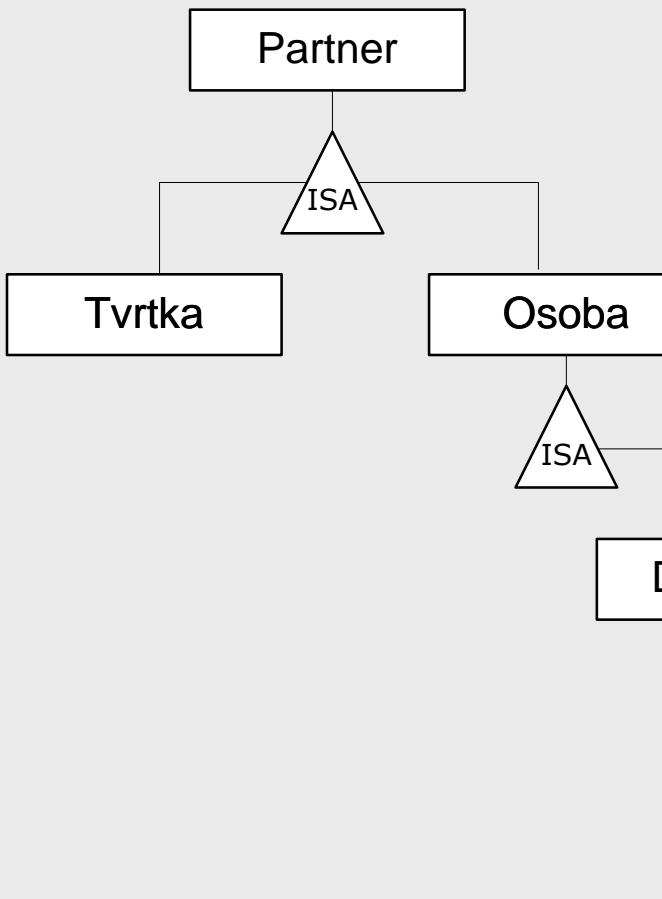
Izrada ERD analizom izjava korisnika

"Kemičar ili član osoblja kemijskog laboratorija može podnijeti zahtjev za jednom ili više kemikalija. Zahtjev može biti udovoljen ili dostavom pakiranja kemikalije koja se već nalazila na zalihi kemijskog laboratorija ili upućivanjem narudžbe za novim pakiranjem kemikalije od vanjskog dobavljača. Osoba koja upućuje zahtjev mora imati mogućnost pretraživanja kataloga kemikalija vanjskog dobavljača dok sastavlja narudžbu. Sustav mora pratiti status svakog zahtjeva za kemikalijama od trenutka kad je ispunjen do trenutka kad je udovoljen ili otkazan. Također, mora pratiti povijest svakog pakiranja kemikalija od trena kad stigne u kompaniju do trenutka kad je potpuno upotrijebljen ili odbačen."



Primjer: hijerarhija specijalizacija

□ Diskusija: Može li bolje ?



□ Diskusija: Može li (još) bolje ?

Logičko modeliranje podataka

Pretvorba modela E-V u relacijski model

□ Entiteti

- entitet (skup entiteta) → relacija, npr. Mjesto, Osoba

□ Atributi

- kardinalnost 0 → opcionalna vrijednost (null)
- kardinalnost 1 → zahtijevana vrijednost (not null)
- kardinalnost N → više značni atribut (npr. Osoba.Telefon)
- atribut → atribut, npr. Osoba.Prezime
- izvedeni atribut → atribut pohrane ili se izostavlja, npr. Osoba.Staz
- složeni atribut → atribut (grupiranjem), npr. (dan, mjesec, godina) → datum

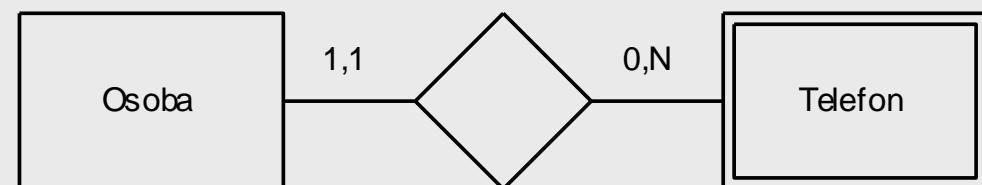
Osoba
IdOsobe
Prezime
Ime
Adresa
DatRod
Staz

Pretvorba višeznačnih atributa

- višeznačni atribut →

- skup odgovarajućih atributa, npr. Osoba.Telefon → Osoba.TelefonNaPoslu, Osoba.TelefonKodKuce, Osoba.MobilniTelefon ili
- slabi entitet, npr. Osoba.Telefon → Telefon (IdOsobe, VrstaTelefona, BrojTelefona)

Osoba
IdOsobe
Prezime
Ime
TelefonNaPoslu
TelefonKodKuce
MobilniTelefon
Adresa
DatRod
Staz



Telefon
VrstaTelefona
BrojTelefona

Diskusija: Koje rješenje, kada ?

Pretvorba ključeva

□ Ključevi

- ključ → primarni ključ, npr. Osoba.IdOsobe, Mjesto.SifMjesta
- alternativni ključ → indeks nad jedinstvenim vrijednostima (unique index) + oznaka zahtijevane vrijednosti (not null), npr. Mjesto.PostBr

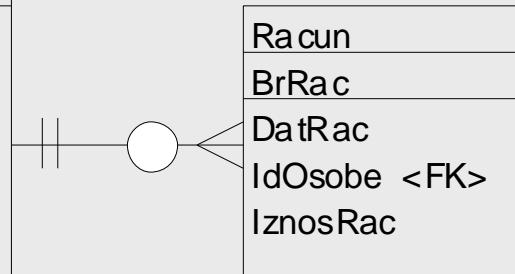
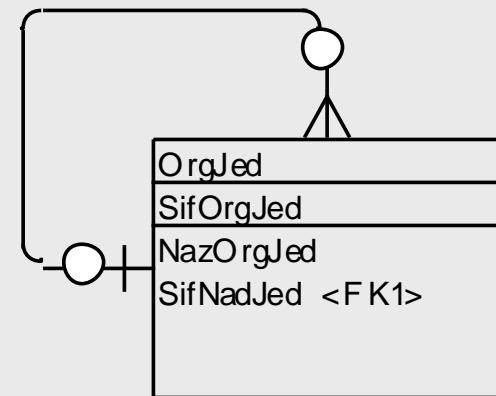
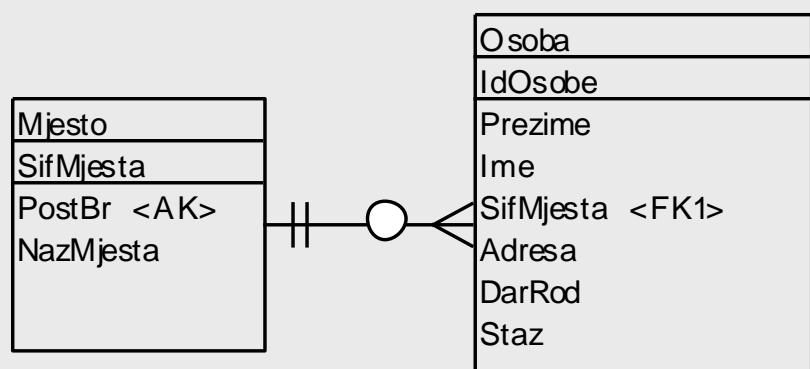
OsobaSKljuce m
IdOsobe
Prezime
Ime
Adresa
DatRod
Staz

Mjesto
SifMjesta
PostBr <AK>
NazMjesta

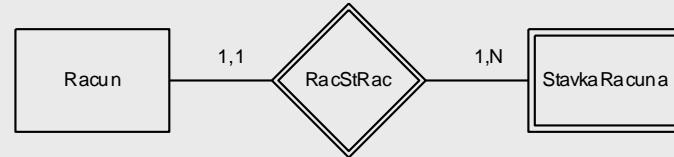
Pretvorba binarnih veza

□ Binarna veza 1:N → strani ključ

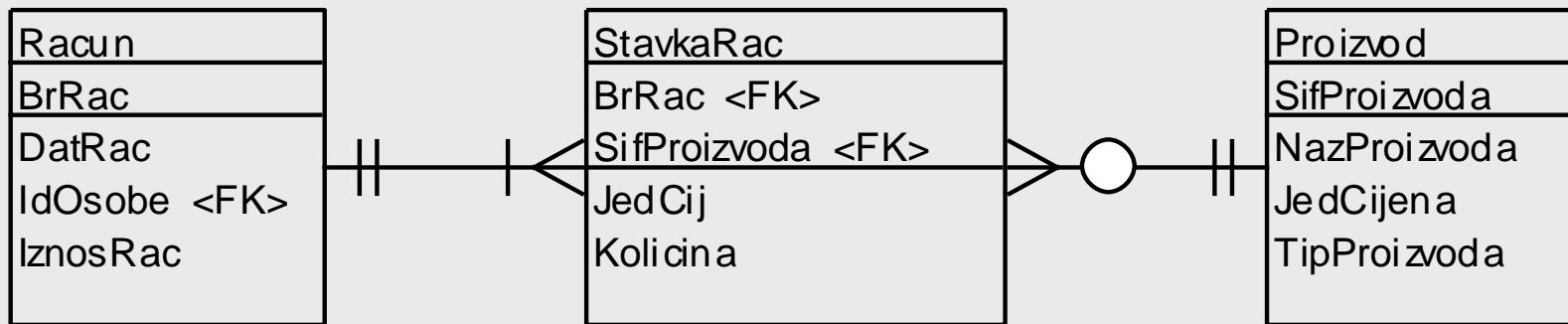
- egzistencijalni slabi entitet → obični strani ključ
 - npr. Stanuje → Osoba.SifMjesta, Pripada → OrgJed.SifNadJed, RacunOsoba → Racun.SifOsobe



Pretvorba identifikacijskih veza



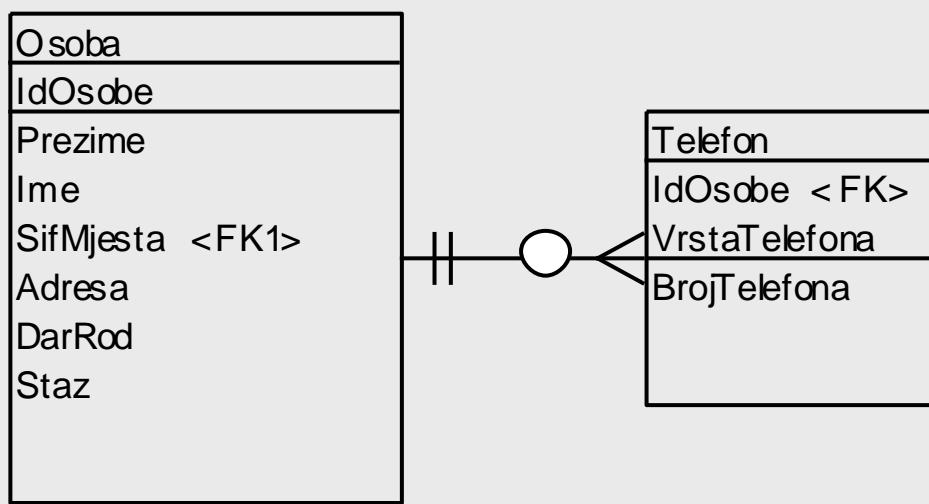
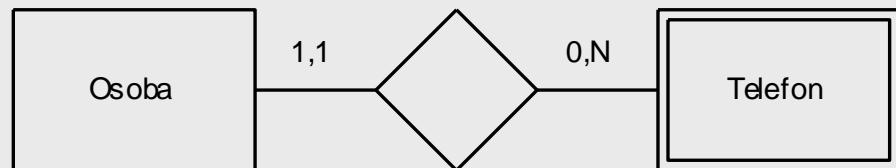
- identifikacijski slabi entitet → nasljeđuje ključ jakog entiteta
 - spojni ključ (compound key), npr. StavkaRacuna (BrRacuna, SifProizvoda, JedCijena, Kolicina)
» ili
 - kompozitni ključ (composite key), npr. StavkaRacuna (BrRacuna, RbrStRac, SifProizvoda, JedCij, Kolicina)



Diskusija: Koje rješenje, kada ?

Pretvorba identifikacijskih veza (2)

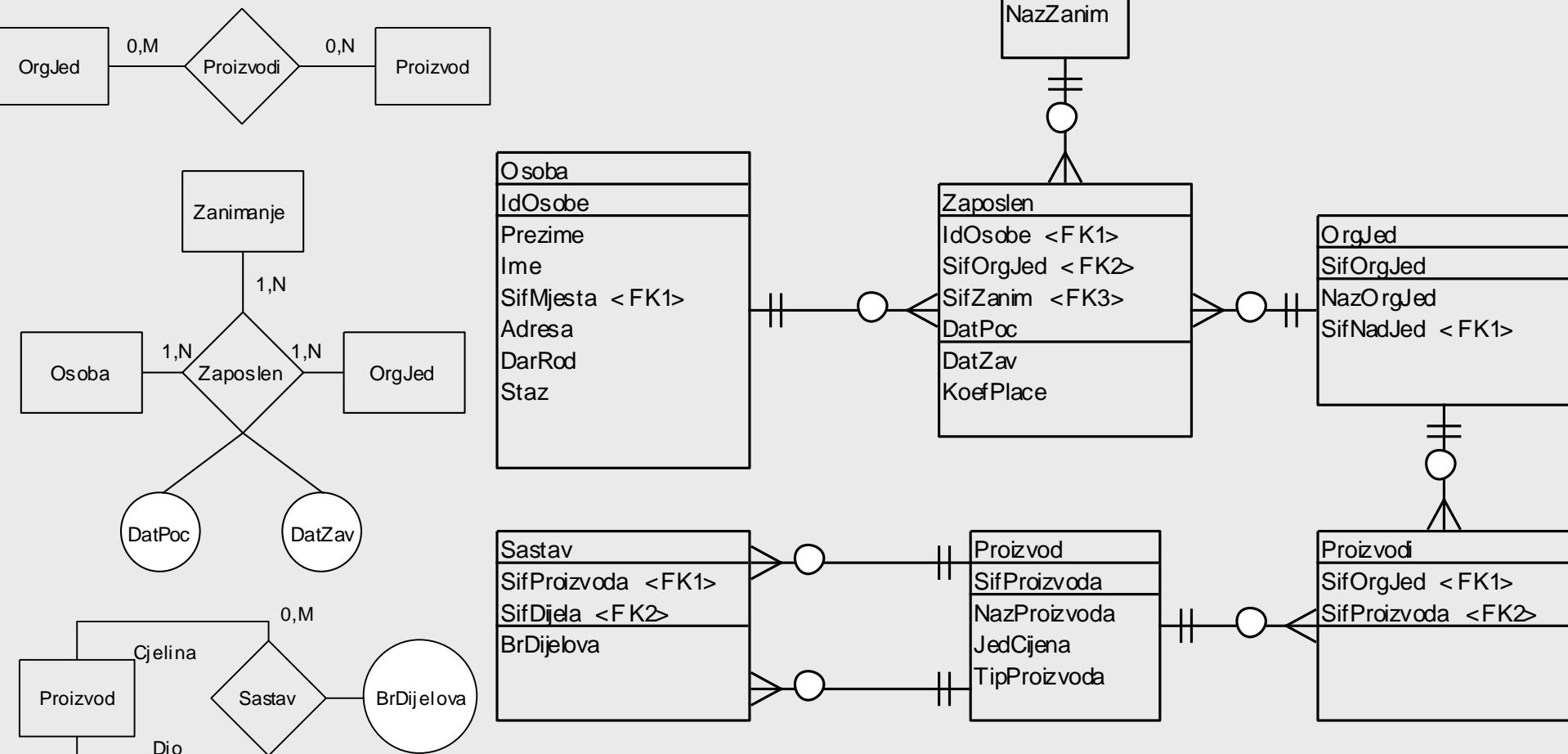
- ❑ Koliko i kakvih telefona osoba može imati ?
- ❑ Uvesti kompozitni ključ (pogledati prethodni slajd) ?
- ❑ Elegantnije rješenje ?



Pretvorba nespecifičnih veza

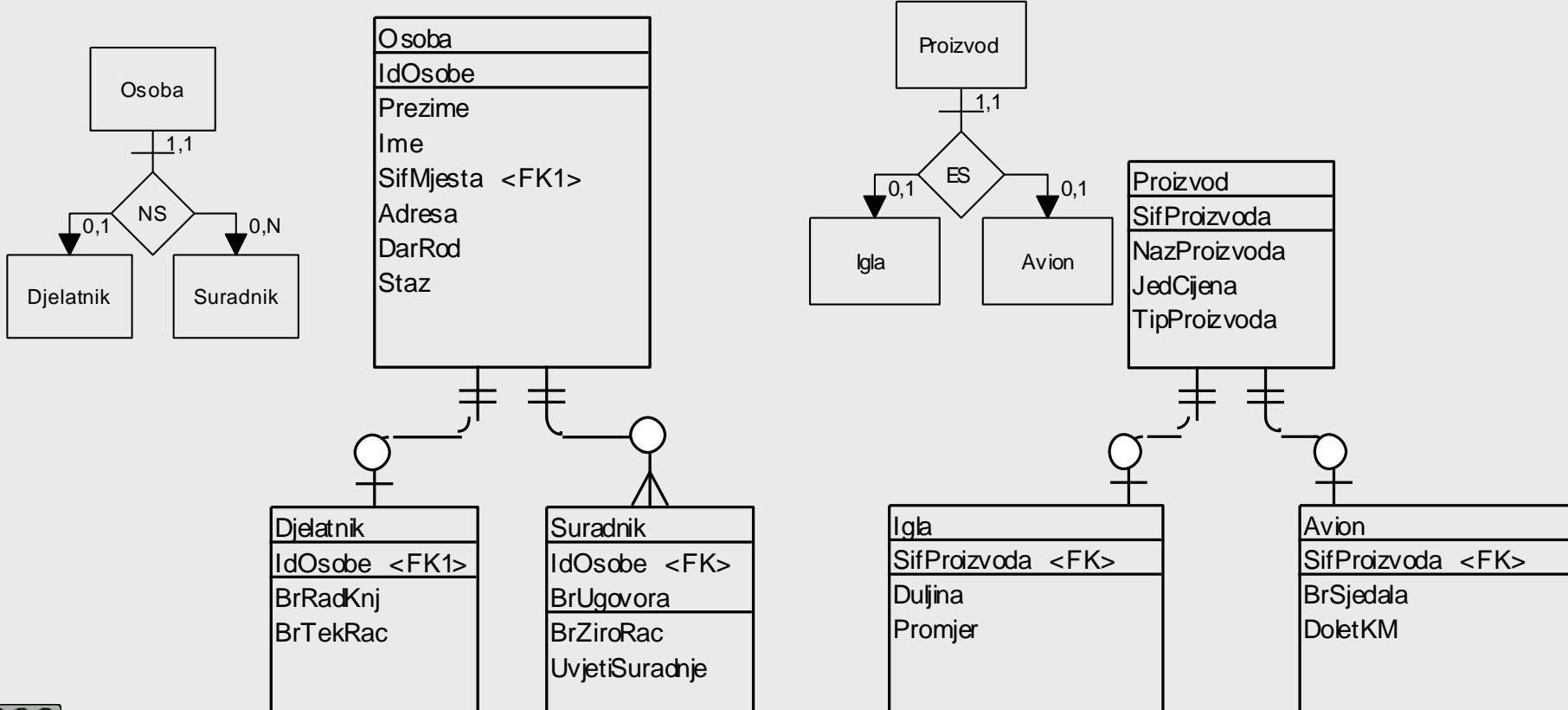
□ Nespecifične veze → asocijativni entitet + n binarnih veza 1:N

- ključ asocijativnog entiteta = unija ključeva entiteta spojenih vezom
 - određivanje ostalih ključnih atributa, ako nije bilo obavljeno ranije
- primjer: Zaposlen, Proizvodi, Sastav



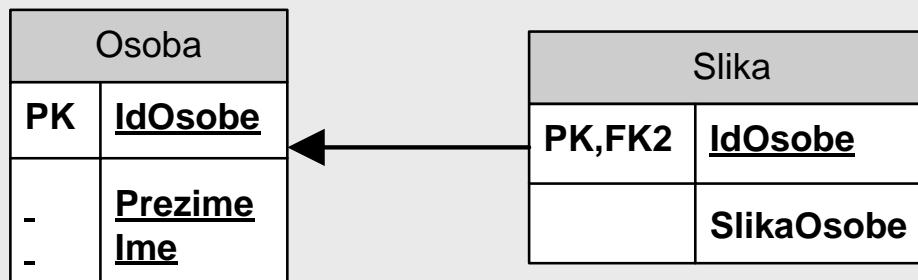
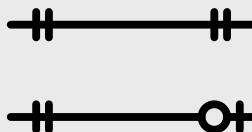
Pretvorba specijalizacija

- Specijalizacija nadtipa u n podtipova $\rightarrow n$ binarnih veza
 - nadtip \rightarrow (jaki) entitet, kojemu se po potrebi određuje klasifikacijski atribut, npr. Proizvod.TipProizvoda
 - podtip \rightarrow (identifikacijski) slabi entitet, npr. Igla, Avion, Djelatnik, Suradnik
- Da li i kada koristiti klasifikacijski atribut (teorijski, ne bi trebalo) ?

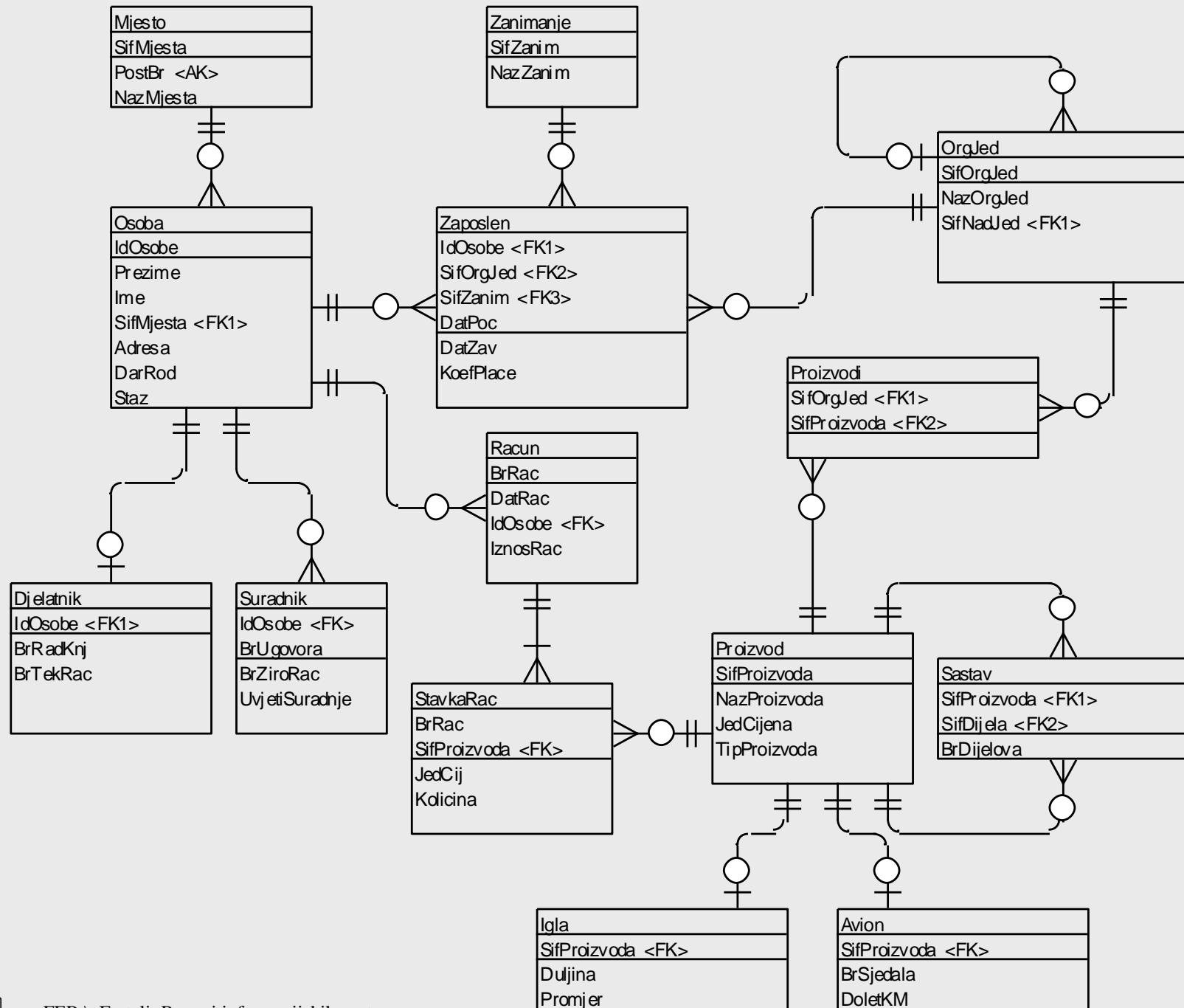


Pretvorba ostalih binarnih veza

- Binarna veza 1,1:0,1 i binarna veza 1,1:1,1 → strani ključ
 - identifikacijski slabi entitet bez deskriptora



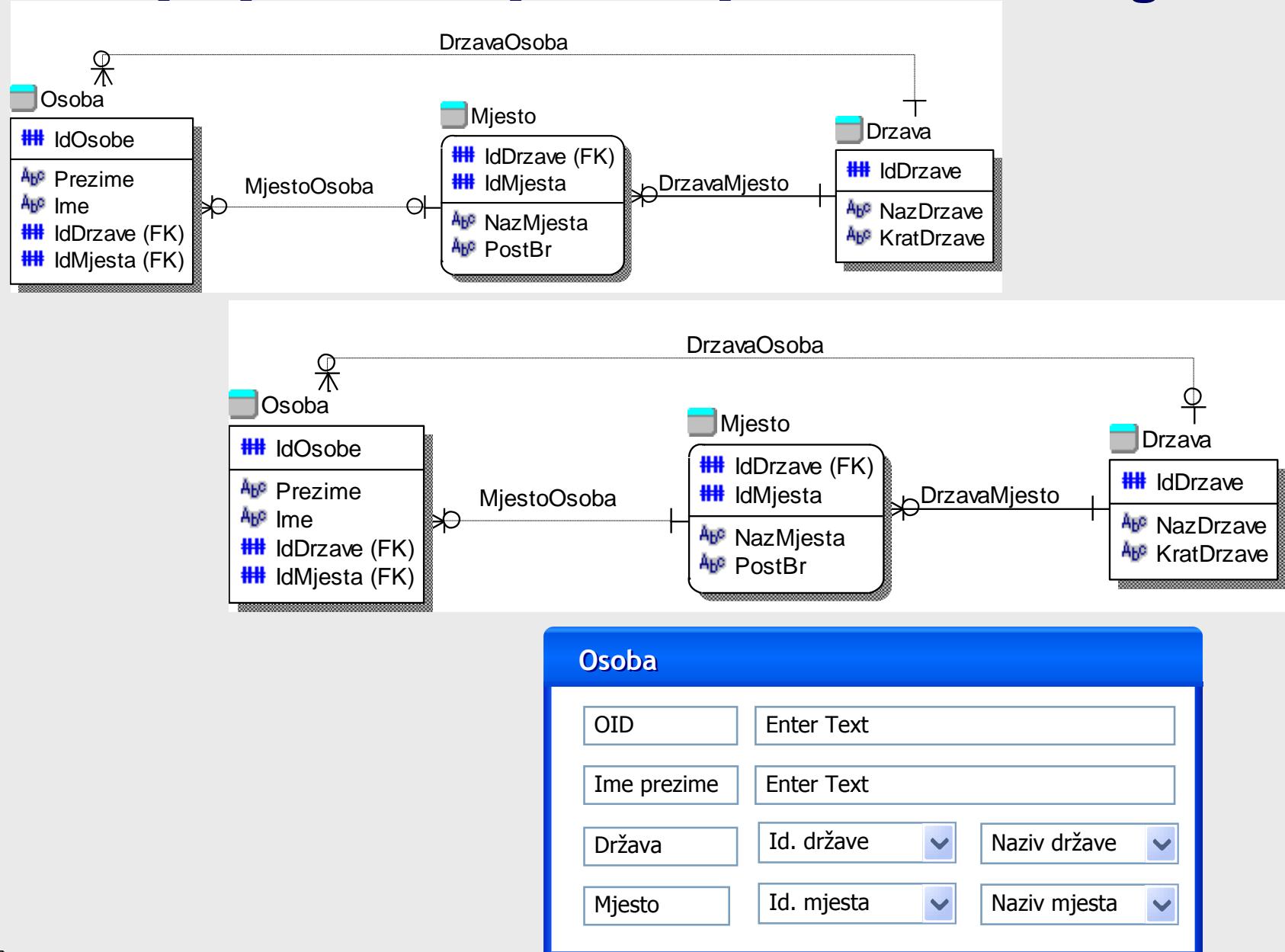
(bez diskusije, bude u nastavku)



Preporuke za izradu modela podataka

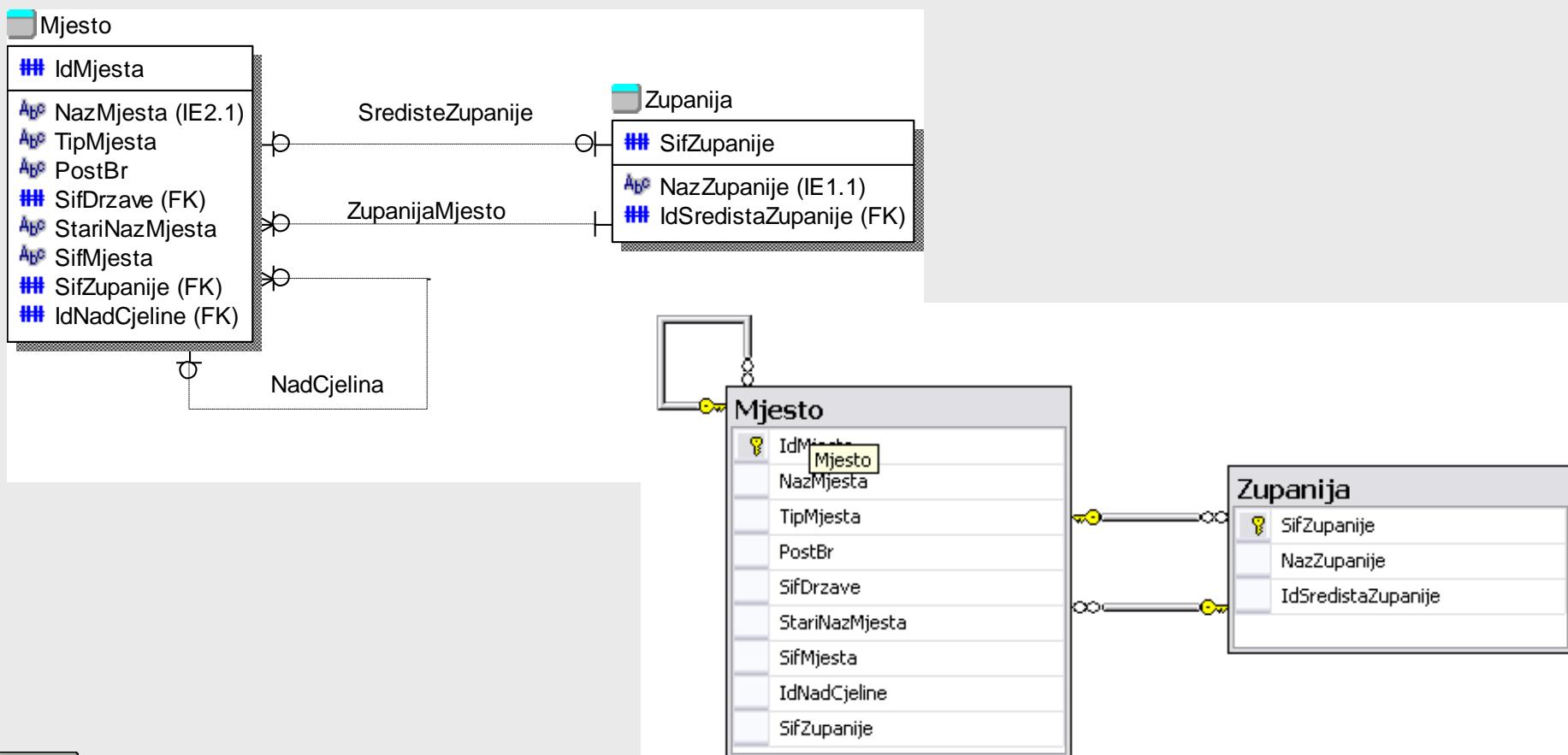
- **Sinkronizacija s modelom procesa**
 - spremišta podataka u modelu procesa su entiteti u modelu podataka
- **Provjera modela**
 - pripaziti na sinonime (različite nazine istih objekata)
 - pripaziti na homonime (jednake nazine različitih objekata)
 - provjeriti sumnjive i redundantne veze i po potrebi ih ukloniti (oprez!)
 - uklanjanje neke od paralelnih veza
 - uklanjanje veza koje se daju izvesti iz drugih (tzv. trijade) - pripaziti na kardinalnost veza u lancu
 - ukloniti balansirane veze 1,1:1,1 (ukoliko je moguće!)
 - pripaziti na nebalansirane veze 0,1:1,1 (zavisnost entiteta)
 - cirkularne reference
 - izbaciti izvedene atribute
 - mogući gubitak informacije o potrebnim izvedenim/izračunatim poljima

Uklanjanje veza koje se daju izvesti iz drugih ?



Problem paralelnih i cirkularnih veza - ukloniti?

- primjer: Zupanija 1:N Mjesto i Zupanija 1:1 MjestoSrediste
- slično: OrgJedinica 1:N Djelatnik i OrgJedinica 1:1 DjelatnikUpravitelj
- problem: zahtijevanost vrijednosti stranih ključeva
- problem: realizacija veze 1:1, redundantna središta županija



Balansirane i nebalansirane binarne veze

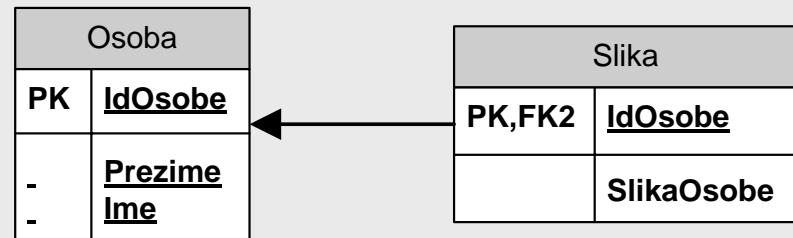
□ Udruživanje u zajedničku tablicu

- Binarna veza 1,1:1,1 – slika NOT NULL
- Binarna veza 1,1:0,1 – slika NULL



□ Razlozi za udruživanje

- jednostavniji dohvati podataka
- ubrzanje upita



□ Razlozi za razdvajanje

- ušteda diskovnog prostora
- utjecaj na fizički raspored podatka
- tablice u vezi 1:1 s prevelikim brojem stranih ključeva

Šifarski sustav

Određivanje ključeva

Šifarski sustav

□ Serijske šifre

- brojevi koji se slijedno pridjeljuju svakoj novo dodanoj instanci entiteta
- u modernim SUBP mogu se generirati uz opcionalna ograničenja
- primjer: SQL Server IDENTITY [(seed , increment)]

□ Blok šifre

- slično serijskim šiframa, s tim da su serijski brojevi grupirani prema značenju
- primjer satelitskih TV kanala: 100-199 PAY PER VIEW, 200-299 CABLE CHANNELS, 300-399 SPORT, 400-499 ADULT, 500-599 MUSIC-ONLY, ...

□ Alfanumeričke oznake

- ograničeni skup znakovnih oznaka, često kombiniranih s brojevima
- primjer, oznake država: HR, DE, IT, SI

Šifarski sustav (nastavak)

□ Samogovoreće šifre (significant position codes)

- svaka znamenka ili grupa znamenki opisuje neko svojstvo instance
- primjer: JMBG, a često se koriste i u skladišnoj evidenciji (dimenzije automobilske gume, električne žarulje)

□ Hijerarhijski kodovi

- podjela u grupe, podgrupe itd.

□ Šifre s kontrolnom znamenkom

- Zadnja znamenka se izračunava iz svih ostalih
- Detektira se jednostruka pogreška u znamenci ili zamjena mesta dvije znamenke
- JMBG, Tekući račun, Žiro račun, poziv na broj, ...

Primjeri šifrarnika

□ Primjer, studentska prehrana

- Pravilnikom o studentskoj prehrani propisan je status upisa potreban za ostvariranje prava na subvencioniranu prehranu.
- Npr. mora biti upisan u tekuću akademsku godinu na redovnom studiju ili studiju za osobne potrebe.

Šifra statusa	Opis statusa	Pravo prehrane
R	redovan studij	DA
P	paralelni studij	NE
U	studij uz rad	NE
V	vanredni studij	NE
L	paralelni studij s pravom prehrane	DA
O	studij za osobne potrebe	DA

□ Primjer, IUCN kriteriji ugroženosti

IdIUCN	NaziUCN	NaziUCNENG
A	Redukcija populacija (smanjivanje broja jedinki)	Reduction in population size
A1	reverzibilna, razumljiva, obustavljena	reversible AND understood AND ceased
A1a	a-direktno opažanje	a-direct observation
A1b	b-indeks učestalosti	b-an index of abundance
...		
B1biii	iii- područja, obima i/ili kvalitete staništa	iii-area, extent and/or quality of habitat
B1biv	iv- broja lokaliteta ili subpopulacija	iv-number of locations or populations
B1bv	v- broja zrelih individuumu	v-number of mature individuals
B1c	kolebanje	fluctuations
...		

Primjer jedinstvene šifre

□ JMBAG - jedinstveni broj od upisa do kraja studija

- student upisan na dvije ili više ustanova zadržava JMBAG koji je dobio na prvoj
- broj se generira automatski
- JMBAG ima deset znamenki podijeljenih u dvije grupe te kontrolnu znamenku:
 - Prve četiri znamenke – matična ustanova vlasnika
 - Sljedećih 5 - oznaka vlasnika u ustanovi (matični broj vlasnika ili slijedno)
 - Unutar ustanove JMBG i matični broj (broj indeksa) su također jedinstveni.

□ Broj kartice generira se automatski, a u sebi sadrži 6 grupa znamenki:

- A. jedinstveni broj u međunarodnom kartičnom poslovanju (IIN)
 - uvijek **601983** i prema standardu ISO/IEC 7812 na jedinstven način identificiraju Studentsku karticu u međunarodnom sustavu kartičnog poslovanja
- B. oznaka vrste kartice (1 znamenka)
 - npr: 1 - student i 4 - privremena kartica
- C. redni broj kartice koju je student dobio (1 znamenka)
- D. JMBAG (10 znamenki)
- E. Kontrolna znamenka (1 znamenka),



Izrada šifarskog sustava

□ Izrada šifarskog sustava

- treba biti smislen i prikladan da dodavanje novih šifara bude jednostavno
- gdje je moguće treba preuzeti postojeće šifrarnike
 - ustanova ili sustava (državni zavodi, nacionalne institucije, ...)
- oznake definirane zakonom ili drugim propisima treba preuzeti i prilagoditi
- ostale šifrarnike definirati tako da se naknadno mogu nadograđivati
- izbjegavati samogovoreće šifre

□ Primjer: IPISVU

- šifre djelatnika kao redni brojevi, honorarci oblika 9999*
- dobavljači iz državne riznice, nadopunjeni vlastitim

Dizajn baze podataka

Ugađanje i (de)normalizacija

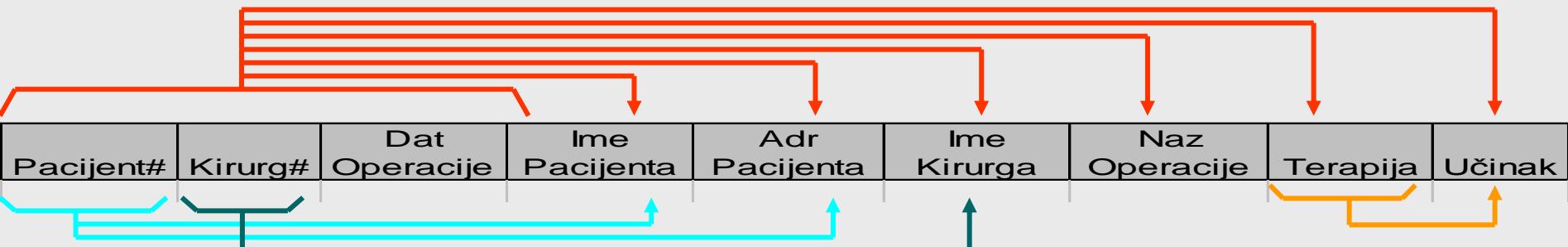
Normalizacija

□ Normalizacija

- postupak strukturiranja sheme relacijske baze podataka tako da se ukloni što više neodređenosti (zalihosti)
- stupanj normalizacije povećava se od 1NF do 5NF
 - većina dizajnera zaustavlja se na 3NF ili na BCNF (Boyce-Codd NF)

□ svodi se na ispunjenje tzv "relacijske zakletve" [Finkelstein, 1989]:

- Ključ
 - 1NF: nema ponavljajućih grupa, definiran primarni ključ
- Cijeli ključ
 - 2NF: svi neključni atributi u potpunosti zavisni o čitavom PK
- Ništa drugo nego ključ
 - 3NF: svaki neključni atribut je neposredno zavisan samo o PK
- Tako mi Codd pomogao! (Dr. E. F. Codd - otac relacijske teorije BP)
 - BCNF: ne postoji kompozitni i/ili preklapajući kandidati ključeva



Denormalizacija

- **Denormalizacija - redukcija modela podataka u nižu NF**
 - ne smije se miješati s pogreškom nedovođenja u jaču NF, lošim dizajnom
 - u tom slučaju denormalizacija mogla dovesti do još većih problema !
- **Razlozi za denormalizaciju**
 - poboljšanje performansi, npr. ubrzanje upita eliminacijom vanjskih spajanja
 - olakšanje pristupa nekim podacima koje je komplikirano dohvatiti
- **Denormalizaciju treba obaviti samo tamo gdje je to stvarno nužno i na takav način da ne ugrožava integritet podataka**
 - održavanje redundancije podataka zahtjeva aplikacijsko upravljanje integritetom podataka koje je "skupo"

Preporuke za denormalizaciju

□ Poznavati vlastitu bazu podataka

- kako je strukturirana te kako ju aplikacije koriste
- poznavanje frekvencije obrade podataka

□ Denormalizirati mjestimično i oprezno

- Prije denormalizacije ili za usporedbu probati druge tehnike
- izračunati/virtualni stupci (computed column), funkcije u bazi podataka

□ Analizirati postupke pristupa podacima

- indeksi, optimizacija upita, forsiranje plana izvođenja upita

□ Procijeniti fizičke resurse

- povećanje ili bolja raspodjela radne memorije
- dodavanje ili zamjena procesora
- fizička organizacija podataka (raspored datoteka, segmentacija, ...)
- kvaliteta i optimizacija diskovnog prostora (OS, BP, LOG, BAK)

Podešavanje (umjesto denormalizacije)

□ Postavljanje indeksa

- primarni ključevi implicitno indeksirani
 - ali im treba eksplicitno kreirati sortirani, tzv. clustered indeks
- strani ključevi indeksirani ?
- najčešće korištena polja – pretraživanje, grupiranje, sortiranje
- složeni ključevi "brzi" samo po prvom polju
- masovne obrade - ukloniti indekse, pa kasnije vratiti

□ Ubrzanje i optimizacija upita

- forsira redoslijed spajanja naveden u upitu
 - primjer: SELECT ... OPTION (FORCE ORDER)
- primoravanje nekorištenja indeksa primjenom neškodljive funkcije nad poljem nad kojim se uobičajeno koristi indeks
 - primjer: WHERE NULLIF (polje , “”) = uvjet
- optimizacija za određeni broj prvih zapisu
 - primjer: SELECT ... OPTION (FAST n_rows)
- izbjegavanje "ILI" uvjeta

Uklanjanje nul-vrijednosti

□ **NULL vrijednost (razlikovati od broja 0), primjer Ispit.Ocjena:**

- Ne postoji uopće (predmet iz kojeg nema ispita)
- Još ne postoji (ispitivanje još nije završeno)
- Ne zna se (nastavnik nije upisao)
- Neće je ni biti (student nije izašao na prijavljeni ispit)

tumačenje ?

□ **Indeksi ne pomažu u slučaju nul-vrijednosti**

- opisna polja - zahtijevana vrijednost + prepostavljena vrijednost

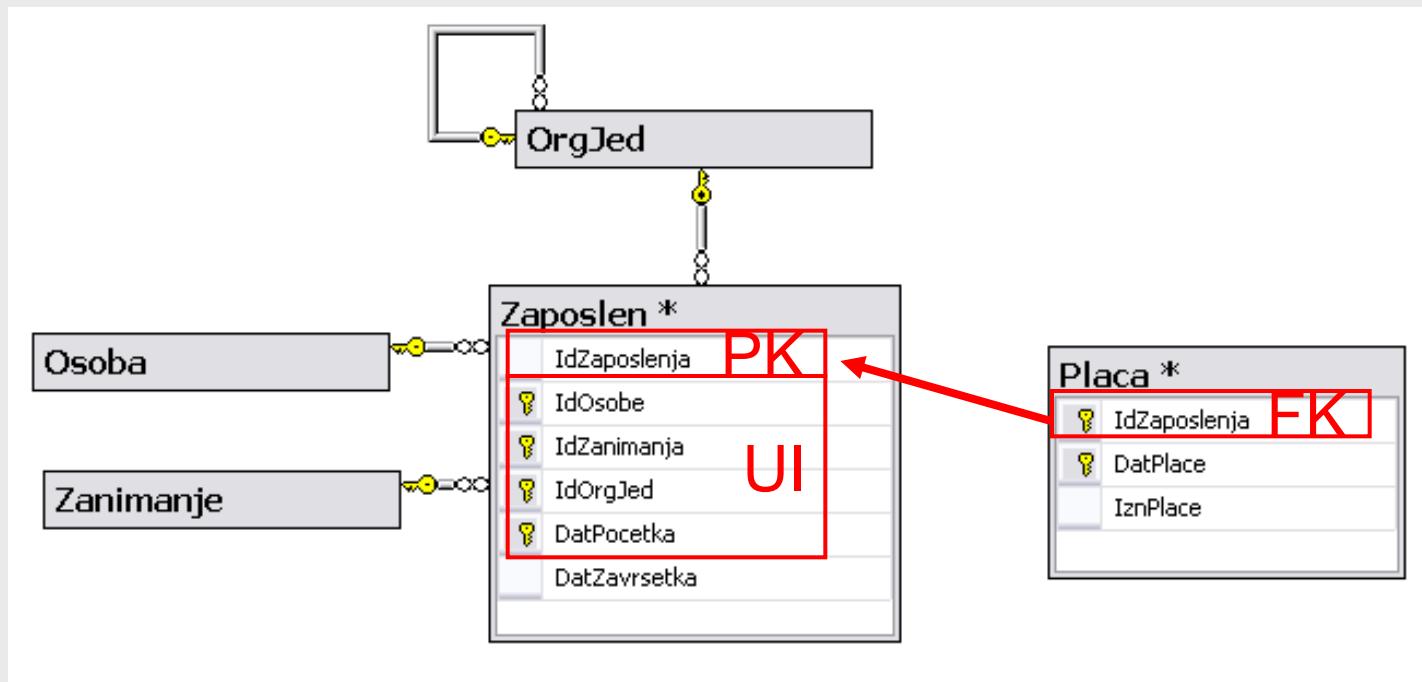
□ **NULL strani ključevi - vanjska spajanja koja ne koriste indekse**

- strane ključeve postaviti na NOT NULL
- definirati posebne vrijednosti u šifrarnicima
- primjer: 0-nepoznata vrijednost, ..., 999-nepostojeća vrijednost

Nadomjesni (surogatni) ključevi

☐ Nejasna ili izmjenjiva poslovna pravila, pojednostavljenje veza

- ispred ključa složenog od većeg broja atributa (npr. ≥ 3) umeće se ključ sa samopovećavajućim vrijednostima (serial), a na originalni ključ postavlja se jednoznačan (unique) kompozitni indeks
- teorijski se ne preporuča za asocijativne entitete, koji nasleđuju ključeve svojih roditelja, jer se time gubi smisao identifikacijske veze
- praktično, nadomjestak treba ugraditi kada je tablica u koju se ugrađuje referencirana iz drugih tablica (npr. Placa referencira Zaposlen)



Nadomjesni ključevi (aplikacijski razlozi)

- Problem odabira vrijednosti složenog stranog ključa u padajućim listama koje evidentiraju samo jednu vrijednost odabranog retka
- Primjer: Prethodni dokument (RPPP)

The screenshot shows the Microsoft Visual Studio IDE interface. At the top, there is a navigation bar with icons for file operations like New, Open, Save, and Print. Below the navigation bar, the main workspace displays a Windows Form with several controls: a dropdown menu labeled 'Prethodni:', a numeric up-down control labeled 'Porez:' with a value of '0 - 1', and a page navigation section with buttons for 'First', 'Previous', 'Next', and 'Last'. To the right of the form, a 'ComboBox Tasks' window is open, containing configuration settings for the dropdown control:

Setting	Value
Use data bound items	<input checked="" type="checkbox"/>
Data Binding Mode	
Data Source	prethDokumentBindingSource
Display Member	NazDokumenta
Value Member	IdDokumenta
Selected Value	dokumentBindingSource - I

At the bottom of the screen, there are status bars for 'leDbDataAdapterArtikl', 'oleDbConnection1', and 'oleDbData'. The bottom right corner of the slide contains the number '39'.

"Čisti" dizajn

- **Takozvani "čisti" dizajn – dosljedna ugradnja nadomjesnog ključa sa samopovećavajućim vrijednostima u sve tablice baze podataka**
 - Prednost:
 - pojednostavljenje ugradnje
 - Nedostatci:
 - umnožavanje ključeva (nadomjesni primarni, originalni alternativni)
 - nepreglednost izvornog stranog ključa
 - primjer: umetanje Valuta.IdValute (serial 876) ispred OznValute (string HRK)
 - za posljedicu ima, npr. Dokument.IdValute umjesto Dokument.OznValute

Izvedeni podaci (derivable data)

□ Dodavanje atributa za vrijednosti koje se daju izračunati iz drugih

- atribut pohrane za vrijednost koja se može izračunati, na primjer:
 - iznos dokumenta kao suma iznosa stavki
 - oznaka zbirnog stanja kada se vrijednosti pojedinih stanja nalaze u tablici s velikim brojem zapisa (stanje skladišta, saldo na računu)

□ Primjer: iznos dokumenta

- aplikacijski, pisanjem programskog koda za izračun (pogledati RPPP)
- pohranjenom procedurom, po potrebi uz IdDokumenta kao argument uz:
 - WHERE Dokument.IdDokumenta = @IdDokumenta
- okidačem tablice Stavka uz:
 - WHERE Dokument.IdDokumenta IN Select (IdDokumenta FROM inserted)

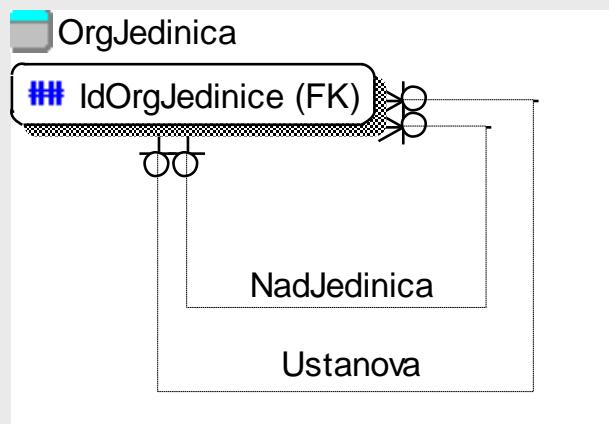
```
UPDATE Dokument SET IznosDokumenta =
  (SELECT SUM(KolArtikla*JedCijArtikla*(100-PostoRabat)/100)
   * (1+Dokument.PostoPorez)
  FROM Stavka WHERE Stavka.IdDokumenta=Dokument.IdDokumenta)
```

Izvedeni podaci (2)

- Redundantna vrijednost koja se inače dohvaća složenim i/ili sporim a često postavljanim upitima
- Primjer, osoba s identifikatorom zadnjeg zaposlenja
 - može naravno i drugim operatorima, npr. $>= \text{ALL}$

```
SELECT Osoba.IdOsobe,
       IdZaposlen = (SELECT IdZaposlen FROM Zaposlen
                      WHERE Zaposlen.IdOsobe=Osoba.IdOsobe
                      AND DatPocetka = (SELECT Max(DatPocetka)
                                         FROM Zaposlen AS Z
                                         WHERE Z.IdOsobe=Osoba.IdOsobe) )
FROM Osoba
```

- Primjer: redundantni strani ključ na vrh hijerarhije



Prethodno povezane tablice (prejoined tables)

- **Izrada osnovne tablice strukture spojnog upita**
 - kada se dvije ili više tablica često povezuje u različitim a (pre)sporim upitima
- **Alternativa**
 - u slučaju postavljanja većeg broja dovoljno brzih spojnih upita – VIEW
- **Varijanta : Kombinirane tablice (combined tables) u vezi 1:1**
 - SlikaOsobe (pogledati "Balansirane i nebalansirane binarne veze")
 - Firma: Partner+Tvrтka+Osoba
 - Poduzeće : Proizvod+Igla+Avion
- **Primjer, neovisno o tipu veze (1:1 ali i 1:N)**
 - primarni i strani ključ s atributima koji se ne indeksiraju
 - izgledniji bi bio onaj složenog stranog ključa, ali taj se rješava surogatom
 - Osoba = @IdOsobe BIT(5000) + Prezime + ...
 - Slika = @IdSlike + SlikaOsobe + IdOsobe BIT(5000)
 - preslikava se u
 - **OsobaSaSlikom = @IdOsobe BIT(5000) + Prezime + ... + SlikaOsobe**

Tablica izvješća (report table)

□ Tablica izvješća

- Može kombinirati i druge tehnike denormalizacije
- Česta primjena – TEMP TABLE

□ Primjer

```
INSERT INTO Report
    (ControlBreak, VrDokumenta, BrDokumenta, Promet)
SELECT 'dokument', VrDokumenta, BrDokumenta, IznosDokumenta
    FROM Dokument
UNION ALL
SELECT 'ukupno', VrDokumenta, NULL, SUM(IznosDokumenta)
    FROM Dokument GROUP BY VrDokumenta
UNION ALL
SELECT 'sveukupno', 'Z', NULL, SUM(IznosDokumenta)
    FROM Dokument
```

□ SELECT FROM Report može zahtijevati sort, ovisno o tome kuda UNION smješta zapise s NULL (na vrh ili na dno grupe)

- ORDER BY VrDokumenta, ControlBreak
- u primjeru je radi sorta uvedena i izmišljena vrsta dokumenta 'Z'

Zrcalne tablice (mirrored tables)

□ Zrcalne tablice

- kopije originalnih tablica
- jednake strukture ili dodavanje dodatnih atributa – npr. vremenske dimenzije

□ Primjer:

- transfer u analitički sustav ili dio baze podataka za izvješćivanje
- da se izbjegne opterećenje ili zaključavanje izvornih tablica

□ Primjer:

- evidencija obrisanih zapisa

Dijeljenje, cijepanje tablica (table splitting)

□ Dijeljenje, cijepanje tablica

- vertikalno ili horizontalno – izvorna tablica u dvije ili više
- ako se zadrži izvorna – varijanta zrcaljenja
- razlog: lakše ili brže rukovanje podacima
- implicira kasnije kreiranje upita (pogleda) za agregaciju dijelova
 - UNION ALL za horizontalnu podjelu
 - INNER JOIN za vertikalnu podjelu

□ Kriteriji dijeljenja i primjeri

- fizički - po jedna tablica za svaki terminal
- prostorno - po državama, županijama
- vremenski - po jedna tablica za svaki mjesec ili godinu
- proceduralno - po jedna tablica za svaki korak poslovne procedure
- administrativno - po jedna tablica za svaki odjel umjesto za čitavo poduzeće

□ Neki problemi

- horizontalna podjela – konflikt primarnog ključa
- vertikalna podjela – simultano kreiranje zapisa s istim ključem u više tablica

Redundantni podaci (redundant data)

□ Redundantni podaci

- udruživanje čitave referencirane tablice u zavisnu
- primjer: Osoba + Spol gdje je Spol = @OznSpola INT, NazSpola CHAR
- anti primjer: Osoba + Mjesto

□ preporuke, uvjeti primjene:

- primjena samo na mali broj stupaca
- primjena samo na vremenski stabilne podatke
- pisanje programskog koda za održavanje

Ponavljajuće grupe (repeating groups)

□ Ponavljajuće grupe (repeating groups)

- izvorno: Naplata = @IdKupca, NazKupca, @Kategorija, Iznos
- denormalizirano: Naplata = @IdKupca, NazKupca, Iznos1, ..., IznosN

□ preporuke, uvjeti primjene:

- kada je broj stupaca ponavljajuće grupe mali i stabilan
- kada se grupi pristupa kolektivno a ne individualno, tj. kada je grupa suvisla cjelina a ne skup vrijednosti pretvoren u atribut
- kada se želi evidentirati da neka grupa nedostaje ili je nepoznata (NULL)
- nije potrebno agregirati grupe unutar retka

Preopterećeni tipovi podataka (overloaded datatypes)

□ Udruživanje šifrarnika

- Sifrarnik<i> = @Sifra<i> <datatype>, Naziv<i>
 - pr. Mjesto = @PostBr INT, NazMesta VARCHAR
 - pr. Valuta = @KratValute VARCHAR, NazValute VARCHAR
 - pr. Koeficijent = @SifraKoeficijenta, VrijKoeficijenta DECIMAL(5,2)
- integriramo uvođenjem oznake šifrarnika (Vrsta) te nastaje
- Sifrarnik = @Vrsta VARCHAR, @Sifra VARCHAR, Naziv VARCHAR

□ prednost:

- izrada jedne (zajedničke) komponente za održavanje šifrarnika

□ problemi: glavni je, naravno, narušavanje 1NF

- konverzija tipa podatka izvornih ključeva i zavisnog atributa u VARCHAR
- validacija unosa
- strani ključevi zavisnih tablica
- ograničenje na šifrarnike s jednostavnim ključem
- sigurnost (kontrola pristupa)
- **povećanje prostora potrebnog za pohranu**

Preopterećeni tipovi podataka - validacija

□ Rješenje: CHECK CONSTRAINT (produžuje validaciju!)

```
CREATE TABLE Sifrarnik (..., CHECK  
    (CASE WHEN Vrsta = <vrsta 1> AND <validation rule 1>  
          THEN 1  
          ...  
          WHEN Vrsta = <vrsta n> AND <validation rule n>  
          THEN 1  
          ELSE 0  
      END = 0) , ...
```

□ Načelno, udruživanje šifrarnika treba izbjegavati

□ Preopterećenje tipa podatka zavisnih atributa

- sličan, doduše, manji problem
- nastavak slijedi (u narednom poglavlju)

Meta-modeliranje

**Rječnik podataka
Meta-baza
Primjena meta-modela u izradi aplikacija**

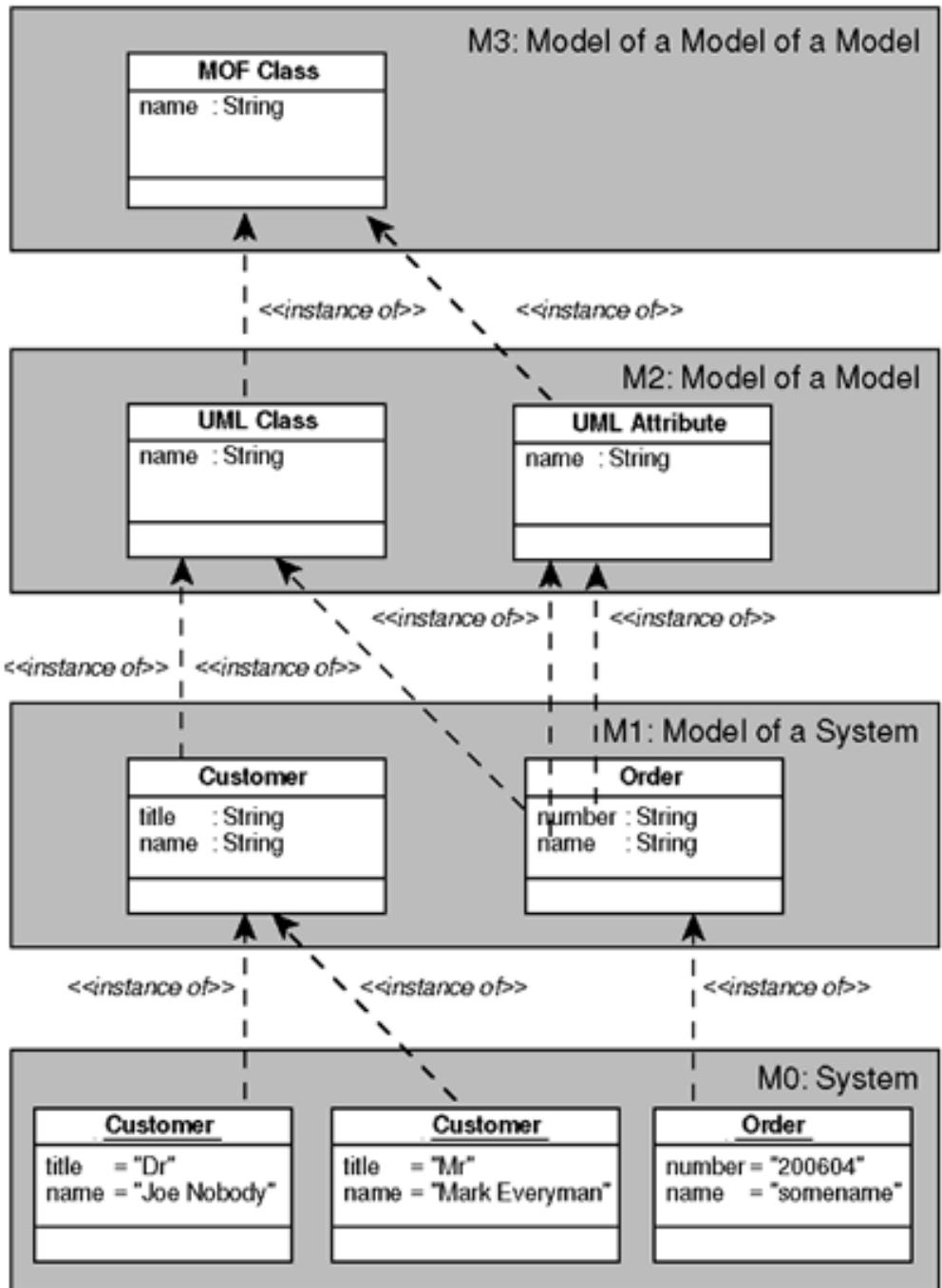
Meta-modeliranje

□ Meta-modeliranje = modeliranje podataka o podacima

- oznaka "meta" označava višu razinu apstrakcije
- metapodaci - podaci o podacima
- metamodel - model modela

□ Primjene

- opisivanje (baze) podataka
- generiranje aplikacija, podataka, meta podataka, ...
- dinamička prilagodba aplikacije podacima, supstitucija sličnih tablica, ...
- pohrana i razmjena podataka, npr. eLearning Digital Object



Opisivanje i pohrana (meta)modela

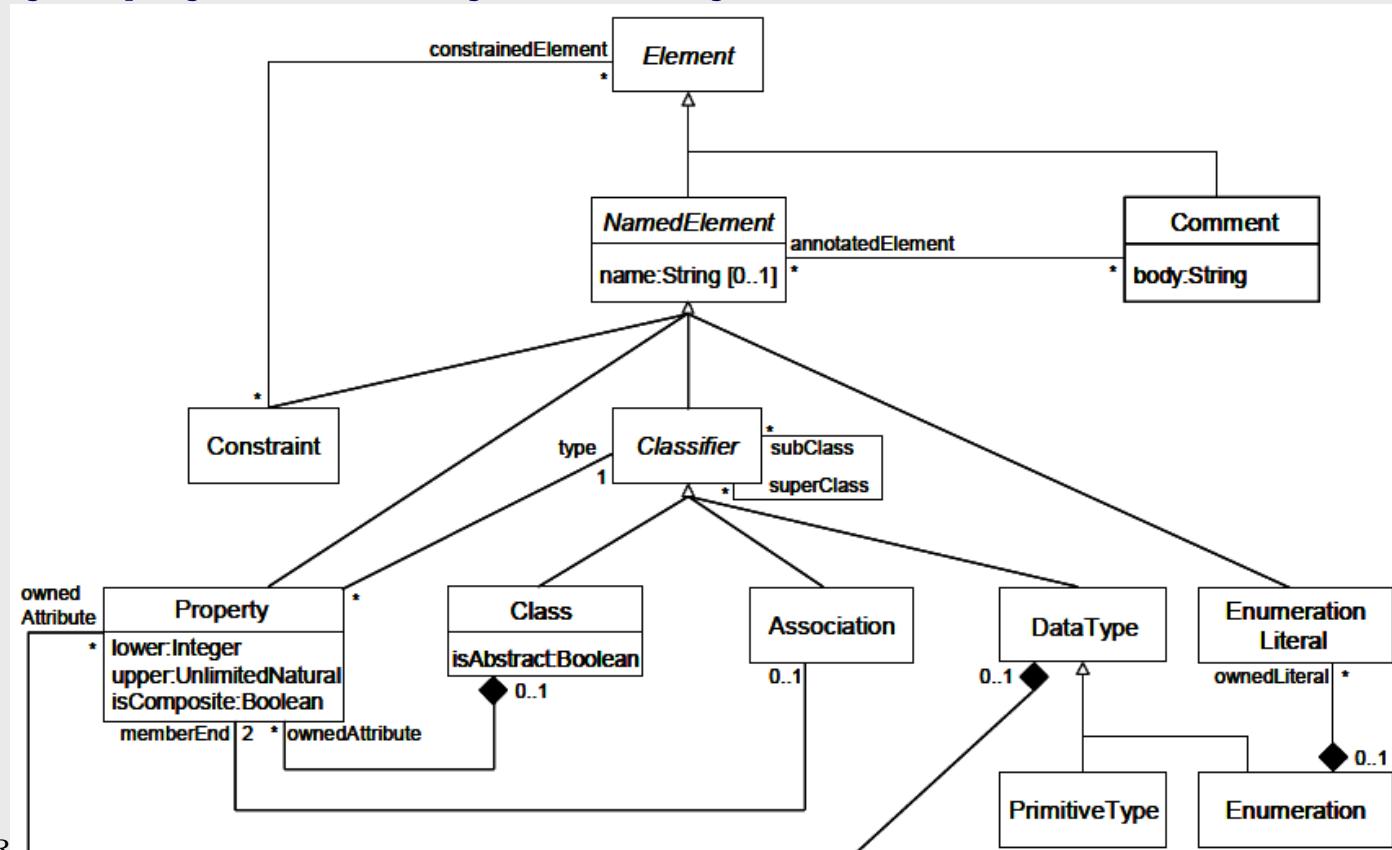
- Metamodel – model (za opisivanje drugih) modela
- Metabaza – baza podataka za pohranu modela
 - rječnik podataka, repozitorij, riznica
- Rječnik podataka – dio SUBP s podacima za upravljanje BP
 - definicija strukture, evidencija dozvola, ...
- Riznica, repozitorij – dio CASE alata za pohranu specifikacija
- Shema – instanca metasheme
- Metashema – instanca meta-metasheme
- Metamodeliranje – definicija i upotreba meta-metashema

Meta Object Facility (MOF)

□ MOF - Object Management Group (OMG)

- Standard za modelom vođeno inženjerstvo
- Proširivi modelom vođeni okvir, najpoznatija meta-metashema
- CMOF (complete) – druge metasheme kao instance MOF (npr. UML2)

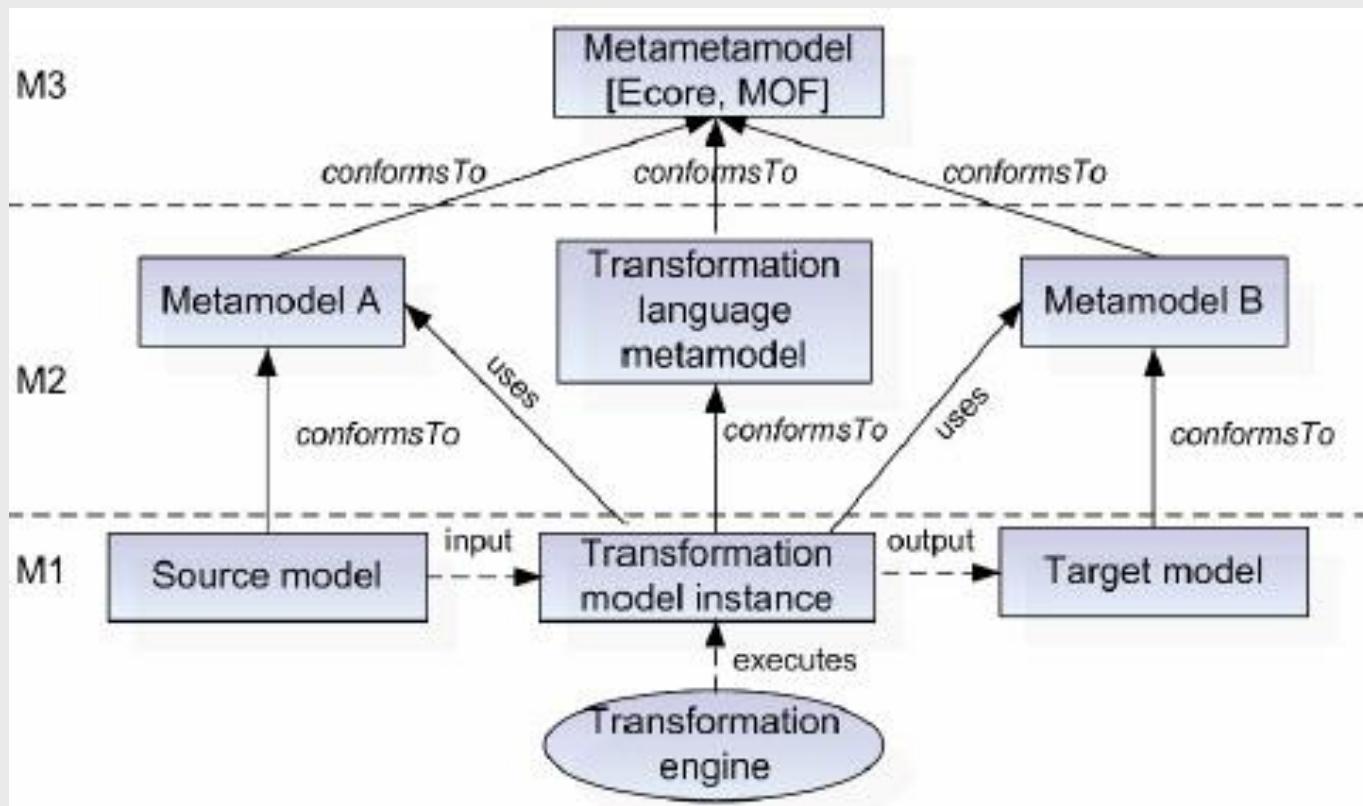
□ Primjer, pojednostavljena verzija MOF



Modelom vođena arhitektura / razvoj

□ Model Driven Architecture (Design)

- PIM – Platform Independent Model
 - UML kao PIM: obični (essential) i izvršni (executable)
- PSM – Platform Specific Model
- Transformacija PIM – PSM - kod



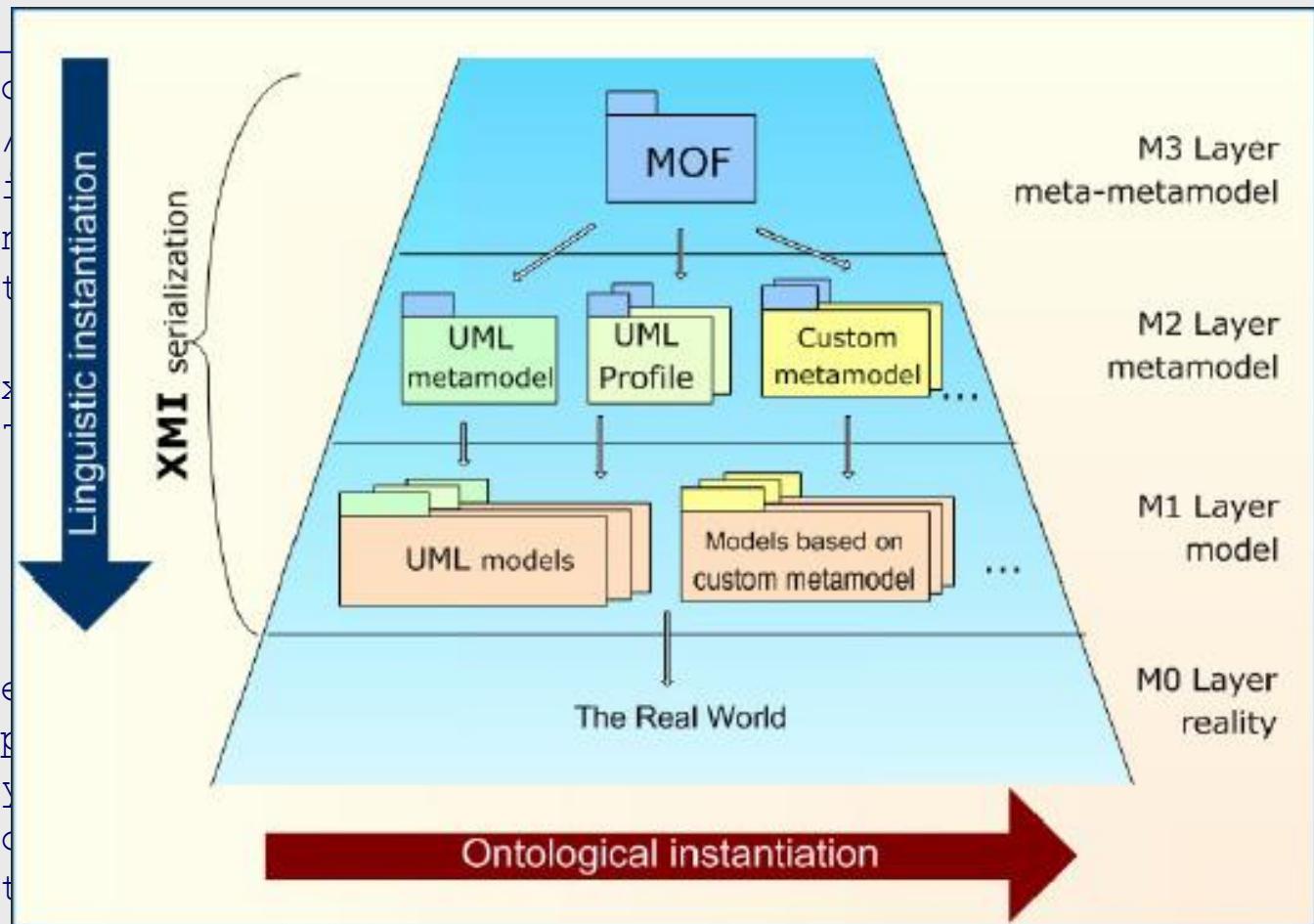
XML Metadata Interchange (XMI)

□ XML Metadata Interchange (XMI)

- OMG standard za razmjenu metapodataka XMLom
- Sadrži opis fizičke reprezentacije entiteta i veza

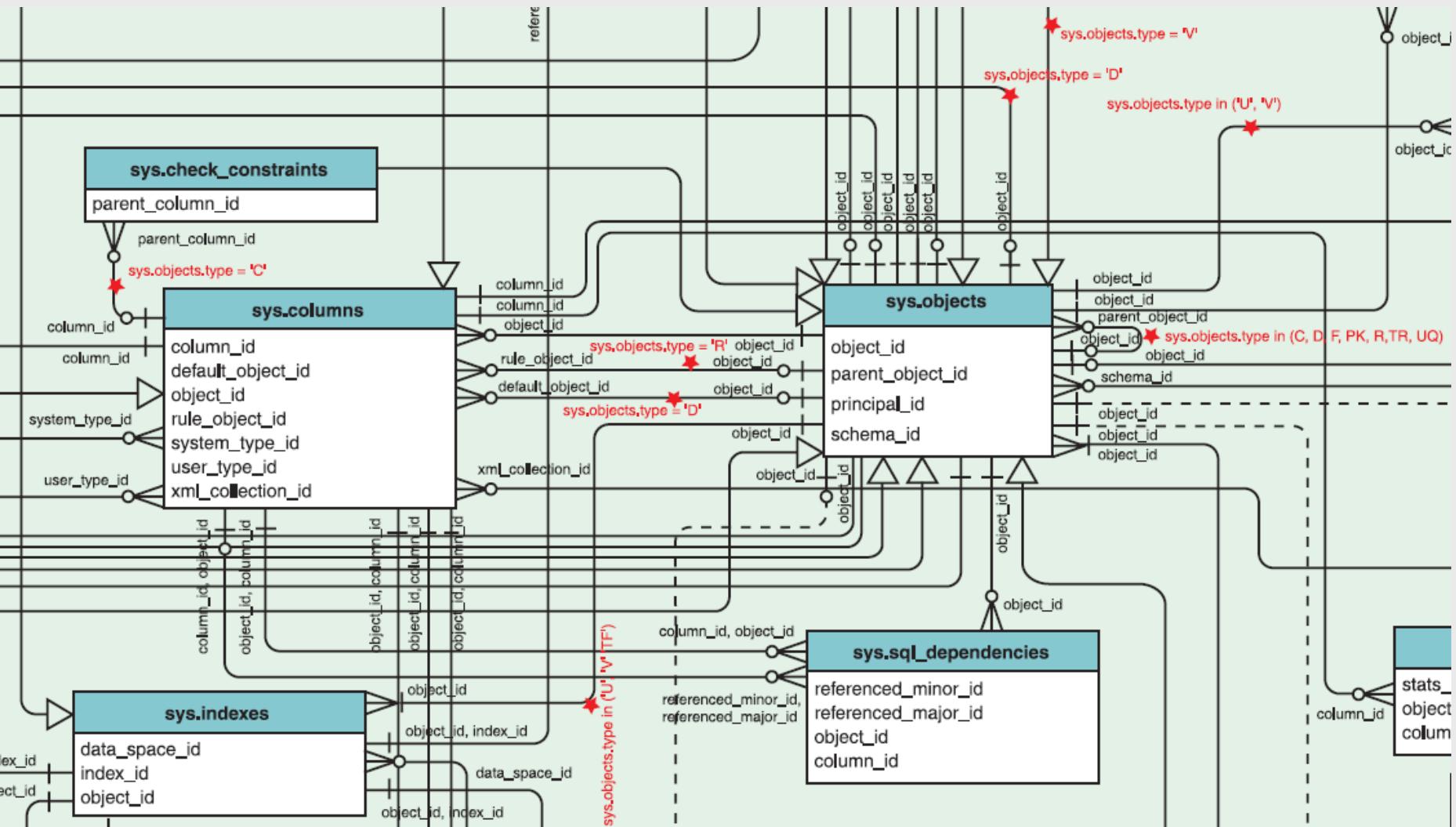
□ Primjer, XMI

```
<xmi:XMI xmi:version="2.0">
  xmlns:xmi = "http://www.omg.org/XMI"
  <OriginalBook xmi:id="1">
    language = "en"
    <isbn nil = "true"/>
  </OriginalBook>
  <TranslatedBook xmi:id="2">
    isbn = "84,727,250,1"
    ...
    <Person xmi:id = "3">
      ...
      <DataType xmi:id="4">
        <ownedAttribute lower = "1" upper = "1">
          <type xmi:type="xsd:string" href="doc#Color"/>
        </ownedAttribute>
      </DataType>
    </Person>
  </TranslatedBook>
</xmi:XMI>
```



Shema rječnika SUBP

□ Primjer: [Resursi\SQL2005_Sys_VIEWS.pdf](#)



Sistemske tablice i objekti

□ Sistemske tablice

- objects
- columns
- indexes
- constraints
- ...

□ Pohranjene procedure za rad s rječnikom

- sp_Help
- ...

■ Vrste objekata

```
SELECT name AS 'ObjectName'  
, 'ObjectType' = CASE xtype  
WHEN 'C' THEN 'CHECK'  
WHEN 'D' THEN 'DEFAULT'  
WHEN 'F' THEN 'FOREIGNKEY'  
WHEN 'L' THEN 'Log'  
WHEN 'FN' THEN 'ScalarFunction'  
WHEN 'IF' THEN 'InlineTableFunction'  
WHEN 'P' THEN 'StoredProcedure'  
WHEN 'PK' THEN 'PRIMARYKEY'  
WHEN 'RF' THEN 'ReplicationFilterStoredProcedure'  
WHEN 'S' THEN 'SystemTable'  
WHEN 'TF' THEN 'TableFunction'  
WHEN 'TR' THEN 'Trigger'  
WHEN 'U' THEN 'UserTable'  
WHEN 'UQ' THEN 'UNIQUEConstraint'  
WHEN 'V' THEN 'View'  
WHEN 'X' THEN 'ExtendedStoredProcedure'  
END  
FROM sysobjects -- isto što i sys.objects  
ORDER BY xtype, name;
```

Korištenje rječnika

□ Primjer: baza podataka FirmaRIS, procedura ap_CountALL

```
CREATE PROCEDURE ap_CountALL
AS

DECLARE @tablename varchar(30)
DECLARE @sql varchar(75)
DECLARE cu_names CURSOR FOR SELECT name FROM sysobjects
    WHERE type = 'U'
OPEN cu_names
FETCH NEXT FROM cu_names INTO @tablename
WHILE (@@fetch_status <> -1)
BEGIN
    IF (@@fetch_status <> -2)
    BEGIN
        SELECT @sql = 'SELECT ''' + @tablename
                      + ''', COUNT(*) FROM ' + @tablename
        EXEC (@sql)
    END
    FETCH NEXT FROM cu_names INTO @tablename
END
DEALLOCATE cu_names
GO
```

	Results	Messages
1	Mjesto	9120
1	Partner	554
1	Osoba	63
1	Tvrđka	491
1	sysdiagrams	1
1	Dokument	857
1	Artikl	1531
1	Država	239

INFORMATION_SCHEMA

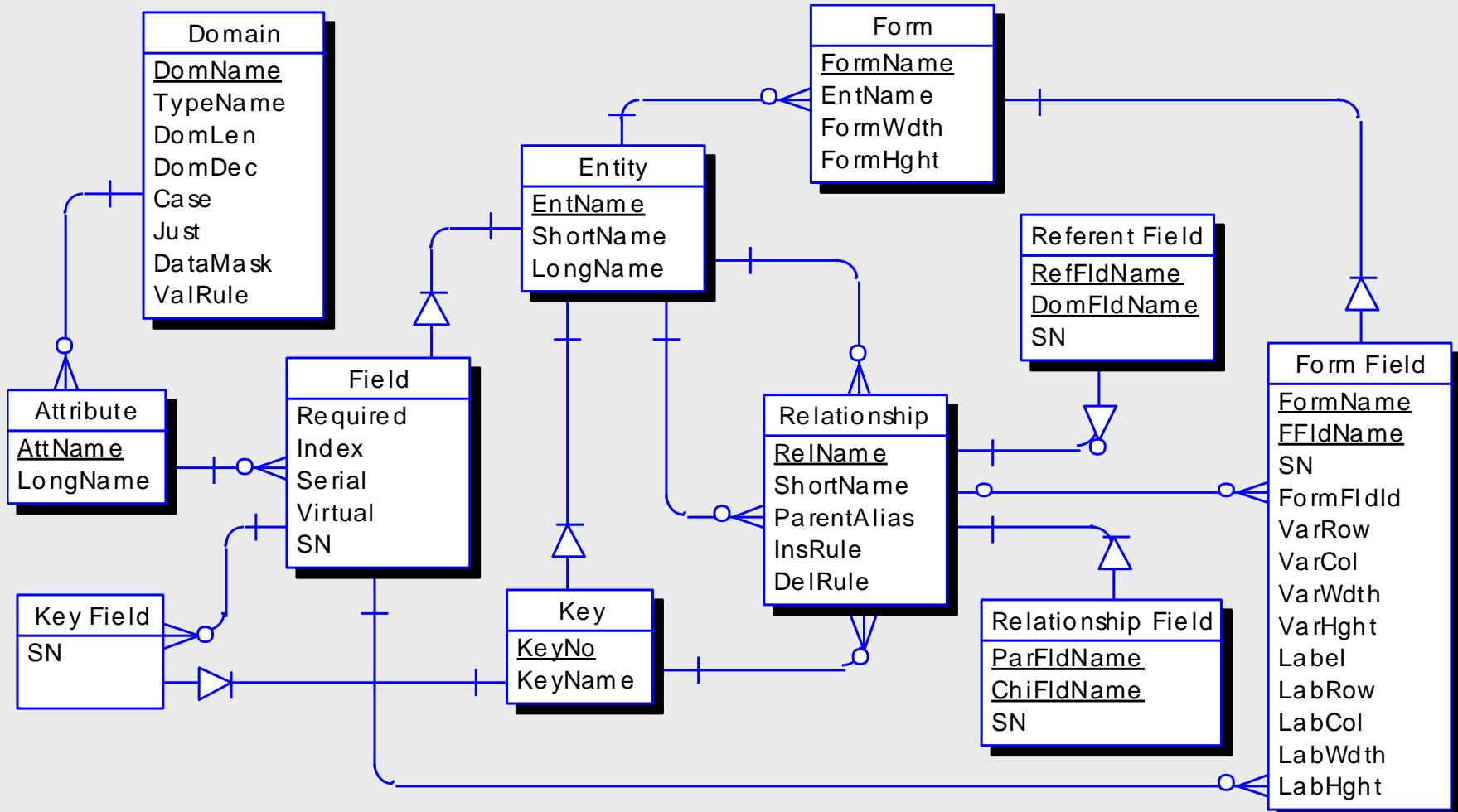
□ Sistemski nezavisani pogled na metapodatke - ISO standard

CHECK_CONSTRAINTS	REFERENTIAL_CONSTRAINTS
COLUMN_DOMAIN_USAGE	ROUTINES
COLUMN_PRIVILEGES	ROUTINE_COLUMNS
COLUMNS	SCHEMATA
CONSTRAINT_COLUMN_USAGE	TABLE_CONSTRAINTS
CONSTRAINT_TABLE_USAGE	TABLE_PRIVILEGES
DOMAIN_CONSTRAINTS	TABLES
DOMAINS	VIEW_COLUMN_USAGE
KEY_COLUMN_USAGE	VIEW_TABLE_USAGE
PARAMETERS	VIEWS

□ Primjer:

```
SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,  
       COLUMN_NAME, COLUMN_DEFAULT  
FROM FirmaRIS.INFORMATION_SCHEMA.COLUMNS
```

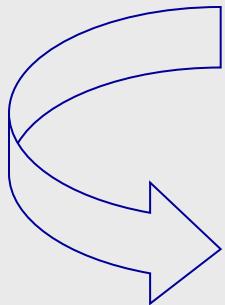
Model meta-baze generatora aplikacija



Primjer generiranja programskog koda

```
ForEach Ent -> sEnt
<<
CREATE TABLE >> Print sEnt.EntName <<  (>>

ForEach Fld
    Where Fld.EntName=sEnt.EntName -> sFld
    <<
    >> Print sFld.Fldname, " ";
...
...
```



```
CREATE TABLE Drzava
(
    OznDrz CHAR(2) NOT NULL,
    NazDrz CHAR(25) NOT NULL,

    PRIMARY KEY (OznDrz) CONSTRAINT pk_drzava
);
...
```

Meta-modeliranje aplikacije

□ Primjer: Firma

- Zahtjevnica, Upit, Ponuda, Narudžbenica, Otpremnica, Primka, Račun ...
 - Svi slične ili jednake strukture kao npr. Račun na slajdu „Pretvorba identifikacijskih veza“
- Zamijenimo ih s Dokument = VrstaDokumenta + BrojDokumenta
 - Vrsta Dokumenta $\in \{ Z,U,P,N,O,T,R \}$

□ Prednosti

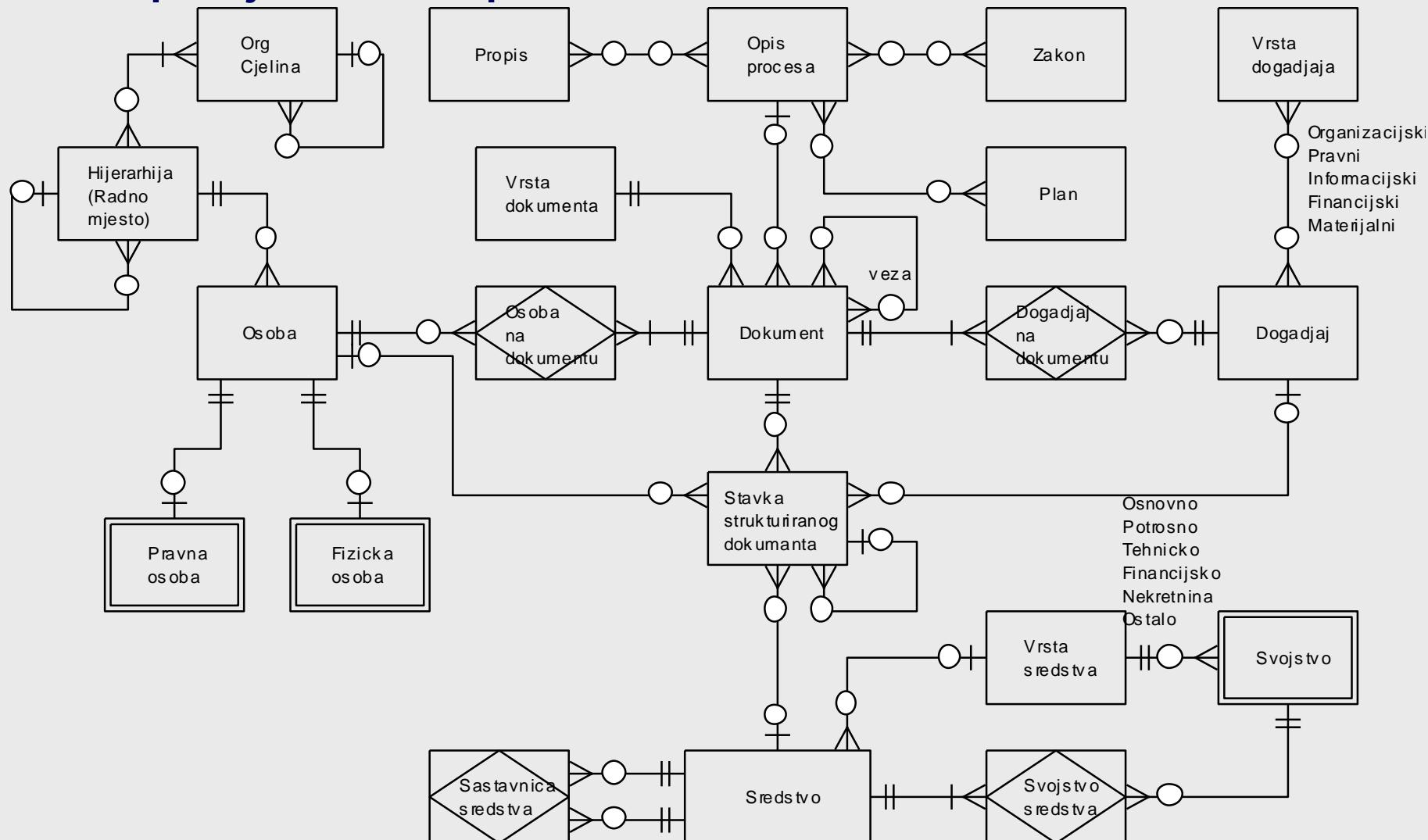
- Pojednostavljanje modela
- Smanjenje broja objekata

□ Nedostatci

- Otežano razumijevanje
- Problem preslikavanja realnih objekata u objekte metamodela

Ideja aplikacijskog meta-modeliranja (1)

Poopćenje strukture podataka



Ideja aplikacijskog meta-modeliranja (2)

□ Primjer, narudžba

Narudžba MBB-Nabava za osnovno tehničko sredstvo prema poduzeću IMB
Br. xxxx od xx.xx.200x.

Veza: Ponuda poduzeća IMB br. xxxx od xx.xx.200x.

Narudžbu izradio: Pero

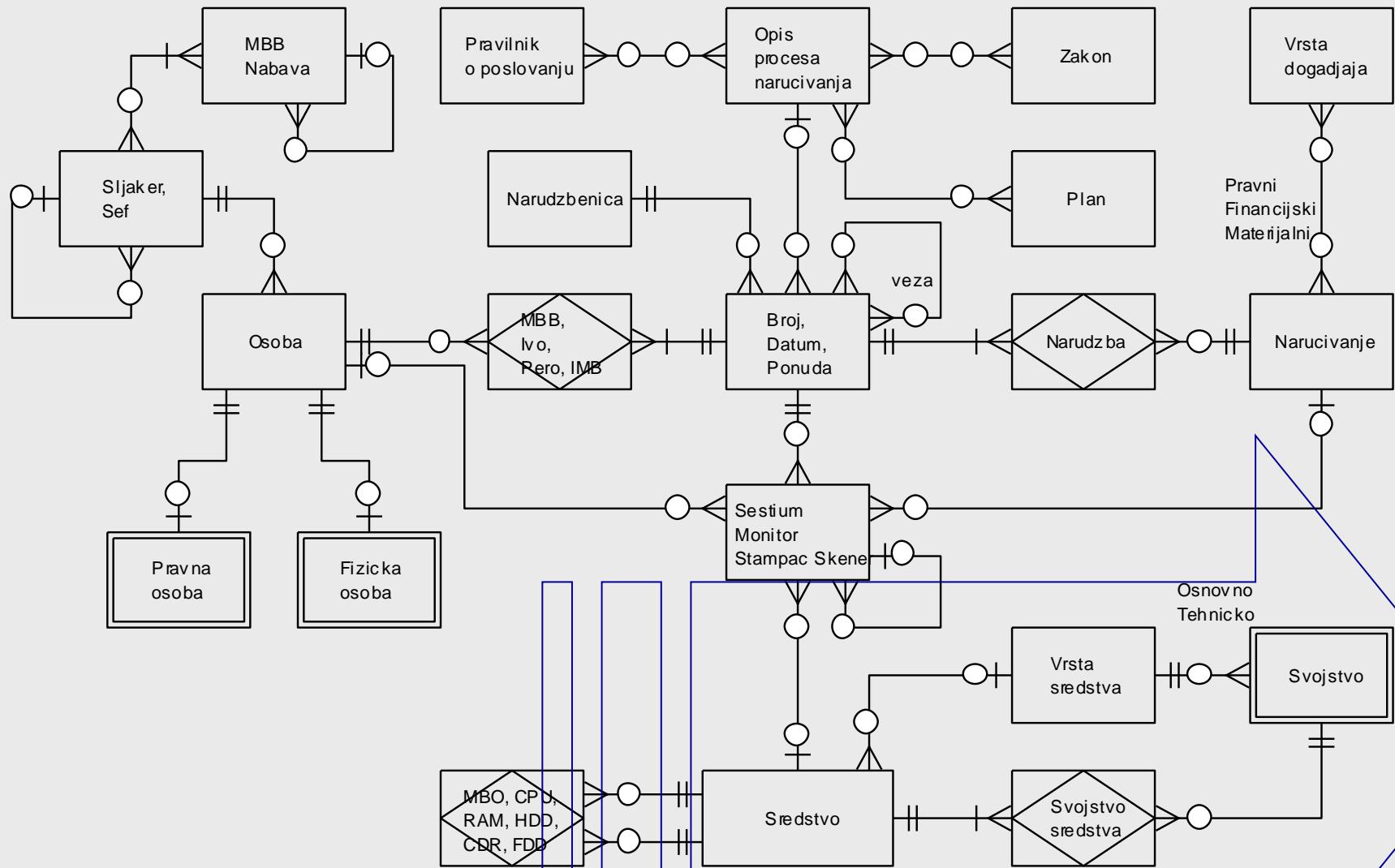
Odobrio njegov šef: Ivo

Stavke:

Računalo IMB Šestium:	xx kom
MBO, CPU, RAM, HDD, CDR, FDD, CASE	
Monitor 15"	xx kom
Štampač	xx kom
Skener	xx kom

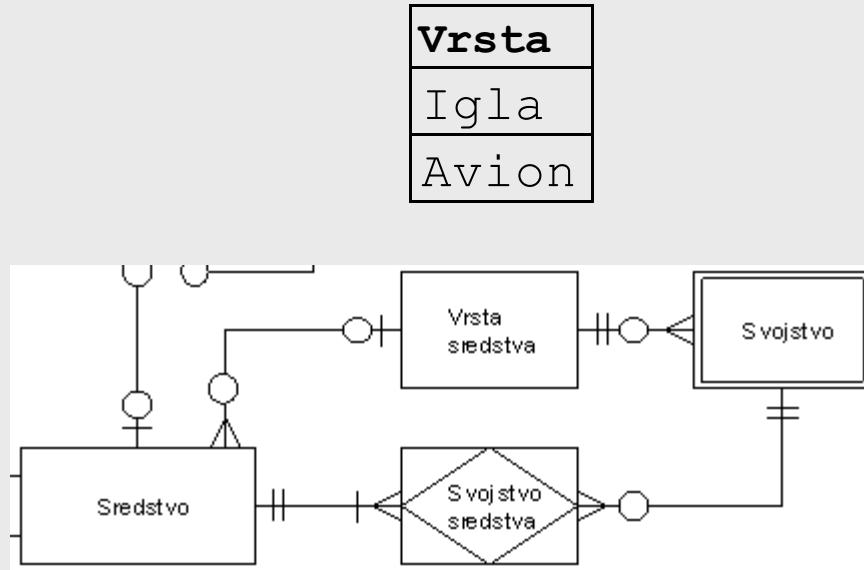
Ideja aplikacijskog meta-modeliranja (3)

□ Primjer, narudžba



Primjena na opisivanje podataka

□ Primjer: opisivanje podataka



Sredstvo	Vrsta
A380	Avion
B2	Avion

Vrsta

Igra
Avion

Vrsta

Vrsta	Svojstvo
Avion	Broj Sjedala
Avion	Dolet
Igra	Duljina
Igra	Promjer

Sredstvo

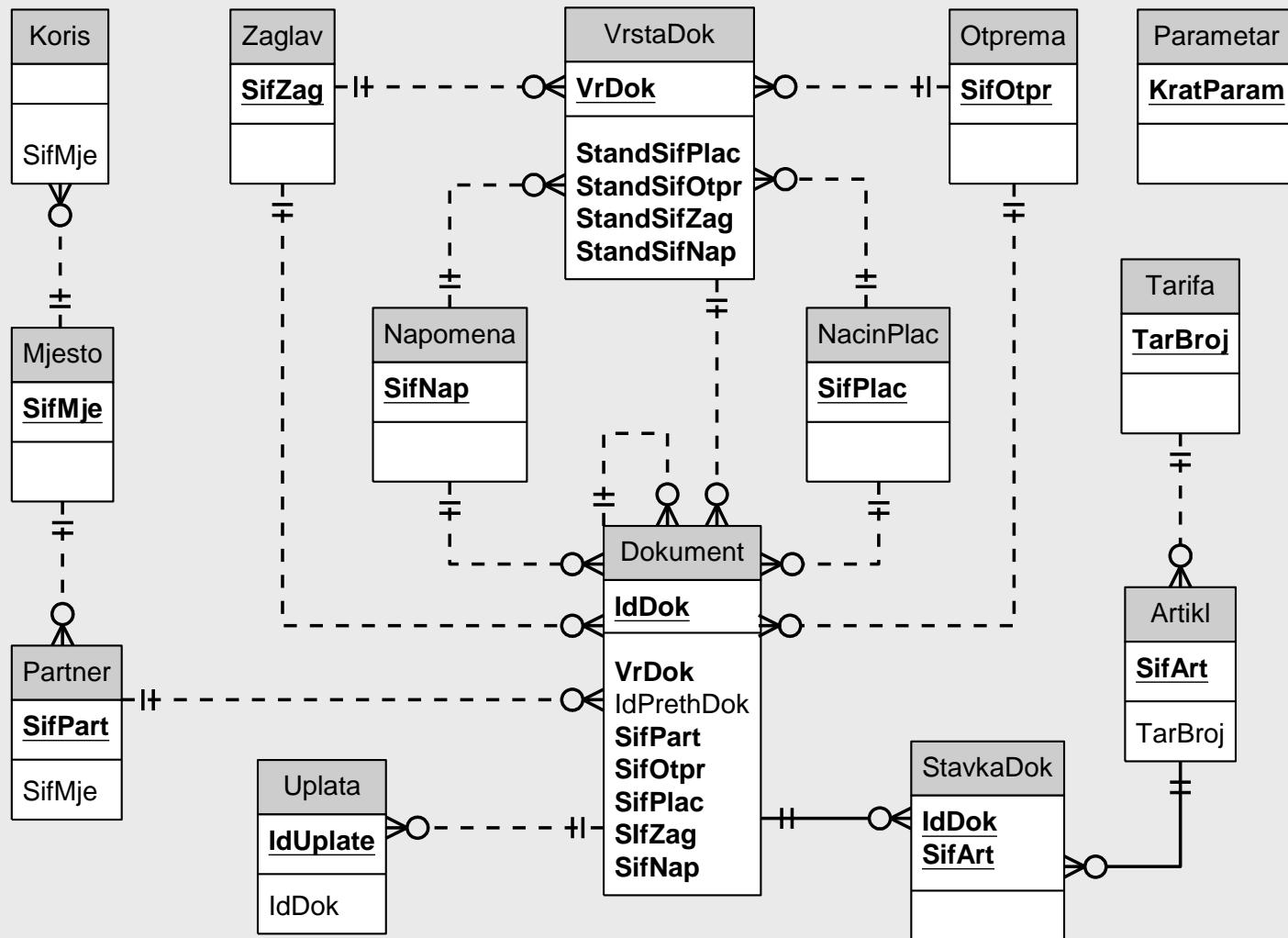
Sredstvo	Svojstvo	Vrijednost
B2	Sjedala	2
B2	Dolet	18000
A380	Sjedala	853
A380	Dolet	15200

□ Primjer: unos podataka

□ Što s tipovima vrijednosti ?

Praktična primjena meta-modeliranja

- Supstitucija većeg broja tablica jednake ili slične strukture
 - Upit, Ponuda, Narudžba, Otpremnica, Primka, Račun, ... sa stavkama



Primjer meta-podataka

□ Definiranje dokumenata, modeliranje poslovnih pravila

Vrsta	Naziv	Naziv opcije	Mat.pred	Fin.pred	Stand.plać	Stand.otpr	Stand.zag	Stand.nap
S	Korekcija salda	Korekcije &salda	0	1	1	2	13	10
K	Korekcija skladišta	&Korekcije	1	0	1	2	13	10
N	Narudžba	&Narudžbe	0	0	1	2	13	10
I	Otpremnica	O&tpremnice	-1	0	1	2	13	10
O	Otpremnica-Račun	&Otpremnice-Računi	-1	1	1	2	13	10
D	Predračun	Pre&dračuni	0	0	1	2	13	10
P	Primka	&Primke	1	0	1	2	13	10
R	Račun	&Izlazni računi	-1	1	2	2	13	10
U	Ulazni račun	&Ulazni računi	0	-1	1	2	13	10
▶		Šifra zag			Tekst zaglavlja			
		▶ 0						
		13						
		18 Na temelju Vaše narudžbenice zaračunavamo Vam slijedeće:						
		20 Na temelju našeg ugovora o kupoprodaji video kaseta naručujemo:						
		25 Na temelju našeg dogovora o prodaji knjiga, molimo da nam pošaljete:						
		26 Na temelju Vaše pismene narudžbe zaračunavamo Vam slijedeće:						
		27 Na temelju našeg dogovora o tisku radnih listova, šaljemo Vam narudžbu :						
		28 Na temelju Vaše tel. narudžbe zaračunavamo Vam :						
		29 Na temelju Vaše usmene narudžbe zaračunavamo Vam :						
		30 Na temelju našeg obračuna prodaje radnih listova molimo Vas da nam fakturirate :						
		31 KOMISIONA PRODAJA						
		32 Na temelju Vašeg izvještaja o komisionoj prodaji zaračunavamo Vam :						
		33 Na temelju Vaše telefonske narudžbe šaljemo Vam uzorke :						
		34 Reklamni uzorci						
		35 Zamjena knjiga:						
		36 Na temelju našeg dogovora o kompenzaciji šaljemo Vam račun:						
		37 Zamjena kaseta :						

Primjer aplikacije nad meta-modelom

□ Primjer, prilagodba funkcionalnosti

Dokument

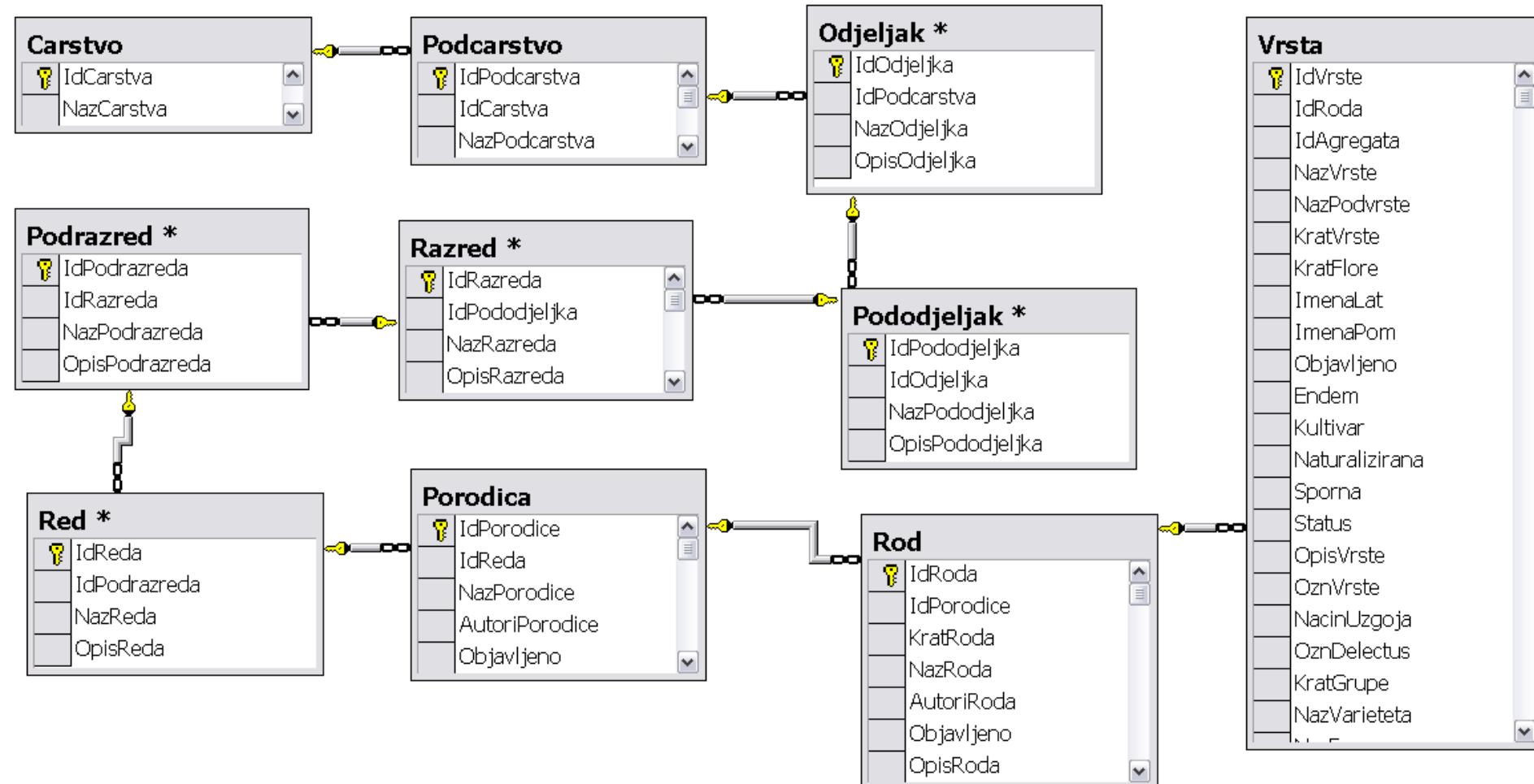
Vrsta	Predračun	Broj	102	Datum	09.09.96	Prethodni		Broj	
Partner	Dječji vrtić "Vrapčići"								
Datum valute	20.09.96	Dana valute	12	10 % Rabat	Račun PP	<input type="checkbox"/>	Br. ulaznog dokumenta		
Otprema	HPT	Plaćanje	Virman	Ukupno	432.00	Datum storna		Datum ažuriranja	
Ostali troškovi	manipulacija + poštarnica								
Slovima	četristo pedeset kuna								
Zagлавje	Napomena	Virman	Ček	Gotovina	Kreditno pismo	Otkupnina		Datum ispisa	09.09.96
ESC									
Šifra	Naziv	Kolicina	Cijena	Iznos	%Rab	Rabat	%PP	Porez	Ukupno
1	Pokaži što znaš	8.00	20.00	160.00	10	16.00	0	0.00	144.00
2	Pokušaj nešto novo	8.00	20.00	160.00	10	16.00	0	0.00	144.00
3	Učimo opažati	8.00	20.00	160.00	10	16.00	0	0.00	144.00
Stavka [◀◀] 1 od 3 [▶▶] 480.00 48.00 0.00									
Dokument [◀◀] 1 od 6 [▶▶] Brisanje 0 Ažuriranje Upute Storno									
F3 - Brisanje F5 - Tablica F7 - Ispis F8 - Unos F9 - Dohvat F10 - Spremi ESC - Kraj									

□ Primjer: pojednostavljenje izračuna

```
SELECT Artikl.SifArt, Artikl.NazArt,
       SUM(StavkaDok.Kolicina*VrstaDok.PredMat)
  FROM Dokument INNER JOIN VrstaDok ON Dokument.VrDok=VrstaDok.VrDok
                INNER JOIN StavkaDok ON StavkaDok.IdDok = Dokument.IdDok
                INNER JOIN Artikl ON StavkaDok.SifArt = Artikl.SifArt
 WHERE VrstaDok.PredMat <> 0
       AND Artikl.IdeSklad <> 0
       AND Dokument.DatSklad IS NOT NULL
       AND Dokument.DatStorno IS NULL
 GROUP BY Artikl.SifArt, Artikl.NazArt
```

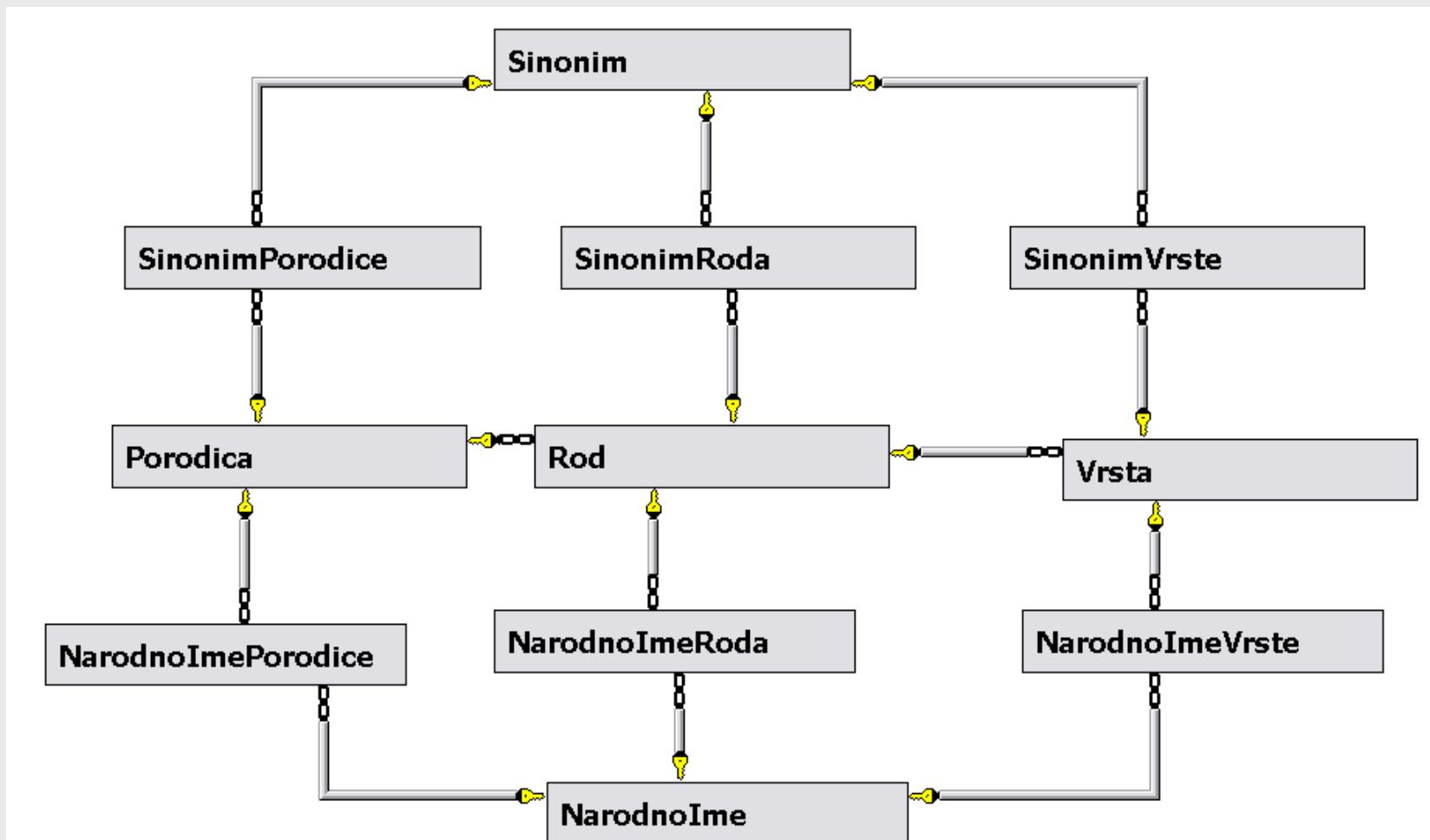
Primjer: kaskada sličnih entiteta

- Svi imaju identifikator i naziv, neki imaju opis i objavljeno ...
- Moguće rješenje: generalizacija (klasa) s 9 specijalizacija



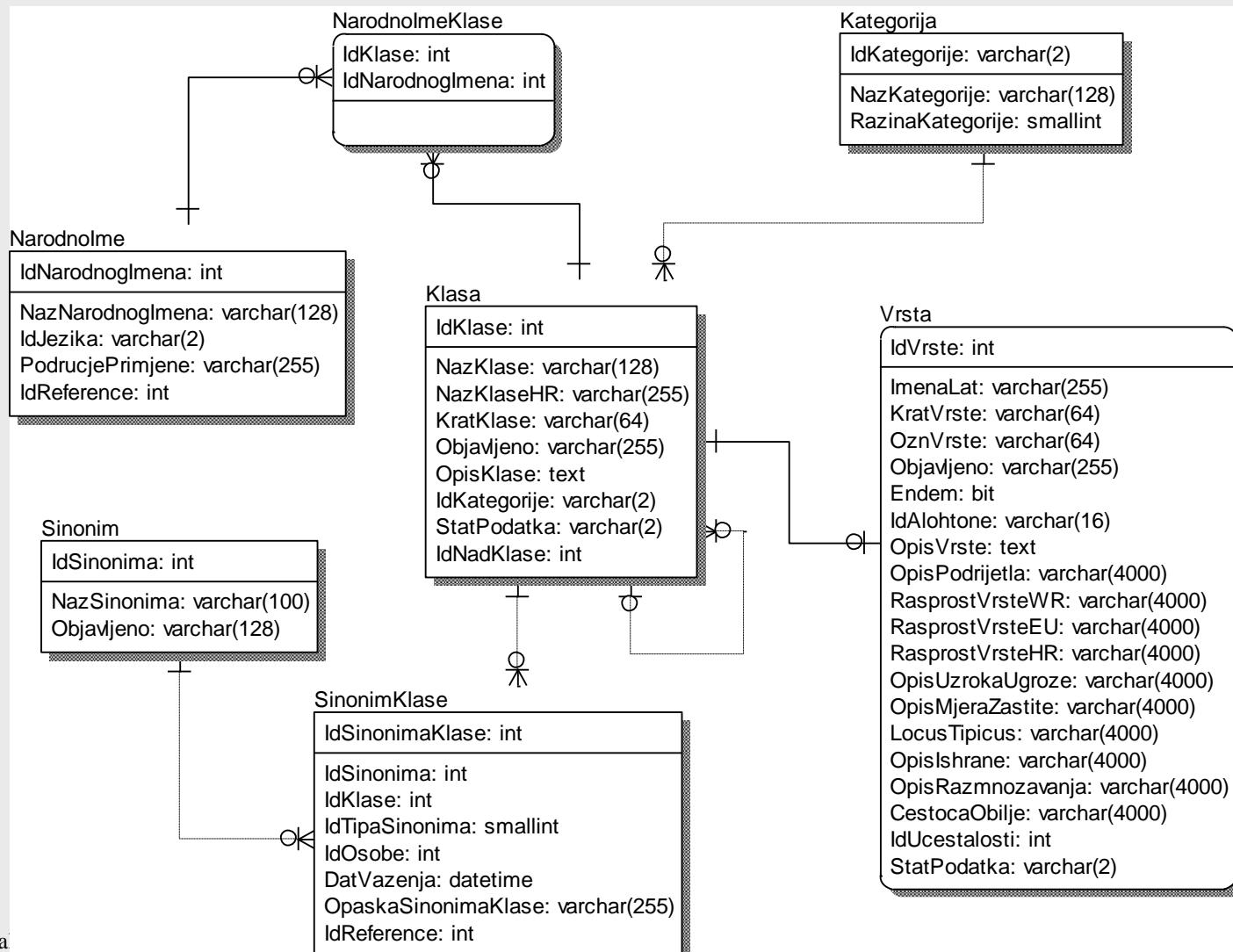
Primjer: kaskada sličnih entiteta (2)

- ❑ Neki entiteti međutim imaju i spojne tablice na narodna imena i sinonime, približno jednake strukture
- ❑ Rješenje: još tri specijalizacije ?

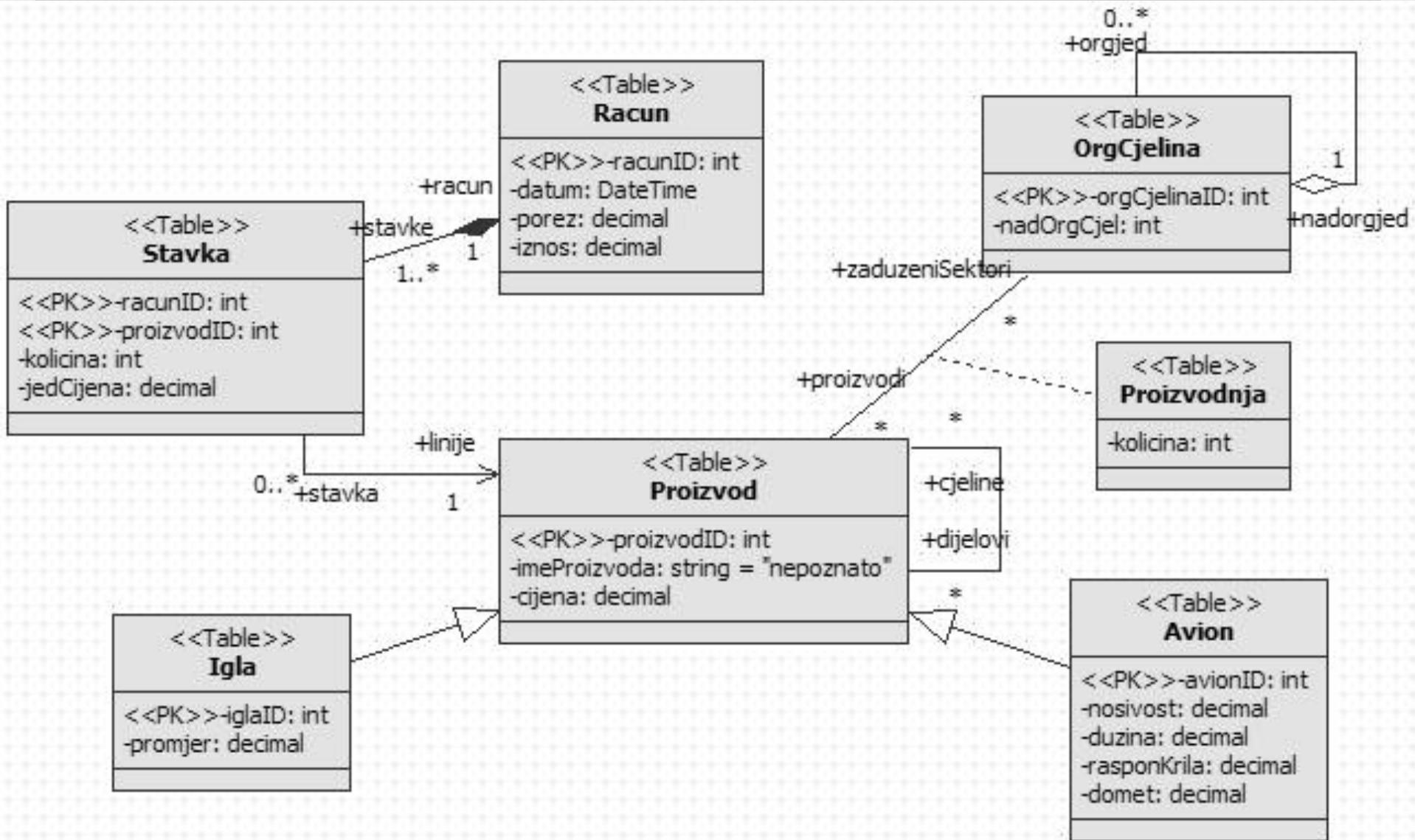


Primjer: kaskada nakon modeliranja

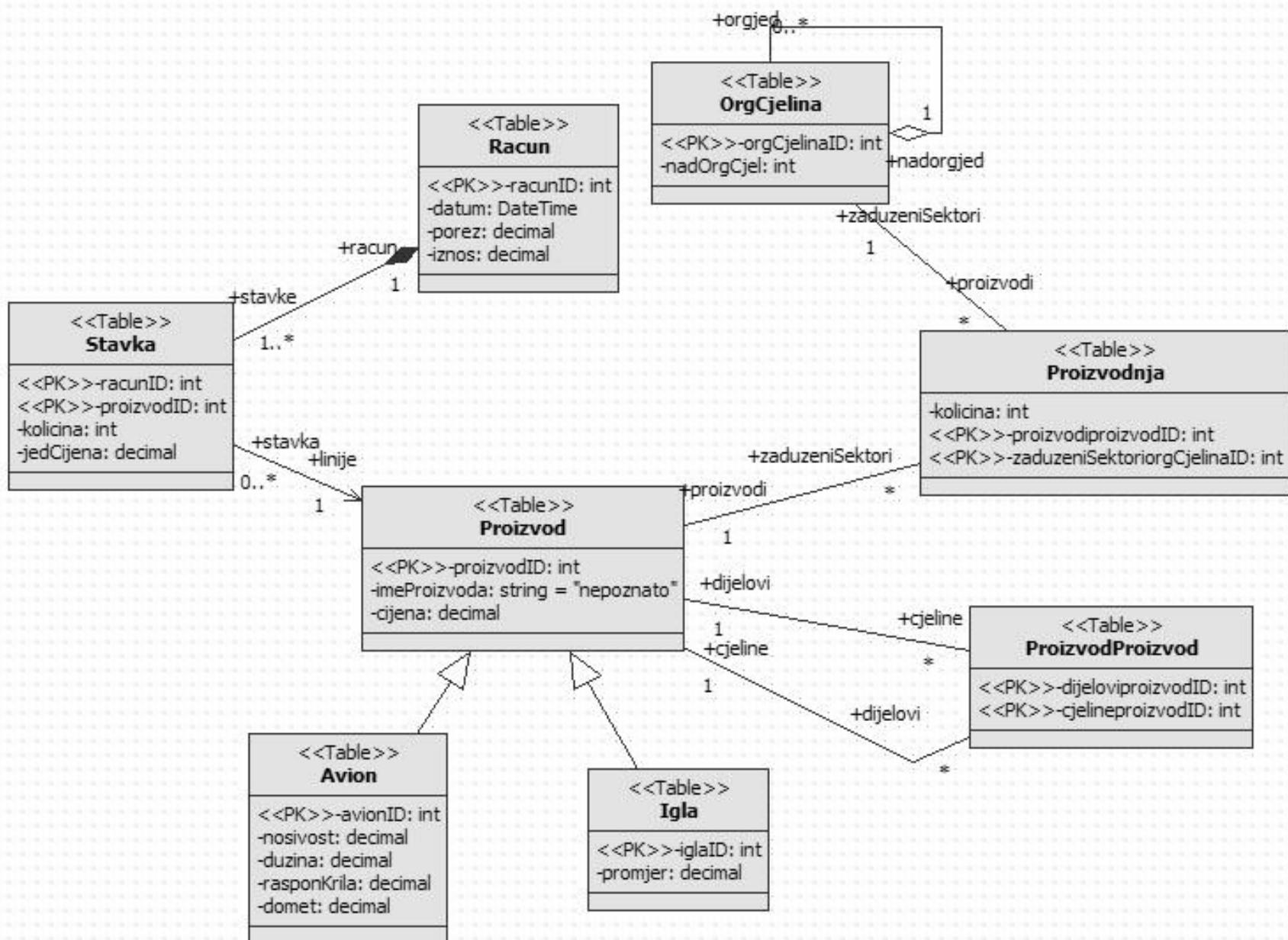
- rekurzivna hijerarhija, izjednačene strukture, jedna specijalizacija te spojni entiteti na narodna imena i sinonime (samo za entitete koji ih stvarno imaju)



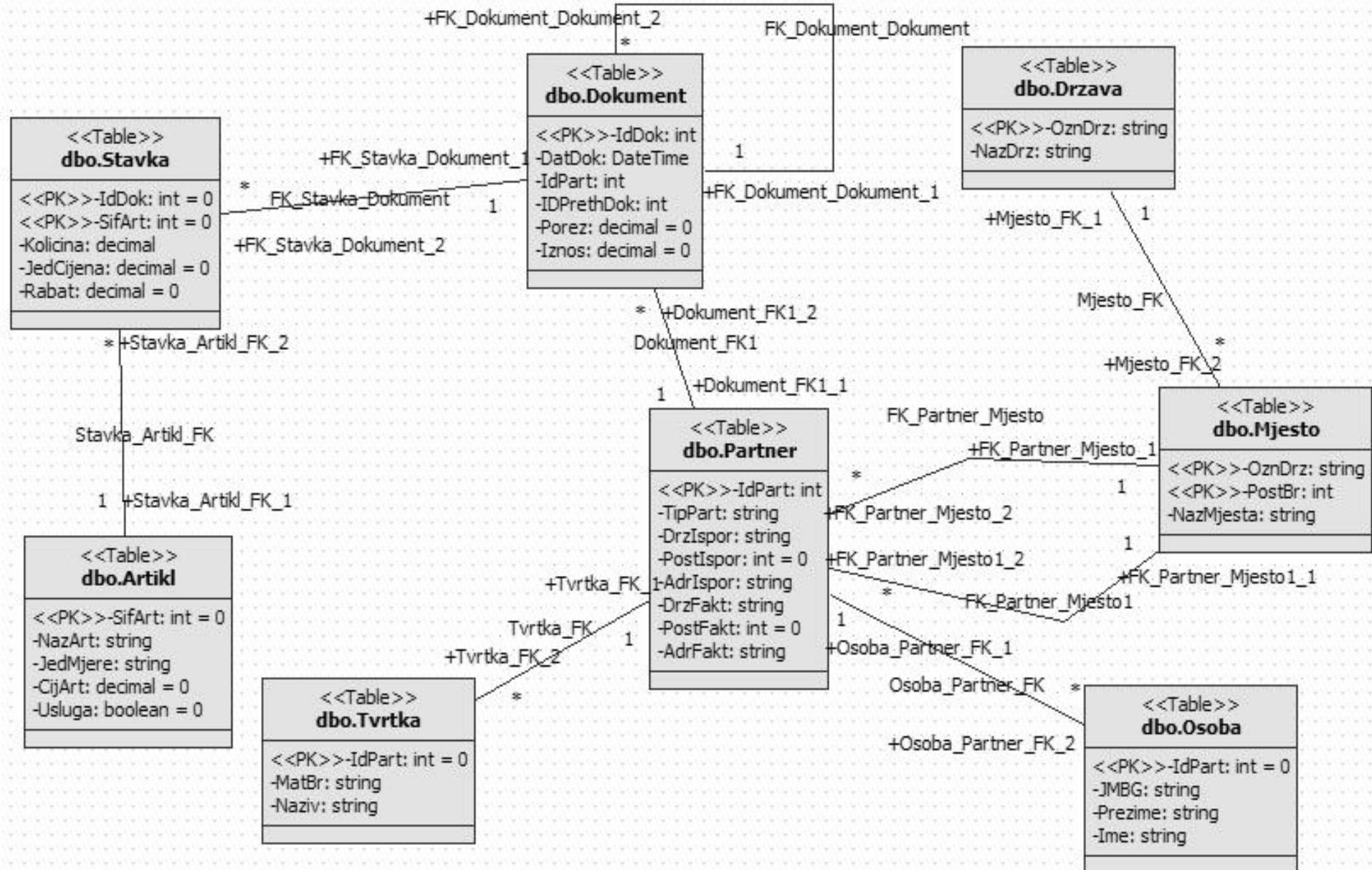
Za sladokusce: konceptualni UML model ...



... model nakon transformacije ...

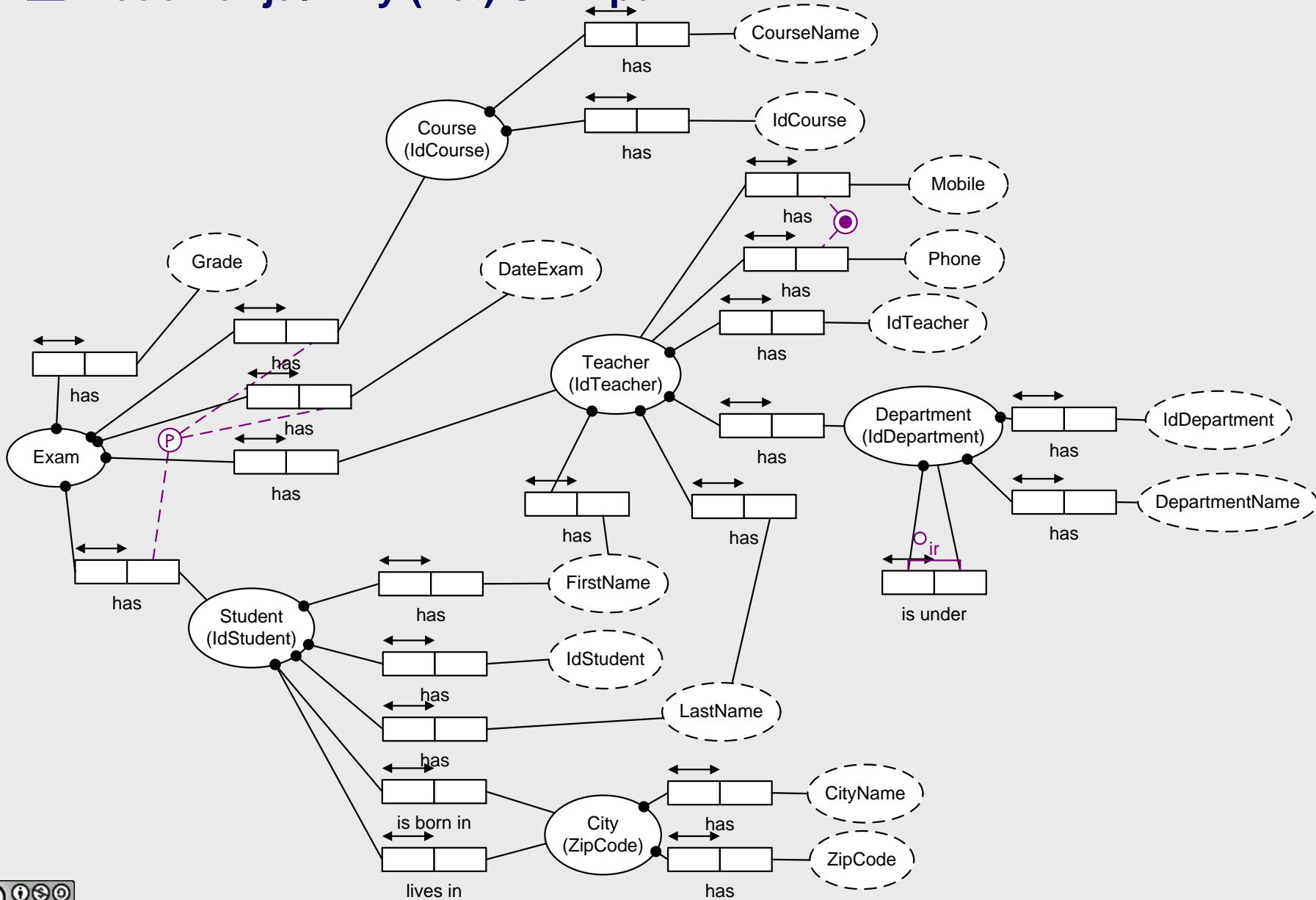


... i klasični relacijski model



Object Role Modeling (ORM)

Modeliranje\ Why (not) ORM.pdf



Primjeri i zadaci



□ Primjer: Resursi\BazePodataka

- Poduzeće – temeljem ER modela s predavanja
- Organizacija – model iz jednog projekta
- FirmaRIS – primjeri aplikacija na narednim predavanjima
- Projekt – podloga aplikacija za naredna predavanjima

□ Dodatak: Dodaci\RIS05-dodatak

- ER notacija, razvoj modela

□ Zadaci za razgibavanje

- ER model cjenika u restoranu brze prehrane
 - Različite cijene samostalnog proizvoda i proizvoda u paketu
- ER model videoteke (Član, Posudba, Rezervacija, Video)
 - Na posudbu se plaća zakasnina



□ Domaća zadaća - čime crtati ?

- Primjer: Modeliranje\RIS-preporuke

[neke] Reference

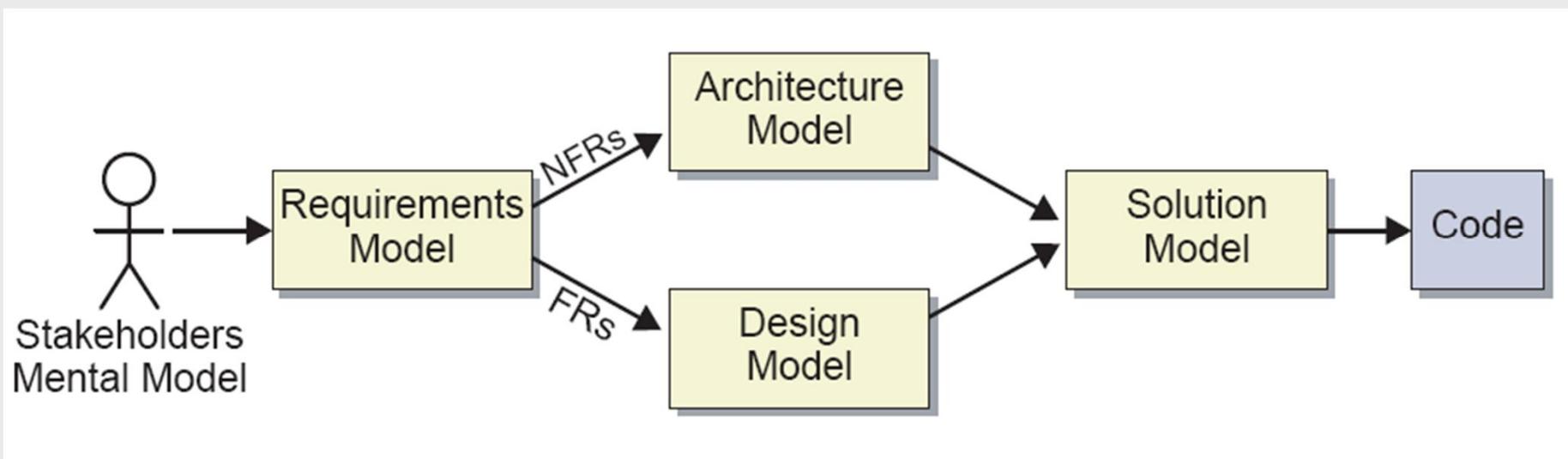
- A. Olive: **Conceptual Modeling of Information Systems**, Springer, 2007.
- A UML Profile for Data Modeling
 - <http://www.agiledata.org/essays/umlDataModelingProfile.html>
- Entity Relationship Modeling with UML
 - http://www.ibm.com/developerworks/rational/library/content/03July/2500/2785/2785_uml.pdf
- A Repository Model - Business Rules
 - <http://www.tdan.com/view-articles/4978/>
- OMG's MetaObject Facility
 - <http://www.omg.org/mof>

Objektno orijentirani razvoj



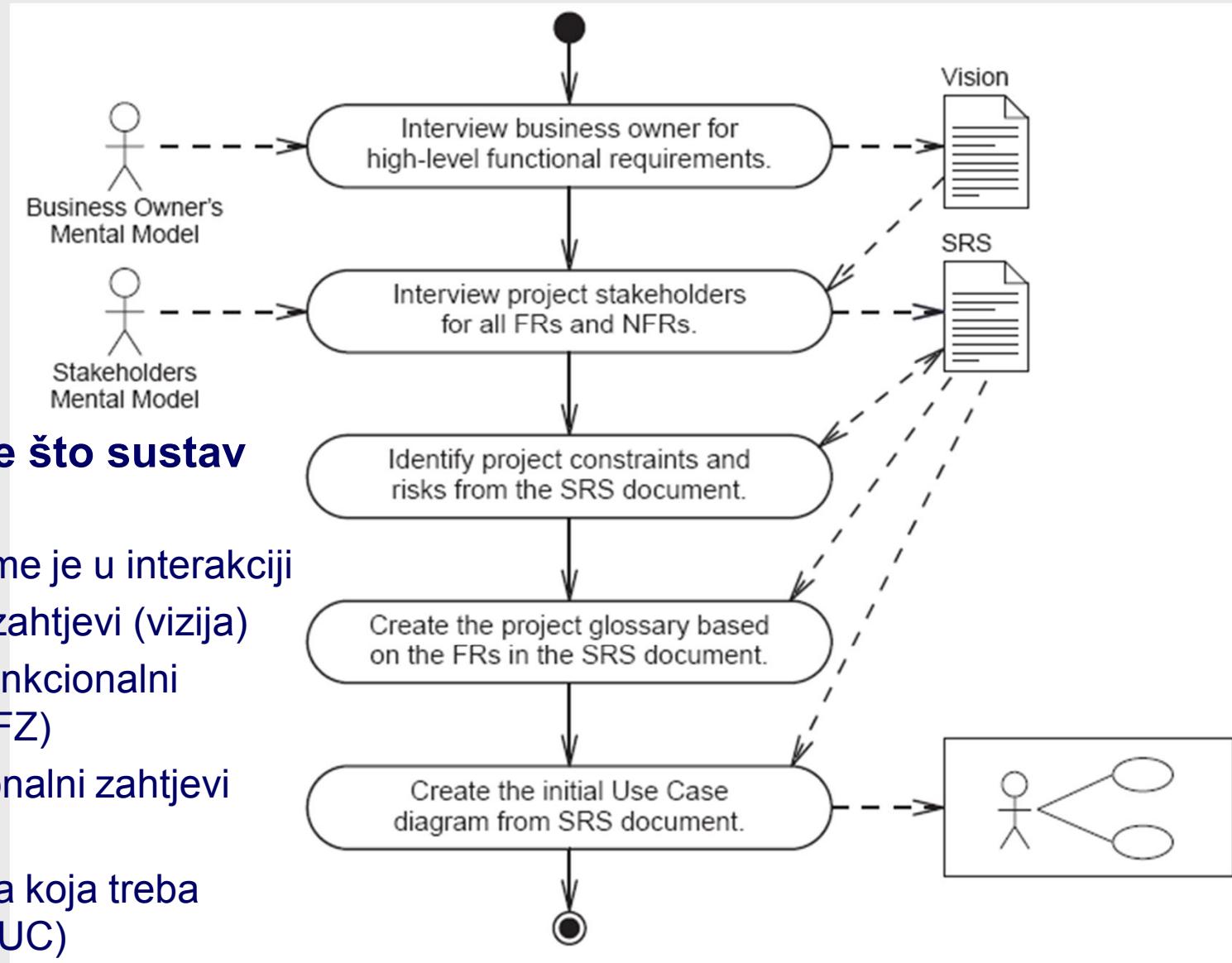
Objektno orijentirani razvoj softvera

- niz slijednih transformacija od mentalnog modela dionika do koda
- NFRs – programskim okvirima, ponovno iskoristivim knjižicama, ...
- FRs – projektiranjem prema zahtjevima korisnika



© Object-Oriented Analysis and Design Using UML
Sun Microsystems, Inc.

Radni tok prikupljanja zahtjeva



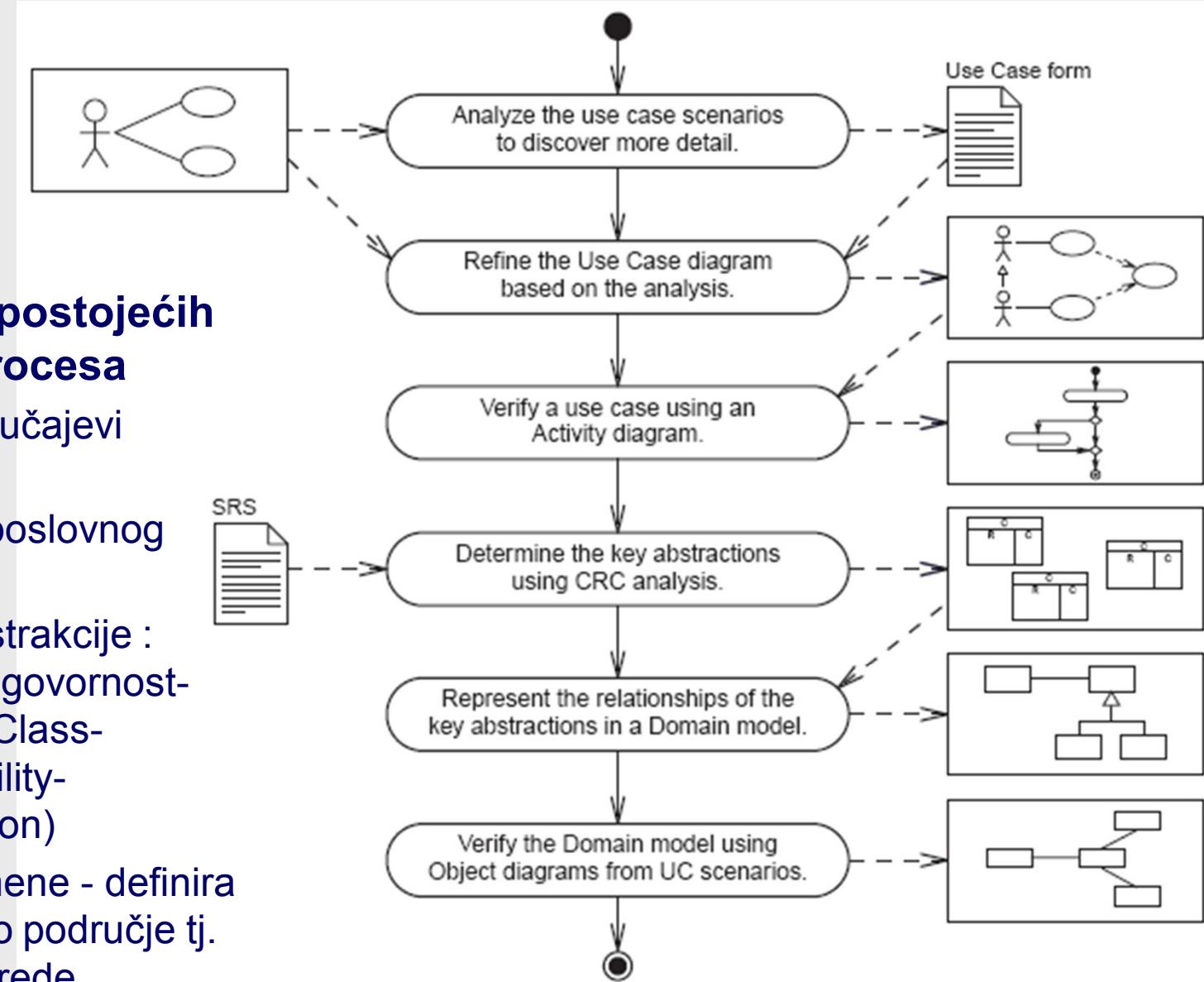
□ Određivanje što sustav treba činiti

- s kime/čime je u interakciji
- poslovni zahtjevi (vizija)
- detaljni funkcionalni zahtjevi (FZ)
- nefunkcionalni zahtjevi (NZ)
- ponašanja koja treba podržati (UC)

Radni tok analize zahtjeva (→ modeliranje funkcija)

□ Modeliranje postojećih poslovnih procesa

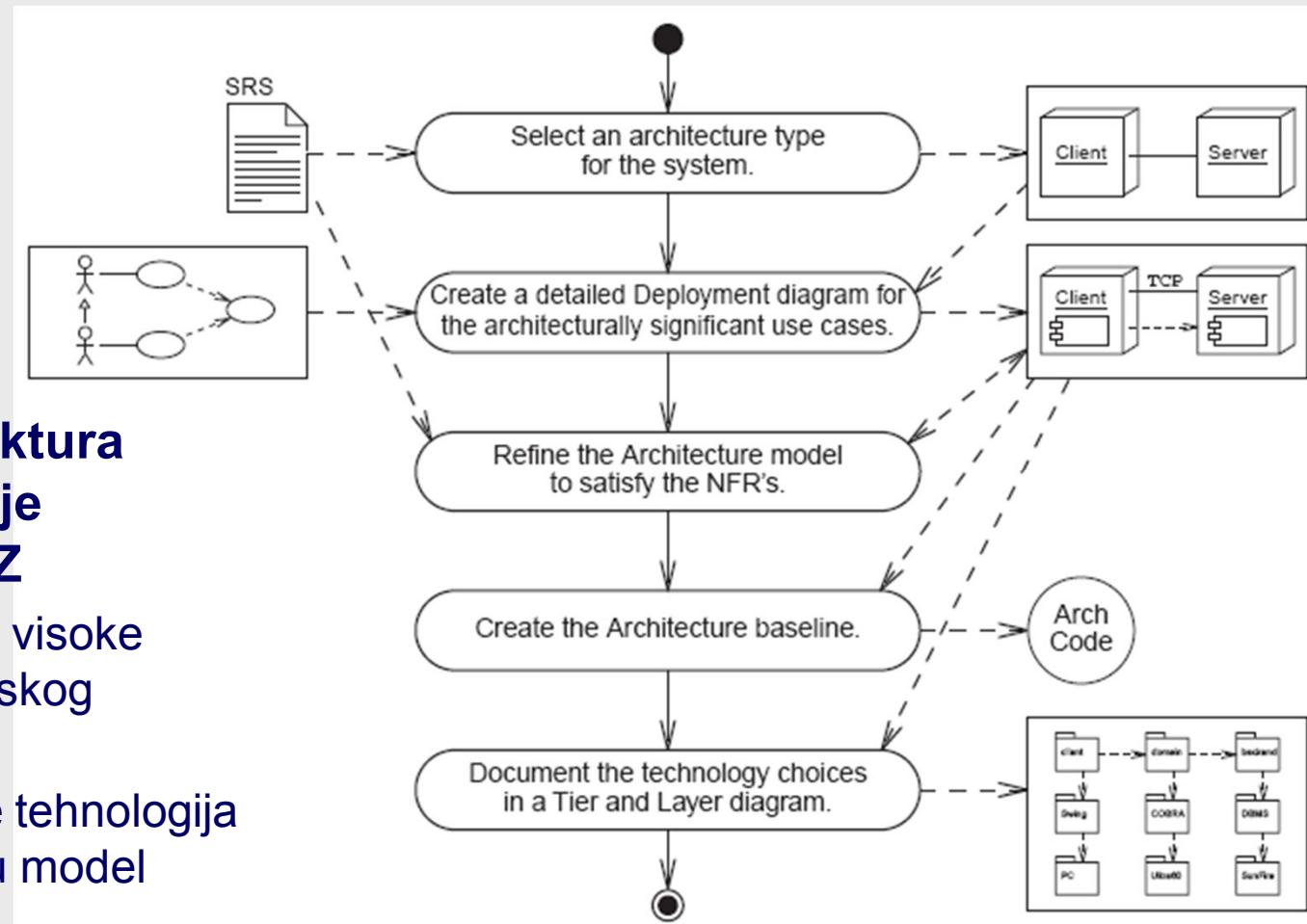
- rafinirani slučajevi korištenja
- aktivnosti poslovnog procesa
- ključne apstrakcije : Razred-Odgovornost-Suradnja (Class-Responsibility-Collaboration)
- model domene - definira problemsko područje tj. ključne razrede



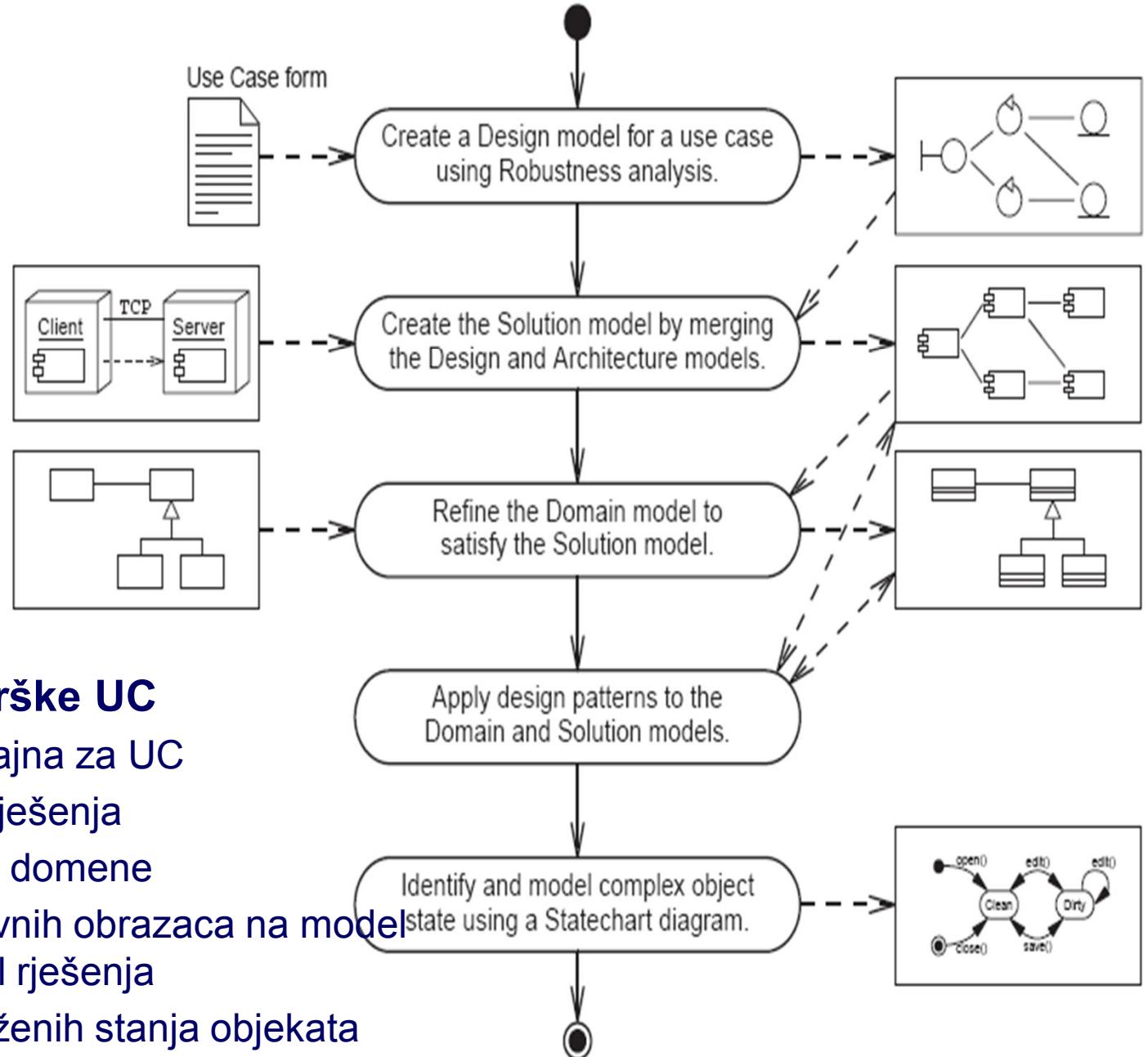
Radni tok modeliranja arhitekture

□ Modeliranje struktura visoke razine koje zadovoljavaju NZ

- razvoj struktura visoke razine programskog rješenja
- ustanavljanje tehnologija koje podržavaju model arhitekture
- pojašnjenje modela arhitekture obrascima arhitekture koji zadovoljavaju NZ



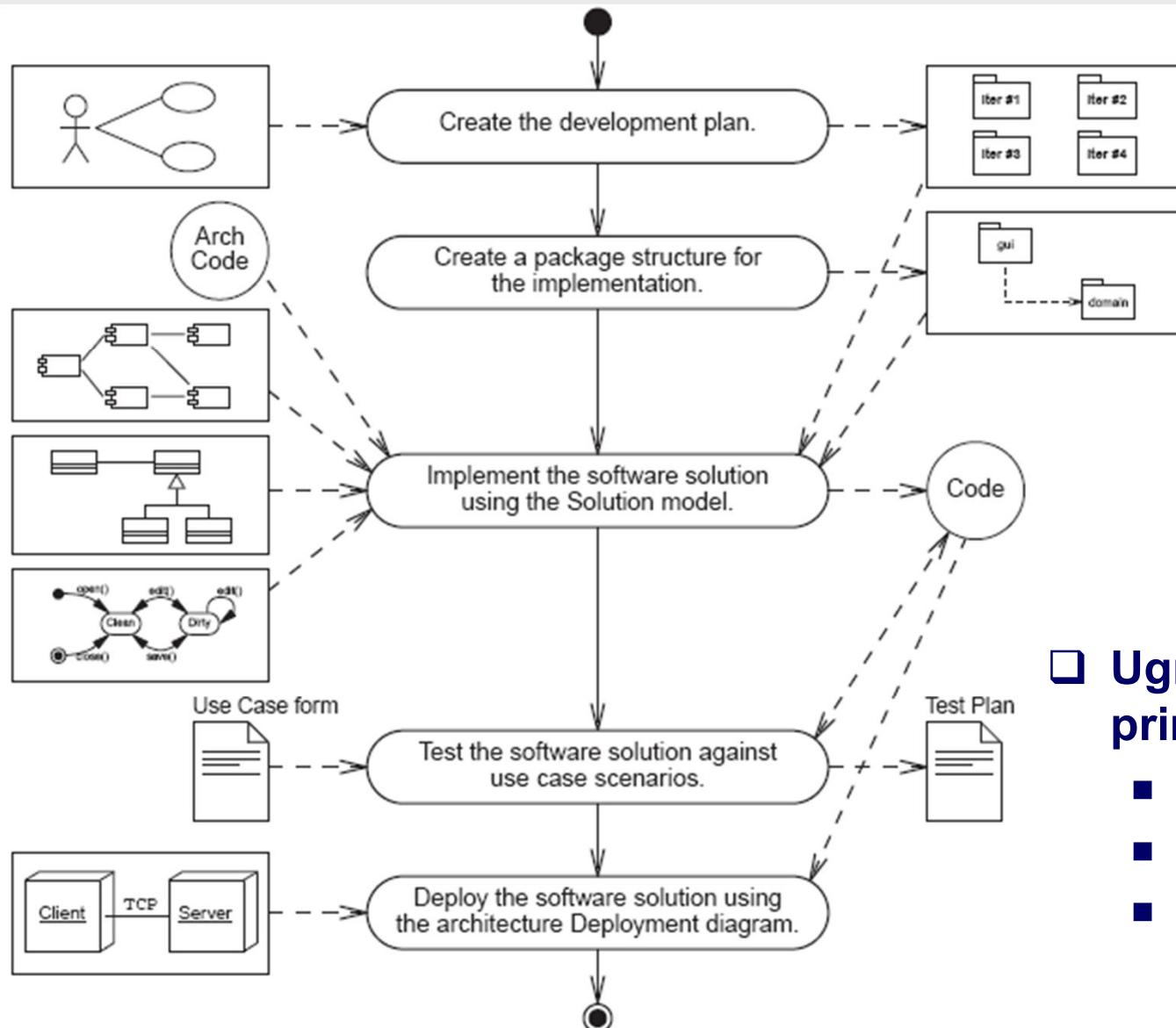
Radni tok dizajna



□ Modeliranje podrške UC

- modeliranje dizajna za UC
- izrada modela rješenja
- razrada modela domene
- primjena oblikovnih obrazaca na model domene i model rješenja
- modeliranje složenih stanja objekata

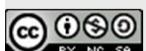
Radni tok konstrukcije



□ Ugradnja, provjera i primjena sustava

- implementacija rješenja
- testiranje
- primjena u radnom okruženju

Dijagrami aktivnosti



Dijagram aktivnosti

□ Prikazuje

- ili tok rada (workflow) između objekata nekog slučaja korištenja
- ili tok upravljanja za neku operaciju

□ Osnovni simbol: aktivnost ili akcija

- akcija – jednostavna, nedjeljiva obrada
- aktivnost – skup akcija

□ Sadrži tok aktivnosti, ali ne i objekte

- sva (ili većina) stanja su stanja akcija
- sva (ili većina) prijelaza obavlja se po dovršetku akcija u stanju

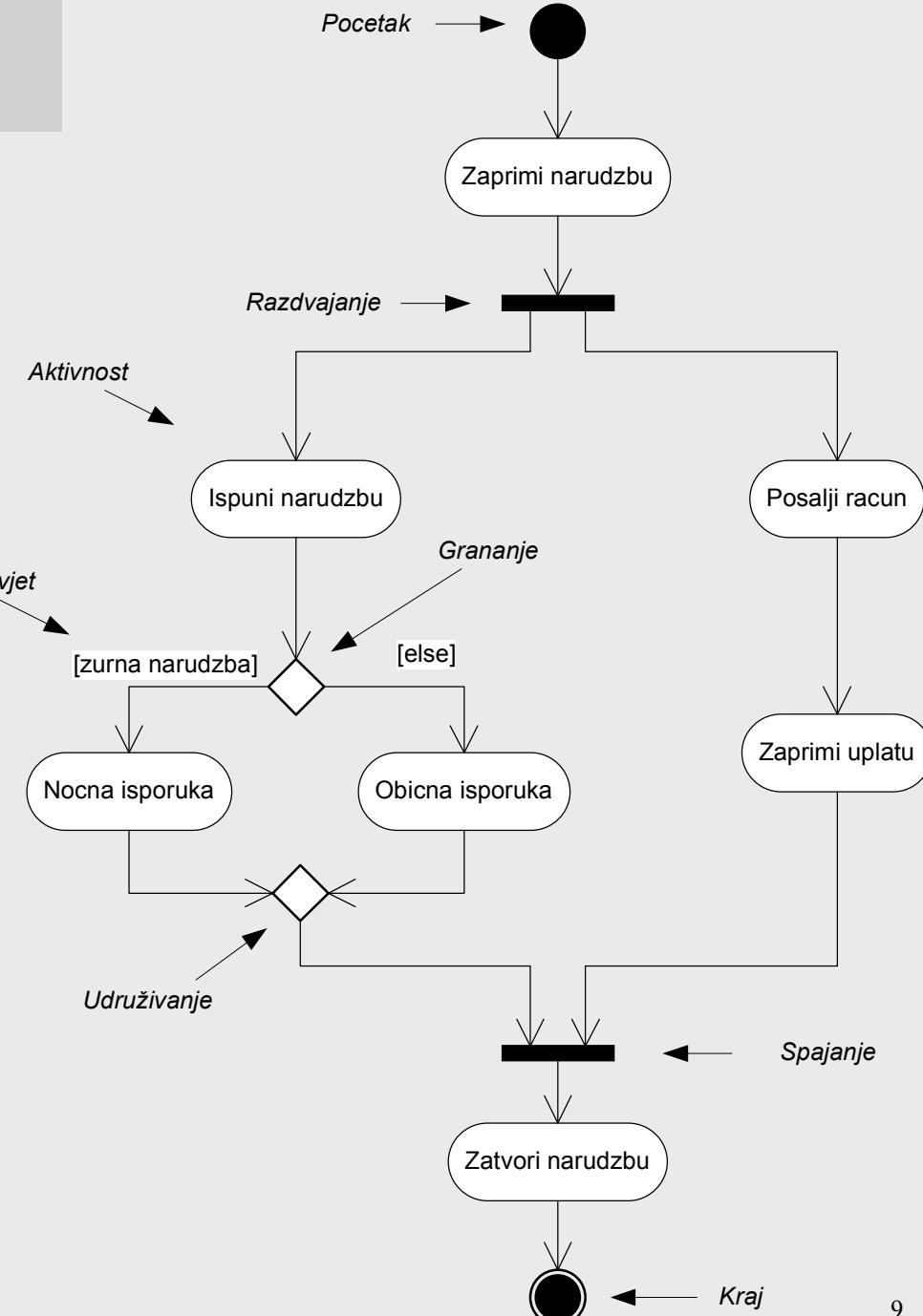
□ Odluka - uvjetovano ponašanje

- grananja (branches)
- udruživanja (merges)

□ Paralelizam, istovremenost procesa

- razdvajanje, račvanje (fork)
- spajanje, pripojenje (join)
- grana predstavlja nit izvođenja (thread)

□ Uvjet grananja (guard condition)



Paralelizam i uvjetne niti

□ Oznake za razdvajanje i spajanje moraju se poklapati

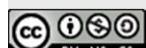
- za svako razdvajanje mora postojati spajanje niti nastalih razdvajanjem

□ Dodatna pravila (sljedeći slajd)

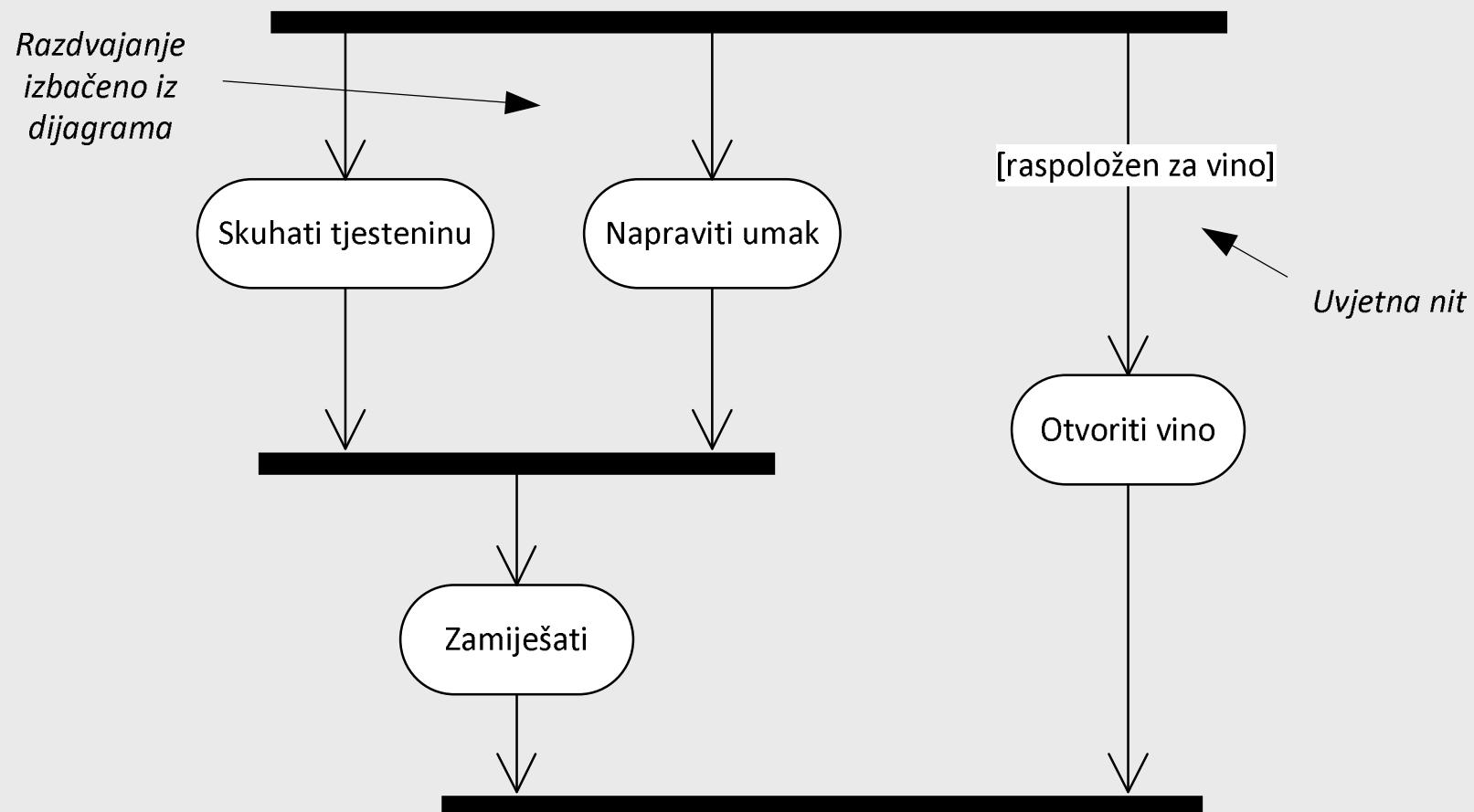
- razdvojena nit može se i sama razdvojiti, npr. 1:2,3 zatim 3:4,5
- novonastale niti spajaju se obrnutim redom, npr. spoj 4,5 pa 2,3
- ako nit koja izlazi iz razdvajanja ide direktno u novo razdvajanje, dozvoljeno je izostaviti drugo razdvajanje a niti premjestiti da izlaze iz prvog
- analogno, mogu se udružiti neposredna slijedna spajanja
- postoji posebno stanje, tzv. sinkronizacijsko stanje (sync state) koje omogućava sinkronizaciju na mjestima na kojima bi pravilo o parovima razdvajanja i spajanja inače sprječavalo napredak

□ Uvjetna nit (conditional thread)

- ako uvjet u niti koja izlazi iz oznake razdvajanja nije ispunjen, smatra se da je ta nit završila



Primjer skraćenog prikaza i uvjetnih niti



Razlaganje aktivnosti

□ Dekomponiranje aktivnosti

- složena aktivnost - u podaktivnosti

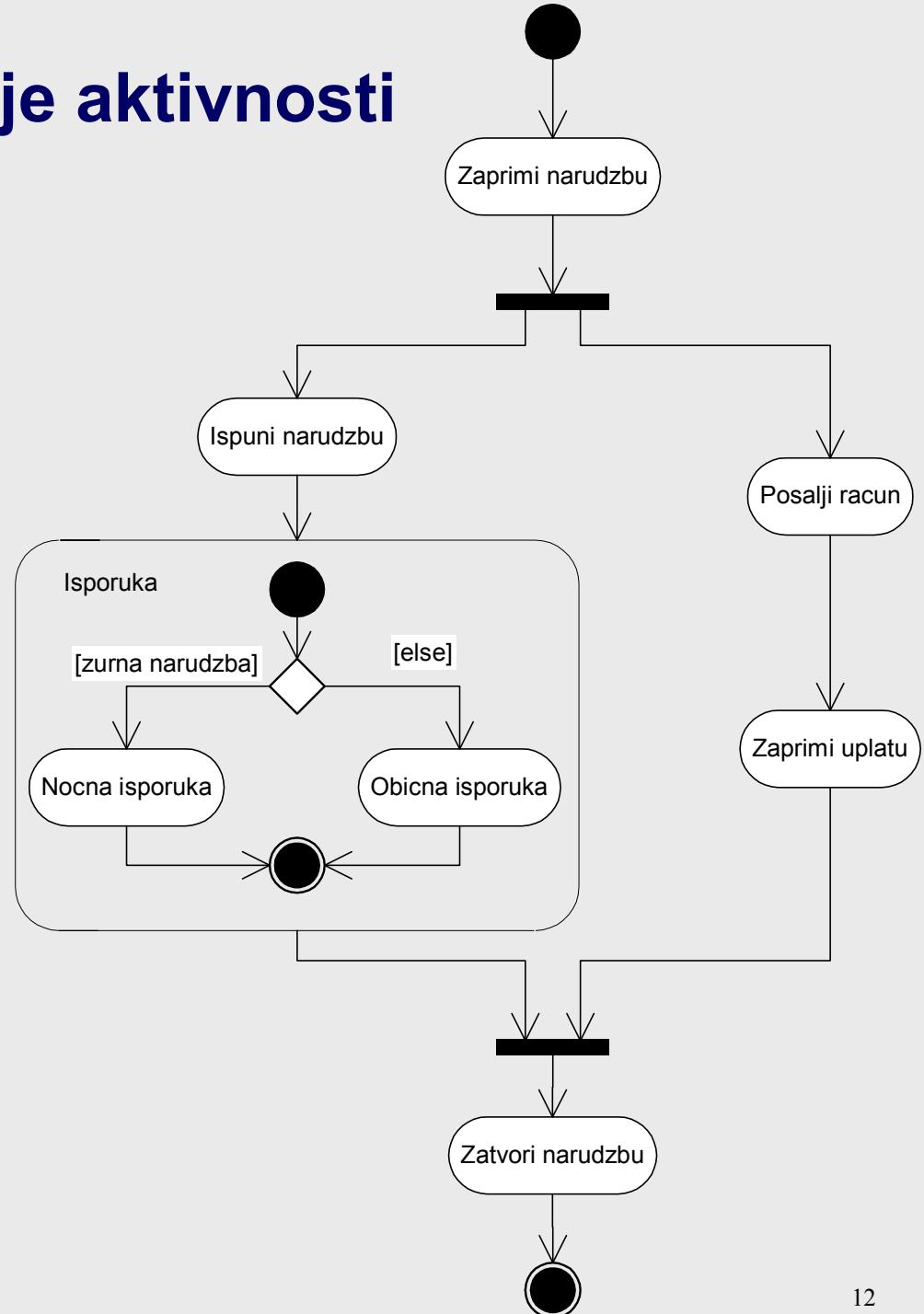
□ Alternative prikaza složene aktivnosti

- prikaz s podaktivnostima (na slici)
- prikaz bez podaktivnosti (vidjela bi se samo aktivnost Isporuka)

□ Unutar razložene aktivnosti mogu se prikazati točke početka i kraja

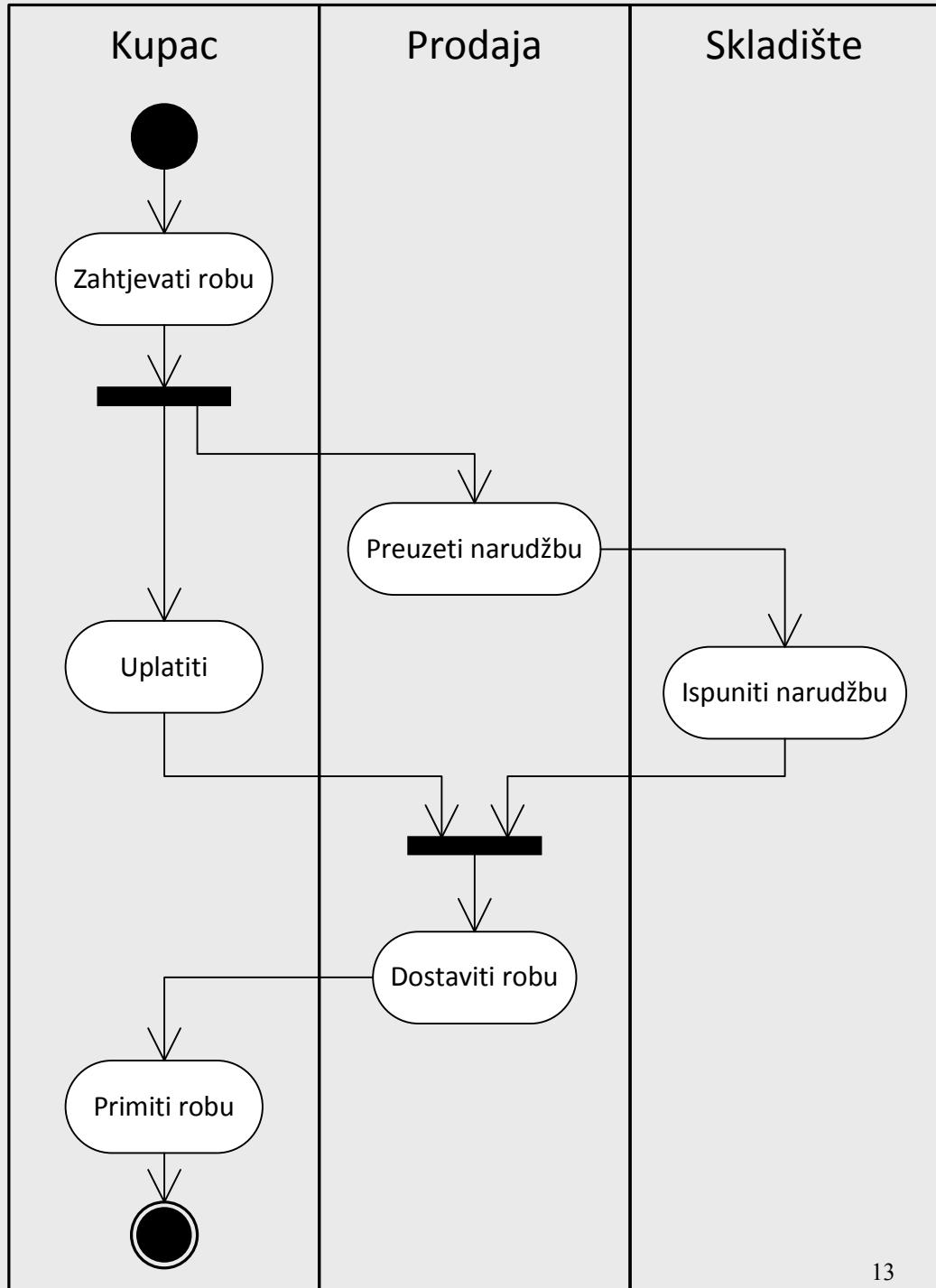
- ako se ne prikazuju tada prijelazi između stanja aktivnosti idu direktno u dijagram, odnosno iz poddijagrama.

□ Zašto je to važno ?



Plivaće staze

- Osnovni DA pokazuje koje se aktivnosti obavljaju i kojim redoslijedom ali ne i tko što radi
- Staze (swimlanes)
 - staze predstavljaju **odgovornost pojedinog razreda ili sudionika**-osobe, a najčešće su to organizacijske jedinice
- Svaka aktivnost pridružena je jednoj stazi
 - tokovi mogu prelaziti kroz više staza



Primjena dijagrama aktivnosti

□ Analiza slučaja korištenja

- DA dobro pokazuje slijed akcija **za nekoliko objekata i nekoliko UC**
- za prikaz ponašanja objekata za jedan UC bolji je dijagram interakcija
- na konceptualnoj razini nije ključno dodjeljivanje aktivnosti objektima
 - dobiti opću sliku ponašanja sustava, te odrediti međuzavisnosti u ponašanju – objekti se vezuju kasnije

□ Razumijevanje radnog, poslovnog procesa

- modeliranje radi analize procesa (mogućeg poboljšanja)
- potrebno uključivanje korisnika poznaje poslovni proces

□ Opisivanje složenih slijednih algoritama (obrade podataka)

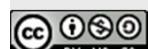
- u ovom slučaju DA samo zamjena za dijagrama toka (tzv. blok dijagram)

□ Opisivanje višenitnih aplikacija



Modeliranje funkcija

Functional modeling



Modeliranje funkcija

□ Modeliranje funkcija (functional modeling)

- Opisuje poslovne procese i interakciju IS s okolinom
- Dokumentiranje zahtjeva i vanjskog ponašanja sustava

□ Logički modeli - problemskog područja (problem domain models)

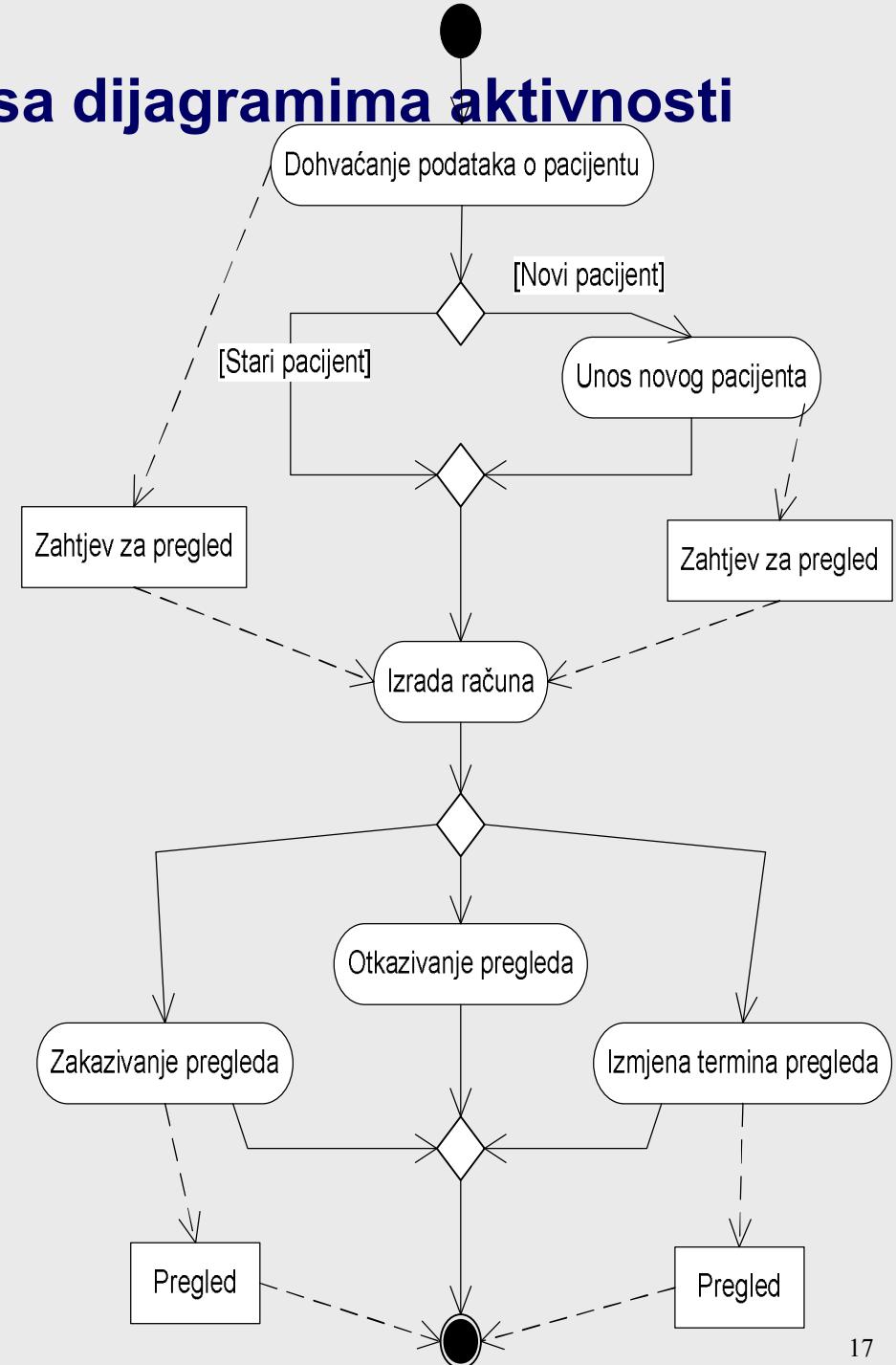
- opis aktivnosti bez detalja provedbe
- ne mora se znati izvode li se računalom ili ručno !
- **Dijagram aktivnosti**
 - logičko modeliranje poslovnih procesa i tokova rada
 - prikaz procesa ili aktivnosti te kolanja podataka
- **Slučaj korištenja**
 - prikaz aktivnosti korisnika
 - vanjski ili funkcionalni pogled na IS - interakcija sustava i okoline

□ Fizički modeli

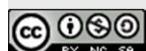
- temeljem logičkih modela za vrijeme faze oblikovanja
- informacije potrebne za izgradnju sustava

Modeliranje poslovnih procesa dijagramima aktivnosti

- "sofisticirani dijagram toka podataka"
 - dodatni elementi AD
- Objekt (object node)
 - razred objekata povezan s tokom
- Upravljački tok (control flow)
 - puna strelica
 - slijed izvršenja
 - pridruživanje samo akcijama ili aktivnostima
- Podatkovni tok (object flow)
 - protok objekata između aktivnosti
 - isprekidana strelica
 - povezuje objekt i aktivnost



Slučajevi korištenja



Slučajevi korištenja

□ Analiza slučajeva korištenja (Use Case Analysis)

- tehnika analize poslovnih procesa iz perspektive korisnika

□ Use case = Slučaj korištenja (primjene)

- skup scenarija sa zajedničkim ciljem korištenja sustava

□ Scenarij

- niz koraka koji opisuju interakciju korisnik-sustav

□ Tipični sadržaj

- kako kreće i završava slučaj
- normalni tok događaja
- varijabilni (alternativni) ili iznimni tok događaja

□ Standard izrade UC-a ne postoji (nije dio UMLa) !

- alternative i iznimke mogu biti zasebni slučajevi korištenja



Primjer scenarija

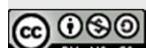
Use Case: Posudba

Tipični tok

Aktivnosti sudionika	Odziv sustava
1. Član pristupa pultu s medijima koje želi posuditi.	
2. Član daje identifikacijsku karticu Djelatniku, koji u sustavu provjerava valjanost.	3. Prikaz informacije o članu i status prethodnih posudbi ako postoje, uključujući kašnjenja.
4. Za svaki primjerak Djelatnik unosi stavku evidencije o posudbi.	5. Prikaz liste posuđenih medija po datumima te iznos naknade i zakasnine.
6. Djelatnik informira člana o ukupnom iznosu za plaćanje.	
7. Član plaća gotovinom ili kreditnom karticom.	
8. Djelatnik evidentira plaćanje.	9. Za plaćanje karticom obavlja se autorizacija. 10. Ispis liste posudbe i računa.
11. Djelatnik izdaje račun i listu posudbe.	

Alternativni tok

- 7a. Član nema dovoljno gotovine. Zahtijeva se plaćanje karticom, opoziva prethodna transakcija ili smanjuje broj primjeraka za posudbu.
- 7b. Postoji neplaćena zakasnina. Član ju mora platiti prije posudbe ili se transakcija prekida.
9. Autorizacija kartice ne uspijeva (usluga nedostupna ili blokirana kartica). Plaćanje gotovinom.



Modeliranje funkcija slučajevima korištenja

- UC opisuju funkcije sustava iz perspektive korisnika
 - "Što sustav treba napraviti?"
- UC opisuje samo jednu funkciju interakcije korisnika i sustava
- UC može uključivati različite scenarije
 - Npr. scenariji za isti UC: pretraživanje knjiga po autoru, po naslovu, ...
- Svaki UC smije biti povezan s jednom i samo jednom ulogom
 - Npr. liječnička "sestra" dogovara preglede, ispisuje uputnice i recepte, naplaćuje participaciju ...
- Različiti fizički korisnici mogu biti u istoj ulozi
 - stoga se slučajevi korištenja povezuju s ulogama a ne korisnicima

Tipovi slučajeva korištenja

□ Prema svrsi, namjeni:

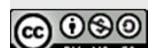
- pregledni (overview) – usaglašavanje zahtjeva s korisnikom
 - na početku prikupljanja zahtjeva, sadrže samo osnovne informacije
 - identifikator, primarna uloga, tip, kratki opis
- detaljni (essential) – sve informacije potrebne za razumijevanje

□ Prema količini informacija koje sadrži:

- osnovni (essential) – analitički, implementacijski nezavisni
- stvarni (real) – opis specifičnih koraka
 - dovoljno detaljan da odredi ponašanje sustava kada bude ugrađen

Primjer opisa slučaja korištenja 1/3

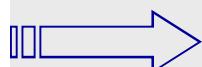
Naziv slučaja korištenja: Zakazati pregled	ID: 2	Prioritet: Visok
Glavni sudionik: Pacijent	Tip slučaja korištenja: Detaljni, osnovni	
Sudionici i interesenti: Pacijent – želi zakazati, promijeniti ili otkazati pregled <u>Liječnik</u> – želi obraditi pacijenta u prihvativom vremenu		
Kratki opis: slučaj korištenja opisuje kako zakazati, promijeniti ili otkazati pregled.		
Okidač: Pacijent pozivom zakazuje novi pregled te mijenja ili otkazuje postojeći pregled. Tip: Vanjski		
Veze: Asocijacija (association): Pacijent Uključivanje (include): Izrada računa Proširenje (extend): Unos novog pacijenta Generalizacija (generalization):		
Tok događaja: 1. Pacijent kontaktira ordinaciju radi progleda		



Elementi opisa slučaja korištenja

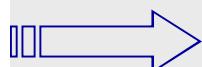
■ Pregled (overview) - osnovni podaci o slučaju korištenja

- naziv, tip, sudionici, kratak opis, prioritet
- glavni, primarni sudionik – pokreće scenarij
- okidač – događaj pokretanja (vanjski, unutarnji, vremenski ...)



■ Veze - odnos s drugim slučajevima korištenja

- Asocijacija (association) – veza između slučaja korištenja i sudionika
- Proširenje (<<extend>>) – proširenje funkcionalnosti uz dodatna pravila
 - npr. *Unos novog pacijenta*, izvodi se uvjetno, nije dio normalnog toka
- Uključivanje (<<include>>) – uključivanje drugog UC, prije <<uses>>
 - omogućuje funkcionalnu dekompoziciju (slično DTP)
 - npr. *Izrada računa* smatra se dovoljno složenom da bude samostalan UC
- Generalizacija (generalization) – poopćenje i nasljeđivanje UC
 - npr. *Zakazivanje pregleda starom pacijentu* i *Zakazivanje pregleda novom pacijentu* imaju generalizirano ponašanje u *Zakazivanje pregleda*



Primjer opisa slučaja korištenja 2/3

Uključivanje (include): Izrada računa

Proširenje (extend): Unos novog pacijenta

Generalizacija (generalization):

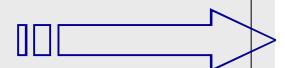
Tok događaja:

1. Pacijent kontaktira ordinaciju radi pregleda.
2. Pacijent službeniku daje svoje podatke.
3. Službenik potvrđuje da pacijent postoji u bazi pacijenata.
4. Službenik izvršava slučaj korištenja "Izrada računa".
5. Službenik pita pacijenta želi li zakazati novi pregled, promijeniti ili otkazati postojeći pregled.
 - Ako pacijent želi zakazati novi pregled,
izvodi se podtok T-1: novi pregled.
 - Ako pacijent želi otkazati postojeći pregled,
izvodi se podtok T-2: otkaži pregled.
 - Ako pacijent želi promijeniti postojeći pregled,
izvodi se podtok T-3: promijeni pregled.
6. Službenik obavještava pacijenta o rezultatu transakcije.

Podtokovi:

T-1: Novi pregled

1. Službenik pita pacijenta za moguće termine pregleda.
2. Službenik uspoređuje željene termine pacijenta i dostupne termine te zakazuje novi pregled.



Elementi opisa slučaja korištenja (nastavak)

- Tok događaja (flow of events)
 - Opis pojedinih koraka unutar poslovnog procesa
 - normalni tok – standardni, uobičajeni koraci
 - podtokovi – zasebni slučajevi korištenja
 - definirani da bi se pojednostavio normalan tok
 - slijede iz grananja dijagrama aktivnosti
 - mogu se oblikovati uključivanjem u normalan tok (<<include>>)
 - alternativni/izuzetni tokovi – mogu se dogoditi, ali nije uobičajeno
 - npr. potreba za unosom novog pacijenta prije nego mu se zakaže pregled

□ Navedeni su minimalni elementi, "zaboravljeni je":

- procjena vremena potrebnog za realizaciju
- specifični tokovi podataka
- atributi, ograničenja ili operacije u vezi sa slučajem korištenja
- prepostavke (pre-conditions) i posljedice (post-conditions)
- ...

Primjer opisa slučaja korištenja 3/3



□ Primjer: Dizajn\UCprimjer

6. Službenik obavještava pacijenta o rezultatu transakcije.

Podtokovi:

T-1: Novi pregled

1. Službenik pita pacijenta za moguće termine pregleda.
2. Službenik uspoređuje željene termine pacijenta i dostupne termine te zakazuje novi pregled.

T-2: Otkaži pregled

1. Službenik pita pacijenta za termin zakazanog pregleda.
2. Službenik pronađe zakazani pregled i otkazuje ga.

T-3: Promijeni pregled

1. Službenik izvodi podtok T-2: otkaži pregled.
2. Službenik izvodi podtok T-1: novi pregled.

Alternativni/izuzetni tokovi:

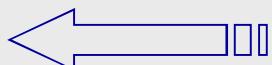
3a: Službenik izvodi slučaj korištenja "Unos novog pacijenta".

T-1, 2a1: Službenik u skladu s dostupnim terminima predlaže alternativne termine pregleda.

T-1, 2a2: Pacijent se odlučuje za jedan od predloženih termina ili odlučuje ne zakazati pregled.

Preporuke za pisanje opisa slučajeva 1/2

- Napisati svaki korak u obliku **Subjekt-Predikat-Objekt**
- Razjasniti tko je pokretač (subjekt), a tko primatelj (objekt) u koraku
- Utvrditi da je skup koraka slučaja smislen (vidi primjer UC 2/3)
 - glavni sudionik pokreće izvršenje slučaja slanjem zahtjeva (podataka) u sustav (npr. koraci 1 i 2)
 - sustav brine da je zahtjev valjan (korak 3)
 - sustav obrađuje zahtjev te moguće mijenja svoje unutarnje stanje (koraci 4 i 5)
 - sustav vraća rezultate obrade (korak 6)



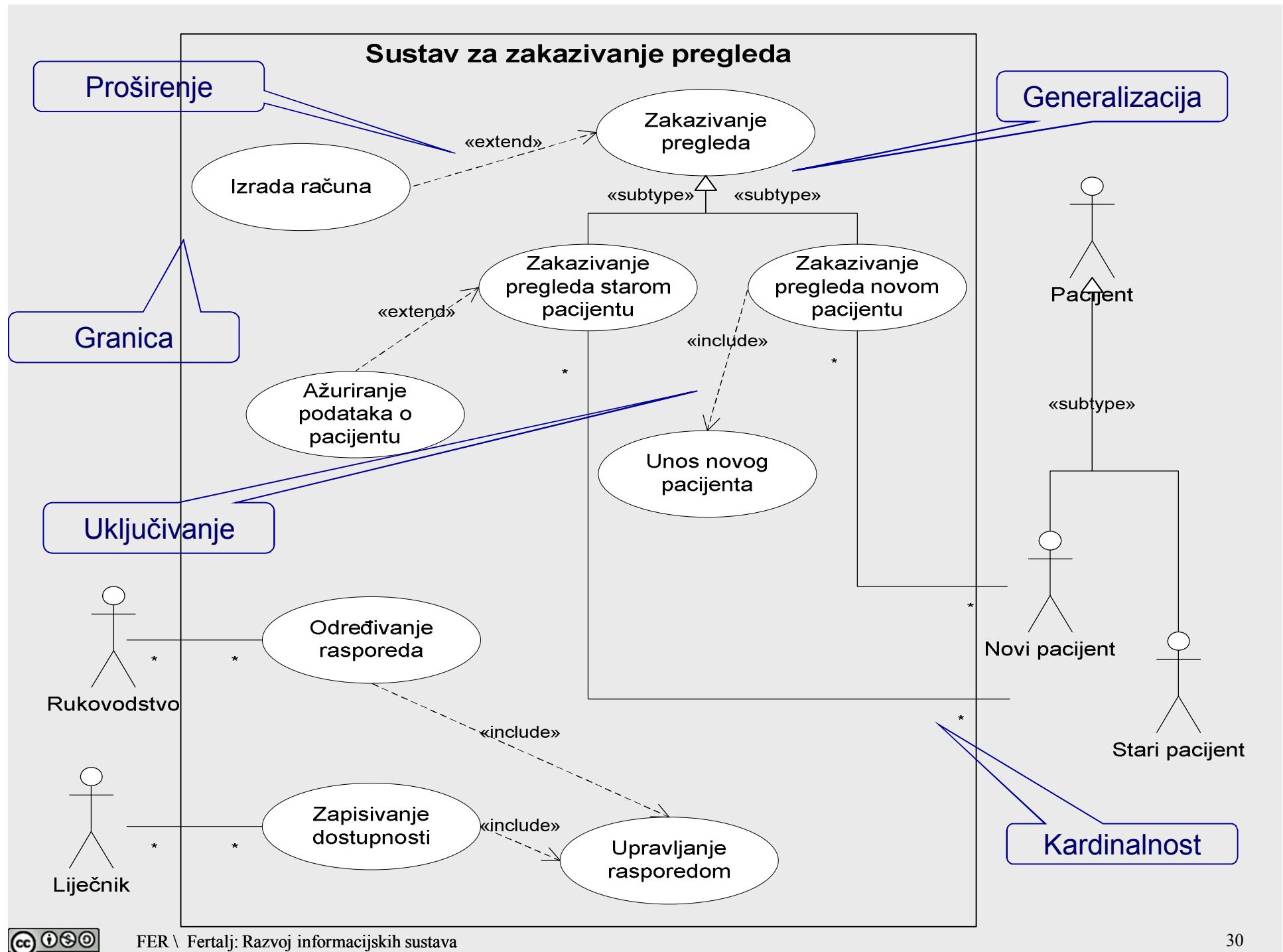
Smjernice za pisanje opisa slučajeva 2/2

□ Pravilo KISS (Keep It Simple, Stupid / Keep it Short and Simple)

- presložen ili predug slučaj treba podijeliti u više manjih
- ako normalni tok postane složen, treba ugraditi podtokove
- ali ne pretjerivati s dekompozicijom !

□ Za korake koje treba ponavljati – napisati uputu

- Primjer: Novi pregled
 - Službenik pita pacijenta za mogući termin pregleda
 - Službenik provjerava je li termin dostupan
 - **Ponavljati korake 1 i 2 sve dok se ne pronađe dostupan termin**
 - Službenik unosi datum i vrijeme sastanka
 -



Komentar dijagrama i daljnji koraci

□ Granica sustava (subject boundary)

- razdvaja vanjske i unutarnje dijelove
- primijetiti da na prethodnoj slici nedostaje *Službenik*, jer je dio sustava (unutarnji sudionik)

□ Mogući daljnji koraci

- Rafiniranje veličine projekta UC točkama - slično funkcionskim točkama
- Modeliranje struktura - u nastavku

□ Primjeri: Dizajn\UC*

Modeliranje struktura

Structural modeling



Modeliranje struktura

□ Model strukture (structural model)

- formalna prezentacija objekata kreiranih i korištenih u poslovnom sustavu
- opisuje strukturu podataka koji prate poslovni proces

□ Modeliranje strukture

- modelira se iterativno od konceptualne do detaljne razine
- u fazi analize bez ulazeњa u detalje pohrane podataka

□ Primjenjive tehnike

- CRC kartice
- dijagrami razreda
- dijagrami objekata

CRC kartice

Razred-Odgovornost-Suradnja (*Class-Responsibility-Collaboration*)

CRC kartica

- modelira odgovornosti jednog razreda i njegovu suradnju s drugim razredima
- može se koristiti umjesto dijagrama suradnje
- preporuča se da sadrži najviše tri odgovornosti visoke razine
 - u protivnom – razmisliti o razdvajanju razreda u nekoliko manjih

Odgovornosti

- odgovornost znanja (podaci)
- odgovornost radnje (operacije, postupci)

Suradnja

- klijent (client) – instanca koja šalje zahtjev drugoj instanci za izvođenjem
- poslužitelj (server) – instanca koja prima zahtjev
- ugovor (contract) – formalizacija interakcije između klijenta i poslužitelja
 - npr. pacijent dogovara pregled s liječnikom
 - oba sudionika imaju obvezu dolaska na pregled
 - ukoliko se pacijent ne pojavi, otpada plaćanje računa

Primjer CRC kartice

Prednja stranica:

Razred: Pacijent	ID: 3	Tip: Konkretni, Domenski
Opis: Pojedinac koji treba ili prima medicinsku pomoć.		Slučajevi korištenja: 2
Odgovornosti Zakazati pregled _____ Odrediti zadnji posjet _____ Promijeniti status _____ Dohvatiti povijest bolesti _____ _____ _____		Suradnici Pregled _____ _____ Povijest bolesti _____ _____

Stražnja stranica:

Atributi: Iznos (double) _____ Nositelj osiguranja (text) _____ _____ _____	_____ _____ _____
--	-------------------------

Veze: Generalizacija (a-kind-of): Osoba _____ Agregacija (has-parts): Povijest bolesti _____ Ostale veze : Pregled _____



Koraci modeliranja strukture

□ Izrada CRC kartice analizom opisa UC

- određivanje razreda, atributa, operacija i asocijacija

□ Analiza skupne liste objekata (common object list approach)

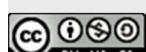
- opipljive stvari (tangible things) – npr. oprema, materijal, roba
- incidenti (incidents) – događaji, npr. sastanci, letovi, akcidenti
- uloge (roles) – uloge osoba, npr. liječnici, pacijenti, administratori
- interakcija (interaction) – transakcije, npr. akcija prodaje

□ Glumljenje CRC karticama (role-play the CRC cards)

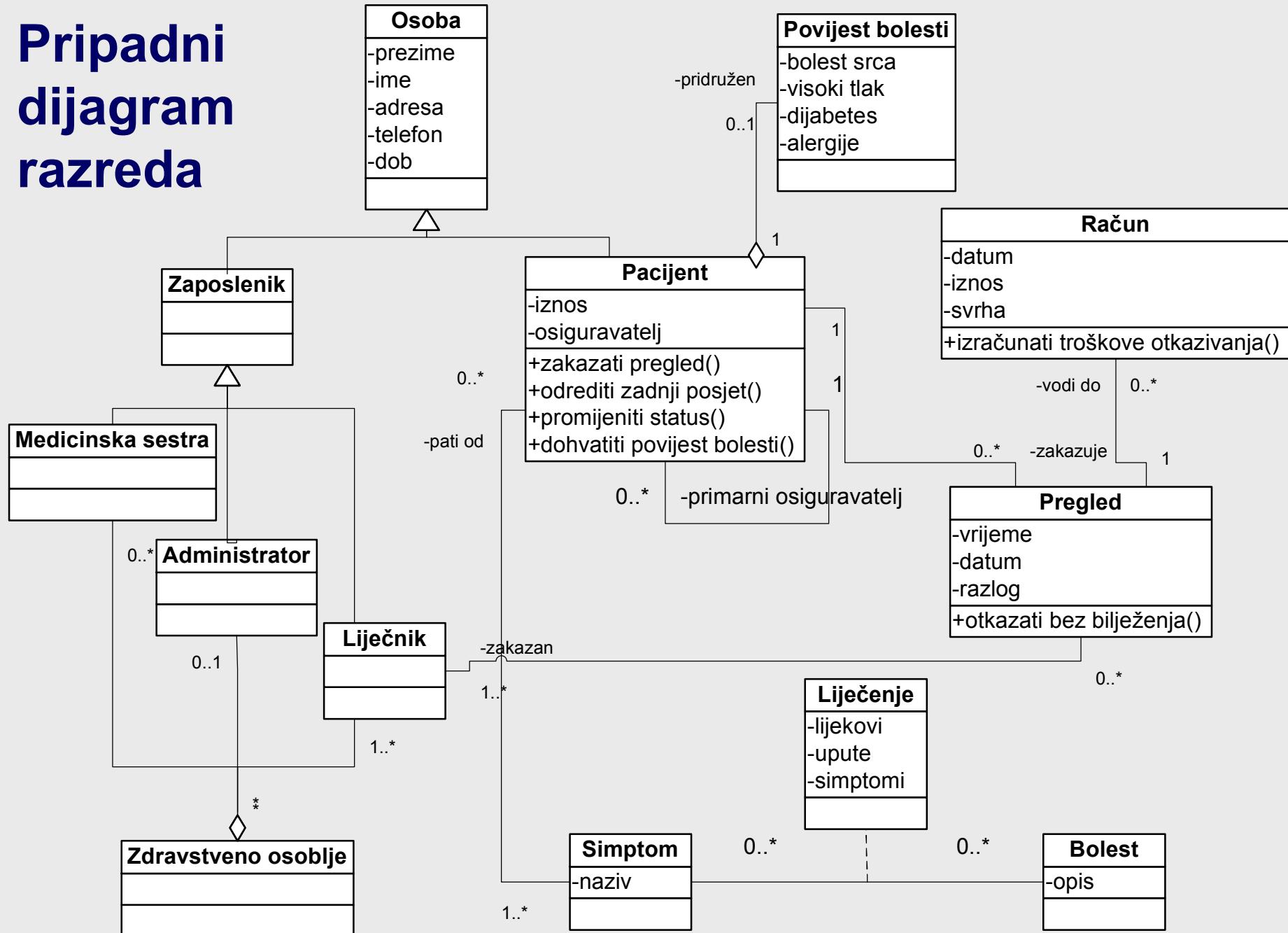
- testiranje UC – antropomorfizam – igrač uzme karticu i glumi razred
 - Tko si ili što si ? Što znaš ? Što možeš ?

□ Primjena obrazaca (incorporate patterns)

- "kako se to radi"
 - analitički obrasci, npr. *Partner-Dokument-Stavke-Artikl*
- "kako realizirati", "kako ponovno iskoristiti"
 - **oblikovni obrasci, aplikacijski okviri, knjižnice razreda**



Pripadni dijagram razreda



Dizajn vođen odgovornostima



Aktivnosti objektnog dizajna općenito

□ Dodatna specifikacija razreda

- potrebni atributi i postupci
- vidljivost atributa i postupaka
- *signatura* postupaka - naziv, argumenti, tip povratne vrijednosti
- ograničenja – npr. raspon vrijednosti, preduvjeti, posljedice, ...
- rukovanje iznimkama

□ Restrukturiranje dizajna

- Faktoriranje
 - postupak izdvajanja novog modula (razreda, metode) iz postojećih
- Normalizacija
 - ustanovljavanje razreda koji nedostaju
 - pretvorba asocijacijskih razreda i agregacija u atribute

□ Optimiranje dizajna (**refaktoriranje - u fazi kodiranja**)

- skraćenje pristupnih putova između objekata
- smanjenje broja metodom poslanih poruka
- redoslijed izvršenja naredbi (npr. redoslijed pretraživanja)
- uvođenje izvedenih atributa umjesto ponavljanog izračuna



Dizajn vođen odgovornostima

□ Responsibility-Driven Design

- postupak oblikovanja softvera koji naglašava modeliranje uloga i odgovornosti objekata te suradnje objekata

□ Načela dizajna vođenog odgovornostima

- Maksimiziranje apstrakcije
 - početno se skriva (zanemaruje) razlika između podataka i ponašanja
 - odgovornosti objekta su "znanje", "činjenje", "odlučivanje"
- Distribucija ponašanja
 - poboljšavanje odabrane upravljačke arhitekture
 - pametni objekti – inteligentno ponašanje, a ne samo pohrana podataka
- Očuvanje elastičnosti, fleksibilnosti
 - projektiranje objekata (razreda) koje se može brzo i lako mijenjati

Pristup oblikovanju vođenom odgovornostima

Postupak oblikovanja vođenog odgovornostima

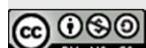
- Analiza slučajeva korištenja ili korisničkih priča
- Izrada liste potencijalnih poslovnih objekata i njihovih veza
- Određivanje (jedinstvene) odgovornosti objekata
- Postupnim pročišćavanjem nastane dizajn razreda
- Temeljem istih slučajeva korištenja projektira se relacijski model
- (kodiranje, refaktoriranje)

Analiza slučajeva korištenja ili korisničkih priča

- **Objekti postoje da bi podržali specifični UC ili korisnički scenarij**
 - posljedično, objekti sadrže poslovnu logiku i podatke koje korisnik treba da bi ispunio neki zadatak – slučaj korištenja
 - ovaj zahtjev može umanjiti ponovnu iskoristivost (reusability) !
- **Svaki zadatak je slučaj korištenja – dekompozicija**
 - veće zadatke treba razdijeliti u manje od kojih je svaki zaseban UC
 - neki će slučajevi stoga ovisiti o drugima (*uses*, *extends*)
- **Krajnji cilj**
 - svaki slučaj korištenja niske razine (elementarni slučaj) opisuje jedan samostalan zadatak, tj. scenarij koji može biti jasno zapisan

Objekti s odgovornostima

- Nakon što su određeni elementarni slučajevi ustanovljavaju se objekti potrebni za implementaciju slučaja korištenja
- Pažnja se usmjerava na to što objekt radi a ne na podatke !
- Koji je posao ili uloga objekta unutar slučaja korištenja ?
 - preuzimanje (unos) podataka, provjera podataka, izračun, ...
- Dizajn jednostrukе odgovornosti
 - Pojedini objekt treba imati jednu jednostruku odgovornost unutar UC
 - Objekt može imati više ponašanja (svojstava, postupaka), ali takvih da objekt podupire jedinstvenu odgovornost



Zavisnost objekata i normalizacija ponašanja

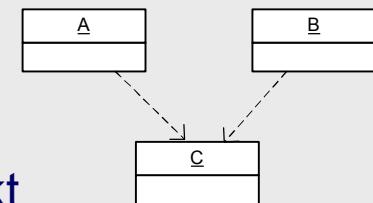
□ Npr. Objekt A i Objekt B podupiru svaki svoj UC

- A treba neko ponašanje od B (coupling)
- svaki od objekata podržava vlastiti UC
- rizik da će se B promijeniti radi promjene zahtjeva na UC koji podržava



□ Normalizacija ponašanja

- postupak postizanja ponovne iskoristivosti i kontrole kopčanja
- izdvajanje koda koji se želi ponovno iskoristiti u zaseban objekt
- npr, ponašanje iz B izdvaja se u C, te A i B nastavljaju nezavisno kolaborirati



□ Factoring – tehnika OO restrukturiranja

- postupak izdvajanja novog modula (razreda, metode) iz postojećih
- npr. da u primjeru ordinacije nije bio identificiran Zaposlenik, izvjesno bismo ga izlučili iz Sestra, Administrator ili Doktor

□ Ponovnu iskoristivost forsirati samo ako se može izbjegći kopčanje !

Primjer dizajna vođenog odgovornostima

Modeliranje analizom više slučajeva korištenja



Slučajevi korištenja sustava evidencije projekata

□ Dodavanje projekta (provodi upravitelj projekta)

- podaci o projektu sadrže naziv, opis, datum početka i datum završetka
- projekt ima jedinstveni broj, ali se korisnik ne treba njime posebno baviti
- projekt se identificira nazivom, pa je to zahtijevana vrijednost
- ostale vrijednosti su opcionalne
- moguće je dodati projekte koji još nisu započeti
- započeti projekt ne mora imati datum završetka
- datum početka mora biti manji od datuma završetka
- svaki projekt ima listu sudionika (pogledati Pridjeljivanje sudionika)
 - pojednostavnjena evidencija - općenitije dodavali bi se i drugi resursi

□ Ažuriranje projekta (provodi upravitelj projekta)

- moguće je ažurirati postojeći projekt odabirom s liste projekata
- može se ažurirati datum početka, datum završetka i opis
- moguće je ažurirati listu sudionika (pogledati Pridjeljivanje sudionika)

□ Brisanje projekta (upravitelj projekta ili administrator)

- brisanje se obavlja odabirom projekta s liste projekata

Evidencija sudionika

Dodavanje osoba (upravitelj projekta ili rukovoditelj)

- pohranjuju se ime i identifikator zaposlenika, uz obvezan unos
- treba omogućiti evidenciju projekata osobe (zaduženja) pri dodavanju osobe (Pridjeljivanje sudionika)

Ažuriranje osobe (upravitelj projekta ili rukovoditelj)

- dozvoliti izmjenu imena

Brisanje osobe (upravitelj projekta ili rukovoditelj)

- zaposlenik može prekinuti posao ili otići u drugi dio organizacije
- brisanje se obavlja odabirom osobe s liste osoba

Pridjeljivanje sudionika (upravitelj projekta)

- proces se pojavljuje u više drugih procesa, pa je izdvojen u zaseban slučaj
- osoba se dodjeljuje projektu
- evidentira se uloga osobe, odabirom s liste uloga, te datum zaduženja
- treba dozvoliti promjenu uloge osobe
- osoba može sudjelovati u više projekata
- treba omogućiti brisanje zaduženja (pogledati Brisanje osobe)

Evidencija uloga (administrator)

- omogućiti održavanje uloga u odgovarajućem šifrarniku



Projektiranje objekata (Object Design)

- kombiniranjem dekompozicije ("traženjem imenica") i CRC karticama
- Početni dizajn - ustanovljavanje potencijalnih entiteta
 - neki od nabrojanih entiteta nisu objekti nego npr. uloge korisnika sustava (Upravitelj projekta, Administrator, Rukovoditelj), a neki su atributi

Upravitelj
Naziv projekta
Administrator
Osoba
Rukovoditelj
Lista uloga
Lista osoba

Projekt
Datum početka
Lista projekata
Naziv zaposlenika
Lista projekata osobe
Zaduženje
Lista sudionika projekta

Broj projekta
Datum završetka
Zaposlenik
Id. zaposlenika
Uloga
Datum zaduženja

- Lista entiteta se reducira te se dodaju odgovornosti i suradnici
- Napomena: dalje se nećemo baviti aspektima sigurnosti i zaštite

Razredi potencijalnih objekata

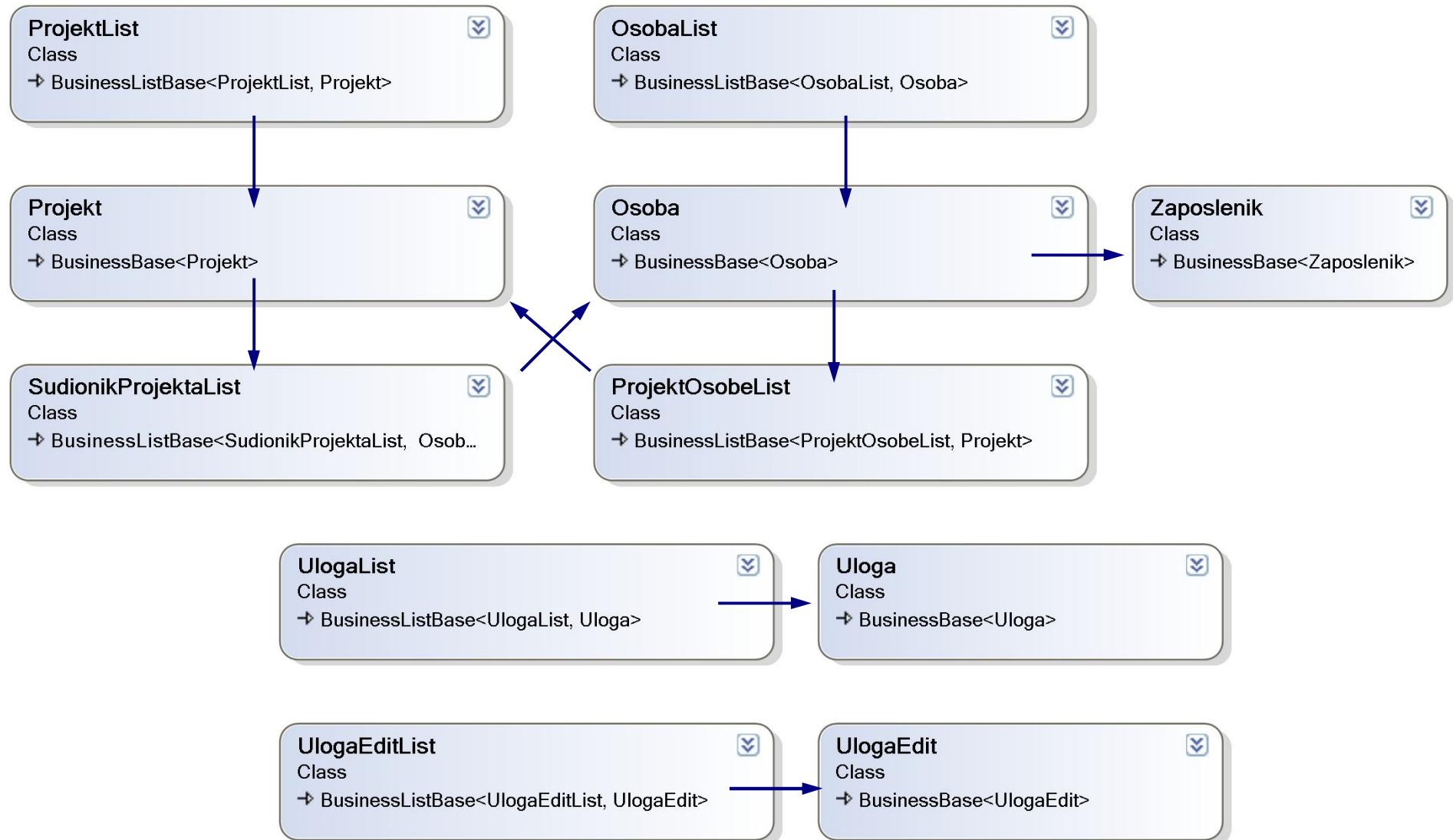
□ CRC kartica visoke razine za potencijalne objekte:

- odgovornost treba biti izražena kratko i precizno
- ako nije, to može značiti da je riječ o složenom objektu koji treba rastaviti

Potencijalni razred	Odgovornost	Suradnici
Projekt	dodaje i ažurira ispravan projekt	SudionikProjektaList
Osoba	dodaje i ažurira ispravnu osobu	ProjektOsobeList, Zaposlenik
Zaposlenik	dodaje i ažurira ispravnog zaposlenika	
ProjektList	dobavlja read-only listu projekata	Projekt
OsobaList	dobavlja read-only listu osoba	Osoba
SudionikProjektaList	održava listu osoba dodijeljenih projektu	Osoba, UlogaList
ProjektOsobeList	održava listu projekata na kojima je osoba	Projekt, UlogaList
UlogaList	dobavlja read-only listu uloga	Uloga
Uloga	read-only vrsta uloge	
UlogaEditList	održava listu uloga	UlogaEdit
UlogaEdit	dodaje i ažurira ispravnu ulogu	



Dijagram potencijalnih razreda



Revizija dizajna

□ Osoba i Zaposlenik su potencijalni duplikati

- nema naznaka da osoba dodaje bilo što zaposleniku
- Osoba i Zaposlenik mogu se udružiti

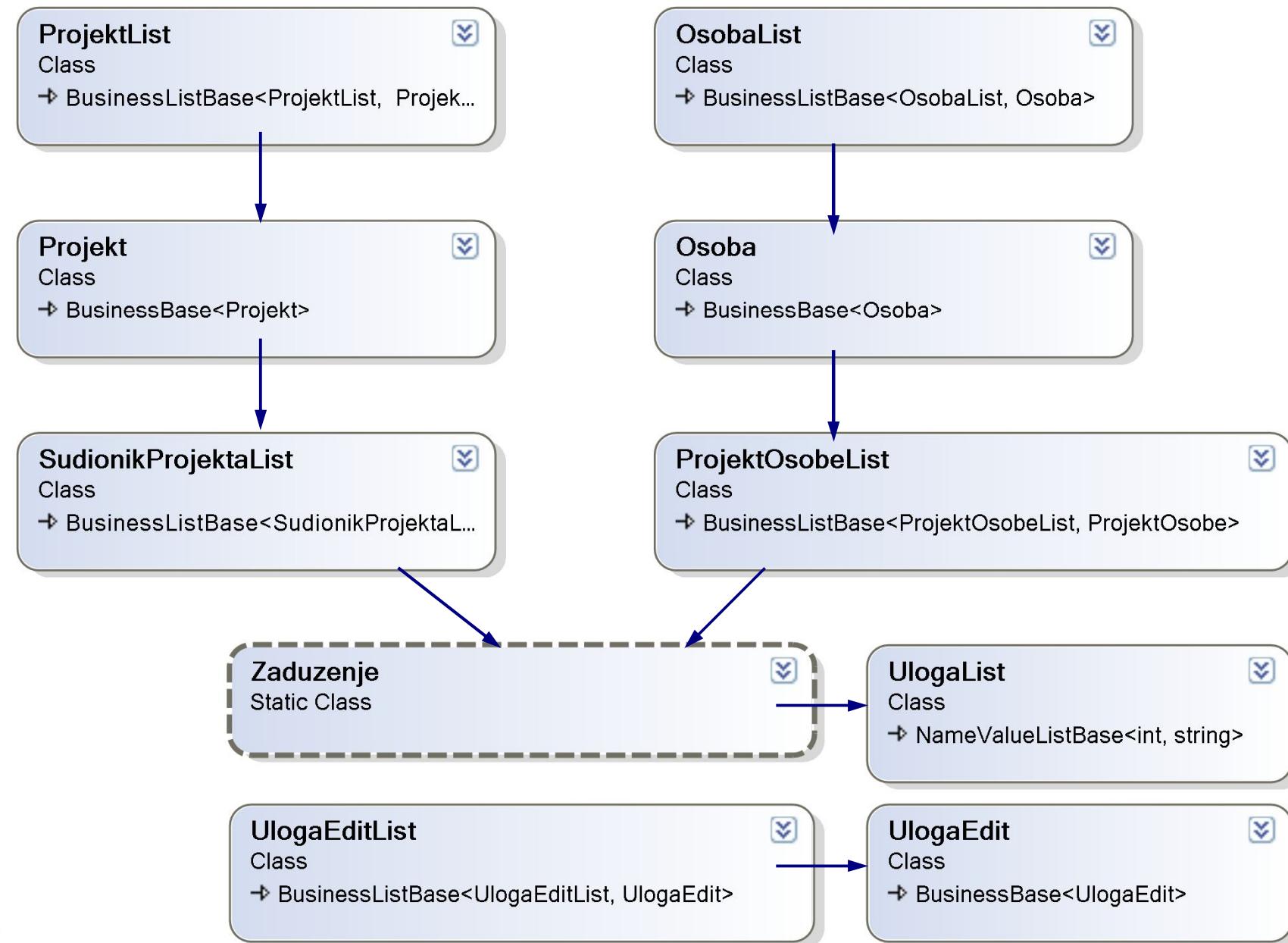
□ UlogaList i Uloga

- UlogaList je jednostavna lista oblika šifra/naziv (tzv. name/value list)
- Uloga je jednostavan šifrarnički zapis (tzv. name/value placeholder)
- može se kasnije pojednostavniti odgovarajućim razredom (npr. Csla.NameValueListBase)

□ Projekt, SudionikProjektaList, Osoba, ProjektOsobeList

- komplikirane, cirkularne veze – potencijalne beskonačne petlje
- nedostaje podatak o ulozi osobe na projektu
 - osoba treba sudjelovati na više projekata ili u više uloga
 - uloga za sada uloga nepovezana
- rješenje: dodavanje novog razreda Zaduzenje

Revidirani dijagram



Problem zaduženja

□ Razred **Zaduzenje** koristi se u dva konteksta

- s liste sudionika projekata i s liste projekata osobe
- zajedničko dijete različitih kolekcija komplicira ugradnju i testiranje

□ Odgovornost razreda **Zaduzenje** ? - nejednoznačno

- pridruživanje osobe projektu
- pridruživanje projekta osobi

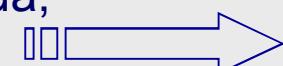
□ Problem s podacima

- Projekt koristi SudionikProjektaList da dohvati listu resursa projekta
 - podrazumijeva da Zaduzenje sadrži informaciju o osobi
- Osoba koristi ProjektOsobeList da bi dohvatio listu projekata osobe
 - podrazumijeva da Zaduzenje sadrži informaciju o projektu

□ Očigledno postoji konflikt ponašanja ali i konflikt podataka

□ Moguća rješenja

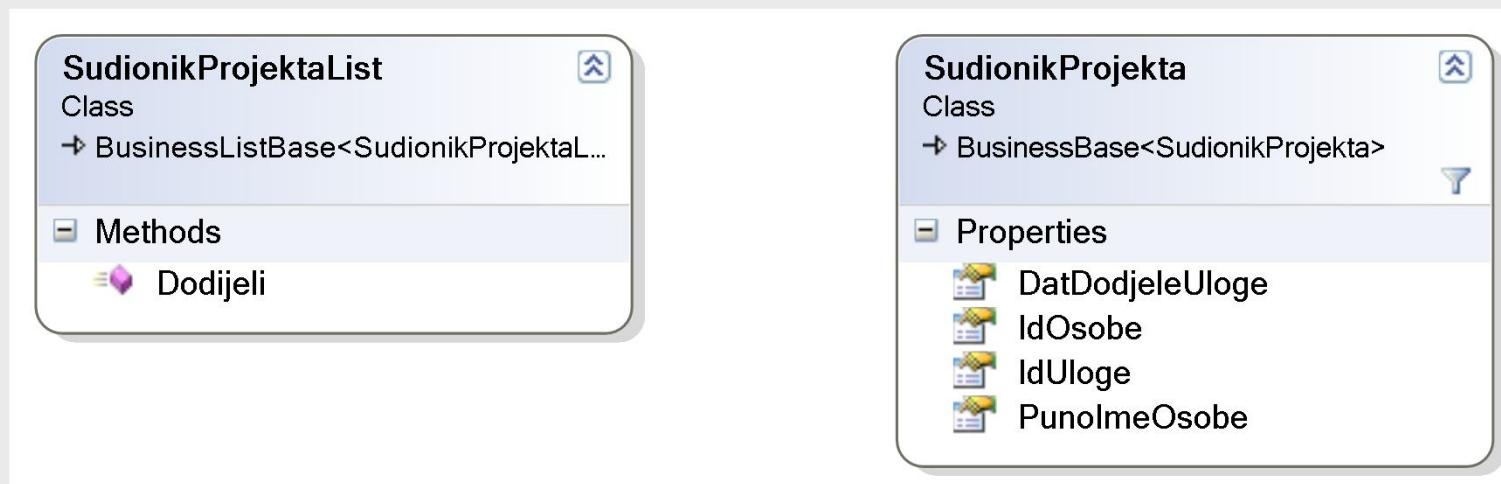
- udruživanje SudionikProjektaList i ProjektOsobeList u npr. ZaduzenjeList
 - problem bi se izgledno preselio razinu više (pa se nećemo time baviti)
- alternativa: dijeljenje razreda Zaduzenje u dva zasebna razreda,
 - SudionikProjekta i ProjektOsobe



Alternativa (1.dio) : Dodjela osobe projektu

- Temeljem UC, korisnik odabere projekt i dodaje osobu projektu
 - podrazumijeva da projekt ima kolekciju pridijeljenih osoba

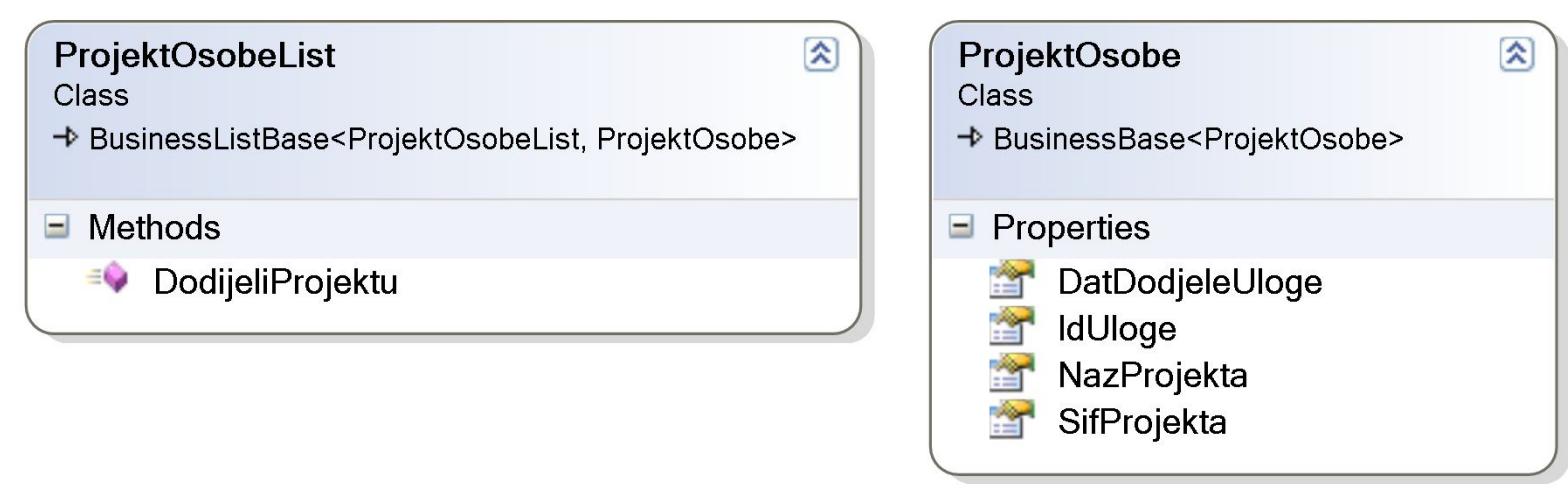
- Koja ponašanja (odgovornosti) očekivati od SudionikProjektaList ?
 - vratiti listu osoba pridruženih projektu
 - pridruživanje nove osobe projektu, npr. postupkom Dodijeli (IdOsobe)
 - davanje podataka za pregled i uređivanje



Alternativa (2.dio) : Dodjela projekta osobi

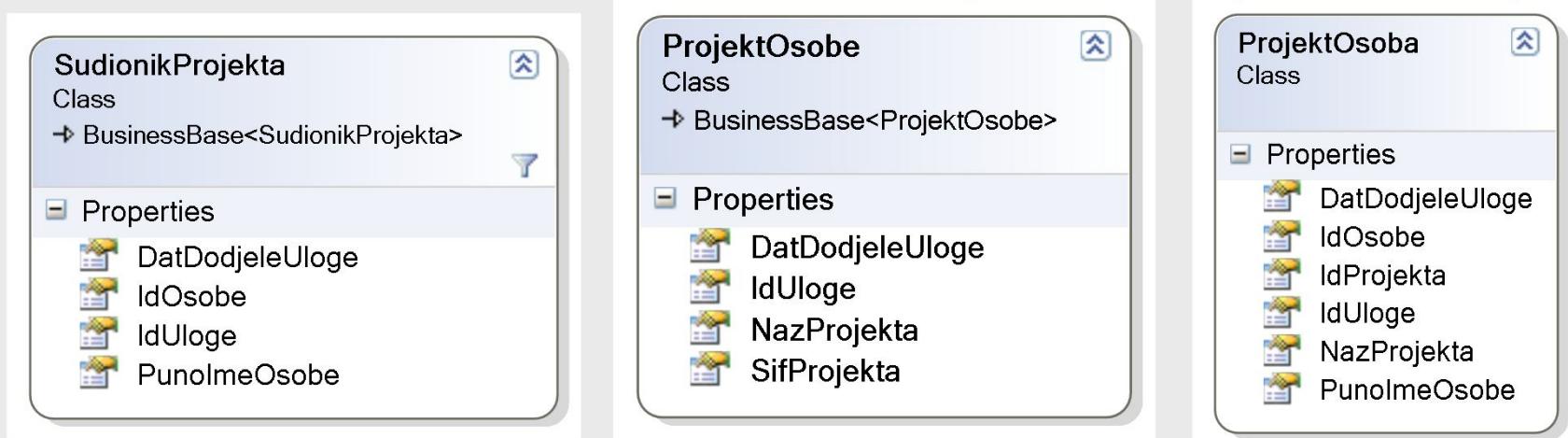
- Analogno prethodnoj dodjeli, dodaje se projekt odabranoj osobi
 - podrazumijeva da osoba ima kolekciju pridijeljenih projekata

- Koja ponašanja (odgovornosti) očekivati od `ProjektOsobeList` ?
 - vratiti listu projekata pridruženih osobi
 - pridruživanje projekta osobi, npr. postupkom `DodjeliProjektu()`
 - davanje podataka za pregled i uređivanje



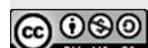
Što dalje: Udruživanje sličnih razreda ?

- SudionikProjekta i ProjektOsobe sadrže slične podatke, ali im je ponašanje različito
- Smislenost udruživanja u zajednički razred ProjektOsoba ?
 - Naziv projekta nema smisla kada se objekt referencira iz konteksta projekta
 - slično vrijedi za neka druga svojstva, ovisno o kontekstu
- Rješenje?
 - poslovna logika koja će baciti iznimku ukoliko se objektu ProjektOsoba, ovisno o kontekstu, pristupa na pogrešan način
 - ostaje, međutim, problem da ProjektOsoba ima različite odgovornosti
 - *Spaghetti code – povjesno dokazano loša implementacija (dakle ne tako)*



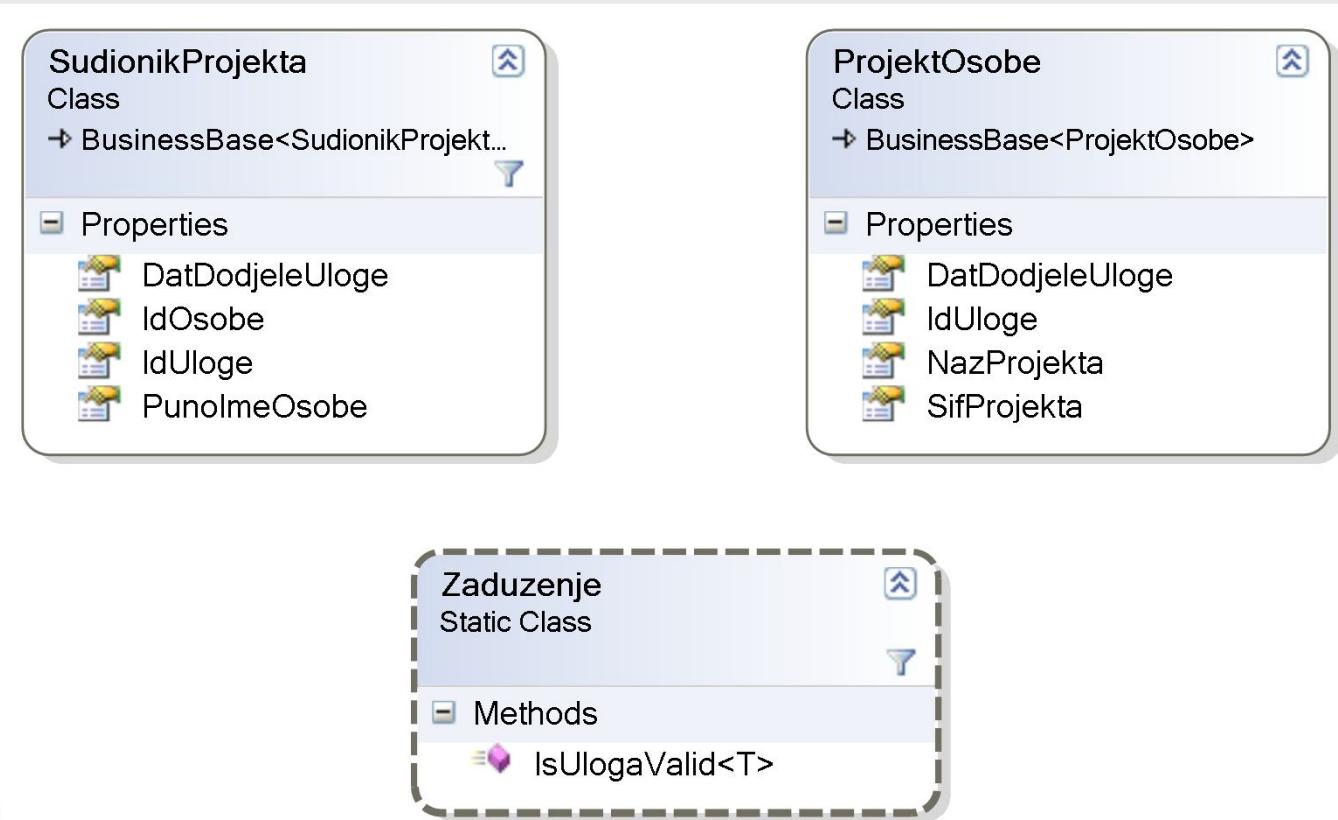
Vladanje zajedničkim ponašanjem i informacijama

- Kako koristiti iste podatke bez umnažanja poslovne logike ?
 - normalizacija ponašanja
- Osnovna ideja relacijskih baza podataka je smanjenje zalihosti
 - svaki elementarni podatak smije postojati samo jednom
- Cilj objektnog dizajna je osiguranje **jedinstvenog ponašanja**
 - svako ponašanje mora postojati samo jednom u modelu, ali smije biti korišteno od strane različitih objekata



Vladanje zajedničkim informacijama

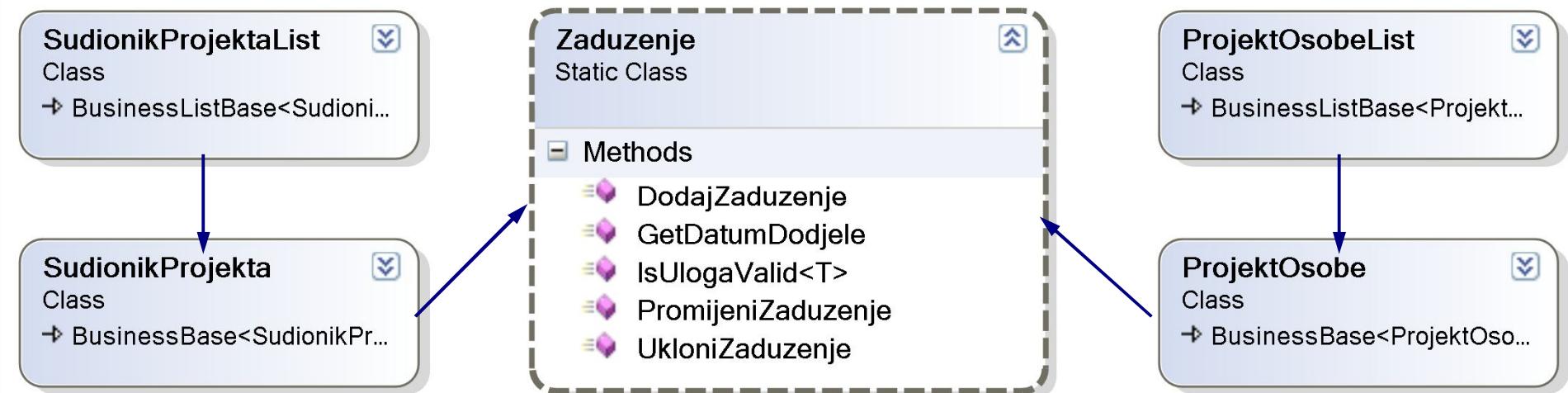
- **DatDodjeleUloge i IdUloge ne treba smještati u zajednički objekt**
 - jer ne razmišljamo relacijski, ne normaliziramo podatke
- **Pitanje je da li ti atributi imaju zajedničko ponašanje (poslovno pravilo ili poslovnu logiku) koje se može staviti u zajednički objekt**
 - Primjer: validacija dodijeljene uloge



Vladanje zajedničkim ponašanjima

□ Odgovornost razreda `Zaduzenje` je upravljanje vezom između projekta i osobe kao resursa projekta

- `Zaduzenje` treba podržati pridruživanja osobe projektu i obrnuto
- Posao dodjele vodit će `SudionikProjekta` i `ProjektOsobe`
- kolekcijski razredi samo će dodavati i uklanjati ove objekte
- naravno, koristiti i druge postupke razreda `Zaduzenje`



Poboljšanje performansi

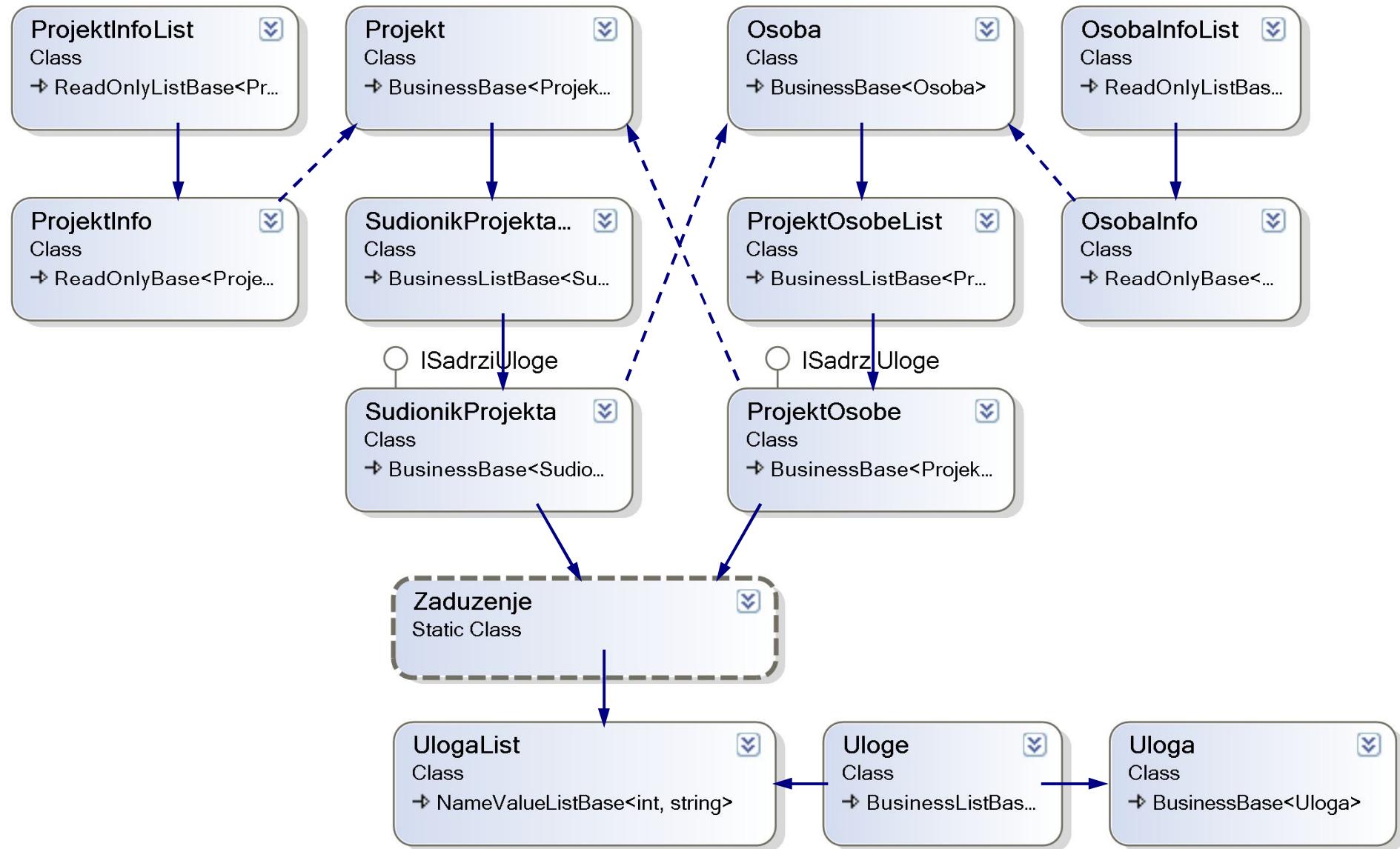
- SudionikProjektaList i ProjektOsobeList dohvaćaju kolekcije poslovnih objekata Projekt i Osoba samo da bi ih prikazali u listi
 - učitavanje objekta Projekt ili Osoba povlači učitavanje lista zaduženja ... (kaskadno)
- Umjesto učitavanja hijerarhija - formirati **read-only** liste
 - Uvode se potrebni **read-only** objekti za prikaz osoba i projekata ProjektInfoList, ProjektInfo, ...



Ponovna procjena dizajna

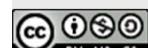
- **Završni korak - usporedba konačnog modela s izvornim UC**
- **U nastavku je prikazan dijagram koji**
 - ilustrira suradnju objekata (pune poveznice)
 - pokazuje navigaciju između objekata (isprekidane poveznice)
 - npr. s `ProjektInfo` objekta moguća je navigacija prema objektu `Projekt`, postupkom `GetProjekt()`
 - primijetiti da je `ProjektList` zamijenjen s `ProjektInfoList`, a `OsobaList` s `OsobaInfoList`
- **Slijedi konačna lista objekata i njihovih odgovornosti**

Konačni objektni model evidencije projekata



Konačna lista objekata i odgovornosti

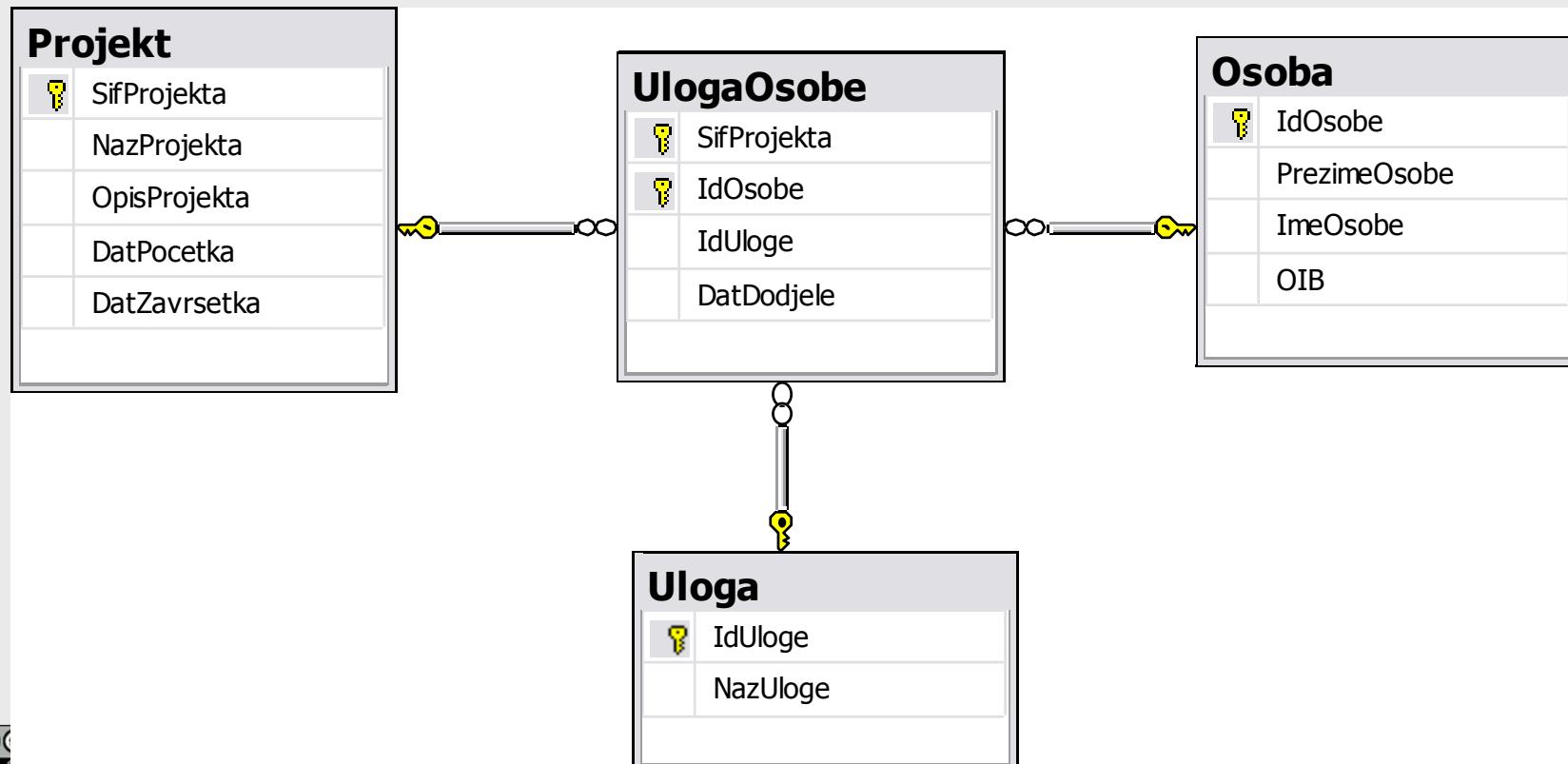
Razred	Odgovornost	Suradnici
Projekt	dodaje i ažurira ispravan projekt	SudionikProjektaList
SudionikProjektaList	održava listu osoba dodijeljenih projektu	SudionikProjekta
SudionikProjekta	upravlja pridjeljivanjem osobe projektu	Zaduzenje, Osoba
Osoba	dodaje i ažurira ispravnu osobu	ProjektOsobeList, Zaposlenik
ProjektOsobeList	održava listu projekata na kojima je osoba	ZaduzenjeOsobe
ZaduzenjeOsobe	upravlja pridjeljivanjem projekta osobi	Zaduzenje, Projekt
Zaduzenje	održava vezu projekta i osobe	UlogaList
ProjektInfoList	dobavlja read-only listu projekata	ProjektInfo
ProjektInfo	read-only informacija o projektu	Projekt
OsobaInfoList	dobavlja read-only listu osoba	OsobaInfo
OsobaInfo	read-only informacija o osobi	Osoba
UlogaList	dobavlja read-only listu uloga	
Uloge	održava listu uloga	Uloga, UlogaList
Uloga	dodaje i ažurira ispravnu ulogu	





Na kraju

- Relacijski dizajn i objektno-orientirani dizajn su dva različita procesa koji rezultiraju različitim modelima !
 - Usklađivanje ovih modela provodi se objektno-relacijskim mapiranjem
- Object Relational Mapping – ORM, OR/M
 - tehnika konverzije podataka između relacijske BP i OO jezika



Prilozi i reference

📁 Dodaci

📁 RIS06-dodatak

📁 Resursi

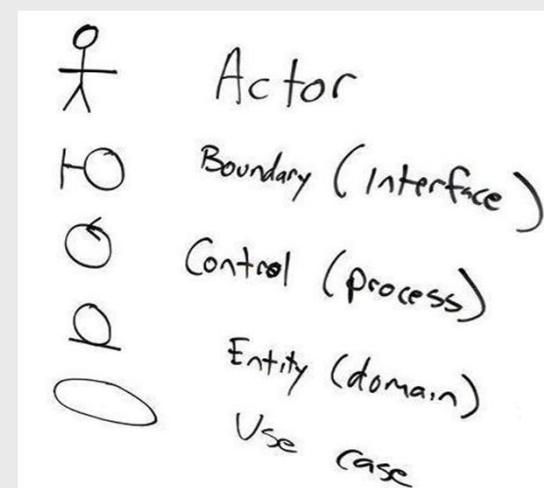
- Rockford Lhotka: Expert C# 2008 Business Objects, Apress, 2008

📁 Dizajn

- Rebecca Wirfs-Brock: A Brief Tour of Responsibility-Driven Design
 - http://www.wirfs-brock.com/PDFs/A_Brief-Tour-of-RDD.pdf

☐ Agilno modeliranje

- <http://www.agilemodeling.com>



UML dijagrami

□ Dijagram ponašanja

- Use Case Diagram – dijagram slučajeva korištenja

□ Dijagrami strukture, Strukturni dijagrami

- Class Diagram – dijagram razreda
- Object Diagram – dijagram objekata
- Component Diagram – dijagram komponenti
- Package Diagram – dijagram paketa
- Deployment Diagram – dijagram ugradnje

□ Dijagrami dinamike

- Sequence Diagram – slijedni dijagram
- Collaboration Diagram – dijagram kolaboracije, suradnje
 - Communication Diagram – dijagram komunikacije (UML 2.0)
- Statechart Diagram, State Diagram, State Machine Diagram – dijagram stanja
- Activity Diagram – dijagrami aktivnosti

□ Ostali

- Composite Structure Diagram
- Interaction Overview Diagram
- Timing Diagram



Projektiranje arhitekture

Aplikacija

- **Aplikacija – skup programskih komponenti koje čine logičku cjelinu**
 - npr. dvoslojni debeli klijenti, višeslojni pametni klijenti, web aplikacije
 - naspram toga SOA model – poslovni sustav sastavljen od aplikacija i servisa
 - korisnička aplikacija i servis kao nezavisne cjeline
 - servis je također aplikacija – XML umjesto GUI ili HTML sučelja
- **Logička arhitektura – razdvajanje tipova funkcionalnosti**
 - npr. sučelje, poslovni sloj, podatkovni sloj
 - sloj logičke arhitekture (layer) : n-layer architecture
- **Fizička arhitektura – fizička reprezentacija glavnih komponenti IS**
 - raspodjela logičkih komponenti na fizičke uređaje
 - sloj fizičke arhitekture (tier) : n-tier architecture

Veza logičke i fizičke arhitekture

- Zabluda – pretpostavka da se logička arh. preslikava u fizičku 1:1**
 - logička arhitektura odnosi se na (samo na) odvajanje funkcionalnosti (npr. korisničko sučelje, pristup podacima)
- Broj slojeva logičke \geq broj slojeva fizičke arhitekture !**
 - više logičkih slojeva može biti smješteno na isti fizički uređaj
- Prilikom kreiranja aplikacije važno je odabrati logičku arhitekturu koja će omogućiti kasniji odabir fizičke arhitekture**
 - sadrži slojeve koji organiziraju komponente u diskrete uloge
 - ima barem toliko logičkih slojeva koliko će imati odabrana fizička arhitektura
- Komunikacija slojeva – odvajanje nije uvijek jednako moguće**
 - npr. BL i DL – najčešće visokooptimirano
 - npr. UI i BL – teže, nepoželjno zbog povezivanja kontrola sučelja (binding)

Fizičke arhitekture i složenost aplikacija

- **Višeslojne fizičke arhitekture smanjuju složenost ako je zadovoljen neki od uvjeta**
 - aplikacija je velika ili složena
 - postoji više jednostavnih aplikacija koje su u kombinaciji velike ili složene
 - velika ili složena okolina (ugradnje, potpore)
- **Višeslojne fizičke arhitekture povećavaju složenost ako je istinito sve navedeno**
 - aplikacija je mala ili relativno jednostavna
 - aplikacija nije dio većeg sustava povezanih aplikacija
 - jednostavna okolina
- **Problem je što jednostavne aplikacije mogu rasti i usložnjavati se !**
→ **skalabilnost**

Karakteristike fizičke arhitekture (kriteriji uslojavanja)

□ Performanse - brzina odziva

- najveća kad su komponente na istom računalu

□ Skalabilnost – prilagodba kapaciteta s obzirom na opterećenje

- npr. opterećenje poslužitelja u dvoslojnoj arhitekturi s puno korisnika
 - smanjenje opterećenja odvajanjem poslovne logike na aplikacijski poslužitelj
- za većinu aplikacija (ipak) nije dovoljan razlog za prijelaz na tri sloja

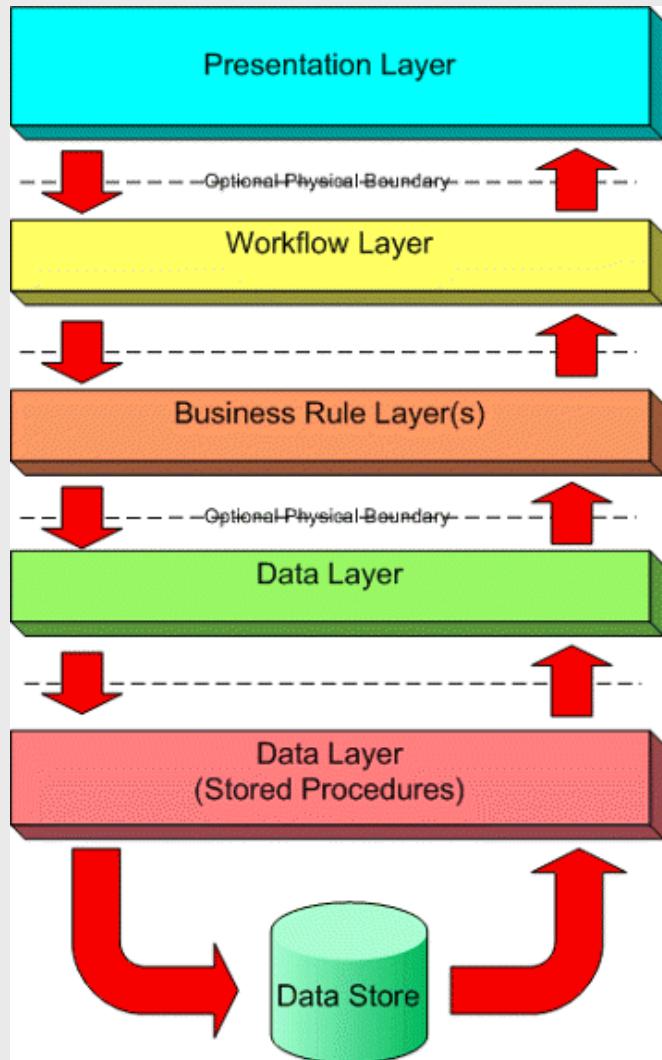
□ Sigurnost – glavni argument

- u dvoslojnoj arhitekturi npr. web poslužitelj mora znati akreditaciju (credentials) za pristup bazi podataka
- troslojna je sigurnija ali unutar iste lokalne mreže teže branjiva – poželjno odvajanje vatrozidom

□ Tolerancija pogrešaka

- bolja ako su fizički dijelovi zamjenjivi, npr. redundantnim hardverom

Petroslojna logička arhitektura



- Korisničko sučelje (UI)**
 - prezentacijski sloj (PL)
- Kontrola sučelja (interface control - IC)**
 - kontrola radnog toka (workflow)
- Poslovni sloj (business layer – BL)**
 - poslovna ograničenja, validacijska pravila
- Sloj za pristup podacima (data layer – DL)**
 - data access layer - DAL
- Sloj za upravljanje pohranom i upravljanje podacima (data storage and management layer – DSML)**
- Fizička pohrana (data storage)**

Slojevi petoslojne logičke arhitekture

□ Korisničko sučelje

- prikaz podataka, unos od strane korisnika
- npr. web preglednik, SOA XML poruka, Windows sučelje
- vrlo slično i blisko kontroli sučelja

□ Kontrola sučelja

- generiranje izlaza, interpretacija / obrada ulaza
- npr. web poslužitelj, WPF ili WF pozadinski kod (code behind)
- vođen događajima (event-driven)
- prihvat unosa korisnika i prosljeđivanje u BL na validaciju i obradu

□ Poslovni sloj

- poslovna pravila, validacija, obrada, autentifikacija, ...
 - "combination of validation edits, login verifications, database lookups, policies and algorithmic transformations that constitute an enterprise's way of doing business"
- mora biti odvojen od sučelja da se olakša održavanje i ponovna iskoristivost
- fizički bliže PL – klijentsko računalo, interaktivne aplikacije, npr. validacija
- fizički bliže DL – nefunkcionalni procesi, npr. obrada plaća, izračun zaliha

Slojevi petoslojne arhitekture (nastavak)

□ Podatkovni sloj

- surađuje sa slojem pohrane i rukovanja podacima
- dohvaća, umeće, ažurira, briše podatke ali ne upravlja i ne pohranjuje
- sučelje između BL i DSML
- fizički se odvaja uslijed istog razloga kao BL – performanse i robusnost
- odvajanje povećava fleksibilnost – neovisnost o sustavu pohrane (BP)
 - npr. ADO.NET, LINQ, EF, ...
- omogućuje objektno-relacijsko preslikavanje (ORM)

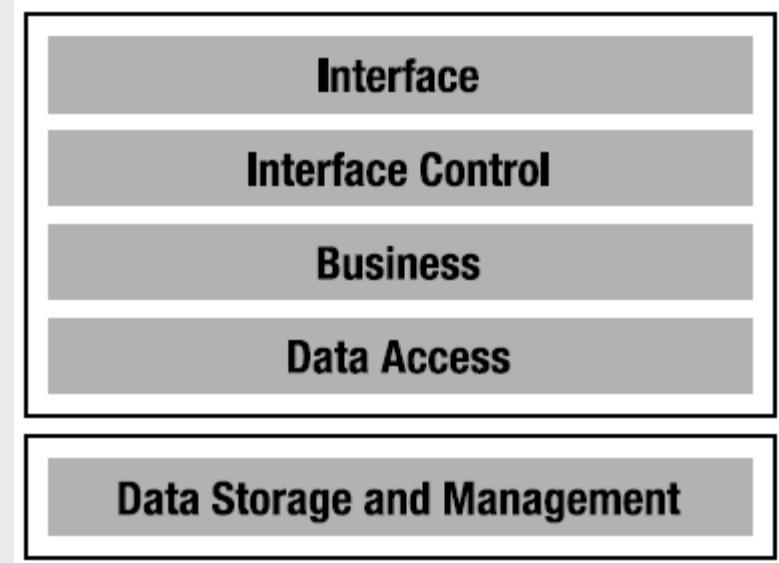
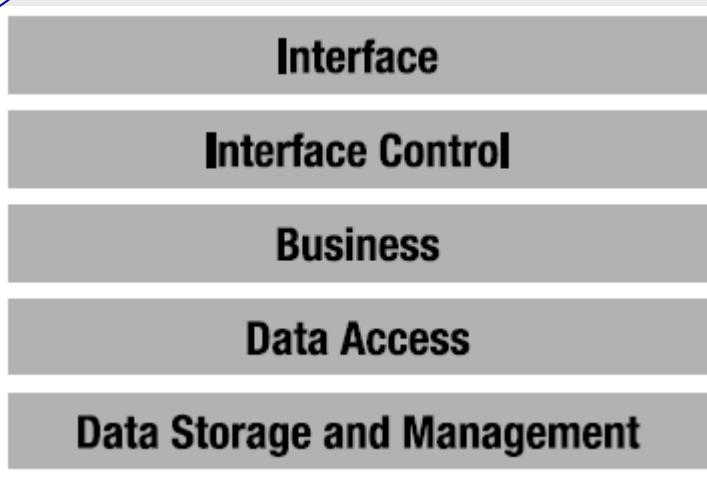
□ Sloj pohrane i rukovanja podacima

- upravlja fizičkim stvaranjem, dohvatom, ažuriranjem i brisanjem podataka
- mehanizmima RSUBP ili XML servisa implementira CRUD mehanizme
- može replicirati dio poslovnih pravila, npr. pravila integriteta koji su varijanta poslovnih pravila

Fizička ugradnja logičke arhitekture

□ Pametni (smart) klijent s optimalnim performansama

- Samostalna aplikacija
 - svi logički slojevi mogu biti smješteni na istom stroju
 - npr. blagajna (Point of Sale – POS) uz periodički transfer podataka
 - u osnovi isto što i
- Debeli klijent
 - odvajanje DSML – osim fizičkog premještaja baze podataka zahtijeva samo promjenu priključka (konekcije) na BP
 - najviše okvirno 300 istovremenih korisnika



Visokoskalabilni pametni klijent u troslojnoj fizičkoj arhitekturi

□ Odvajanje DL na aplikacijski poslužitelj

- dobro pravilo za odvajanje DAL : od 50-100 istovremenih korisnika
- bolja sigurnost – DAL sadrži kôd pristupa BP, koji se seli na aplikacijski poslužitelj pa neće biti izložen na korisničkom računalu

□ Odvajanje BL na aplikacijski poslužitelj

- dobro za neinteraktivne aplikacije
- ipak, većina modernih aplikacija zahtijeva korisničku interakciju

budući da se isti logički slojevi mogu raseliti na više fizičkih ...

□ DL na aplikacijskom poslužitelju, BL na klijentu i poslužitelju



Visokoskalabilni pametni klijent u troslojnoj fizičkoj arhitekturi (2)

□ DL na aplikacijskom poslužitelju, BL na klijentu i poslužitelju

- DL na aplikacijskom poslužitelju - centralizirani pristup BP na jednom stroju
 - bolja optimizacija pristupa (connection pooling) ako korisnici koriste isto korisničko ime, 150-200 istovremenih pristupa s 2-3 konekcije

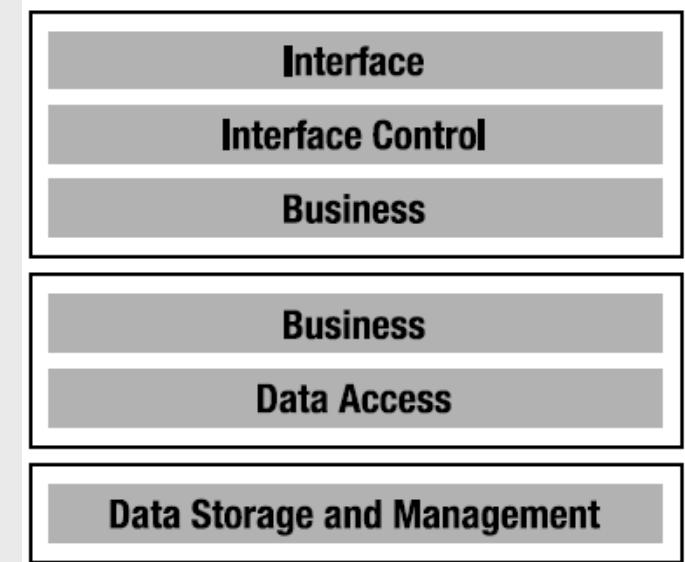
- BL na klijentu i poslužitelju
 - iskorištenje snage oba računala

□ Nedostatak:

- latencija (kašnjenje)
 - uslijed mrežnog prometa
- smanjenje performansi

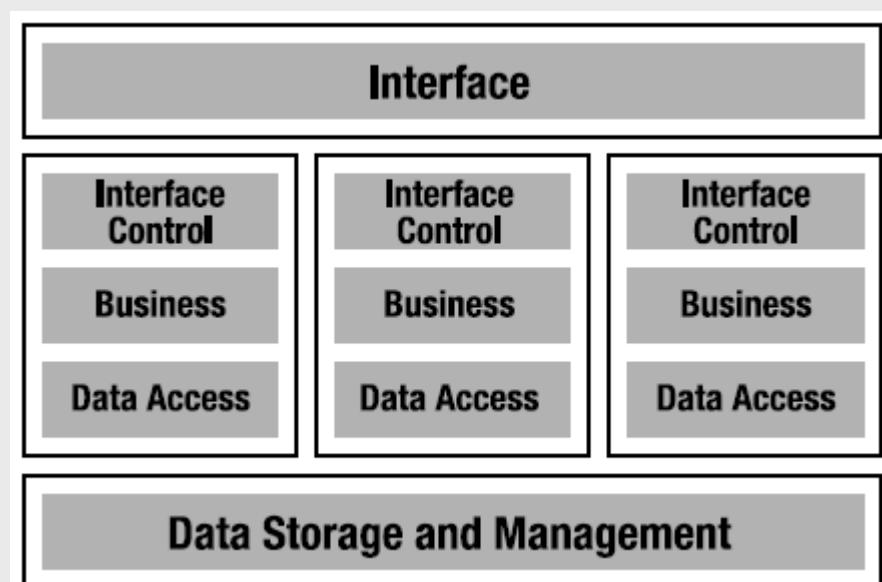
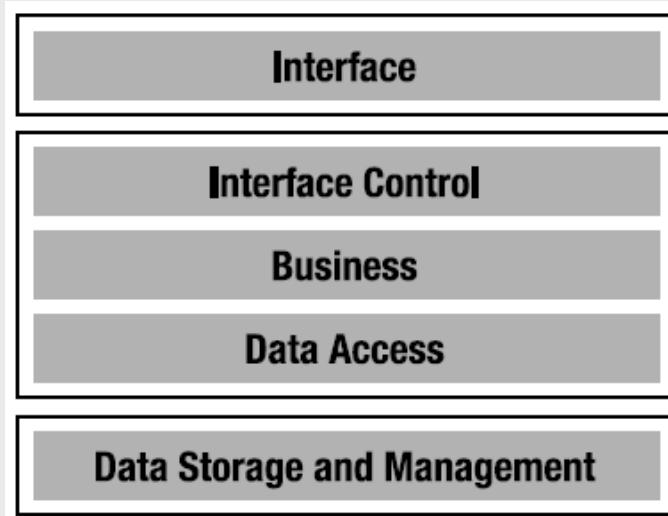
□ Prednost:

- vrlo velika skalabilnost
- preko 1000 istovremenih korisnika



Web-klijent s optimalnim performansama

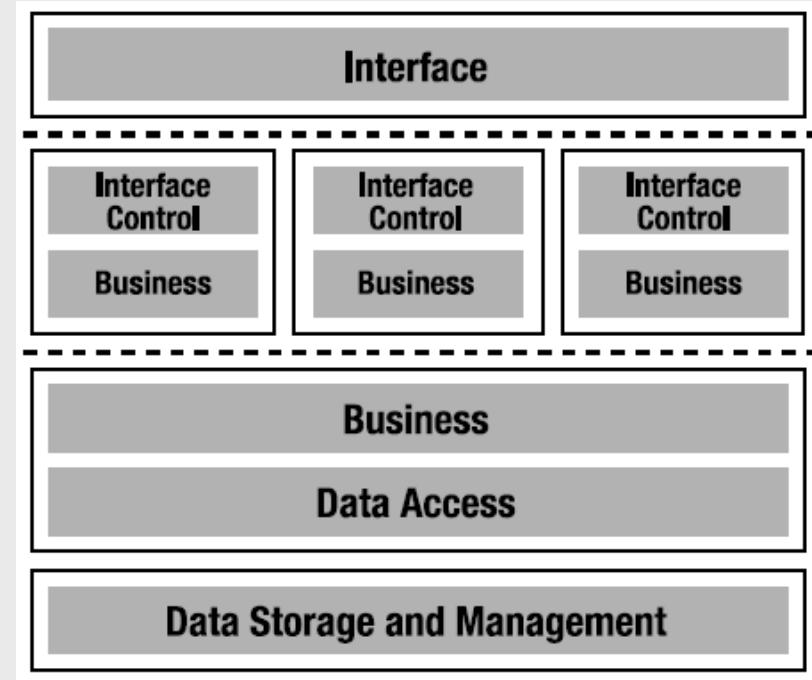
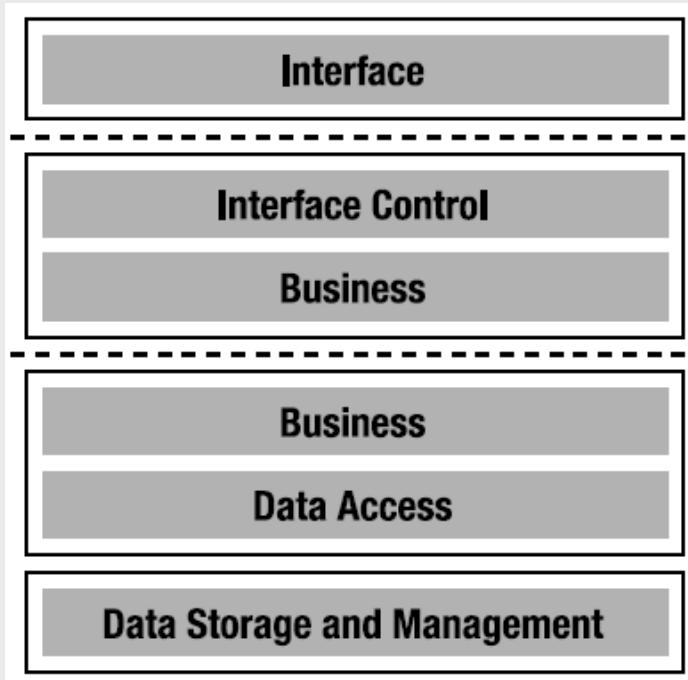
- Najbolje performanse postižu se smanjenjem broja fizičkih slojeva
- Moguće istovremeno poboljšanje performansi i skalabilnosti na uštrb sigurnosti
- Osnovna varijanta
 - IL fizički odvojen jer se izvodi u pregledniku
 - ostali slojevi na istom stroju/procesu
 - manje mrežno opterećenje, bolje performanse
- Poboljšanje
 - povećanje skalabilnosti
 - web farma na kojoj više web poslužitelja izvodi isti kod
 - dobar pristup (pooling) BP jer svaki web poslužitelj obradi stotine korisnika pa proslijeđuje zahtjeve



Vrlo sigurni Web-klijent

□ Web poslužitelj u demilitariziranoj zoni (DMZ)

- stješnjen između vanjskog vatrozida i unutarnjeg vatrozida
- komunicira s drugim poslužiteljem na kojem je BP
- slično skalabilnoj troslojnoj arhitekturi – dobro je imati BL na web poslužitelju i aplikacijskom poslužitelju
 - po cijenu 50% smanjenja performansi !
- povećanje skalabilnosti - web farmom



Odabir arhitekture s obzirom na tip aplikacija

□ Općenito, mogu se primijeniti sljedeći kriteriji:

Arhitekture	Aplikacije
Dvoslojne K/S arhitekture s tankim klijentima	<ul style="list-style-type: none"><input type="checkbox"/> Naslijeđeni, vlastiti (legacy) sustavi gdje je nepraktično i neisplativo odvajanje aplikacijske obrade i upravljanje podacima.<input type="checkbox"/> Procesno ili podatkovno zahtjevne aplikacije (pretraživanje i upiti) s vrlo malo ili bez aplikacijske obrade ("više prikazuju i računaju nego obrađuju i spremaju").
Dvoslojne K/S arhitekture s debelim klijentima	<ul style="list-style-type: none"><input type="checkbox"/> Aplikacije gdje se aplikacijska obrada izvodi na klijentu COTS (<i>Commercial Off-The Shelf Software</i>) programskom podrškom<input type="checkbox"/> Aplikacije koje zahtijevaju računalno zahtjevne obrade podataka (npr. vizualizacija podataka – interaktivno ili izvješćima).<input type="checkbox"/> Aplikacije s relativno čvrstom krajnje-korisničkom funkcionalnošću korištene u okolini gdje je dobro uspostavljeno upravljanje sustavom.
Troslojne ili višeslojne K/S arhitekture	<ul style="list-style-type: none"><input type="checkbox"/> Aplikacije velikog opsega sa stotinama ili tisućama klijenata.<input type="checkbox"/> Sustavi u kojima su i podaci i aplikacije promjenljivi.<input type="checkbox"/> Aplikacije u kojima se integriraju podaci iz višestrukih izvora.

Odabir arhitekture s obzirom na nefunkcionalna svojstva

- **Trošak infrastrukture – trošak hardvera, softvera, mreže**
 - osobna računala jeftinija 100-1000 puta od velikog računala iste procesne moći
- **Trošak razvoja**
 - trošak razvoja klijentsko-poslužiteljskih aplikacija 4-5 puta veći od onog za poslužiteljsko orijentirane aplikacije
 - trošak razvoja debelih klijenata nešto niži radi kvalitetnijih integriranih pomagala za razvoj korisničkog sučelja (npr. Access – forme i reporti)
- **Lakoća razvoja**
 - razvoj serverskih aplikacija (npr. Cobol) je teži, a nedostaje i odgovarajućih kadrova
 - težina razvoja višeslojnih razmjerna broju slojeva
- **Mogućnost sučelja**
 - GUI slabije zastupljen na čisto poslužiteljskim
- **Kontrola i sigurnost**
 - poslužiteljske lakše upravljane
- **Skalabilnost**
 - višeslojne omogućavaju zamjenjivost komponenti i preusmjeravanje opterećenja

Odabir arhitekture s obzirom na nefunkcionalna svojstva

Arhitektura	Poslužiteljska	Klijentska	Klijent-server
Trošak infrastrukture	vrlo visoko	srednje	nisko
Trošak razvoja	srednje	nisko	visoko
Lakoća razvoja	nisko	visoko	nisko-srednje
Mogućnosti sučelja	nisko	visoko	visoko
Kontrola i sigurnost	visoko	nisko	srednje
Skalabilnost	nisko	srednje	visoko

Programski primjeri aplikacija s ORM



RIS07-ORM : Objektno relacijsko mapiranje

- Primjer: Arhitektura \ ORM
 - Linq to SQL (DLINQ)
 - NHibernate
 - Entity Framework
 - XPO

RIS07-LINQ : Debeli klijent nad Dlinq

- Primjer: Arhitektura \ Firma DLINQ



Komponentizacija, ugradnja

Aplikacijski okvir

□ Aplikacijski okvir (Application Framework) – okosnica aplikacije

- A software framework is a re-usable design for a software system (or subsystem).
- Građevni blokovi aplikacije
- Skup osnovnih softverskih rutina koje čine temeljnu strukturu za razvoj aplikacije u koju se ugrađuju aplikacijski specifične komponente (poslovni objekti, poslovna pravila, zaslonske maske, ...)

□ Okvir poduzeća (Enterprise Framework)

- cjelovito okruženje za razvoj i ugradnju složenih informacijskih sustava
- sadrži prethodno napravljene (pre-built) aplikacije i razvojne alate za prilagodbu i integraciju tih aplikacija te pisanje novih
- primjer: okviri za ERP, CRM
- mogu imati komponente za upravljanje radnim tokovima (workflow component)

Primjeri aplikacijskih okvira

- FirmaWin (RPPP)
- CSLA.NET
 - The framework enables developers to leverage the power of object-oriented design as the basis for creating powerful applications.
 - Business objects based on CSLA automatically gain many advanced features that simplify the creation of WPF, ASP.NET MVC, Web Forms, WCF, WF and Web Services interfaces.
- Microsoft Enterprise Library
 - The Microsoft Enterprise Library is a collection of reusable software components (application blocks) designed to assist software developers with common enterprise development cross-cutting concerns (such as logging, validation, data access, exception handling, and many others).
- Ostali : class library, MFC, Struts, AFC, JFC, OWL

Elementi aplikacijskog okvira CSLA.NET

□ Prezentacijski sloj

- sučelje glavne forme
- bazne forme i kontrole
- logika ponašanja aplikacije, komunikacija između formi, komunikacija s glavnom formom

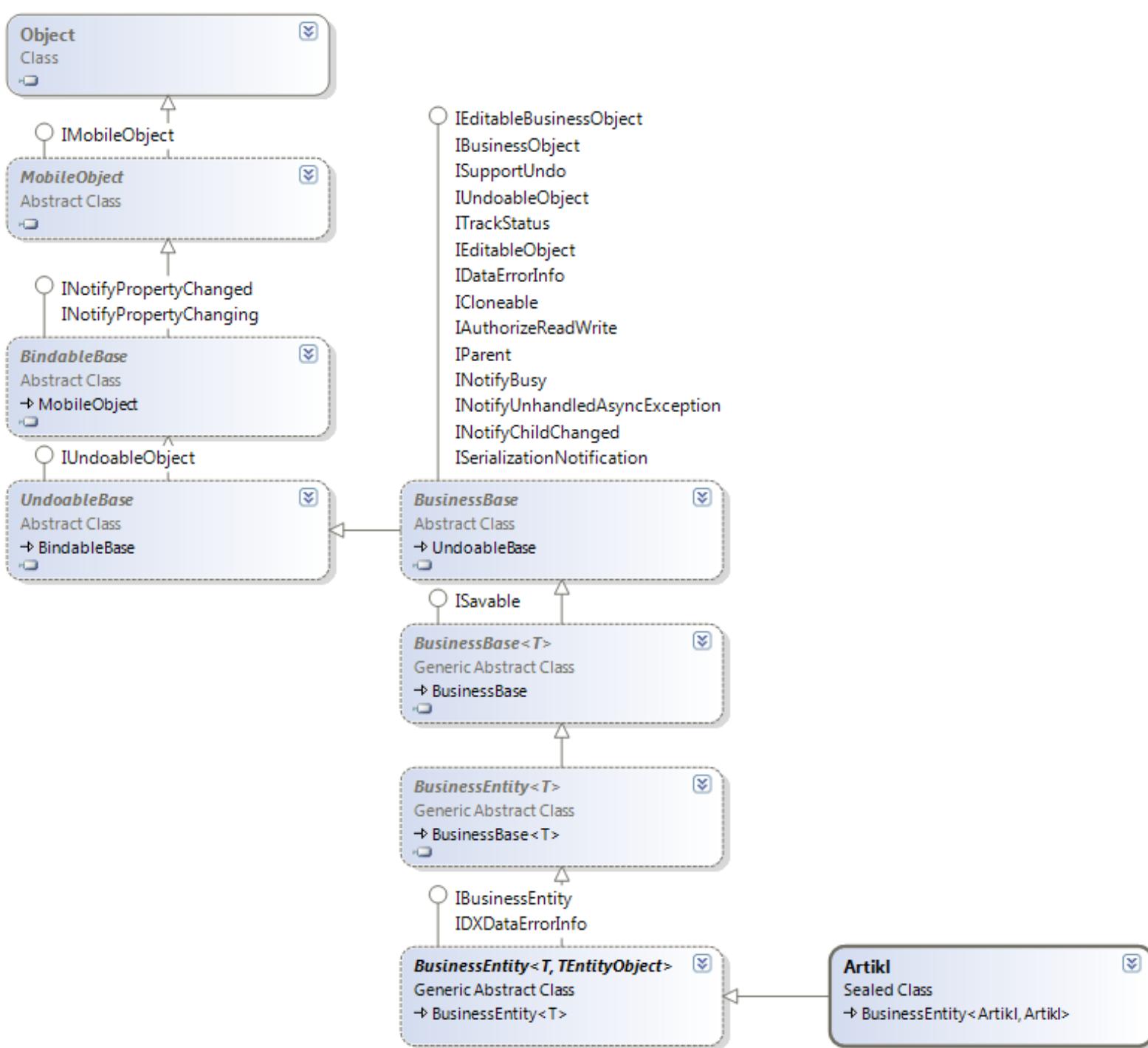
□ Poslovni sloj

- bazne poslovne klase
- pomoćne klase (validacija, kontrola pristupa podacima)
- automatizacija upravljanja poslovnim objektima (stvaranje, uništavanje, prijenos između klijenta i poslužitelja)

□ Podatkovni sloj

- bazne klase za pristup podacima
- automatizacija čitanja, spremanja i brisanja podataka
- upravljanje transakcijama

Primjer: poslovna klasa



Dijagram paketa

Package Diagram

Paketi

□ Paket (Package) = opći mehanizam grupiranja elemenata

- logički povezana grupa elemenata modela

□ Elementi dijagrama

- podsustavi
- drugi (manji) paketi
- realizacija slučajeva korištenja
- sučelja (interfaces)

□ Svojstva

- paketi tvore imenike (namespace)
- pojedini element sadržan je samo u jednom paketu
- nazivi unutar paketa moraju biti jedinstveni
- paketi mogu referencirati druge pakete
- UML podrazumijeva da postoji anonimni paket-korijen (root package)

□ Primjena - u fazi razvoja

- grubi pregled zahtjeva (high-level overview)
- konceptualno projektiranje – dizajn sustava
- organizacija izvornog koda
- logička modularizacija složenih dijagrama

Dijagram paketa

- **Pokazuje pakete i razrede te zavisnosti između njih**
 - Zapravo, dijagram razreda koji prikazuje skupove razreda i njihove zavisnosti, ali naziv "dijagram paketa" naglašava glavne elemente, razrede
- **Notacija, varijante prikaza i alternative prikaza**
 - paket, zavisnost, ugniježdeni razredi
 - moderni alati preferiraju dijagrame povezane u hijerarhiju (primjeri slijede)



- **Naziv paketa koristi se za tvorbu punog naziva razreda**
 - paket definira prostore imena - *namespace* u C# i C++
 - Vanjski paketi ponekad se nazivaju *domene*.
 - npr. puni naziv razreda Racun u paketu Dokumenti je Dokumenti.Racun

Zavisnost paketa

□ Zavisnost (dependency)

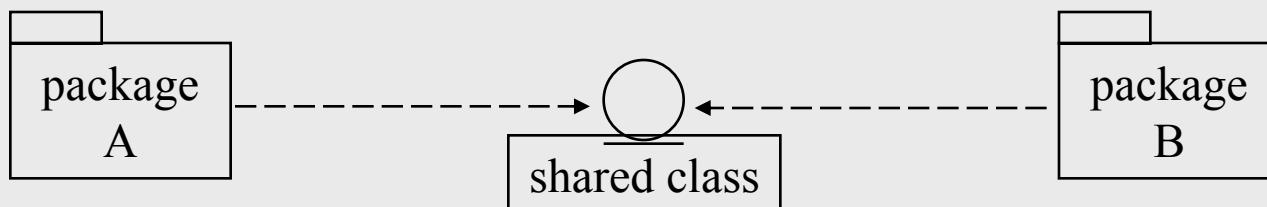
- Zavisnost između paketa postoji ako neki razred iz jednog paketa zavisi o bilo kojem razredu iz drugog paketa
 - npr. cA nasljeđuje cB ili cA poziva postupak od cB
- Zavisnost paketa nije tranzitivna !

□ Postoje različite notacije zavisnosti u UML-u



□ Dijeljeni razredi mogu se:

- staviti u vlastiti paket unutar drugog paketa
- izdvojiti u vanjski paket



Formiranje paketa

□ Kriteriji za grupiranje slučajeva korištenja u pakete

- potpora određene poslovne funkcije
- podrška određenom sudioniku

□ Kriteriji za grupiranje razreda u pakete

- razredi okvira pripadaju u isti paket
- razredi u zajedničkoj hijerarhiji nasljeđivanja
- razredi povezani agregacijom ili kompozicijom
- razredi koji međusobno često surađuju

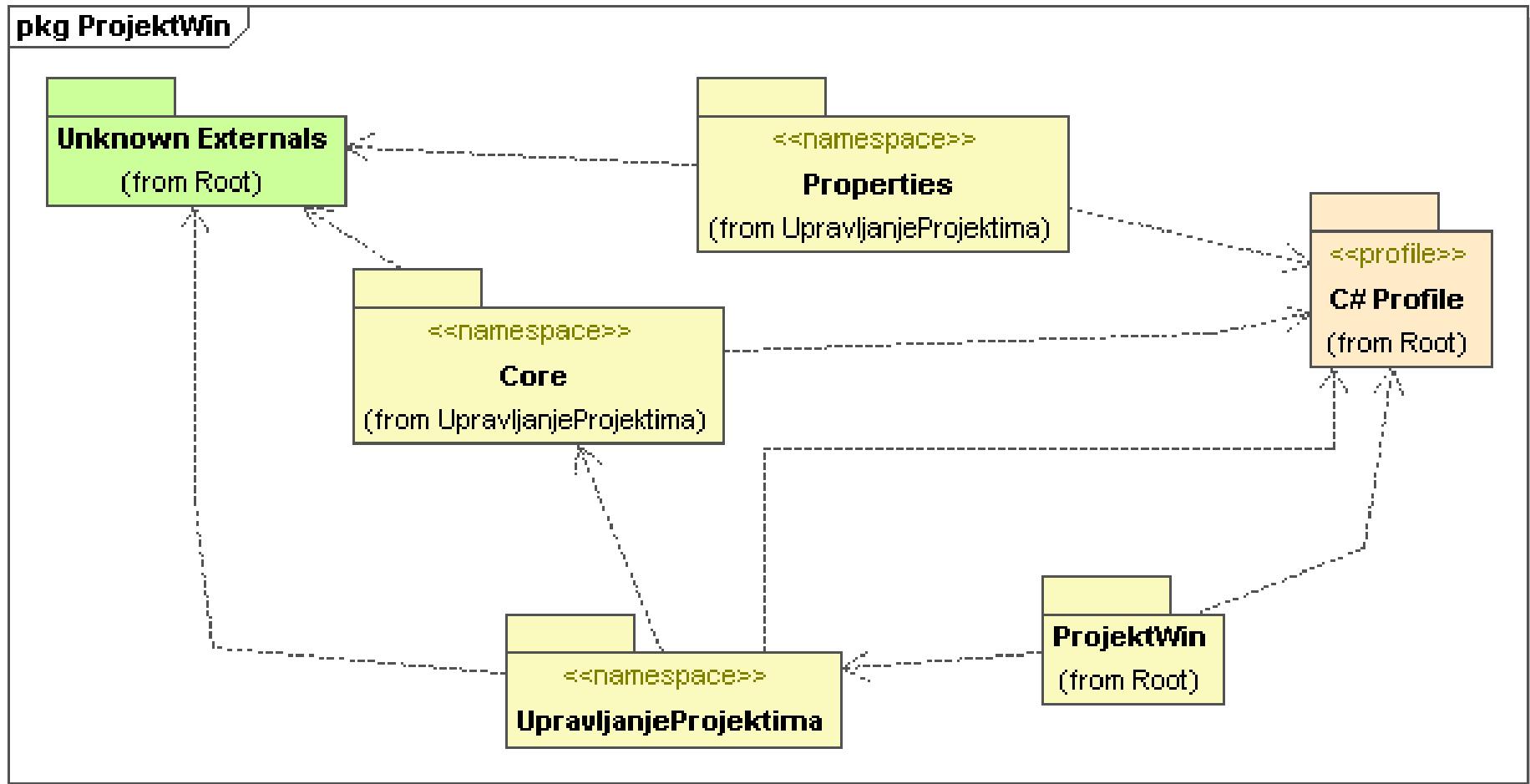
□ Smanjenje zavisnosti postiže se

- smanjenjem broja javnih razreda i njihovih operacija
- delegiranjem ponašanja odgovarajućim razredima
- pročeljima (facades) - dodatni javni razredi za komunikaciju s okolinom
 - nastaju izdvajanjem (iz svih razreda koji ih imaju) podskupova operacija zaduženih za odnose s razredima iz drugih paketa
 - ostali razredi paketa postaju privatni, tj. vidljivi unutar istog paketa

Primjer: Zavisnost paketa *ProjektWin*

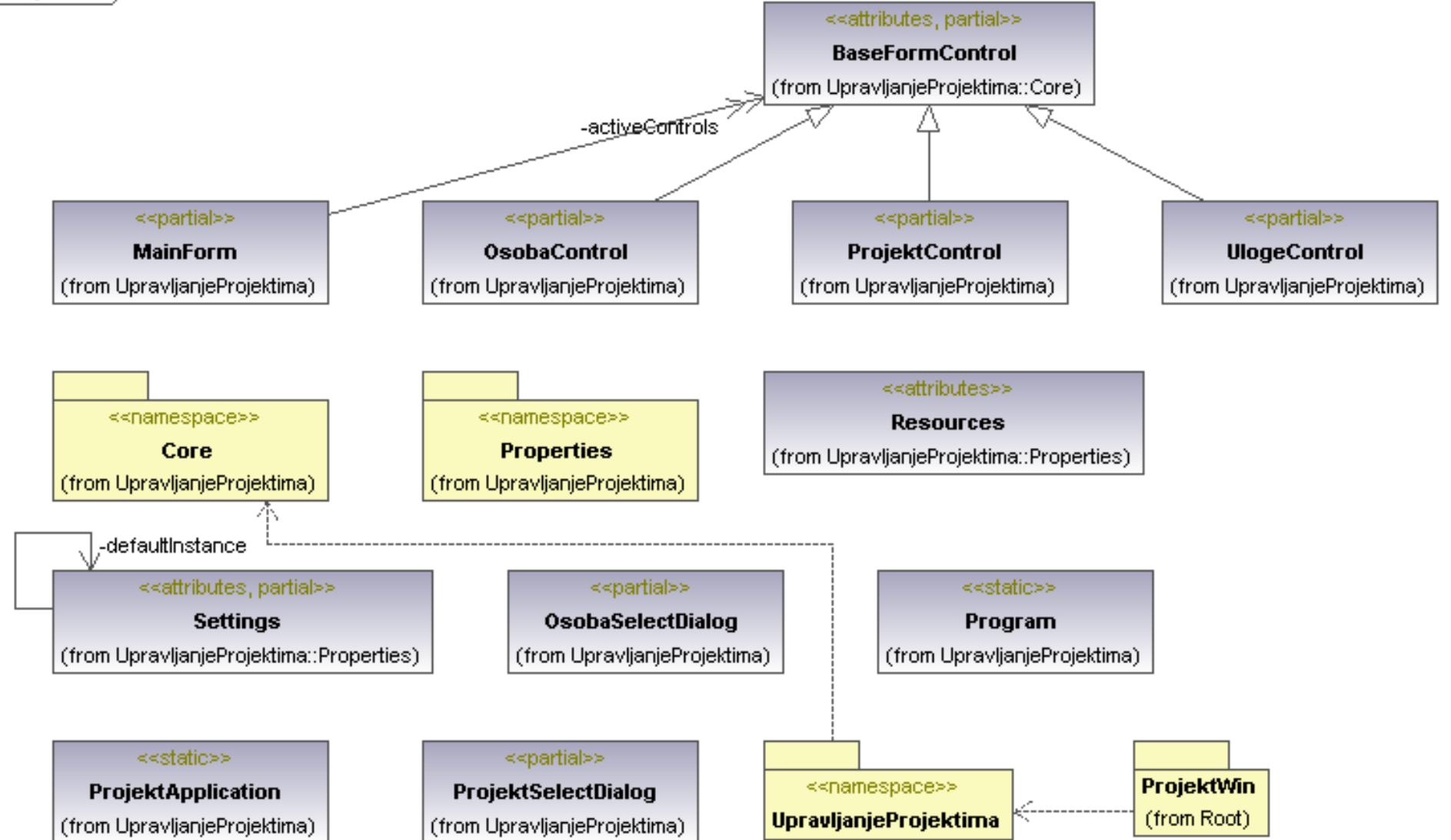
□ Primjer: ProjektCSLA \ ProjektWin

- alt.: VS 2010 - Architecure – Generate Dependancy Graph by Namespace



Paket ProjektWin

ProjektWin



Sadržaj paketa *UpravljanjeProjektima*

pkg UpravljanjeProjektima



<<partial>>
MainForm

<<partial>>
OsobaControl

<<static>>
Program

<<partial>>
OsobaSelectDialog

<<static>>
ProjektApplication

<<partial>>
ProjektControl

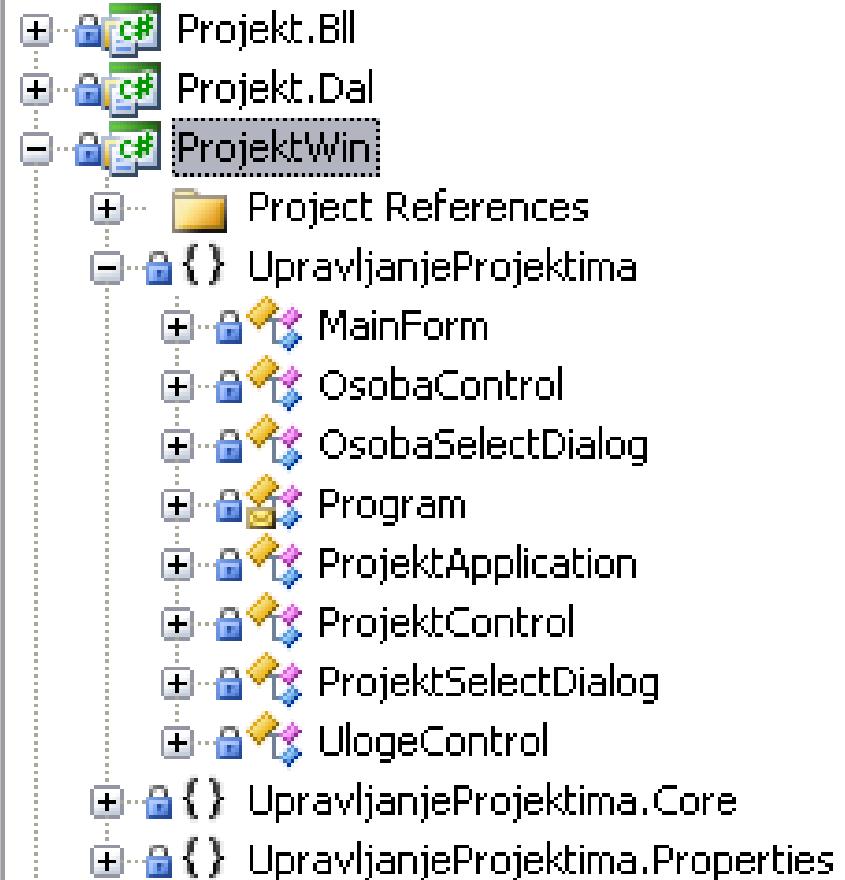
<<partial>>
ProjektSelectDialog

<<partial>>
UlogeControl

Class View



<Search>



Fizički dijagrami

Physical Diagrams

Dijagrami komponenti

□ Dijagram komponenti (Component diagram)

- prikaz organizacije i zavisnosti softverskih komponenti

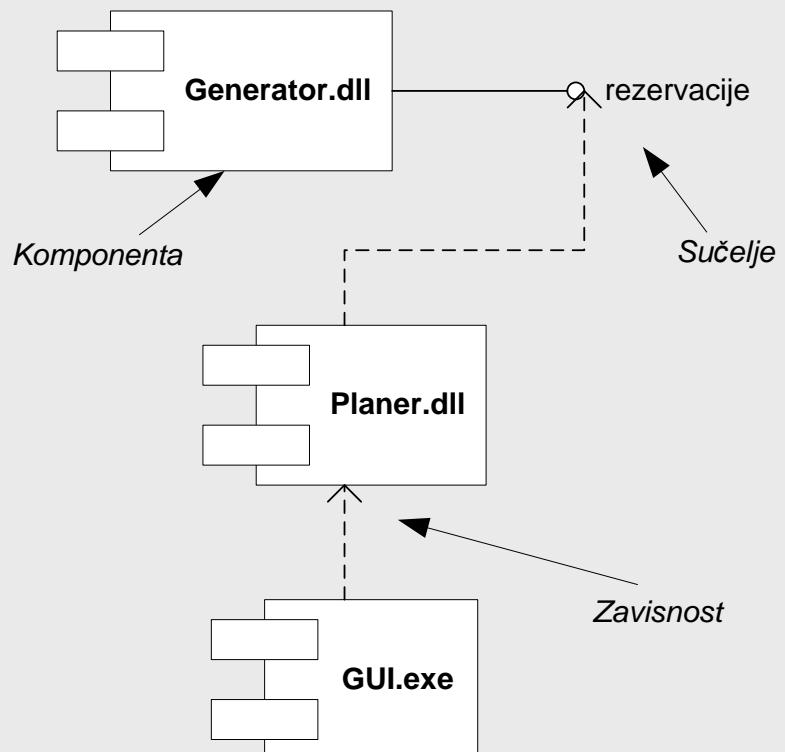
□ Komponenta: fizički modul programskog koda

- izvorna datoteka, pogonska komponenta (run time) ili izvedbeni program (executable)

□ Zavisnost komponenti

- iskazuje na koji način promjena jedne komponente može utjecati na promjenu drugih komponenti
- primjeri: zavisnost pri komunikaciji, zavisnost pri prevođenju

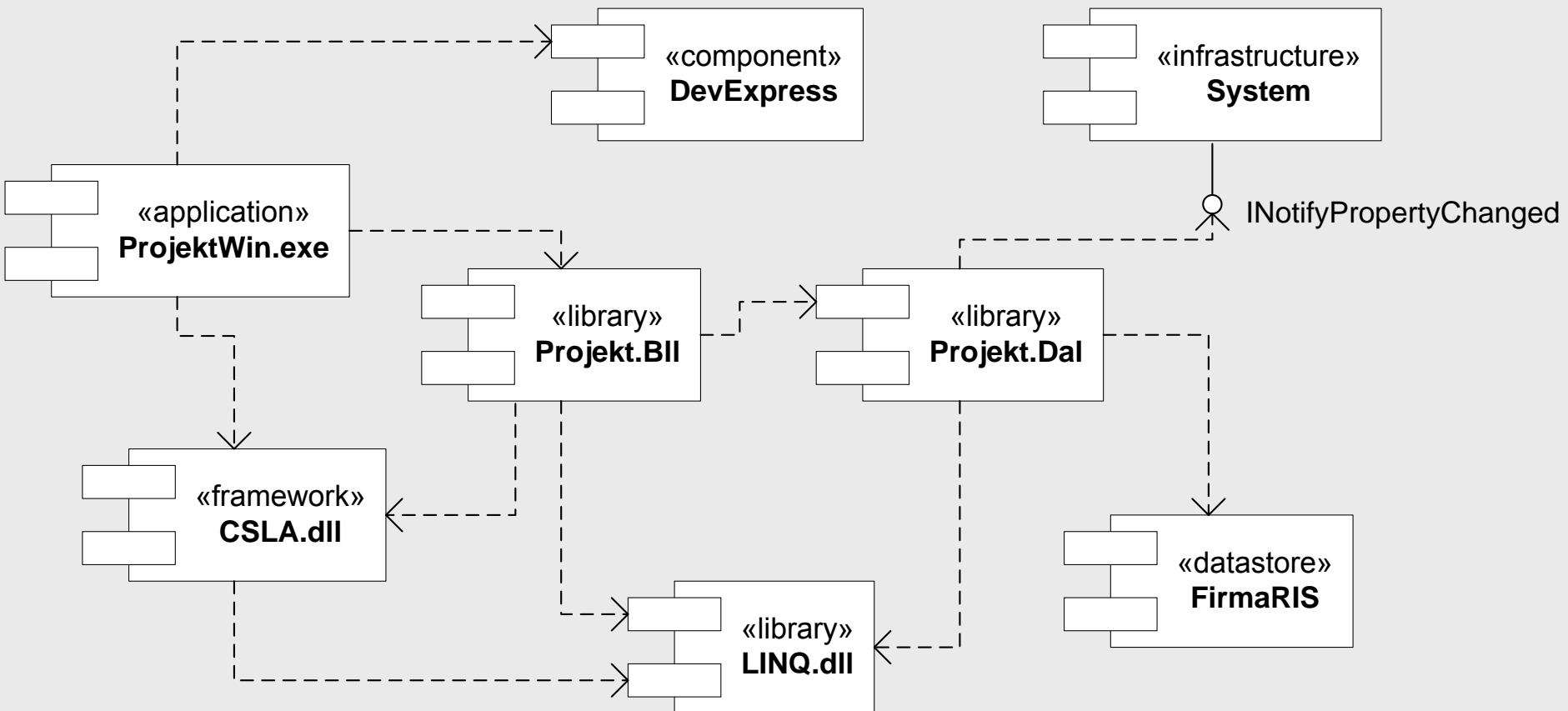
- Naziv tipa komponente je oblika: *tip_komponente*.
- Naziv instance komponente je oblika: *ime_komponente : tip_komponente*.
 - Ako se naziv tipa izostavi izostavlja se i dvotočka.



Primjena dijagrama komponenti

- **Komponenta sliči paketu, ali podliježe drugim pravilima**
 - reprezentira fizičko pakiranje programskog koda
 - pojedini razred može se koristiti u različitim komponentama, ali je definiran u samo jednom paketu
- **Primjena**
 - detaljni dizajn konfiguracije sustava
 - modeliranje tehničke infrastrukture
 - modeliranje poslovne/domenske arhitekture organizacije
- **Označavanje komponenti stereotipovima (pr. Visio – UML – Stereotypes)**
 - <<application>> komponenta prednjeg plana, npr. Web stranice ili Win GUI
 - <<datastore>> trajna pohrana podataka
 - <<document>> papirnat ili elektronički dokument
 - <<executable>> komponenta izvodljiva na fizičkom čvoru
 - <<file>> datoteka (s podacima)
 - <<infrastructure>> tehnička komponenta, npr. servis pohrane ili logger
 - <<library>> knjižnica funkcija
 - <<source code>> datoteka izvornog koda
 - <<table>> tablica u bazi podataka
 - <<web service>> jedan ili više web servisa
 - ...

Primjer dijagrama komponenti ProjektWin



- alt.: VS 2010 - Architecure – Generate Dependancy Graph by Assembly

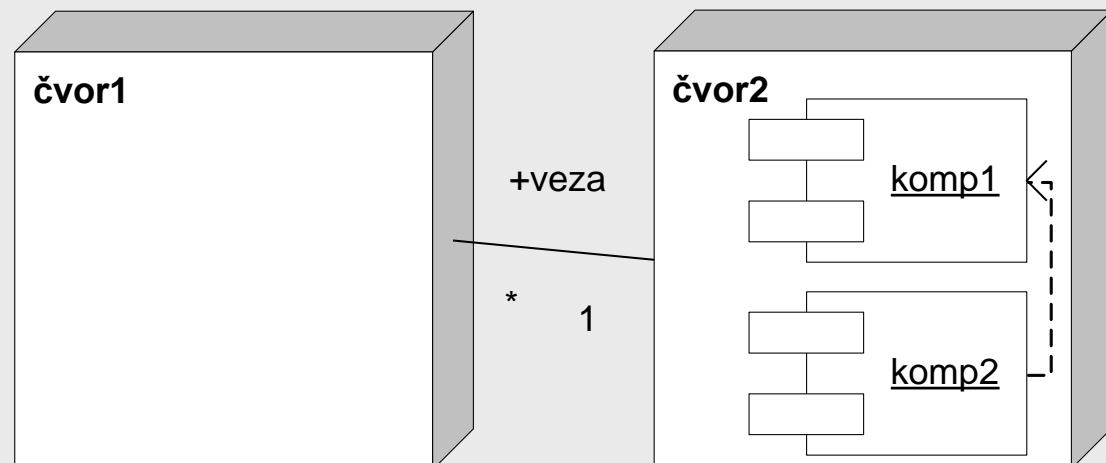
Dijagram ugradnje

□ Dijagram ugradnje (Deployment Diagram)

- Prikaz konfiguracije pogonskih elemenata i softverskih komponenti koji žive u njima.
- Čvorovi (Nodes): izvršni resursi - najčešće sklopljene
- Spojevi (Connections) - komunikacijski putovi, pr. TCP/IP
- Nazivlje: ime_čvora : tip_čvora (opcionalno)

□ Kombiniranje dijagrama komponenti i dijagraama ugradnje

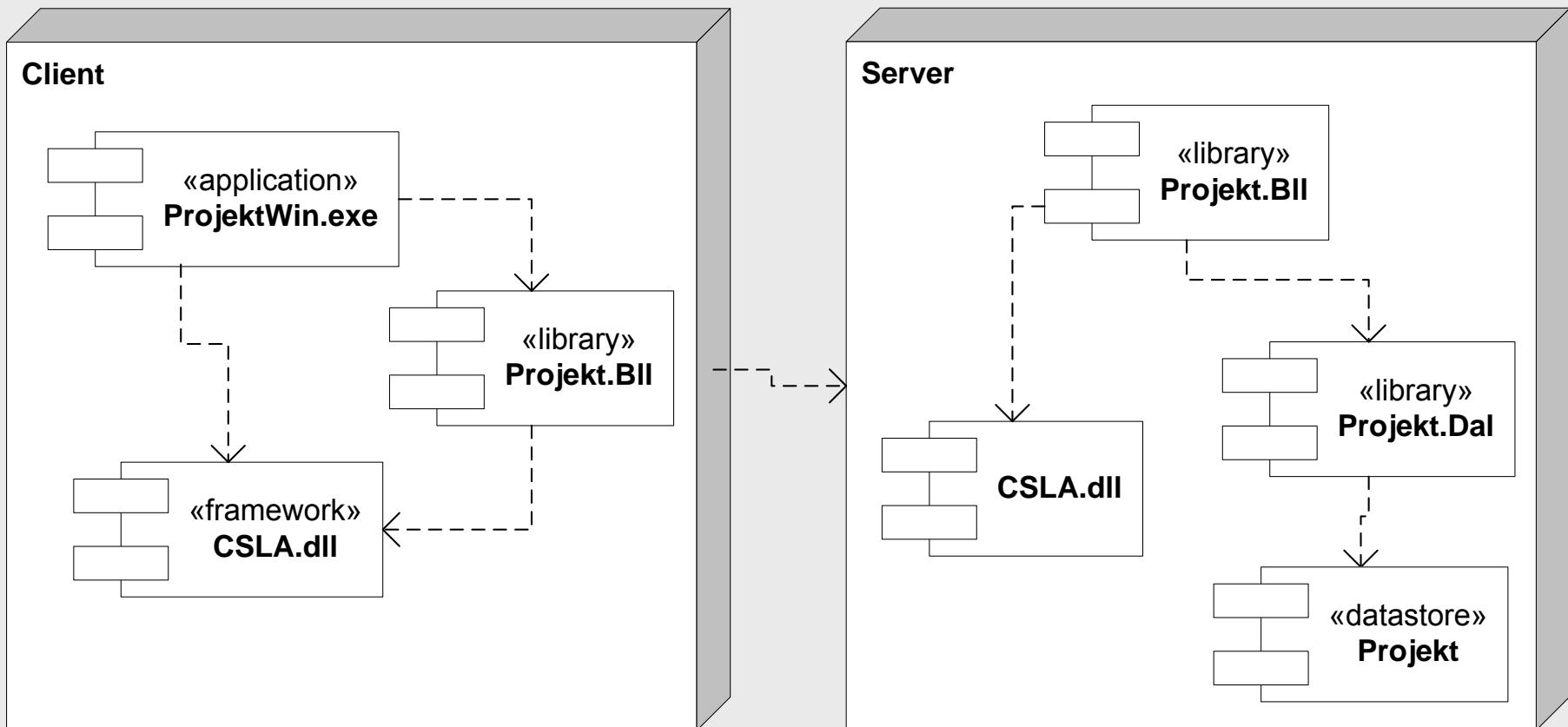
- prikaz fizičke zavisnosti između softvera i hardvera
- lokacije komponenti unutar (distribuiranog) sustava
- instance softverskih komponenti koje predstavljaju pogonsku manifestaciju jedinica programskog koda.



Primjer dijagrama ugradnje

□ Distribuirani klijent-poslužitelj (tzv. smart client)

- odvajanjem baze Projekt na treći poslužitelj BP – troslojno rješenje
- stavljanjem svih komponenti osim BP na klijenta – debeli klijent
- Primjer:  Arhitektura \ DataPortal (client: Step into, server: Start new)



Primjeri projekata s različitim korisničkim sučeljem nad istom osnovicom RIS07-CSLA



Primjeri: Arhitektura \ Projekt CSLA \

ProjektWin
ProjektMvc
ProjektWeb
ProjektWPF



Reference

Resursi

- Rockford Lhotka: Expert C# 2008 Business Objects, Apress, 2009
- Pro ASP.NET MVC 3 Framework

Aplikacijski okviri i obrasci arhitekture

- CSLA: <http://www.cslanet.com/>
- MS EntLib: <http://msdn.microsoft.com/en-us/library/cc467894.aspx>

Web

- <http://www.asp.net/ajax/>
- <http://www.asp.net/ajaxlibrary/AjaxControlToolkitSampleSite/>

WPF

- http://en.wikipedia.org/wiki/Windows_Presentation_Foundation
- Getting Started with Windows Presentation Foundation
 - <http://msdn.microsoft.com/en-us/library/ms752299.aspx>

Upravljanje protokom poslova

2012/13.08

Protok poslova

❑ Poslovni proces

- *skup međusobno povezanih aktivnosti, nastao kao odgovor na neki pokretački događaj u cilju dostizanja određenog rezultata za korisnika ili ostale dionike u procesu [Sharp2008]*

❑ Protok poslova (workflow)

- *automatizacija poslovnog procesa, djelomično ili u cjelini, pri čemu se dokumenti, informacije i zadaci za pojedinu akciju prosljeđuju od jednog sudionika procesa prema drugom prema definiranom skupu proceduralnih pravila [WfMC-TC-1011]*

❑ Upravljanje protokom poslova (workflow management)

- nadgledanje procesa prijenosa informacija, dokumenata i zadataka

Sustav za upravljanje protokom poslova

- **Sustav za upravljanje protokom poslova (SUPP)**
- **Workflow Management System (WMS)**
 - Interpretira formalnu definiciju procesa
 - Kreira i upravlja aktivnostima u procesu
 - Zadužen za interakciju s aplikacijama i sudionicima procesa
- **Cilj:**
 - Osigurati da se određeni zadatak obavio u pravo vrijeme od prave osobe
 - Olakšati odvajanje “kad i kojim redom napraviti” od “što i kako napraviti”

Korist od uvođenja upravljanja protokom poslova

□ Poslovna razina

- Poboljšana učinkovitost – automatizacijom poslovnih procesa
- Bolja kontrola procesa – standardizacijom posla i nadzora
- Poboljšane usluge korisnicima – veća konzistentnost procesa
- Fleksibilnost – lakše preoblikovanje procesa prema poslovnim potrebama
- Poboljšanje poslovnih procesa - racionalizacijom i pojednostavljenjem

□ Razvojna razina

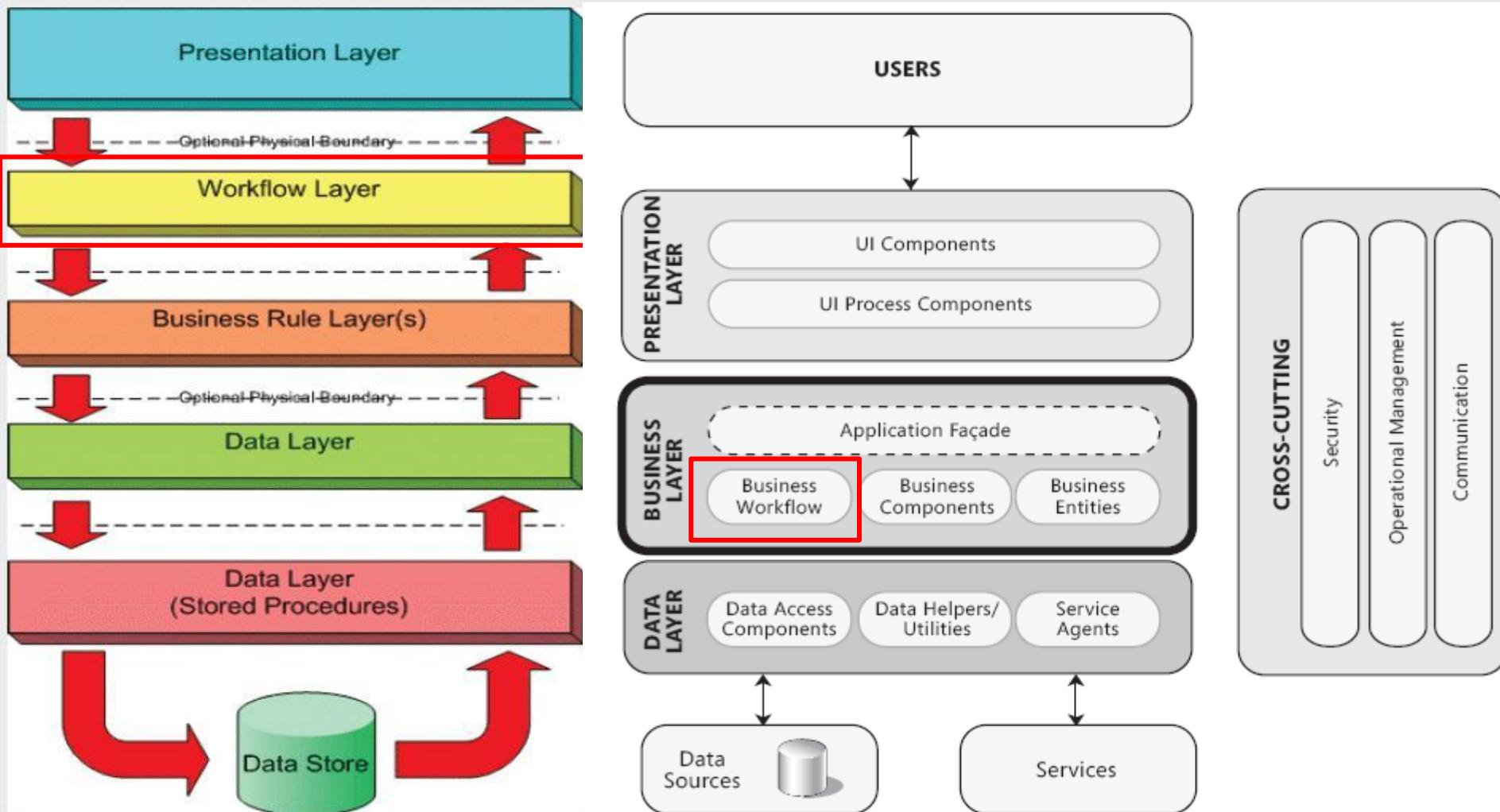
- Smanjen rizik razvoja – zajednički jezik analitičara i razvojnika
- Centralizirana implementacija – promjena poslovnog procesa ne zahtjeva značajne (drastične) promjene u softveru
- Brzi razvoj – slaganje procesa ubrzava razvoj i olakšava održavanje

□ Općenita prednost

- implementacija poslovnog procesa više nije nejasna kombinacija dijelova softvera kroz nekoliko sustava

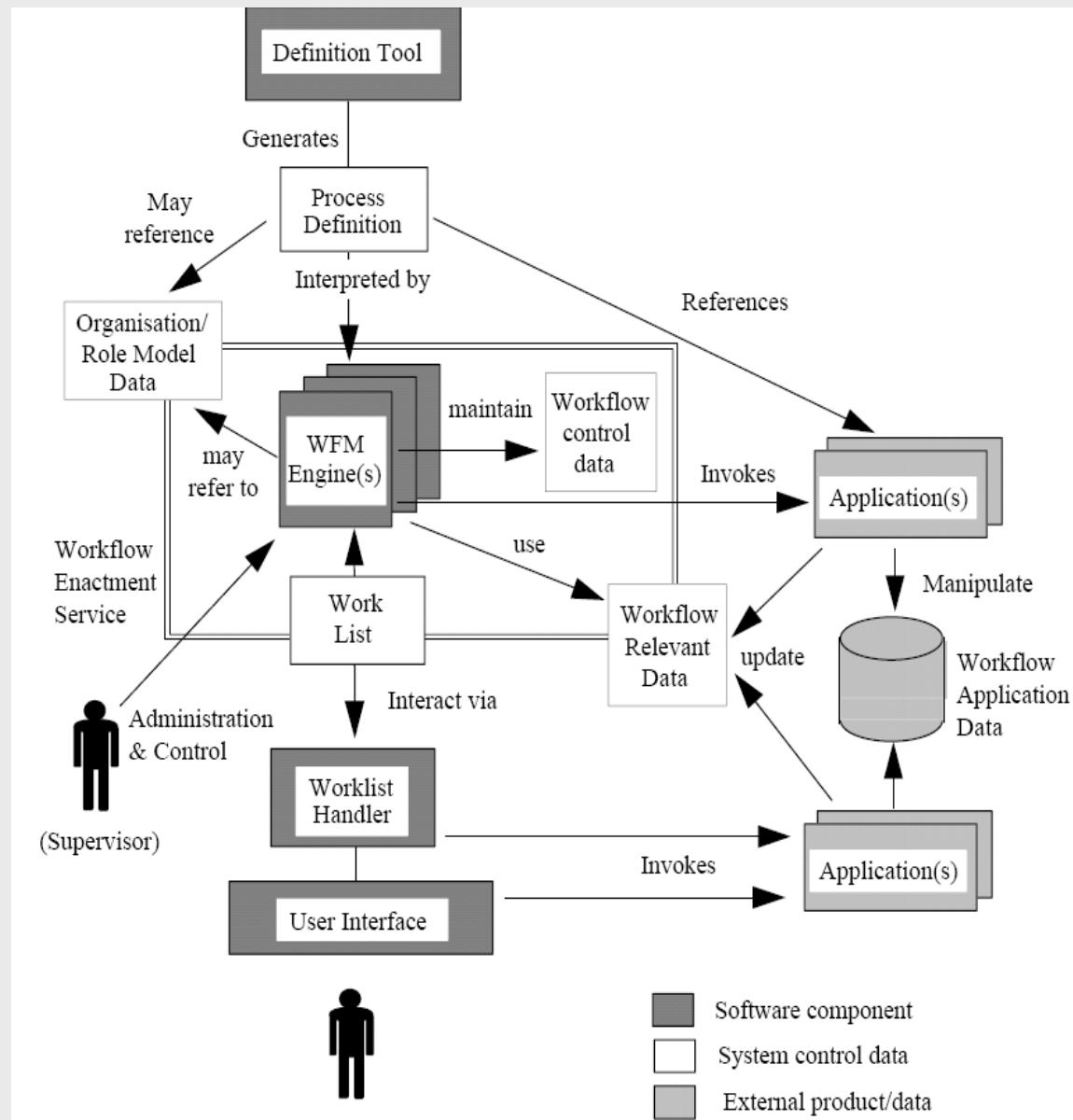
Protok poslova u višeslojnoj aplikaciji

- sloj u petoslojnoj ili dio poslovnog sloja troslojne arhitekture



Generička struktura sustava za upravljanje PP

- **Tri tipa komponenti (WfMC 1995)**
- **softverske komponente**
 - funkcije SUPP
- **sistemske definicije i kontrolni podaci**
- **vanjske aplikacije/podaci**
 - aplikacije i pripadne baze podataka
 - nisu dio samog sustava za ali se pozivaju od strane sustava

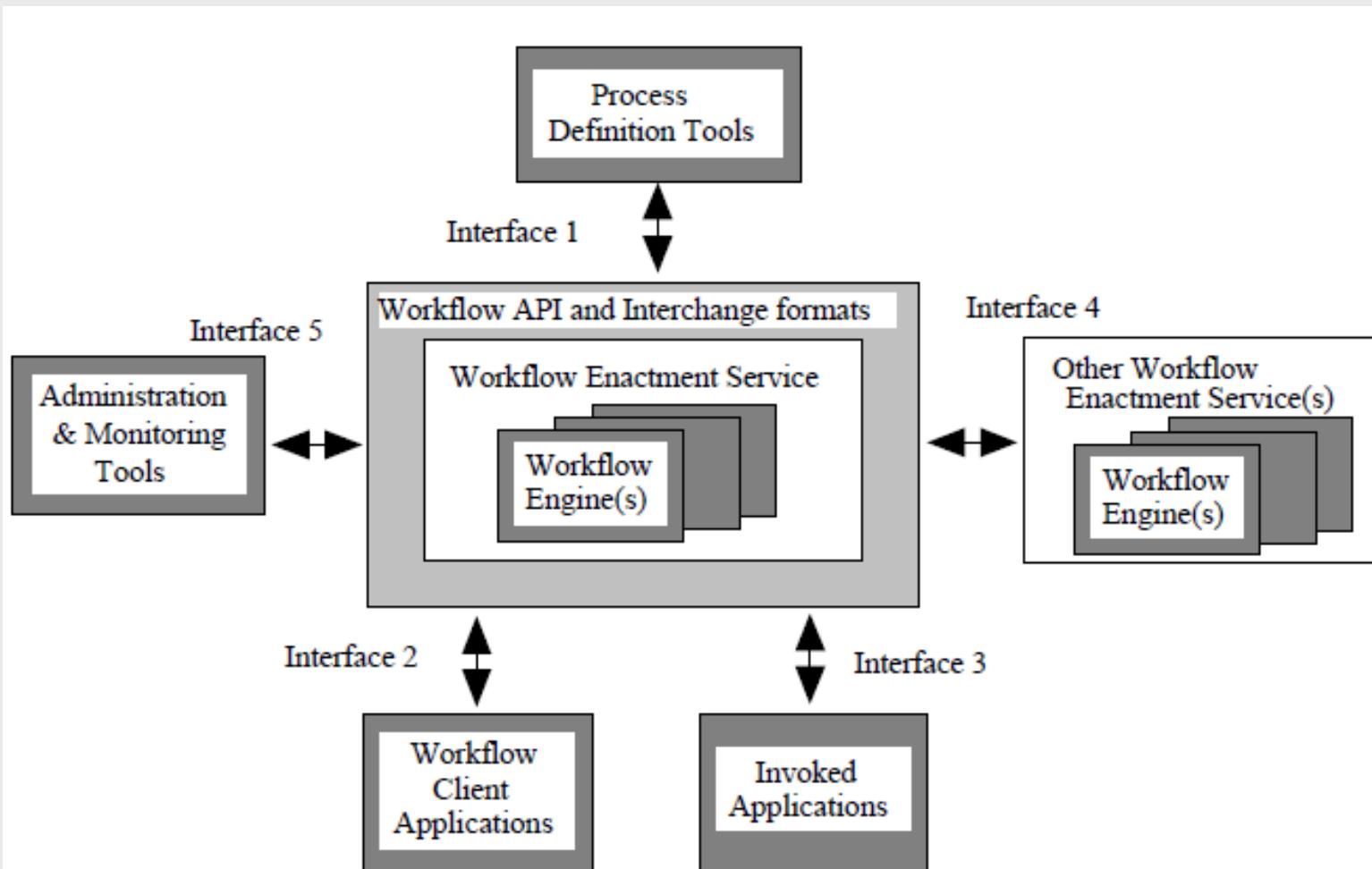


Referentni model

- različiti SUPP, nepostojanje jedinstvenog, zajedničkog standarda

□ Nastojanje: <http://www.wfmc.org/reference-model.html>

- terminologija, funkcionalni opis ključnih komponenti, tehnološki neutralan



Osnovni dijelovi referentnog modela (1)

□ Jezgra sustava za upravljanje protokom poslova

- Workflow Enactment Service - interpretira definiciju procesa
- Uz pomoć jednog ili više pogoniča (engine) stvara i izvršava instance protoka poslova i upravlja aktivnostima unutar instanci
- Inicira pokretanje vanjskih aplikacija

□ Alat za definiranje procesa

- opisivanje poslovnog procesa i prijenos opisa u format prilagođen SUPP
- definicija procesa: prezentacija u obliku koji podržava automatizaciju
 - informacije o aktivnostima, podacima, resursima i povezanim aplikacijama
 - redoslijed aktivnosti, uvjeti početka i dovršetka aktivnosti, uvjeti prijelaza
- različite notacije: BPEL (OASIS), BPMN (OMG), XPDL (WfC), PIF, ...
 - težnja za standardizacijom, relativno neuspješno
- alati za izradu mogu biti različiti i općenito ne moraju biti dio sustava

Osnovni dijelovi referentnog modela (2)

□ Workflow Aplikacija

- općeniti izraz za aplikaciju koja komunicira s jezgrom SUPP

□ Klijentska aplikacija (Workflow Client Application)

- Koristi sadržaje i usluge sustava za upravljanje protokom poslova
- Upravlja listom zadataka i rukuje podacima
- Inicira promjene stanja instance procesa te stanja aktivnosti u instanci procesa

□ Pozvana aplikacija (Invoked Application)

- Omogućava izvršavanje određene aktivnosti
- Pokreće se od strane sustava za upravljanje protokom poslova

□ Alati za administraciju i nadgledanje

□ Drugi sustavi za upravljanjem protokom poslova

Osnovni odnosi

► Process Definition

(*a representation of what is intended to happen*)

Sub-Proceses



Activities

composed of

which may be

or

Manual Activities

(*which are not managed as part of the Workflow System*)

Automated Activities

used to create & manage

Process Instances

(*a representation of what is actually happening*)

via

include one or more

Activity Instances

which include

Work Items

(*tasks allocated to a workflow participant*)

and/or

Invoked Applications

(*computer tools/applications used to support an activity*)

Osnovne strukture u definiciji procesa

□ Poslovni proces

- Sastavljen od međusobno povezanih aktivnosti

□ Aktivnosti povezane prijelazima

□ Prijelaz (transition)

- označava završetak jedne aktivnosti i početak druge aktivnosti
- uvjet prijelaza određen je uvjetima zapisanima u svojstvima prijelaza i vrijednostima varijabli pojedinog procesa.

□ Četiri osnovne vrste prijelaza između aktivnosti

- Slijed
- Odabir
- Paralelno grananje
- Ponavljanje

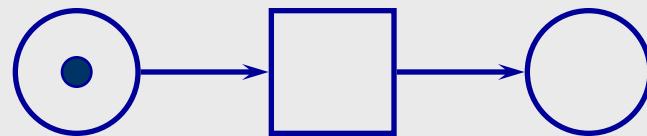
□ Složeniji obrasci (>40 obrazaca)

- <http://www.workflowpatterns.com/patterns/control/index.php>

Slijed

□ Sequence

- prijelaz između dvije aktivnosti,
 - gdje se slijedeća aktivnost izvodi odmah nakon završetka prethodne
- ne postoje posebni uvjeti prijelaza,
 - početak slijedne aktivnosti ovisi samo o možebitnim vanjskim okolnostima,
 - pr. (ne)postojanje resursa koji mogu započeti aktivnost

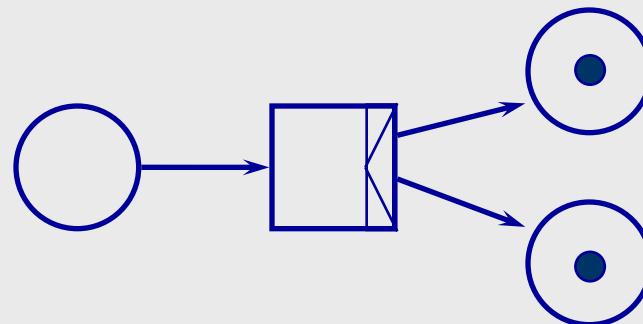


– Prema notaciji iz [Aalst2000] : prijelaz - pravokutnik, aktivnost - krug

Paralelno grananje

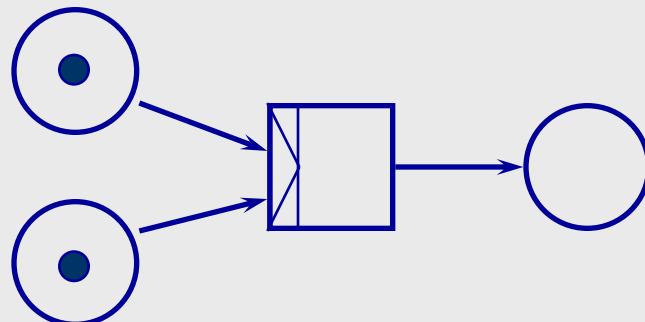
□ Parallel split, AND-split, fork, parallel routing, parallelization

- nakon završetka prethodne aktivnosti dolazi do grananja protoka poslova u dvije neovisne paralelne grane koje se mogu izvoditi istovremeno



□ Paralelne grane moraju se spojiti u sinkronizacijskoj točki (synchronization, AND-Join)

- veze između paralelnih grana nisu dopuštene



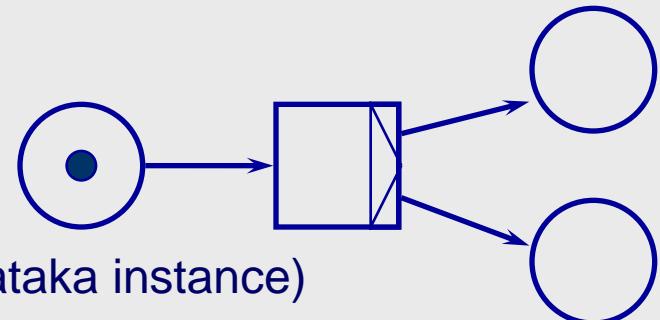
Odabir

□ Choice, selection, XOR-Split, switch, decision point

- nakon završetka prethodne aktivnosti dolazi do odabira jedne od grana

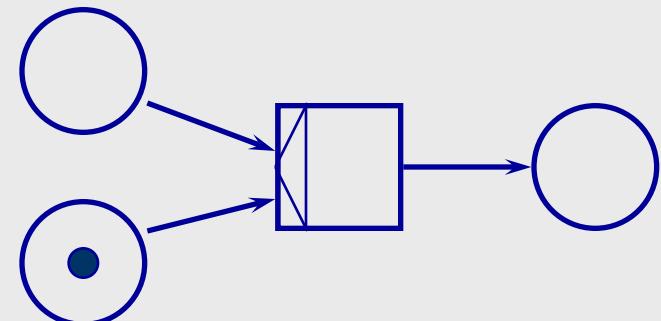
□ Različite varijante:

- Isključivi odabir (jedna i samo jedna grana)
- Višestruki odabir (jedna ili više grana)
- Eksplicitni odabir (na osnovu dostupnih podataka instance)
- Implicitni odabir (potaknut vanjskim događajem)
 - Odgođeni odabir (odabir korisnika)



□ Alternativne grane spajaju se u točki spajanja (simple merge, asynchronous join, OR-join, merge)

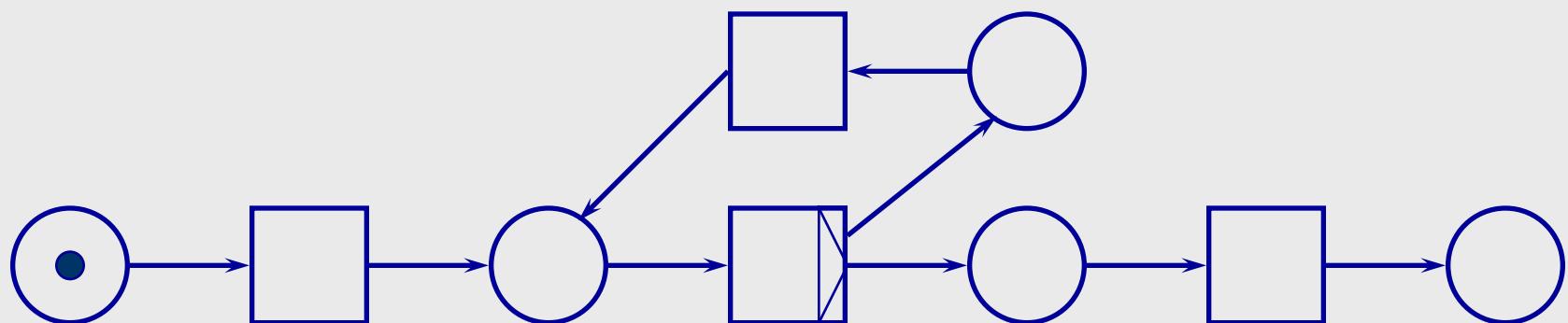
- složenija izvedba ako se radi o višestrukom odabiru



Ponavljanje

□ Loop, iteration, structured loop, arbitrary cycles

- situacija unutar procesa gdje,
- ovisno o stanju vrijednosti i uvjetu prijelaza
- može doći do ponovnog izvođenja niza aktivnosti (i odgovarajućih prijelaza)
- ili se može se prijeći na aktivnost koja bi uobičajeno slijedila iza ovakvog ciklusa

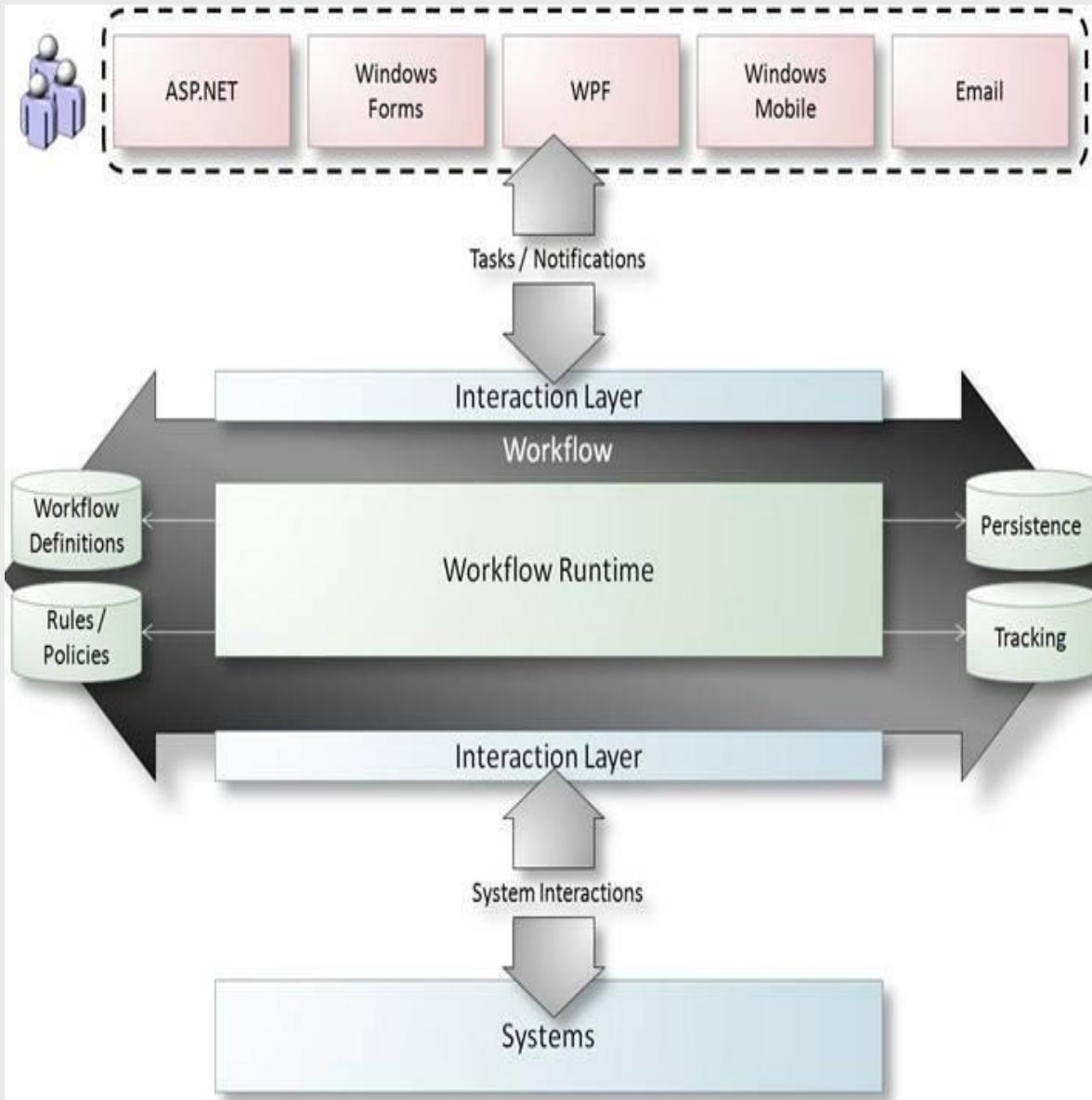


Programski primjeri

 [Workflow \ WFprimjer.zip](#)

Windows Workflow Foundation (WF)

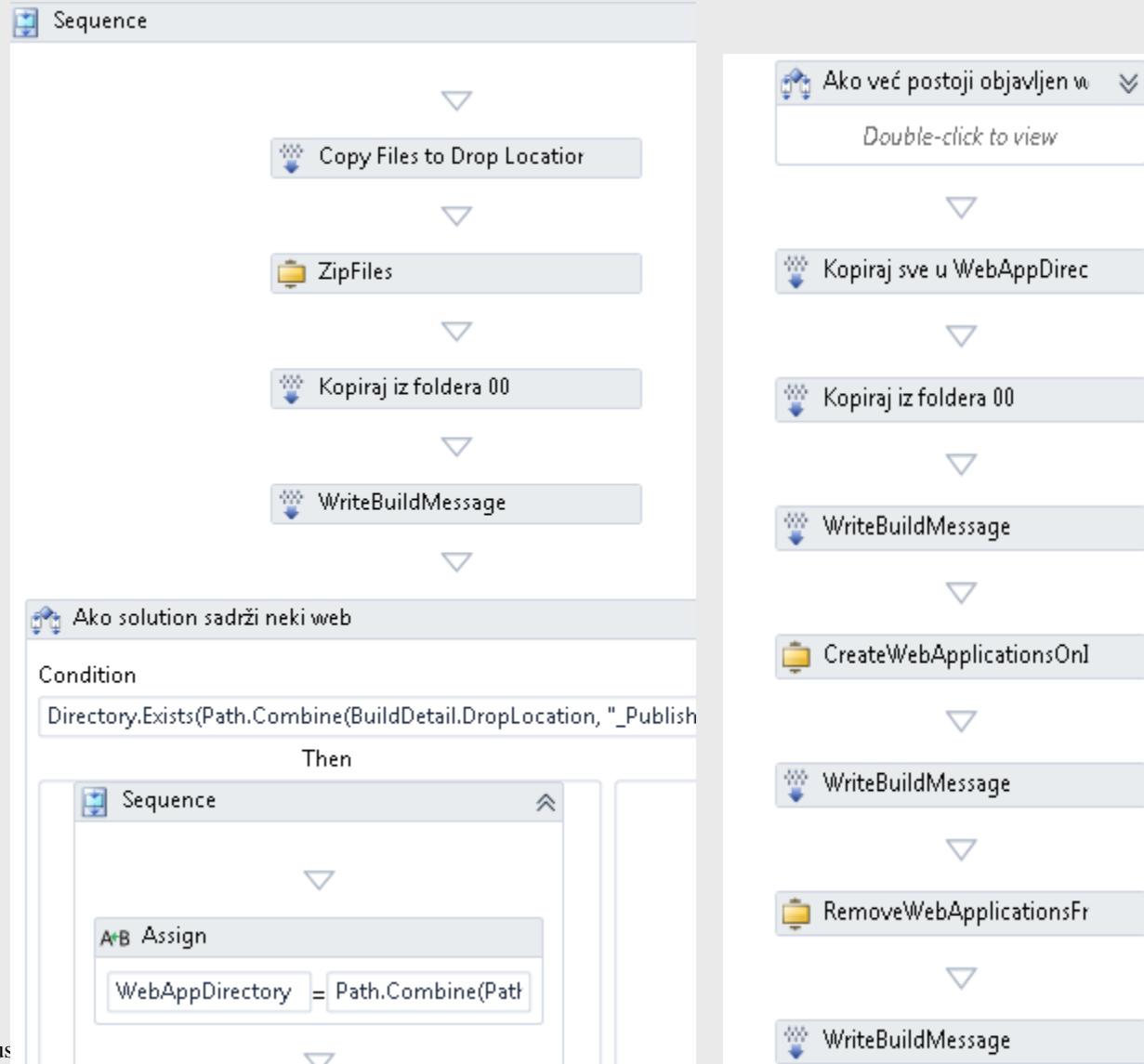
- ❑ Okvir za ugradnju protoka poslova u aplikacije
 - API za *workflow*
 - Nije zasebna aplikacija ili server (kao npr. Biztalk server)
- ❑ Koristi XML za definiciju, ali nije kompatibilan s WfMC standardima
- ❑ Tipovi modela u WF-u
 - Proceduralni
 - Flowchart
- ❑ Ugrađene komponente za praćenje rada i "postojanost"



Primjer upotrebe WF-a (1)

□ Proširenje automatske kompilacije na TFS-u, na RPPP

- Rješenje se komprimira i kopira na lokaciju dostupnu za download
- Ako u rješenju postoji web aplikacija objavljuje se na IIS-u



Primjer upotrebe WF-a (2)

- Postupak izrade putnog naloga
 - WF 3.5 + SharePointServices 3.0 + WCF servisi
- Pokreće se dodavanjem novog dokumenta
 - WF usmjerava daljnji tijek obrade putnog naloga i stvara zadatke korisnicima

radni nalozi Računi kupcima Isplate zaposlenicima Izrada putnog naloga Odobravanje putnih naloga Dozvole ostalim korisnicima

IPISVU > Stanje tijeka rada

Stanje tijeka rada: Obrada putnog naloga

Informacije o tijeku rada

Pokretač:	Boris Milašinović	Dokument:	9d28f7de-4612-415c-9793-0f44edae5048
Pokrenut:	19.7.2010 9:20	Stanje:	U tijeku
Posljednji put pokrenuto:	19.7.2010 9:21		

Dođe do pogreške ili se ovaj tijek rada prestane odazivati, može biti prekinut. Prekidanje tijeka rada postavit će njegovo stanje na Otkazano i izbrisati sve zadatke koje je tijek.
■ Prekinite ovaj tijek rada sada.

Zadaci

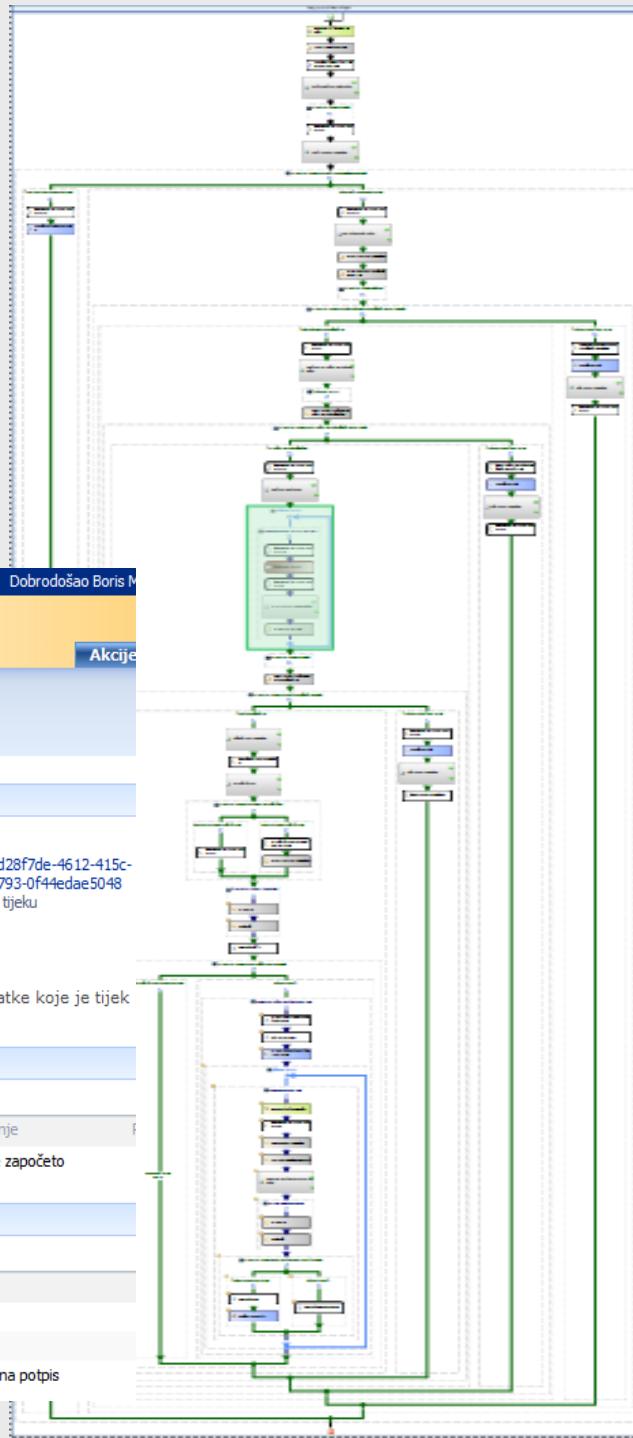
Slijedeći zadaci su pridruženi sudionicima ovog tijeka rada. Pritisnite zadatak kako biste ga uredili. Možete isto tako pregledati ove zadatke na popisu **Zadaci**.

Dodatajeno	Naslov	Krajnji rok	Stanje
Krešimir Fertalj	Putni nalog za djelatnika: Boris Milašinović(Maribor, 12.07.2010.) Broj naloga:12437 Novo!		Nije započeto

Povijest tijeka rada

Sljedeći događaji pojavili su se u ovom tijeku rada.

Datum pojave	Vrsta događaja	ID korisnika	Opis	Rezultat
19.7.2010 9:21	Komentar	Račun sustava	Nalog zaprimljen i poslan na daljnju obradu	Nalogodavac provjeren
19.7.2010 9:21	Komentar	Račun sustava	12437	Putni nalog je spremljen u SAP
19.7.2010 9:21	Komentar	Račun sustava		Putni nalog poslan vlasniku računa na potpis



Značajnije aktivnosti ugrađene u WF (1)

□ **Sequence**

- Kontejner za aktivnosti koje se izvršavaju slijedno

□ **If**

- Sadrži dvije grane za *Then* i *Else* uvjeta upisanog u svojstvu *Condition*

□ **Switch<T>**

- Odabire granu čiji je *Case Value* jednak vrijednosti svojstva *Expression* ili odabire prepostavljenu (*Default*) granu

□ **While i Do-While**

- Ponavlja sadržane aktivnosti ako je zadovoljen uvjet u svojstvu *Condition*

□ **Assign**

- Služi za pridruživanje nove vrijednosti varijabli ili argumentu

□ **InvokeMethod**

- poziv javnog postupka nekog objekta ili javnog statičkog postupka razreda

□ **Delay**

- Zaustavlja izvršavanje (trenutne grane) na određeno vrijeme

Značajnije aktivnosti ugrađene u WF (2)

WriteLine

- Ispisuje tekst na konzolu ili na drugo odredište ovisno o postavljenom svojstvu *TextWriter*

Pick

- Čeka se na prvi od n okidača i izvršava grana za koju se dogodio okidač
- Ostale grane se otkazuju

Send i Receive

- Omogućuju poziv nekog WCF servisa iz modela protoka poslova i izlaganje modela kao WCF servisa
- Obično idu u paru (*ReceiveAndSendReply* i *SendAndReceiveReply*)

Flowchart

- Omogućava direktno povezivanje aktivnosti umjesto korištenja proceduralnih aktivnosti
- Koristi *FlowDecision* i *FlowSwitch<T>* za grananja

Značajnije aktivnosti ugrađene u WF (3)

□ Parallel

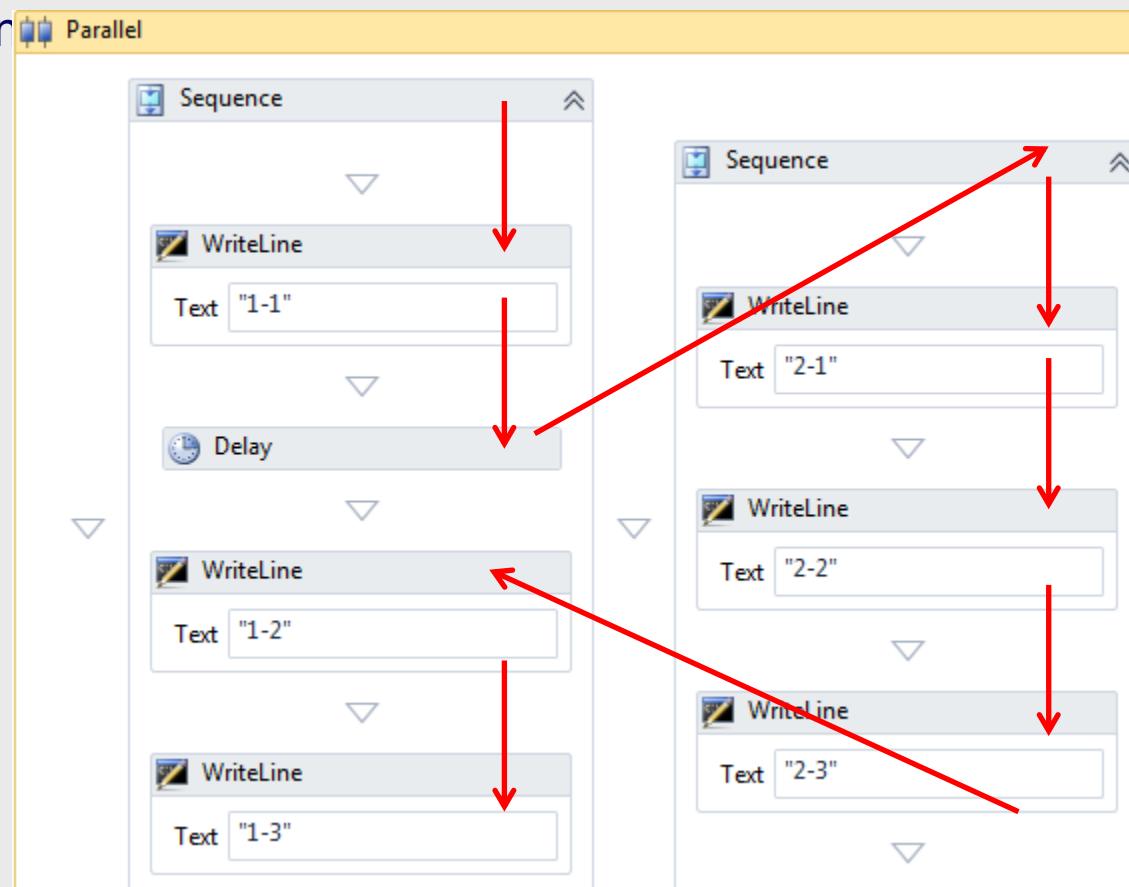
- Koristi se za aktivnosti koje se smiju istovremeno izvršavati

□ Sve grane se istovremeno pripremaju za izvršavanje, mogu se ali ne moraju izvršavati paralelno (nema pravog paralelizma)

- Izvršava se grana po granu
- Pri pojavi aktivnosti koja se izvršava asinkrono izvršavanje se prenosi na sljedeću granu

□ Opcionalno svojstvo *CompletionCondition*

- Provjerava se nakon završetka neke grane
- Ako je istina, preostale aktivnosti se otkazuju



Primjer: upravljanje izvršenjem projekta

□ Pri svakoj promjeni statusa zadatka koriste se sljedeća pravila

- Zadatak može imati sljedeće status: *Nezapočeto, Napreduje, Odgođeno, Čekanje, Pomoć, Procjena, Odbačeno, Dorada, Dovršeno*
- Interno, za svaki zadatak bilježi se povijest promjena

□ Poslovna pravila

- Ako je student postavio status na *Dovršeno* ili *Odbačeno*, promijeniti status na *Procjena* i dodijeliti zadatak nekom od asistenata
- Ako je student postavio zadatak na *Procjena*, dodijeliti zadatak nekom od asistenata
- Nakon procjene asistenta, zadatak se vraća originalnom nositelju
- Dozvoljeni statusi nakon *Procjena* su *Dorada, Odbačeno* ili *Dovršeno*.
- U slučaju nekog drugog statusa vratiti status *Procjena* i dodijeliti zadatak nekom od asistenata
- Ako je student postavio status na *Pomoć*, poslati obavijest (npr. e-mail) svim studentima iz iste grupe
- Nakon što asistent promijeni status na *Dovršeno* poslati na e-mail povijest svih promjena dovršenog zadatka

Komponente rješenja

□ Primjer: WF \ Zadaci # Windows Application

- Windows aplikacija za prikaz i ažuriranje zadataka
- Poziva protok poslova pri svakoj promjeni zadatka

□ Primjer: WF \ WfZadaci # Activity Library

- Modeli protoka poslova i razredi za rad sa zadacima

□ Primjer: WF \ WfService # WCF Workflow Service Application

- Protok poslova izložen kao WCF servis za dohvata popisa zadataka
- Koriste se razred i modeli protoka poslova iz  WF \ WfZadaci

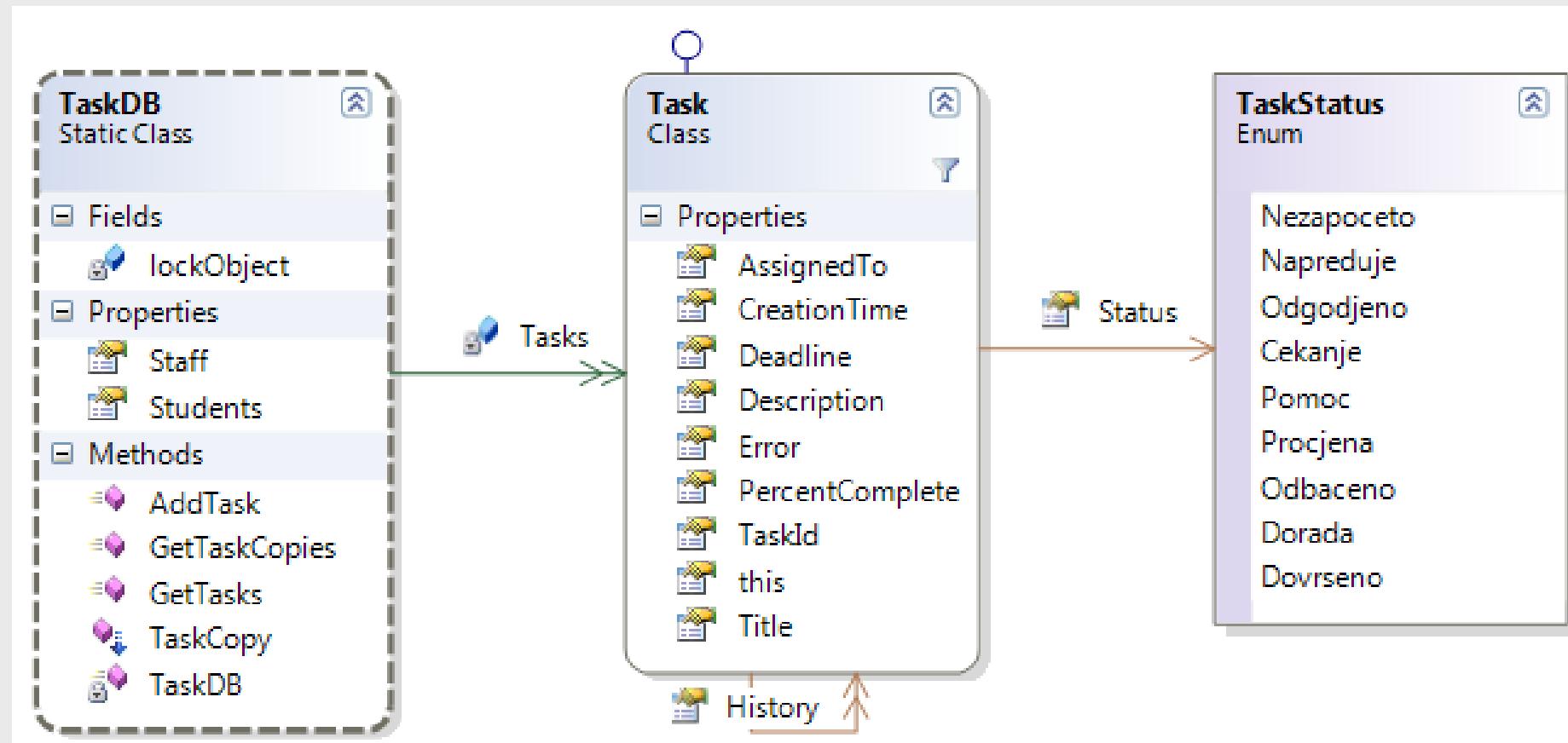
□ Tipovi datoteka

- *.XAML - activity
- *.XAMLX – workflow service

Način pohrane zadatka u primjeru

□ Primjer: WF \ WfZadaci

- Statička klasa i statička kolekcija za rad sa zadacima
- Svaki zadatak ima svoju povijest
- Razred *Task* implementira sučelje *IDataErrorInfo* za validaciju



Korisničko sučelje: lista zadataka

□ Primjer: WF \ Zadaci \ Form1

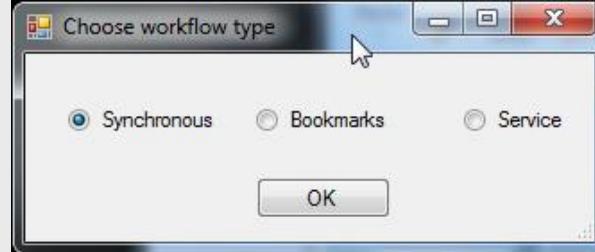
- Dvije liste zadataka (za studente i za djelatnika)
 - Dvostruki klik na ćeliju otvara formu za ažuriranje zadatka
 - Desni klik na zadatak omogućava prikaz povijesti promjena
- Tekstualni okvir za ispis poruka primljenih od instanci protoka poslova

□ Zatvaranjem forme za uređivanje zadatka, ažurirani zadatak se šalje instanci protoka poslova

- Novi nositelj i status se određuju u protoku poslova
- Liste se naknadno osvježavaju
- Primjer: *Form1.ShowEditForm*

Tasks							
Student tasks							
TaskId	Title	Description	AssignedTo	Status	PercentComplete	CreationTime	
2	RIS DZ 04		Pero Perić	Nezapoceto	0	12.05.2013. 15:09	Add

Instanciranje protoka poslova



□ **Synchronous # RunSynchronousWorkflow**

- Pri svakom dodavanju/izmjeni zadatka stvara se nova instanca protoka poslova
- Sinkroni poziv - aplikacija čeka na dovršetak instance nakon čega osvježava listu
- Ulagni argument čine podaci o zadatku, povratna vrijednost je logička vrijednost je li zadatak ispravan ili ne (ovisno o vremenskom roku)

□ **Bookmarks # RunBookmarkWorkflow**

- Demonstrira asinkrono pozivanje protoka poslova
- Nova instanca se stvara pri stvaranju novog zadatka
- Pri promjeni zadatka aktivira se postojeća instanca protoka poslova
- U slučaju da se neki zadatak nije promijenio određeno vrijeme (parametar zadan pri kreiranju nove instance) instanca protoka poslova šalje obavijest aplikaciji
- Za komunikaciju se koristi mehanizam oznaka i proširenja (bookmark, extension)

□ **Service # RunWorkflowService**

- Protok poslova izložen kao WCF servis
- Nova instanca - pozivom WCF servisa u trenutku stvaranja novog zadatka
- Prilikom promjene zadatka poziva se WCF servis
 - na osnovu parametara servisa prepoznaće se potrebna instanca protoka

Modeliranje protoka

- Primjer: WF \ WfZadaci \ WfOnStatusChange.xaml
 - XAML datoteka

□ Grafički prikaz

- Aktivnosti (ugrađene i vlastite) se dovlače s alatne trake

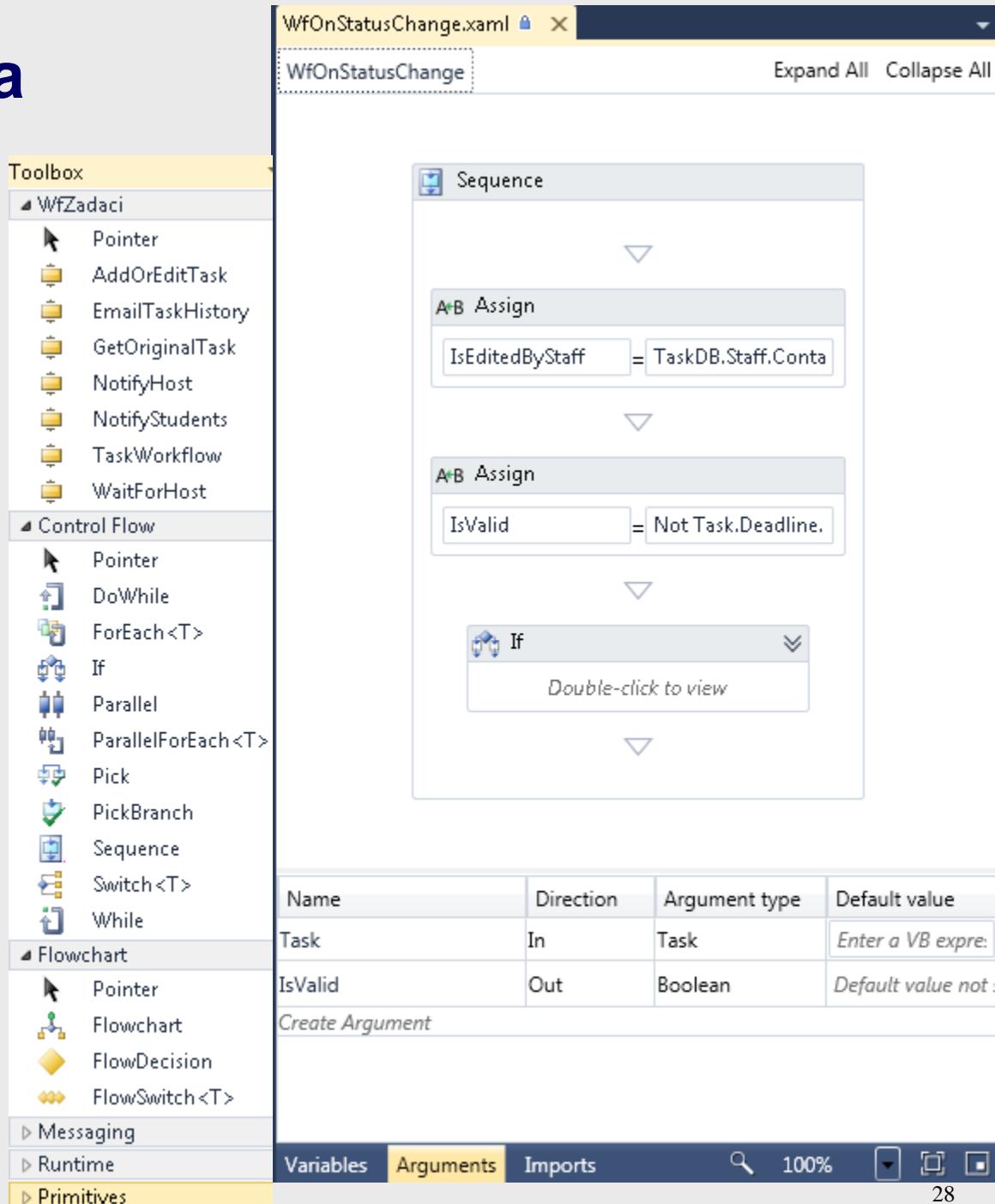
□ Ulazni i izlazni argumenti

□ Interne varijable

- doseg - aktivnost u kojoj je definirana (vidi dijagram Variables, Scope)

□ Izrazi prijelaza i tipovi podataka u VB.Net, od VS 2012 i C#

- pr. Assign IsValid



Varijable i argumenti

- Sve aktivnosti (pa tako i cijeli *workflow*) mogu imati ulazne, izlazne i ulazno-izlazne argumente i variabile (kao na slici)
 - Rezultat (ili izlazni argument) neke aktivnosti se pohranjuje u varijablu koja se koristi kao argument druge aktivnosti
 - Doseg varijable određen aktivnošću u kojoj je definirana
 - Ulazni argumenti se predaju kao *IDictionary<string, object>*
 - Value tipovi i stringovi se mogu prenositi preko imenovanog svojstva
 - Povratne vrijednosti oblika *IDictionary<string, object>*
 - Varijable se definiraju kao *Variable<T>*
 - Posebno spremište za svaku aktivnost
 - CLR varijable - jedno spremište po tipu aktivnosti
 - Moraju se moći serijalizirati (osim ako drugačije nije naznačeno)
-
- The diagram illustrates the flow of variables and arguments within a workflow. It shows a 'Workflow' boundary containing two activities: 'Aktivnost 1' and 'Aktivnost 2'. 'Aktivnost 1' has an incoming argument 'Broj' (value 5) and an outgoing variable 'Temp' (value 13). 'Aktivnost 2' receives 'Temp' and has an outgoing result 'Result' (value 44). A 'Domaćinska aplikacija' (Domestic application) interacts with the workflow: it sends 'Broj' (value 5) to 'Aktivnost 1', receives 'Temp' (value 13) from 'Aktivnost 1', and receives 'Result' (value 44) from 'Aktivnost 2'. A curved arrow at the top right indicates the overall flow of the workflow.

Pisanje vlastitih aktivnosti

- Nasljeđuje se jedan od sljedećih apstraktnih razreda
 - *CodeActivity, AsyncCodeActivity, NativeActivity*
 - Varijante s povratnom vrijednošću **Activity<TResult>* tipa *TResult*
- **CodeActivity, CodeActivity<TResult>**
 - Koristi se za kratke aktivnosti
 - Obavlja se sinkrono na istoj niti kao i *workflow*
 - Implementira se postupak *Execute*
 - Primjer korištenja na slajdu *Korištenje proširenja iz instance*
 -  WF \ WfZadaci \ NotifyHost.cs
- **AsyncCodeActivity, AsyncCodeActivity<TResult>**
 - Koristi se za aktivnosti koje obavljaju posao na posebnoj niti
 - Implementiraju se postupci *BeginExecute* i *EndExecute*
 - Primjer korištenja na slajdu *Aktiviranje oznake*
 -  WF \ WfZadaci \ NotifyStudents.cs

Pisanje vlastitih aktivnosti (nastavak)

□ NativeActivity, NativeActivity<TResult>

- Koristi se za aktivnosti koje traju duže i koje trebaju složenije funkcionalnosti
 - pr. mehanizme koje nije moguće koristiti s prethodna dva tipa aktivnosti
- Implementira se postupak Execute
- Primjer korištenja na slajdu *Aktiviranje oznake*
 -  WF\ WfZadaci \ WaitForHost.cs

□ Za sva tri tipa aktivnosti prilikom izvršavanja

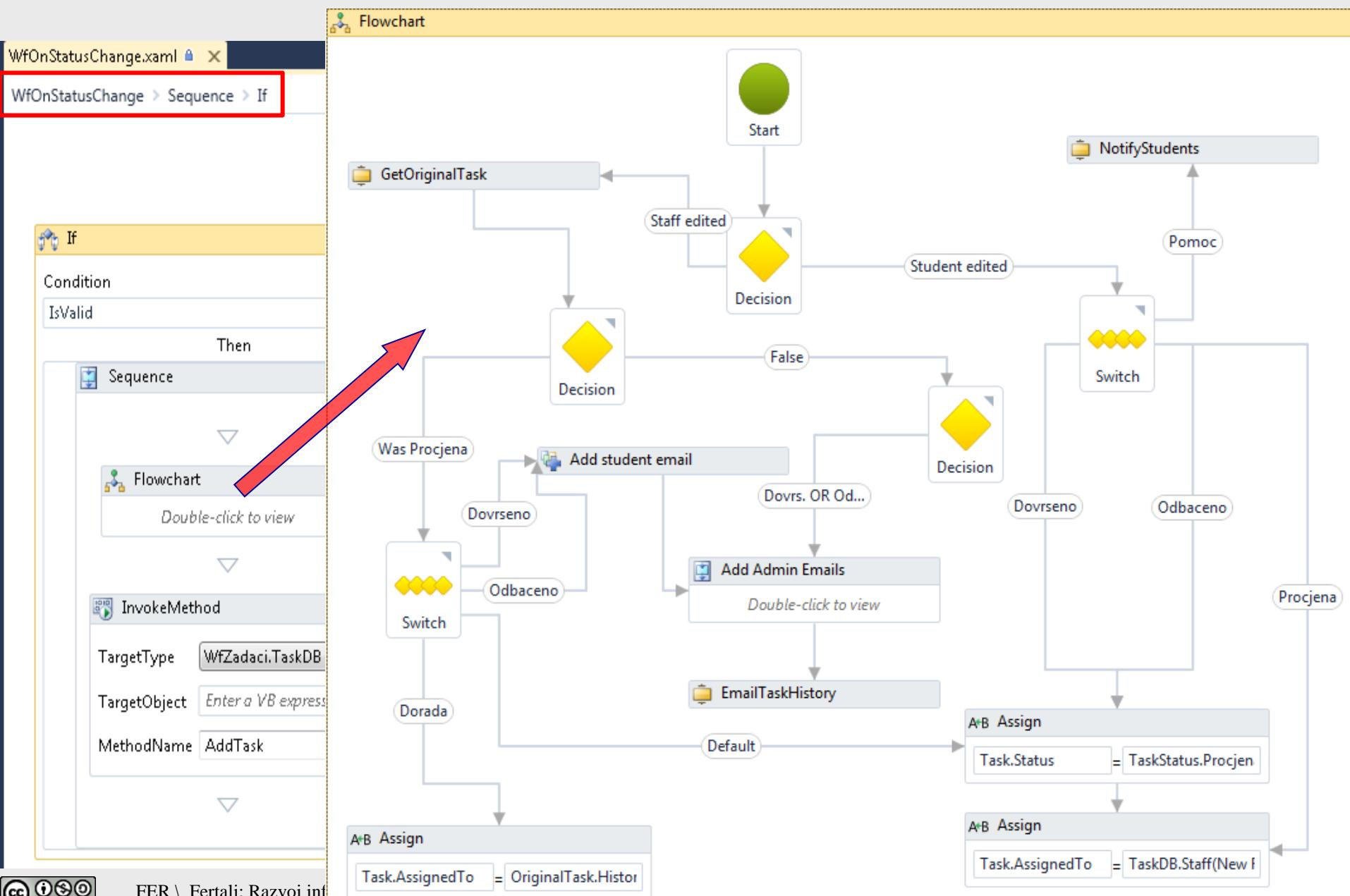
- dostupna je odgovarajuća implementacija razreda ActivityContext
- Omogućava dohvat ulaznih argumenata
 - NazivArgumenta.Get(context)
 - npr. *NotifyHost.cs*: *Task.Get(context);*

□ Korištenje vlastitih aktivnosti

- Toolbox
- pr. *GetOriginalTask*, *NotifyStudents*



Osnovni protok poslova prilikom promjene zadatka



Sinkroni poziv protoka poslova

□ Primjer: WF \ Zadaci \ Form1.cs - RunSynchronousWorkflow

- Pri svakoj promjeni zadatka stvara se nova instanca protoka poslova
- Poziva se protok poslova iz WF \ WfZadaci \ WfOnStatusChange.xaml
- Poziv statickog postupka *Invoke* u razredu *WorkflowInvoker*,

```
var workflow = new WfOnStatusChange();
var inputs = new Dictionary<string, object>();
inputs["Task"] = t;
var results = WorkflowInvoker.Invoke(workflow, inputs);
bool IsValid = (bool)results["IsValid"];
```

- Ulagni skup argumenata kao *Dictionary<string, object>*
 - za pojedini argument koji je value tip ili string se može koristiti imenovano svojstvo (npr. *workflow.NazivArgumenta*)
- Povratni skup vrijednosti kao *Dictionary<string, object>*
- Instanca protoka poslova se izvršava na istoj niti
- Pozivatelj čeka na dovršetak instance i osvježi podatke u listama

Asinhrono pozivanje (1)

□ Primjer: WF \ Zadaci \ Form1.cs - RunBookmarkWorkflow

- protok bude instanciran pri stvaranju zadatka i izvršava se dok se zadatak ne dovrši
-  WF \ WFZadaci \ TaskWorkflow.xaml koji koristi opisani WfOnStatusChange.xaml
- Stvara se objekt tipa *WorkflowApplication* i pridružuje mu se postupak (*WfCompleted*) za obradu događaja *Completed* koji predstavlja završetak instance
- Izvršavanje na posebnoj niti (poziv postupka *Run* na objektu tipa *WorkflowApplication*)

```
var workflow = new TaskWorkflow();
var inputs = new Dictionary<string, object>();
inputs["Task"] = t;
...
var wfApp = new WorkflowApplication(workflow, inputs);
wfApp.Completed = WfCompleted;
wfApp.Run();
```

□ Povratne vrijednosti dohvaćaju se obradom događaja završetka

```
private void WfCompleted(WorkflowApplicationCompletedEventArgs args) {
    IDictionary<string, object> results = args.Outputs;
    ...
}
```

Asinhrono pozivanje (2)

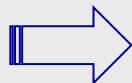
□ Primjer: WF \ Zadaci \ Form1.cs

- Instanca protoka (stvorena kako je prethodno opisano) čuva se da bi se pozvala prilikom promjene postojećeg zadatka

```
Dictionary<int, WorkflowApplication> workflows =  
    new Dictionary<int, WorkflowApplication>();  
  
Dictionary<string, WorkflowApplication> workflowsGuids =  
    new Dictionary<string, WorkflowApplication>();  
  
...  
  
private void RunBookmarkWorkflow(Task t) {  
  
    ...  
  
    workflowsGuids[wfApp.Id.ToString()] = wfApp;
```

□ Svaki zadatak vezan uz pojedinu instancu

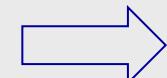
- Uparivanje zadatka i instance obavlja se naknadno primanjem identifikatora novostvorenog zadatka (*TaskId*) od instance (*OnTaskStarted*)
 - *workflows[taskId] = workflowsGuids[workflowInstanceId];*
- Konkretni primjer primanja poruke iz instance i slanja poruka instanci obrađen na slajdu *Implementacija proširenja*



Komunikacija domaćina i instance protoka

(alternative asinkronog poziva)

□ Mehanizam oznaka (bookmarks) i proširenja (extensions)



- Aplikacija šalje poruke instanci protoka poslova korištenjem oznaka
- Instanca protoka poslova šalje poruku domaćinu korištenjem proširenja
- Instancia protoka poslova stvara oznaku i privremeno zaustavlja svoje izvršavanje (ili izvršavanje jedne od paralelnih grana)
- Aplikacija aktivira određenu oznaku

□ Komunikacija putem WCF servisa

- Protok poslova izložen kao servis
- Poziv određene metode WCF servisa
 - aktivira novu instancu
 - nastavlja izvršavanje instance od neke pozicije
- Povratne vrijednosti mogu se vratiti kao rezultat poziva servisa
- Instanca može pozivati neke druge WCF servise

malo kasnije

Kreiranje oznaka

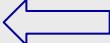
- **Vlastita aktivnost koja nasljeđuje *NativeActivity* ili *NativeActivity<TResult>***
 - Postupak *Execute* stvara oznaku određenog imena
 - (opcionalno) definira postupak koji će se izvršiti aktiviranjem te oznake
 - Svojstvo *CanInduceIdle* postavlja na *true*
- **Nakon stvaranja oznake, aktivnost pasivno čeka na aktiviranje oznake**
- **Primjer:**  WF \ WfZadaci \ WaitForHost.cs

```
public class WaitForHost : NativeActivity<Task> {  
    [RequiredArgument]  
    public InArgument<string> BookmarkName { get; set; }  
    protected override void Execute(NativeActivityContext context) {  
        context.CreateBookmark(  
            BookmarkName.Get(context), OnResumeBookmark);  
    }  
    public void OnResumeBookmark(NativeActivityContext context,  
        Bookmark bookmark, object obj) {  
        Result.Set(context, (Task)obj);  
    }  
    protected override bool CanInduceIdle { get { return true; } }
```

Aktiviranje oznaka

- Aplikacija aktivira oznaku na objektu tipa *WorkflowApplication* pozivom postupka *ResumeBookmark* i slanjem jednog objekta kao parametra:
- *wfApp.ResumeBookmark(naziv oznake, objekt);*

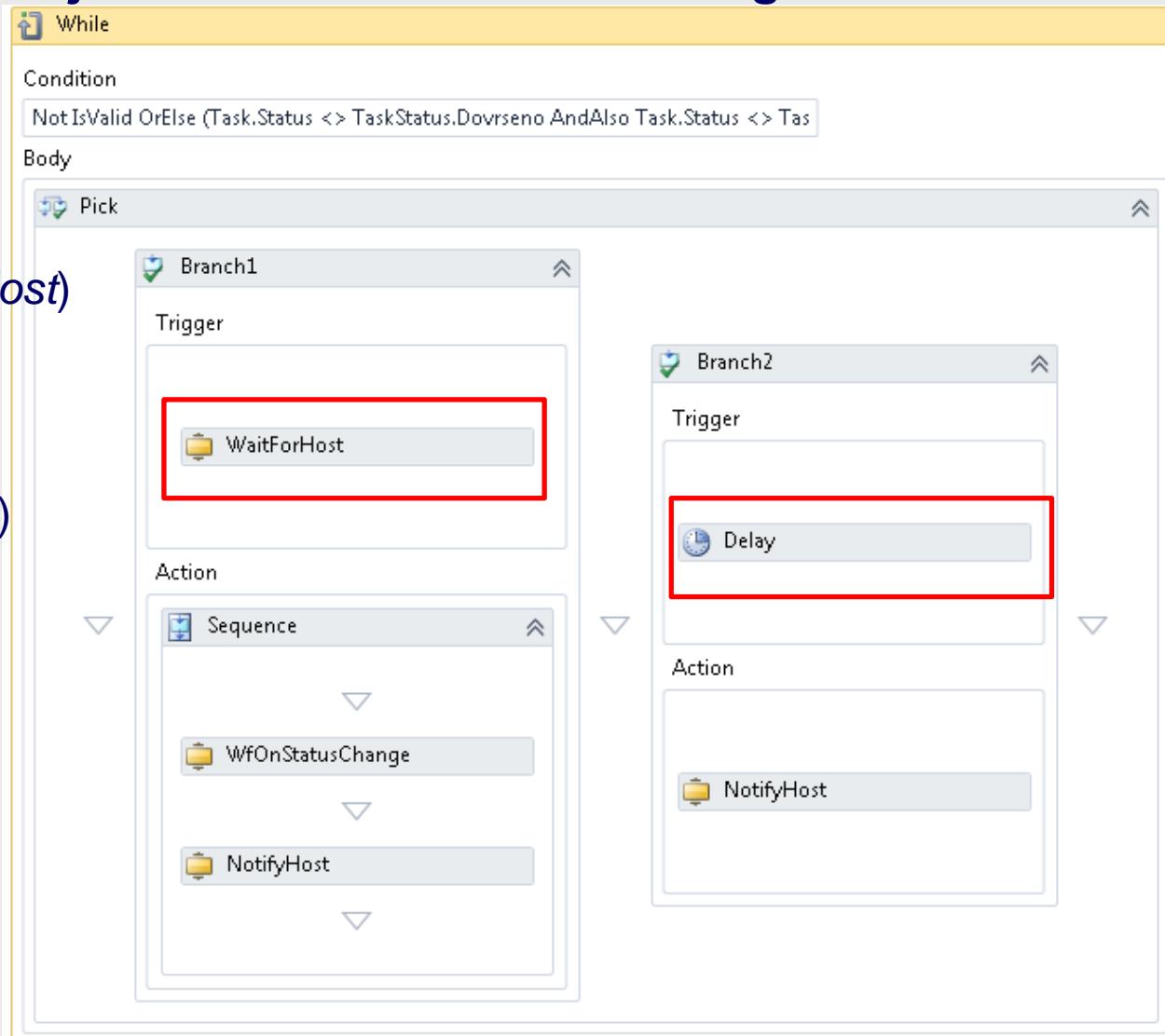
□ Primjer: WF \ Zadaci \ Form1

- Instanca protoka TaskWorkflow kreirana je u *RunBookmarkWorkflow*,
 pogledati "Asinhrono pozivanje (2)"

```
private void ShowEditForm(Task current)
{
    ..
    else if (wfType == WfType.Bookmark) {
        if (f.Selected.TaskId.HasValue // postojeći zadatak
            && workflows.ContainsKey(f.Selected.TaskId.Value)) {
            var wfApp = workflows[f.Selected.TaskId.Value];
            wfApp.ResumeBookmark("Tasks", f.Selected);
        } else { //novi zadatak
            RunBookmarkWorkflow(f.Selected);
        }
    }
}
```

Primjer korištenja oznake (TaskWorkflow.xaml)

- Instanca čeka aktiviranje oznake ili istek vremenskog roka
- Aktivnost *Pick* čeka okidač u jednoj od svojih grana
 - Aktivaciju oznake (aktivnost *WaitForHost*)
 - Protek vremena (aktivnost *Delay* s vrijednošću iz argumenta instance)
- Izvršava se grana za koju se dogodio okidač, a ostale se otkazuju
- Postupak se ponavlja dok se zadatak ne dovrši (*While* aktivnost)



Proširenja

- Omogućuju instanci protoka poslova slanje poruka domaćinskoj aplikaciji
- Početni korak - definira se zajedničko sučelje s postupcima

□ Primjer:  **WF \ WfZadaci \ IHostNotification.cs**

```
public interface IHostNotification {  
    void Notify(string message, Task task);  
    void NotifyTaskStarted(string workflowInstanceId, int taskId);  
    void NotifyTaskChanged(int taskId);  
    void NotifyDelayPassed(int taskId);  
}
```

Implementacija proširenja (1)

□ ... zatim se na strani domaćina vrši implementacija

- Primjer: WF \ Zadaci \ HostEventNotifier.cs

```
public class HostEventNotifier : IHostNotification {  
    ...
```

□ Praktično je u implementaciji umjesto konkretnе obrade poruke podignuti događaj na koji će se aplikacija domaćin pretplatiti

- Događaji definirani u WF \ Zadaci \ DelegatesAndEnums.cs
- Primjer: WF \ Zadaci \ HostEventNotifier.cs

□ Implementacija **NotifyTaskStarted** u posebnoj niti podiže **TaskStarted**

...

```
...  
    public event TaskStartedDelegate TaskStarted;  
  
...  
    public void NotifyTaskStarted(string workflowInstanceId, int taskId) {  
        if (TaskStarted != null) {  
            ThreadPool.QueueUserWorkItem((state) =>  
                TaskStarted(workflowInstanceId, taskId), null);  
        }  
    }
```

Implementacija proširenja (2)

□ ... na koji se forma pretplati

- Primjer: WF \ Zadaci \ Form1

```
private void RunBookmarkWorkflow(Task t) {  
    ...  
    HostEventNotifier extension = new HostEventNotifier();  
    extension.TaskStarted += OnTaskStarted;  
    ...  
}
```

□ ... i obavi osvježavanje liste te uparivanje zadatka i instance

```
void OnTaskStarted(string workflowInstanceId, int taskId) {  
    if (InvokeRequired) {  
        TaskStartedDelegate del = OnTaskStarted;  
        this.BeginInvoke(del, workflowInstanceId, taskId);  
    }  
    else{  
        workflows[taskId] = workflowsGuids[workflowInstanceId];  
        LoadData();  
    }  
}
```

Aktiviranje proširenja

- Prilikom stvaranja objekta tipa **WorkflowApplication** definiraju se podržana proširenja
 - Primjer:  WF \ Zadaci \ Form1 - RunBookmarkWorkflow

```
var wfApp = new WorkflowApplication(workflow, inputs);  
wfApp.Completed = WfCompleted;  
HostEventNotifier extension = new HostEventNotifier();  
extension.TaskStarted += OnTaskStarted;  
...  
wfApp.Extensions.Add(extension);  
wfApp.Run();
```

Korištenje proširenja iz instance

- Na konkretnom kontekstu u kojem se instanca izvršava traži se proširenje vezano uz određeni tip
 - Postupak GetExtension<T>()
 - Tip T može biti konkretna implementacijska klasa, apstraktna klasa ili sučelje
 - Mora se jednoznačno moći identificirati potrebno proširenje
- Primjer:  WF \ WfZadaci \ NotifyHost.cs

```
public sealed class NotifyHost : CodeActivity
{
    public InArgument<Task> Task { get; set; }
    protected override void Execute(CodeActivityContext context)
    {
        IHostNotification extension =
            context.GetExtension<IHostNotification>();
        Task task = Task.Get(context);
        extension.NotifyTaskChanged(task.TaskId.Value);
        ...
    }
}
```

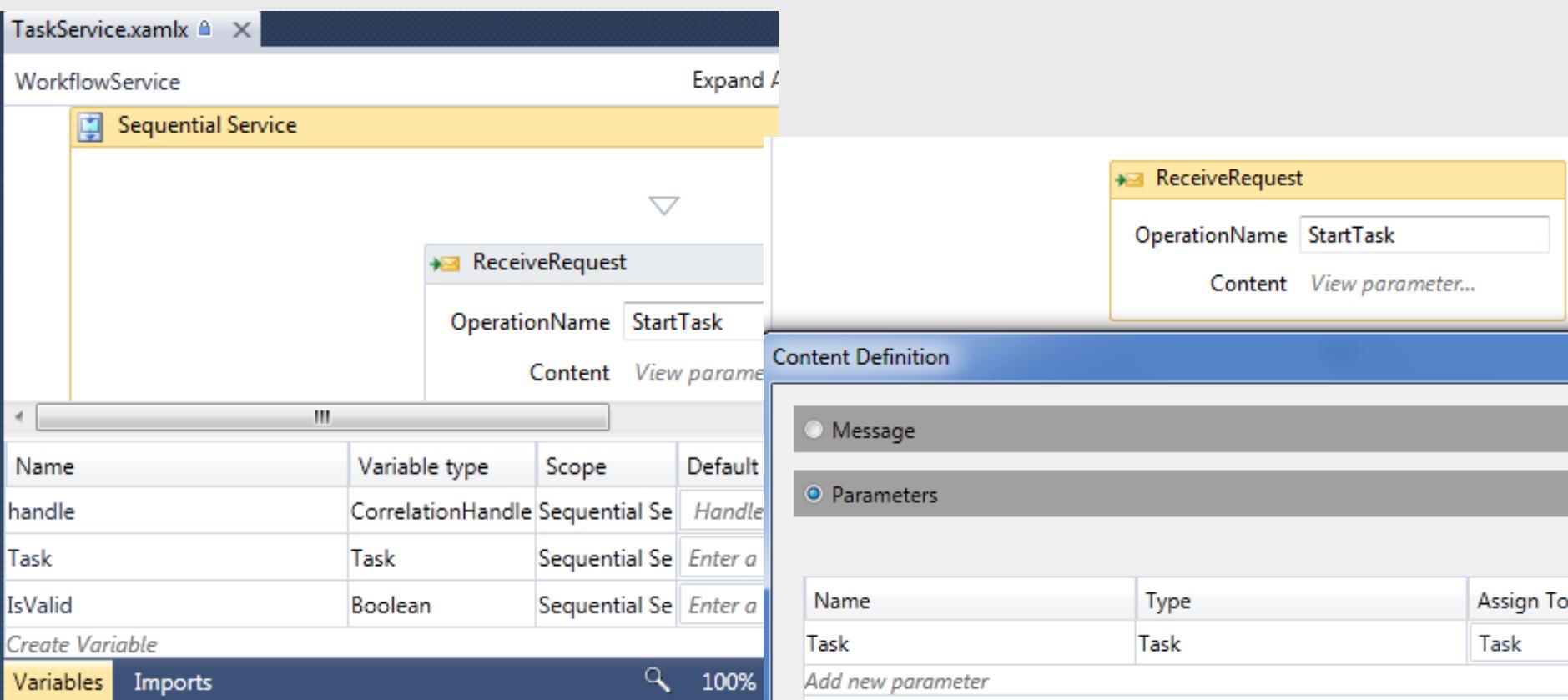
Workflow servisi

- **Primjer:**  **WF \ WFSERVICE \ TaskService.xamlx**
 - Ekstenzija xamlx – protok poslova izložen kao WCF servis
- **Prva aktivnost mora biti tipa *Receive***
 - Svojstvo *CanCreateInstance* mora biti postavljeno na *True*
 - *OperationName* definira naziv WCF postupka
 - *ServiceContractName* definira naziv WCF servisa
 - Parametri servisa podešavaju se dvoklikom na okvir *Content*
 - naziv ulaznog parametra, tip i varijabla kojoj se ulazni argument pridružuje
- **Može se koristiti i predložak *ReceiveAndSendReply***
 - Njemu se automatski dodaje i aktivnost *Send* za slanje odgovora
 - Parametri mu se podešavaju kao onom tipa *Receive*
- **Prepostavljeni tip povezivanja je *BasicHttpBinding***
 - Za drugačije načine izlaganja potrebno je promijeniti datoteku web.config
- **Klijentu se dodaje referenca na web servis (*Add Service Reference*)**
 - Automatski se stvaraju razredi za poziv servisa i unose postavke u app.config
 - *npr. \$|Service References\TaskLoadServiceReference**

Primjer definiranja ulaznih argumenata servisa

□ Primjer: WF \ WFSERVICE \ TaskService.xamlx

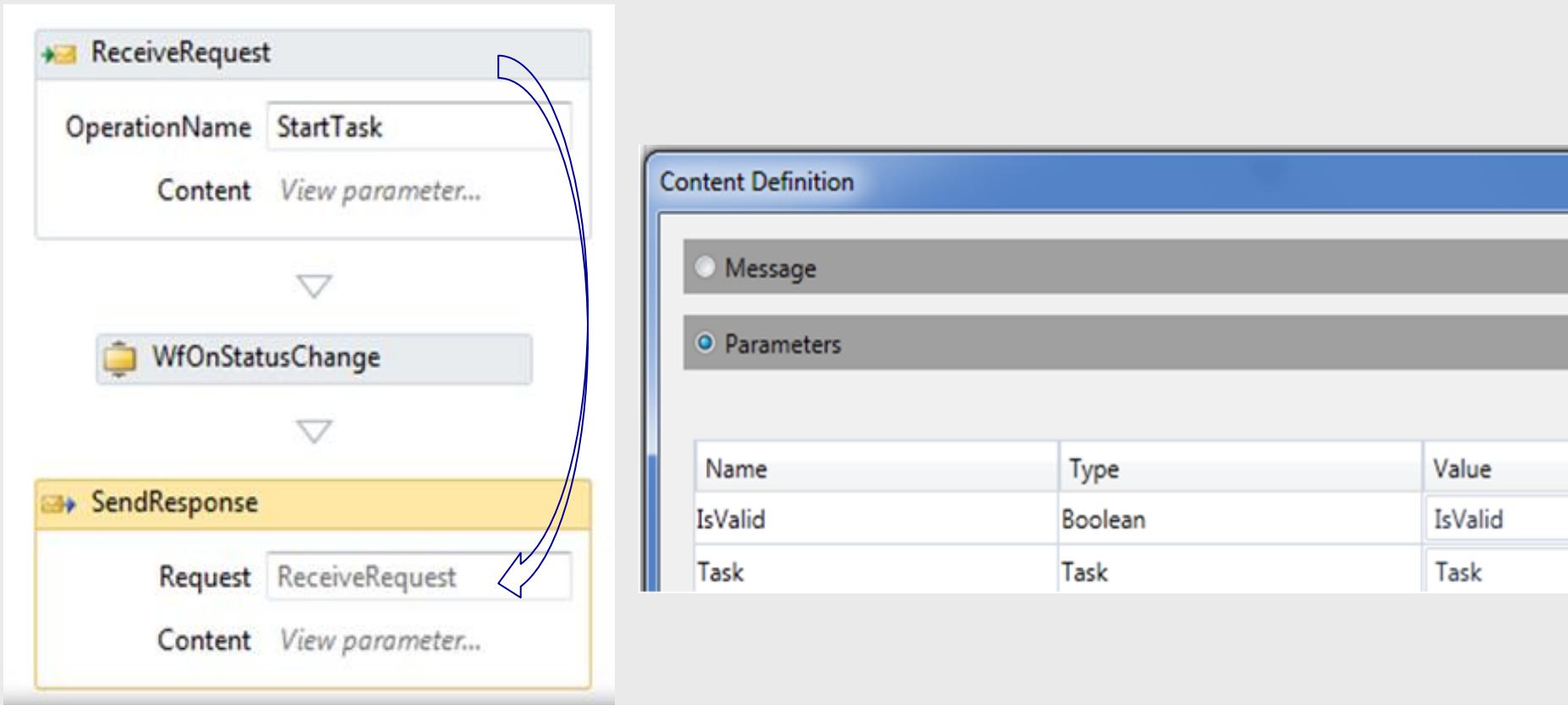
- Unutar protoka definirana varijabla *Task*
- Ulazni argument tipa *Task* nazvan *Task*, pri pozivu pridružen varijabli *Task*
 - pr. *ReceiveRequest – Content – View parameter...*



Primjer definiranja povratnih argumenata servisa

□ Primjer: WF \ WFSERVICE \ TaskService.xamlx

- Aktivnost *Send* povezana s konkretnom aktivnosti *Receive*
- Povratnim argumentima *IsValid* i *Task* pridružene vrijednosti varijabli *IsValid* i *Task*



Primjer poziva *workflow* servisa

- Prilikom dodavanja novog zadatka poziva se postupak **StartTask**
 - Koriste se razredi nastali dodavanjem reference na servis
- Primjer:  **WF \ Zadaci \ Form1.cs**
 - Task je ujedno ulazni i izlazni argument pa je stvoren postupak s *ref* parametrom
 - Identifikator novog zadatka vraćen preko *ref* parametra koristit će se naknadno za usmjeravanje poziva *ChangeTask* na konkretnu instancu protoka poslova

```
private void RunWorkflowService(Task t) {  
    bool isValid = false;  
    if (t.TaskId.HasValue) { //postojeći zadatak  
        isValid = new  
            WorkflowServiceReference.ServiceClient().ChangeTask(ref t);  
    }  
    else { //novi zadatak  
        isValid = new  
            WorkflowServiceReference.ServiceClient().StartTask(ref t);  
    }  
    if (isValid)  
        LoadData();  
    ...  
}
```

Workflow servisi s više pristupnih točaka

□ Jedan **workflow** može sadržavati više aktivnosti *Receive*

- Svaka od aktivnosti definira naziv postupka za koji je vezana
- Barem jedna od *Receive* aktivnosti mora imati postavljeno svojstvo *CanCreateInstance* na *True*
- *Receive* aktivnosti koje su vezane za postupke koji trebaju postojeću instancu svojstvo *CanCreateInstance* moraju imati *False*

□ Pozivom određenog postupka WCF servisa aktivira se odgovarajuća *Receive* aktivnost

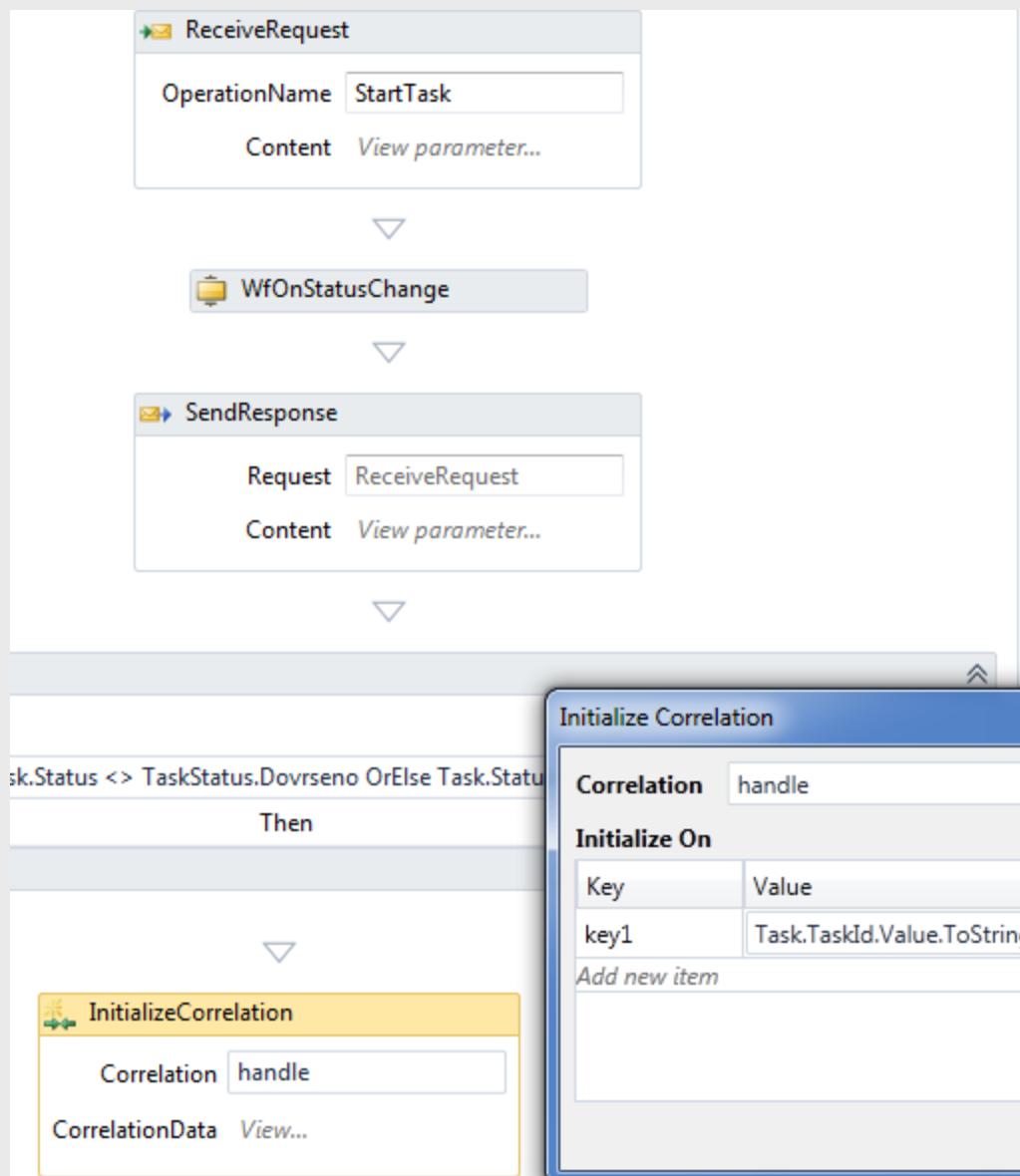
- Ako se ne radi o zahtjevu za početak nove instance, postojeća se pronalazi na osnovu korelacijskih mehanizama
- **Korelacija na osnovu sadržaja** (Content-Based Correlation)
 - Za određivanje konkretnе instance koristi se jedan ili više ulaznih parametara pozvanog postupka
- **Korelacija na osnovu konteksta** (Context-Based Correlation)
 - Koristi se tip povezivanja koji podržava razmjenu konteksta (*WSHttpContextBinding*, *NetTcpContextBinding*, *BasicHttpContextBinding*)



Korelacija na osnovu sadržaja (1)

□ Primjer: WF \ WFSERVICE – TaskService.xamlx

- U workflowu definirana varijabla *handle* tipa *CorrelationHandle*
- Prva aktivnost *Receive* s *CanCreateInstance = True*
- Pozivatelju se kao odgovor na poziv *StartTask* šalje se objekt novostvorenog zadatka (*Task*)



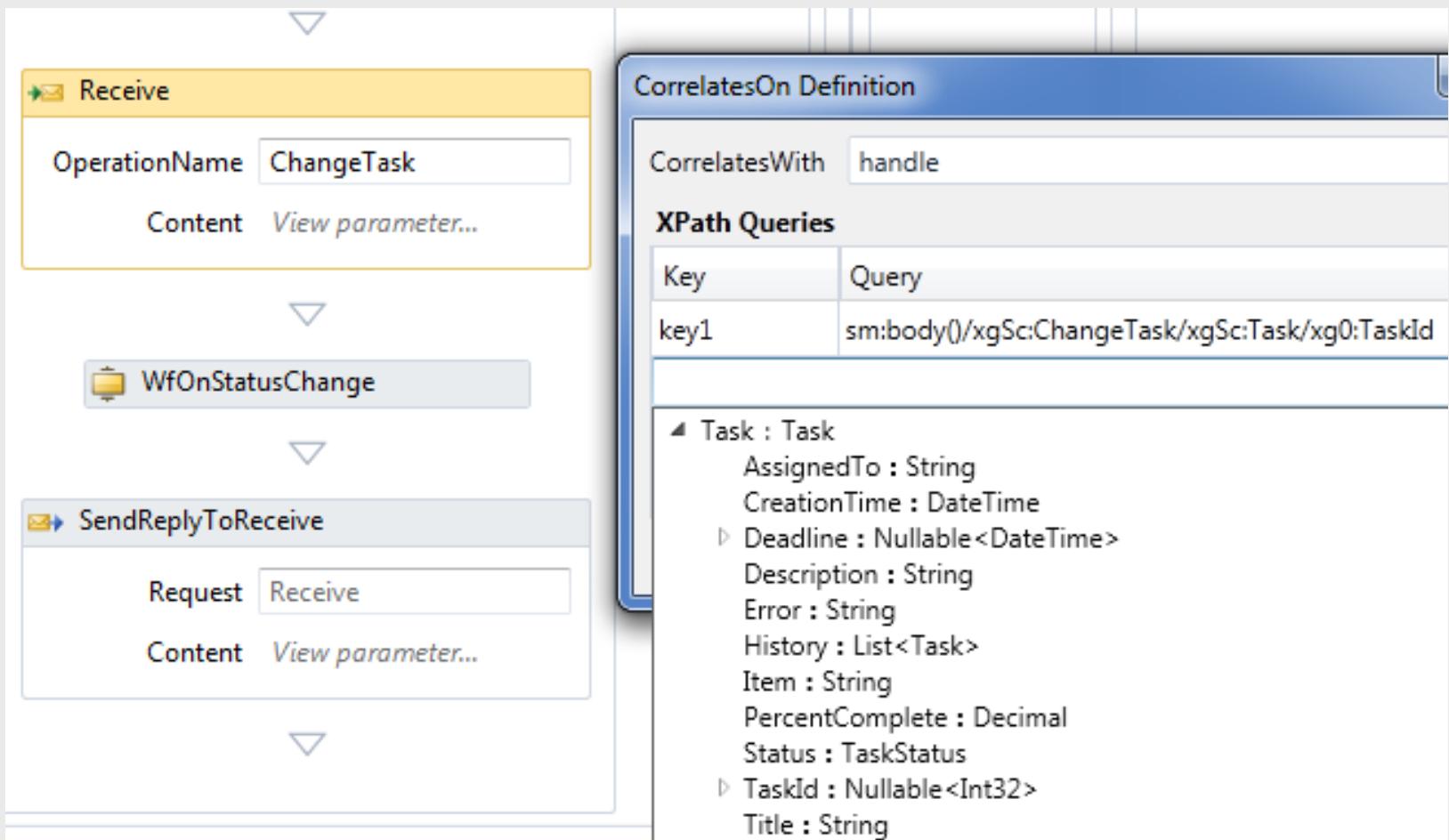
□ Slijedi inicijalizacija korelacije pomoću aktivnosti *InitializeCorrelation*

- Koristi se korelacijska varijabla *handle*
- Za *CorrelationData* odabire se *TaskId*

Korelacija na osnovu sadržaja (2)

□ Primjer: WF \ WFSERVICE – TaskService.xamlx

- Sljedećoj Receive aktivnosti svojstvo *CorrelatesOn* postavljeno na varijablu *handle*
- Za vrijednost odabran parametar *TaskId* iz padajuće liste



Korelacija na osnovu sadržaja (3)

- Prilikom promjene zadatka poziva se postupak *ChangeTask*
- Poslani zadatak sadrži *TaskId* i na osnovu te vrijednosti dolazi do uparivanja s konkretnom instancom protoka poslova
 - Npr. kod stvaranja zadatka poziva se postupak StartTask
 - Protok u svoje spremište (npr. TaskDB) spremi zadatak i dodijeli npr. TaskId = 5
 - Cijeli Task bude vraćen kao rezultat poziva servisa
 - Nakon toga instance protoka nastavlja s radom i inicijalizira korelacijsku varijablu na 5
- Klijent osvježi podatke s *TaskId = 5*
 - pri ažuriranju zadatka poziva se postupak *ChangeTask* i šalje zadatak
 - iz parametra se izdvoji vrijednost *TaskId = 5* i poziv se preusmjeri onoj instanci koja je postavila korelacijsku varijablu na vrijednost 5
- Primjer:  WF \ Zadaci \ Form1.cs

```
private void RunWorkflowService(Task t)    {  
    if (t.TaskId.HasValue) //postojeći zadatak  
        isValid = new  
            WorkflowServiceReference.ServiceClient().ChangeTask(ref t);  
    else //novi zadatak  
        isValid = new  
            WorkflowServiceReference.ServiceClient().StartTask(ref t);  
}
```

Materijali i reference

□ Knjige

- Wil Van der Aalst, Kees Max van Hee. Workflow management. Models, Methods, and Systems. Cambridge, Massachusetts, SAD: MIT Press, 2004. [Aalst2004]
- Sharp, A.; McDermott, P. Tools for Process Improvement and Application Development, Second Edition. Boston, SAD: Artech House, 2008. [Sharp2008]

□ Windows Workflow Foundation

- <http://msdn.microsoft.com/en-us/netframework/aa663328>

□ WMC Terminology & Glossary (Document No. WFMC-TC-1011)

- <http://www.wfmc.org/Download-document/WFMC-TC-1011-Ver-3-Terminology-and-Glossary-English.html>

□ Workflow Patterns - Control-Flow Patterns

- <http://www.workflowpatterns.com/patterns/control/>

□ Business Process Modeling Notation

- <http://www.bpmn.org/>

□ XML Process Definition Language (XPDL)

- <http://www.wfmc.org/xpdl.html>

□ Sustav za automatizirano upravljanje poslovnim procesima

- <http://www.zpr.fer.hr/zpr/projekti/susap>

Izrada sustava

2012/13.09



Sadržaj

Faza izrade

- aktivnosti
- kodiranje i programiranje
- ugovor
- specifikacija postupaka
- pristup programiranju
- očuvanje kvalitete koda

Refaktoriranje

- karakteristike
- tehnike
- reprezentativni predlošci
- preporuke

Testiranje

- ključni pojmovi
- plan testiranja
- stupnjevi testiranja
- razvoj vođen testiranjem
- otklanjanje pogrešaka

Izrada dokumentacije

- vrste dokumentacije
- preporuke

Primjena

- vrste konverzije
- poduka

Održavanje

- sistemska potpora
- odjel pomoći
- vrste održavanja

Faza izrade sustava

□ Implementacija sustava, ugradnja sustava

- faza u kojoj se obavlja izrada novog sustava i isporuka tog sustava u produkciju, to jest svakodnevnu primjenu
- drugi nazivi: konstrukcija, izvedba, provedba

□ Aktivnosti izrade (programiranje, testiranje, dokumentiranje)

- izgradnja i provjera mreža (po potrebi)
- ugradnja i provjera baze podataka
 - kreiranje baze podataka,
 - transfer i generiranje probnih podataka,
 - testiranje operacija nad podacima
- instalacija i provjera nabavljenih softverskih paketa (po potrebi)
- pisanje nove programske podrške i provjera novih programa
 - provodi se prema detaljnem planu programiranja
 - prethodno se stvara izvedbena ekipa i pridjeljuju odgovornosti članovima
- dokumentiranje sustava - izrada tehničke i korisničke dokumentacije

Kodiranje, programiranje

□ Programiranje

- proces pisanja (kodiranja), provjere (testiranja), ispravljanja pogrešaka (debugiranja) te održavanja izvornog programskog koda

□ Kodiranje (iako je sinonim za programiranje, činimo razliku)

- pretvorba detaljnog opisa programa u stvarni program, najčešće pisanje izvornog koda nekog formalnog programskog jezika
- ručno kodiranje
 - neizbjježno zbog veličine stvarnih problema i složenosti procesa
 - sporo i dugotrajno → primjena jezika vrlo visoke razine
 - jezici četvrte generacije (4GL – Fourth Generation Language)
 - objektno zasnovani jezici (Object Based Language)
 - objektno usmjereni jezici (Object Oriented Language)
- automatsko kodiranje
 - generiranje programskog koda, sučelja, sheme baze podataka, ...

Ugovor (Contract Form)

Ugovori dokumentiraju razmjenu poruka između objekata

- Tehnički gledano, ugovor treba napraviti za svaku poruku koja se šalje i prima za pojedini objekt, za svaku interakciju
- U praksi, ugovor se piše za svaku metodu koja prima poruke drugih objekata

Primjer:

Method Name:	Class Name:	ID:
Clients (Consumers):		
Associated Use Cases:		
Description of Responsibilities:		
Arguments Received:		
Type of Value Returned:		
Pre-Conditions::		
Post-Conditions:		

Specifikacija postupaka (Method Specification)

- Dokument s eksplisitnim instrukcijama kako napisati kod metode
 - treba biti jasna i lako razumljiva
 - pišu analitičari i prosljeđuju programerima (?!)

Method Name:	Class Name:	ID:
Contract ID:	Programmer:	Date Due:
Programming Language:		
Triggers/Events:		
Arguments Received: Data Type:	Notes:	
Messages Sent & Arguments Passed: ClassName.MethodName:	Data Type:	Notes:
Argument Returned: Data Type:	Notes:	
Algorithm Specification: (pseudocode)		
Misc. Notes:		

Pristup programiranju

□ Monolitni pristup (build and fix)

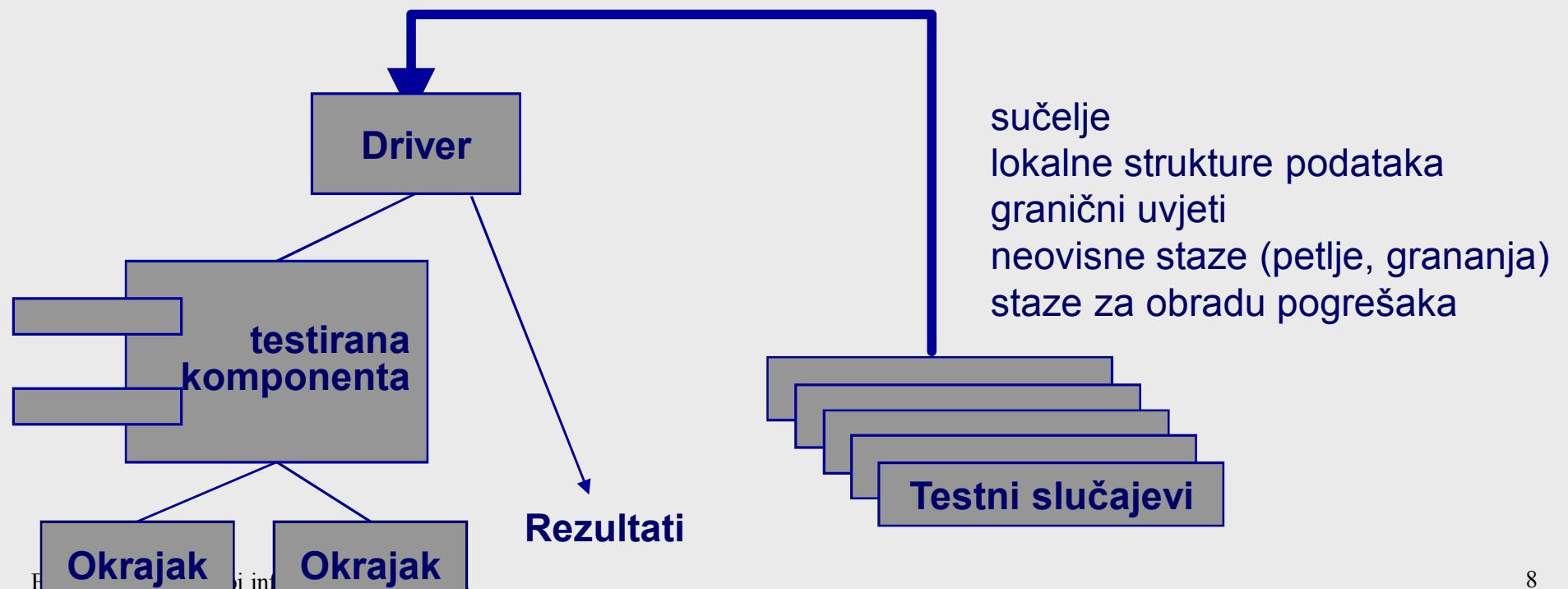
- dugotrajno *kodiranje*, a zatim niz ponavljanja oblika *provjera+ispravak*
- odgađa otkrivanje problema (pogrešaka u kodu i dizajnu)
- prosljeđuje probleme u primjenu i održavanje

□ Inkrementalni pristup (stupnjevito, postupno programiranje)

- niz ponavljanja oblika *kodiranje+provjera+ispravak*
- omogućuje raniju provjeru i izdvajanje pogrešaka (fault isolation)
- omogućuje raniju raspoloživost djelomičnih (nedovršenih) verzija
- omogućuje ravnomjerniju podjelu posla

Inkrementalno programiranje

- Postupno kodiranje i udruživanje razreda i njihovih postupaka
- Odrezak, okrajak (stub)
 - prilikom izrade funkcije koja poziva neke druge funkcije, pozvane funkcije kodiraju se kao odresci ili okrajci (stub), tako da je tijelo funkcije sadrži poruku ("Neimplementirana funkcija X") ili hardkodiranu povratnu vrijednost
- Pogonitelj, pokretač (driver)
 - prilikom izrade funkcije koja će biti pozvana iz neke druge, još neugrađene funkcije, izrađuje se pogonska funkcija (driver)



Očuvanje kvalitete programskog koda

- **Što veće unutarnje prianjanje modula – kohezija (cohesion)**
 - potrebna visoka unutarnja povezanost elemenata
 - svaki modul treba obavljati jednu i samo jednu funkciju
 - postizanje ponovne upotrebljivosti u budućim programima
- **Što manja vanjska zavisnost modula – kopčanje (coupling)**
 - moduli trebaju biti minimalno međusobno zavisni
 - minimizacija utjecaja promjene jednog modula na druge module
- **Iako se navedeni kriteriji pripisuju dizajnu, provode se pri izradi !**

Refaktoriranje

Osnovni pojmovi

□ Refaktoriranje (refactoring)

- promjena interne strukture programske podrške da bi ju se bolje razumjelo i lakše održavalo, uz očuvanje vanjskog ponašanja (Fowler 1999)
- jedna od osnovnih praksi agilnog programiranja
 - karakteristika ekstremnog programiranja je agresivno refaktoriranje
 - proces programiranja sastoji se od niza koraka *kodiraj+refaktoriraj*

□ Primjena - na sve aspekte programske podrške (dokumentacija, programski kod, modeli, testovi, ...)

- programi koji nisu pisani OO jezikom – teže reorganizirati jer su tokovi podataka i kontrolni tokovi čvrsto povezani
- OO programi – također teško radi nasljeđivanja i polimorfizma
- refaktoriranje UML modela - paziti na konzistentnost između različitih modela, kao i između modela i programskega koda, ili dokumentacije

Karakteristike refaktoriranja

□ Lehmanovi zakoni evolucije programske podrške

- povećanje funkcionalnosti razmjerno je smanjenju kvalitete i povećanju složenosti

□ Prednosti refaktoriranja

- sprječava narušavanje strukture programskog koda
 - ako se obavi nakon svake promjene koda, struktura ostaje očuvana, što olakšava buduće promjene
- povećanje razumljivosti i čitljivosti programskog koda
- olakšano otkrivanje bugova
- povećanje produktivnosti
 - kvalitetan, razumljiv, nebugovit kod lakše je mijenjati
 - ušteda vremena znatno veća od vremena utrošenog na refaktoriranje
- smanjenje troškova održavanja pojednostavljenjem dizajna

□ Nedostaci refaktoriranja

- pretjerana primjena smanjuje produktivnost i svodi se na puko refaktoriranje
- neautomatizirano refaktoriranje može biti dugotrajno i mukotrpno
- postojeći alati nisu dovoljno zreli i pouzdani – nepovjerenje programera

Tehnike refaktoriranja

- Obrasci (patterns) refaktoriranja – tzv. *refactorings*
- Composing methods
 - Extract method, inline method, inline temp, replace temp with query, introduce explaining variable, split temporary variable, remove assignments to parameters, ...
- Moving features between objects
 - Move method, move field, extract class, inline class, hide delegate, remove middle man, introduce foreign method, introduce local extension
- Organizing data
 - Self encapsulate field, replace data value with object, change value to reference, replace array with object, encapsulate field, replace subclass with fields, ...
- Simplifying conditional expression
 - Decompose conditional, remove control flag, replace conditional with polymorphism, replace nested conditional with guard, ...
- Making method calls simpler
 - Rename method, parameterize method, remove parameter
- Dealing with generalization
 - Pull up field, pull up method, push down method, extract subclass, extract interface, ...

Reprezentativni predlošci refaktoriranja

- **Rename method**
 - Davanje novog, razumljivijeg imena metodi
- **Extract method**
 - Dio koda se izdvaja u posebnu metodu
- **Replace temp with query**
 - Zamjena varijable koja poprima vrijednost nekog izraza s pozivom metode
- **Move method/field**
 - Metoda/članska varijabla se iz jednog razreda prebacuje u drugi razred
- **Extract class – inline class**
 - Razred koji “radi puno toga” dijeli su više razreda – *extract class*
 - obrnuto – *inline class*
- **Replace conditional with polymorphism**
 - Zamjena uvjeta (*switch*) koji ispituje tip objekta s polimorfizmom, tako što se originalna metoda učini apstraktnom i u podrazredima se primjeni *overriding*
- **Replace type code with state/strategy**
 - Postoji kod koji utječe na ponašanje a ne može se primijeniti *subclassing*

Primjer refaktoriranja

□ Proces refaktoriranja i korišteni uzorci refaktoriranja

- demonstrirani na jednostavnom primjeru evidencije u videoteci
- u realnom projektu se refaktoriranje na ovako kratkom programu ne bi isplatilo, jer bi trud uložen u refaktoriranje znatno nadmašivao korist

□ Primjer iz *Refactoring: Improving the Design of Existing Code*

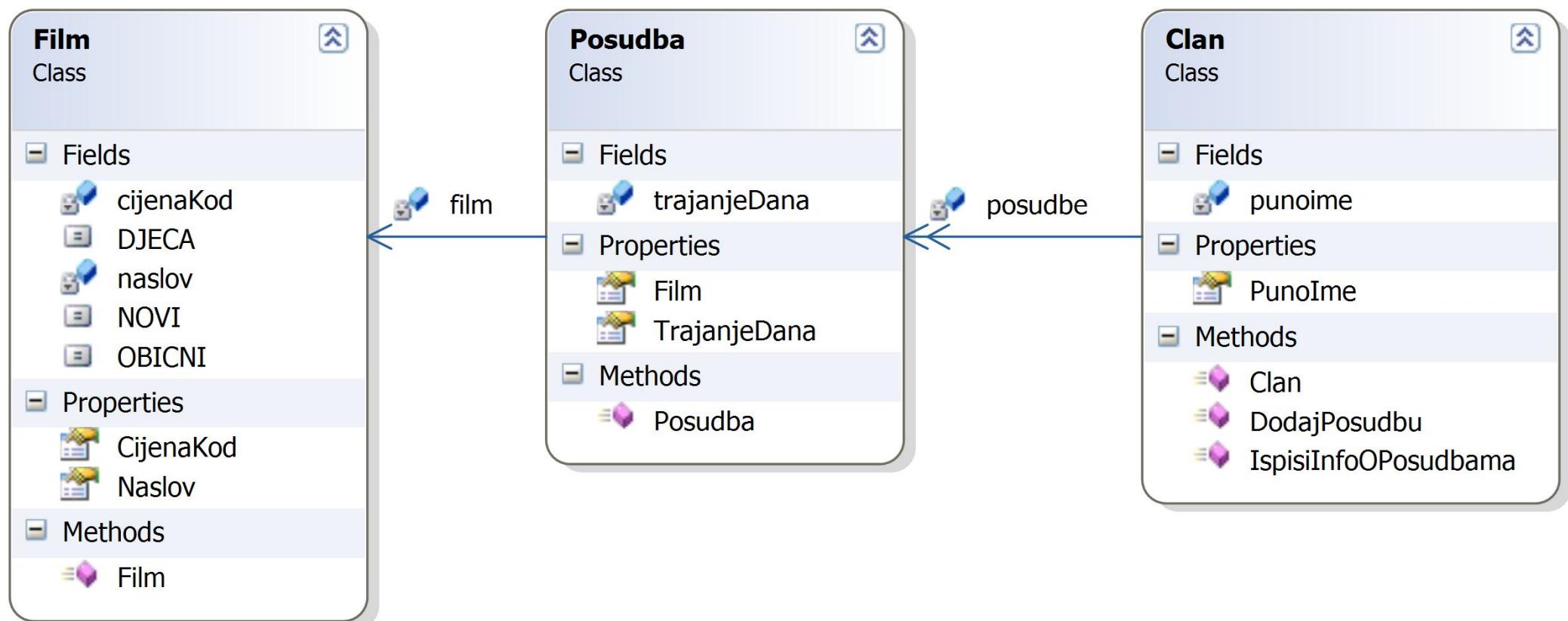
- Ispravno refaktoriranje prate odgovarajući testovi koji su ovdje izostavljeni.
- (o testiranju malo kasnije)

□ Aplikacija

- program evidentira koji je korisnik posudio koji film i na kako dugo
- računa se iznos za naplatu ovisno o tipu filma i trajanju posudbe
- postoje tri tipa filmova: obični, dječji i nova izdanja
- za filme se usput računa učestalost posuđivanja

Početni dizajn

□ Primjer: Izrada \ Refaktoriranje – R0



Razred Film

```
class Film
{
    public const int OBICNI = 0;
    public const int NOVI = 1;
    public const int DJECA = 2;

    private string naslov;
    private int cijenaKod; // OBICNI, NOVI, DJECA

    public Film(string naslov, int cijenaKod)
    {
    ...
    }

    public string Naslov
    {
    ...
    }

    public int CijenaKod
    {
    ...
    }
}
```

Postupak *Clan.IspisiInfoOPosudbama* (1)

```
public string IspisiInfoOPosudbama()
{
    double ukupniIznos = 0;
    int frekvencijaPosudbiBodovi = 0;
    string ispis = "Posudbe za člana " + this.punoime + "\n";

    foreach (Posudba p in this.posudbe)
    {
        double trenutniIznos = 0;

        switch (p.Film.CijenaKod)
        {
            case Film.OBICNI:
                trenutniIznos += 2;
                if (p.TrajanjeDana > 2)
                    trenutniIznos += (p.TrajanjeDana - 2) * 1.5;
                break;
            ...
        }
    }
}
```

Postupak *Clan.IspisInfoOPosudbama* (2)

```
...  
    case Film.NOVI:  
        trenutniIznos += p.TrajanjeDana * 3;  
        break;  
    case Film.DJECA:  
        trenutniIznos += 1.5;  
        if (p.TrajanjeDana > 3)  
            trenutniIznos += (p.TrajanjeDana - 3) * 1.5;  
        break;  
    } // switch  
  
    frekvencijaPosudbiBodovi++;  
  
    // bonus ako je novi film i posuđen barem 2 dana  
    if ((p.Film.CijenaKod==Film.NOVI) && p.TrajanjeDana > 1)  
        frekvencijaPosudbiBodovi++;  
  
...
```

Postupak *Clan.IspisiInfoOPosudbama* (3)

```
...  
  
    //za ispis informacija o posudbi  
    ispis += "\t" + p.Film.Naslov + "\t" +  
            trenutniIznos.ToString("c") + "\n";  
    ukupniIznos += trenutniIznos;  
}  
  
//za ispis footer-a  
ispis += "Iznos dugovanja: "  
        + ukupniIznos.ToString("c") + "\n";  
ispis += "Bodovi zbog frekventnog posudjivanja: "  
        + frekvencijaPosudbiBodovi.ToString();  
  
return ispis;  
} // ispisi info o posudbama
```

Extract Method

- *IspisInfoOPosudbama* je preduga i obavlja više posla nego što bi trebala

□ Problem

- Dodatan zahtjev na ispis informacija o posudbama u HTML formatu, a kasnije možda i u drugim formatima (XML, CSV, ...)

□ Rješenja

- Kopiranje koda u novu metodu i prilagodba formata – zalihost koda
- Uvođenje argumenta-skretnice – komplikiranje ionako prevelike metode
- Izolacija neutralnog, opće primjenjivog koda u novu metodu a zatim ...
 - koji bi to dio koda bio ?

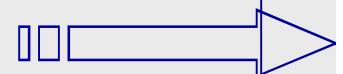
□ Ekstrakcija postupka

- Dio koda koji se može izdvojiti u novu metodu je *switch* dio
- Varijabla *p* se u tom dijelu ne mijenja i nju možemo predati kao parametar
- Vrijednost varijable *trenutnilznos* se mijenja
 - tu ćemo varijablu zato vraćati kao povratnu vrijednost

Stara i nova metoda nakon ekstrakcije

- Primjer: Izsada \ Refaktiranje – R1

```
public string IspisiInfoOPosudbama ()  
{  
    ...  
  
    foreach (Posudba p in this.posudbe)  
    {  
        double trenutniIznos = 0;  
  
        trenutniIznos = IzracunajIznosPosudbe (p) ;  
        ...  
    }  
}
```

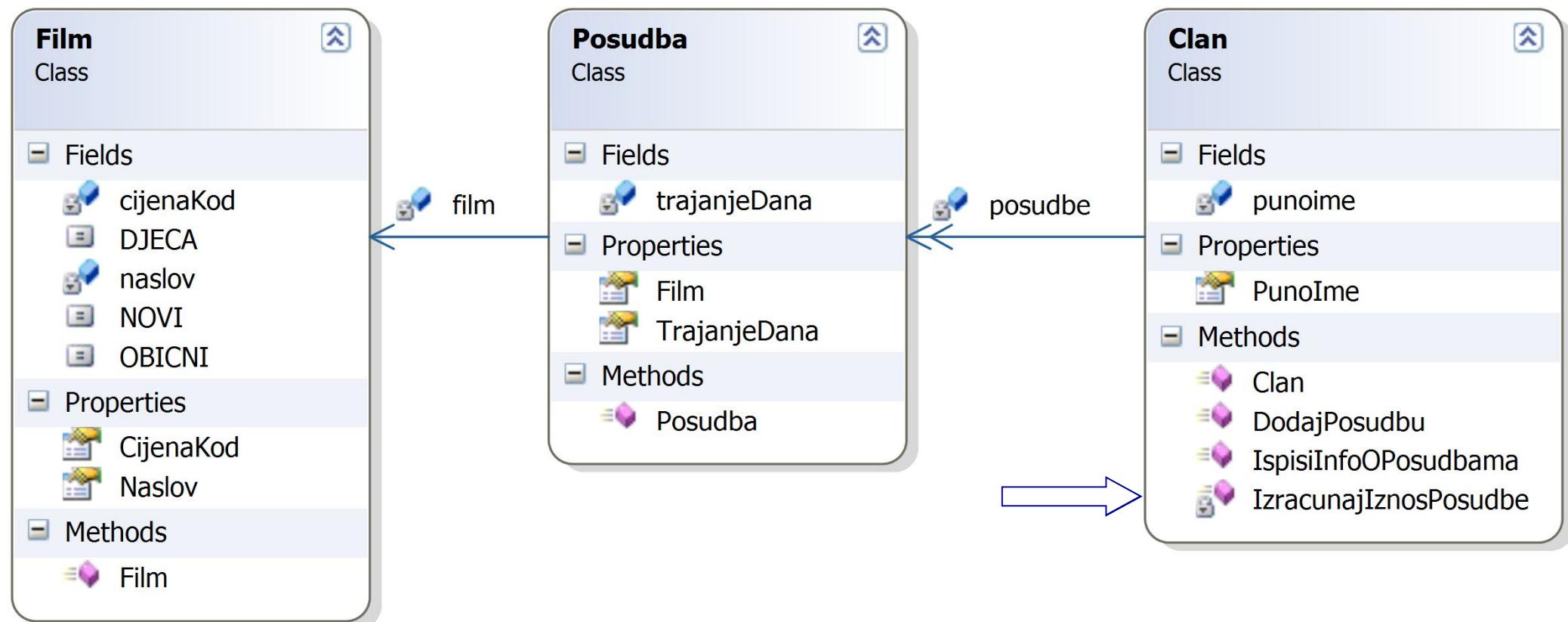


```
private double IzracunajIznosPosudbe(Posudba p)
{
    double trenutniIznos = 0;

    switch (p.Film.CijenaKod)
    {
        case Film.OBICNI:
            trenutniIznos += 2;
            if (p.TrajanjeDana > 2)
                trenutniIznos += (p.TrajanjeDana - 2) * 1.5;
            break;
        case Film.NOVI:
            trenutniIznos += p.TrajanjeDana * 3;
            break;
        case Film.DJECA:
            trenutniIznos += 1.5;
            if (p.TrajanjeDana > 3)
                trenutniIznos += (p.TrajanjeDana - 3) * 1.5;
            break;
    }
    return trenutniIznos;
}
```

Model nakon ekstrakcije

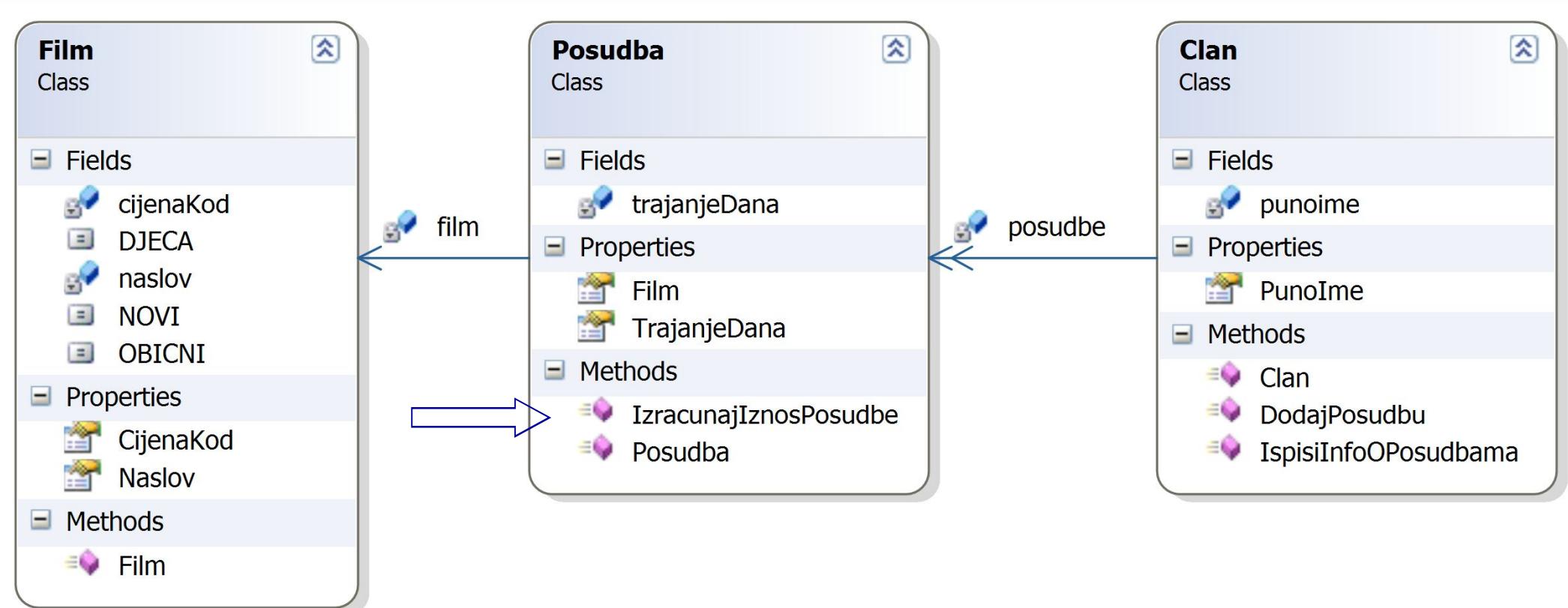
□ Primjer: Izrada \ Refaktoriranje – R1



- **IzracunajIznosPosudbe** metoda klase **Clan** bavi se samo posudbama
 - Metoda se nalazi u krivom razredu te ju treba prebaciti u razred *Posudba*

Move Method

- Metoda *IzracunajIznosPosudbe* se premješta u drugi razred
 - gubi argument *p*, koji zamjenjuje referenca *this*
- Osim toga potrebno je promijeniti sve reference (pozive) metode
 - promijeniti poziv *IzracunajIznosPosudbe* u metodi *IspisiInfoOPosudbama*
- Primjer: Izrada \ Refaktoriranje – R2



Premještena metoda *IzracunajIznosPosudbe*

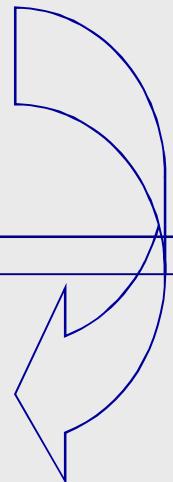
```
public double IzracunajIznosPosudbe()
{
    double trenutniIznos = 0;

    switch (this.Film.CijenaKod)
    {
        case Film.OBICNI:
            trenutniIznos += 2;
            if (this.TrajanjeDana > 2)
                trenutniIznos += (this.TrajanjeDana - 2) * 1.5;
            break;
        case Film.NOVI:
            trenutniIznos += this.TrajanjeDana * 3;
            break;
        case Film.DJECA:
            trenutniIznos += 1.5;
            if (this.TrajanjeDana > 3)
                trenutniIznos += (this.TrajanjeDana - 3) * 1.5;
            break;
    }
    return trenutniIznos;
}
```



Promjena poziva u *IspisiInfoOPosudbama*

```
public string IspisiInfoOPosudbama ()  
{  
    ...  
  
    foreach (Posudba p in this.posudbe)  
    {  
        double trenutniIznos = 0;  
  
        trenutniIznos = IzracunajIznosPosudbe (p) ;  
    }  
}
```



```
public string IspisiInfoOPosudbama ()  
{  
    ...  
  
    foreach (Posudba p in this.posudbe)  
    {  
        double trenutniIznos = 0;  
  
        trenutniIznos = p.IzracunajIznosPosudbe () ;  
    }  
}
```

Replace Temp with Query

- Varijabla **trenutnilznos** u **IspisiInfoOPosudbama** je redundantna
 - zalihost uklanjamo tehnikom Replace Temp with Query

- Primjer:  Izrada \ Refaktoriranje – R2

```
public string IspisiInfoOPosudbama ()  
{  
...  
    foreach (Posudba p in this.posudbe)  
    {  
        double trenutniIznos = 0;  
        trenutniIznos = p.IzracunajIznosPosudbe ();  
  
        frekvencijaPosudbiBodovi++;  
...  
        //za ispis informacija o posudbi  
        ispis += "\t" + p.Film.Naslov + "\t"  
            + trenutniIznos.ToString("c") + "\n";  
        ukupniIznos += trenutniIznos;  
...  
    }  
}
```

Metoda nakon primjene *Replace Temp with Query*

□ Primjer: Izrada \ Refaktoriranje – R3

- lokalna privremena varijabla zamijenjena pozivom metode

```
public string IspisiInfoOPosudbama()
{
...
    foreach (Posudba p in this.posudbe)
    {
        frekvencijaPosudbiBodovi++;

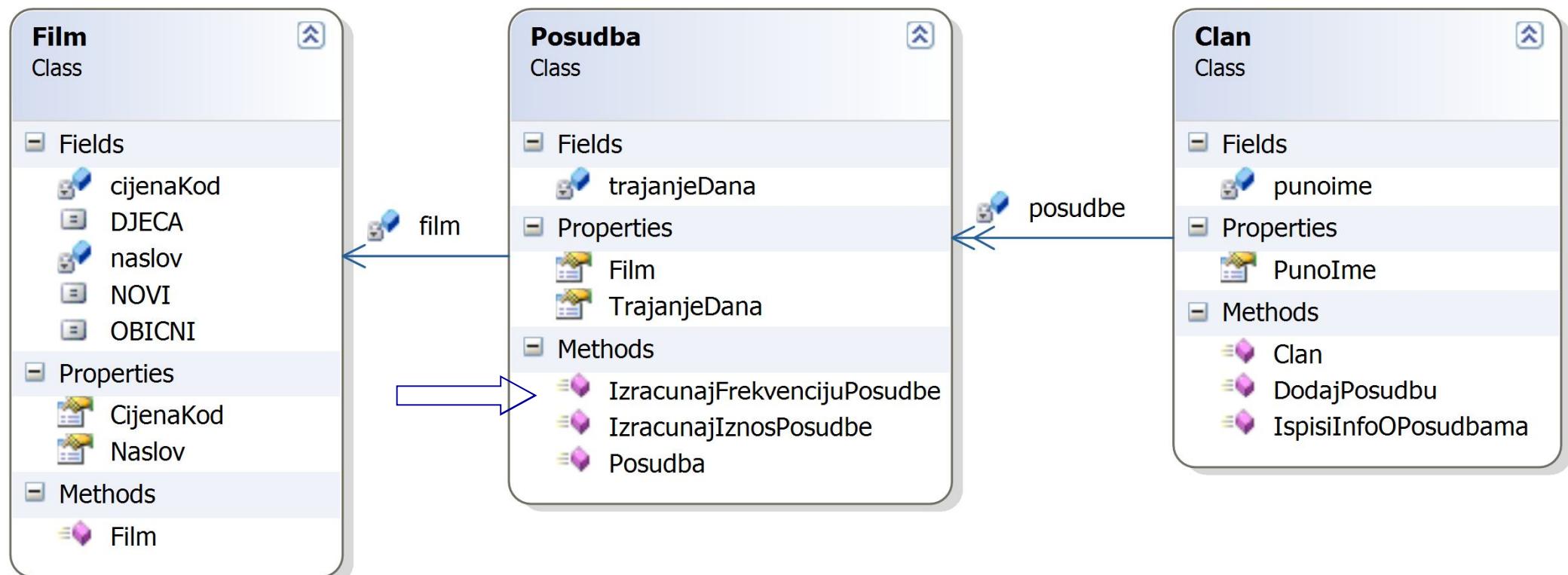
        //za ispis informacija o posudbi
        ispis += "\t" + p.Film.Naslov + "\t"
            + p.IzracunajIznosPosudbe().ToString("c") + "\n";
        ukupniIznos += p.IzracunajIznosPosudbe();
...
}
```

□ Što kada bi u realnom slučaju izračun potrajao ?

- preporuča se optimizaciju odgoditi dok kod ne bude restrukturiran

Ponavljanje *Extract Method* i *Move Method*

- Ako opet pogledamo metodu *IspisiInfoOPosudbama* (R3)
 - uočiti da se kod računanja bodova zbog frekvencije posuđivanja koriste informacije o posudbama, a nalazimo se u razredu *Clan*
 - stoga na dio koda gdje se računa frekvencija posudbi primijenimo *Extract Method*, a zatim i *Move Method*
- Primjer: Izrada \ Refaktoriranje – R4



Nakon *Extract Method* i *Move Method*

- ... u metodi *IspisiInfoOPosudbama* uklanjamo varijable *ukupniIznos* i *frekvencijaPosudbiBodovi*.
 - primjenom *Replace Temp with Query*
- Primjer:  Izsada \ Refaktoriranje – R4 (prije promjene)

```
public string IspisiInfoOPosudbama () {  
    double ukupniIznos = 0;  
    int frekvencijaPosudbiBodovi = 0;  
  
    ...  
    foreach (Posudba p in this.posudbe)  
    {  
        frekvencijaPosudbiBodovi += p.IzracunajFrekvencijuPosudbe ();  
        //za ispis informacija o posudbi  
        ispis += "\t" + p.Film.Naslov + "\t"  
               + p.IzracunajIznosPosudbe ().ToString ("c") + "\n";  
        ukupniIznos += p.IzracunajIznosPosudbe ();  
    }  
    //za ispis footer-a  
    ispis += "Iznos dugovanja: " + ukupniIznos.ToString ("c") + "\n";  
    ispis += "Bodovi zbog frekventnog posudivanja: "  
           + frekvencijaPosudbiBodovi.ToString ();
```

Nakon *Replace Temp with Query* – glavna metoda

□ Primjer: Izrada \ Refaktoriranje – R5

```
public string IspisiInfoOPosudbama()
{
    string ispis = "Posudbe za člana " + this.punoime + "\n";

    foreach (Posudba p in this.posudbe)
    {
        //za ispis informacija o posudbi
        ispis += "\t" + p.Film.Naslov + "\t"
            + p.IzracunajIznosPosudbe().ToString("c") + "\n";
    }

    //za ispis footer-a
    ispis += "Iznos dugovanja: "
        + this.IzracunajUkupniIznosPosudbi().ToString("c") + "\n";
    ispis += "Bodovi zbog frekventnog posuđivanja: "
        + this.IzracunajUkupnuFrekvencijuPosudbiBodovi().ToString();
}
```

□ Primijetiti da je metoda `IspisiInfoOPosudbama` svedena na kod koji se bavi samo ispisom, kako joj i ime govori

Nakon *Replace Temp with Query* – nove metode

□ Primjer: Izrada \ Refaktoriranje – R5

```
private double IzracunajUkupniIznosPosudbi()
{
    double rezultat = 0;
    foreach (Posudba p in this.posudbe)
        rezultat += p.IzracunajIznosPosudbe();

    return rezultat;
}

private int IzracunajUkupnuFrekvencijuPosudbiBodovi()
{
    int rezultat = 0;
    foreach (Posudba p in this.posudbe)
        rezultat += p.IzracunajFrekvencijuPosudbe();

    return rezultat;
}
```

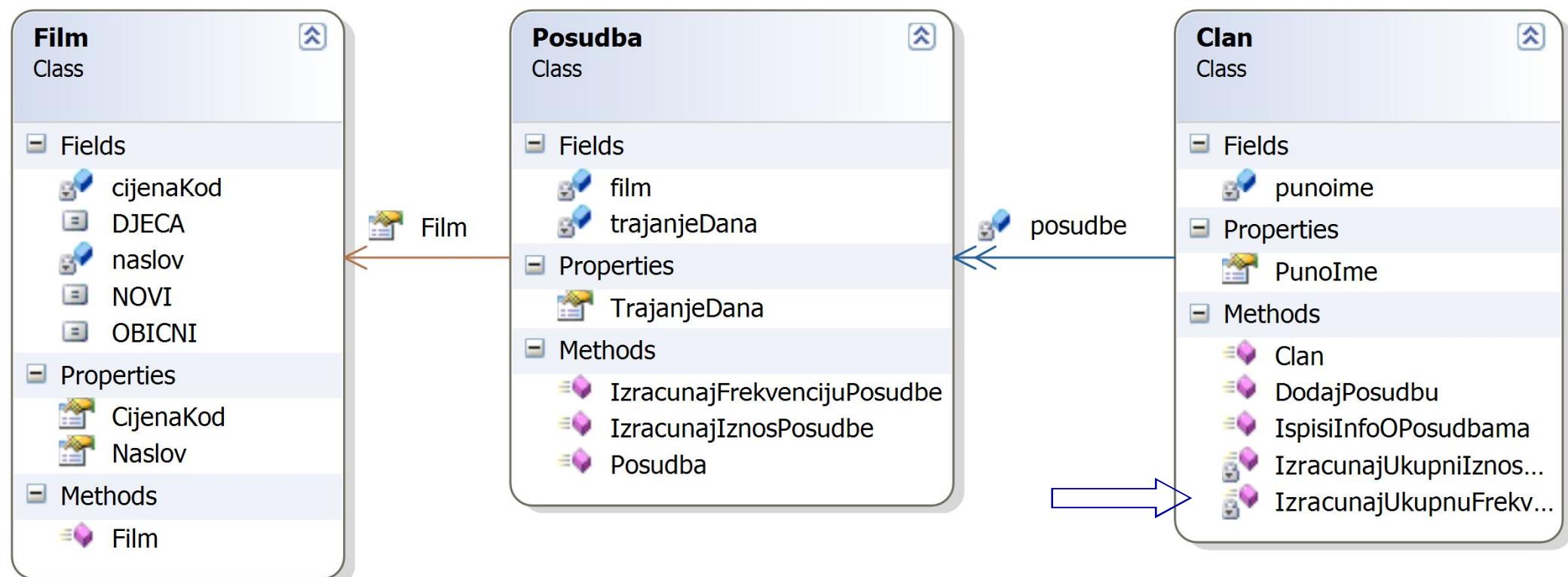
Komentar stanja

□ Primjer: Izrada \ Refaktoriranje – R5

- povećala se količina programskog koda i broj metoda
- potencijalno padaju performanse jer umjesto jedne *foreach* petlje imamo tri

□ Cilj je napraviti kod lakšim za razumijevanje i održavanje

- performansama ćemo se baviti kad količina podataka bude toliko velika da produlji vrijeme obrade i odziva



Dodavanje nove metode s osloncem na postojeće

- (Napokon) dodajemo metodu za formatiranje ispisa u HTML
- Primjer:  Izrada \ Refaktoriranje – R6

```
public string IspisiInfoOPosudbamaHTML()
{
    string ispis = "<h1>Posudbe za člana " + this.punoime + "</h1><p>";

    foreach (Posudba p in this.posudbe)
    {
        //za ispis o posudbi
        ispis += p.Film.Naslov + ": "
                + p.IzracunajIznosPosudbe().ToString("c") + "<br />";
    }

    //za ispis footer-a
    ispis += "<p>Iznos dugovanja: "
            + this.IzracunajUkupniIznosPosudbi().ToString("c") + "</p>";
    ispis += "<p>Bodovi zbog frekventnog posuđivanja: "
            + this.IzracunajUkupnuFrekvencijuPosudbiBodovi().ToString() + "</p>";

    return ispis;
}
```

Dodatni zahtjev

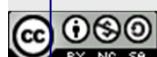
- Uvodi se nova klasifikacija filmova**
- Račun dugovanja vjerojatno će se također mijenjati**
- Jedino rješenje je uvođenje polimorfizma**
 - *switch* dio u metodi *IzracunajIznosPosudbe* obavlja se na temelju atributa *CijenaKod* koji je atribut razreda *Film*
- switch* bi trebalo raditi nad vlastitim podacima**
 - zato metodu *IzracunajIznosPosudbe* prebacujemo u razred *Film* ...
 - ali nećemo primijeniti *Move Method*, jer bi promjene morali raditi i u *Clan*
 - da poziva metodu iz *Film* umjesto *Posudba*
- Umjesto *Move Method***
 - ekstrahiramo tijelo *Posudba.IzracunajIznosPosudbe* u novu *Film.IzracunajIznosPosudbe*
 - u metodi *Posudba.IzracunajIznosPosudbe* pozovemo novu metodu

IzracunajIznosPosudbe prije prebacivanja

□ Primjer: Izrada \ Refaktoriranje – R6

```
public double IzracunajIznosPosudbe()
{
    double trenutniIznos = 0;

    switch (this.Film.CijenaKod)
    {
        case Film.OBICNI:
            trenutniIznos += 2;
            if (this.TrajanjeDana > 2)
                trenutniIznos += (this.TrajanjeDana - 2) * 1.5;
            break;
        case Film.NOVI:
            trenutniIznos += this.TrajanjeDana * 3;
            break;
        case Film.DJECA:
            trenutniIznos += 1.5;
            if (this.TrajanjeDana > 3)
                trenutniIznos += (this.TrajanjeDana - 3) * 1.5;
            break;
    }
    return trenutniIznos;
}
```



Posudba i Film nakon prebacivanja

□ Primjer: Izrada \ Refaktoriranje – R7, Razred Posudba

```
public double IzracunajIznosPosudbe()
{
    return this.Film.IzracunajIznosPosudbe(this.TrajanjeDana);
}
```

□ Primjer: Izrada \ Refaktoriranje – R7, Razred Film

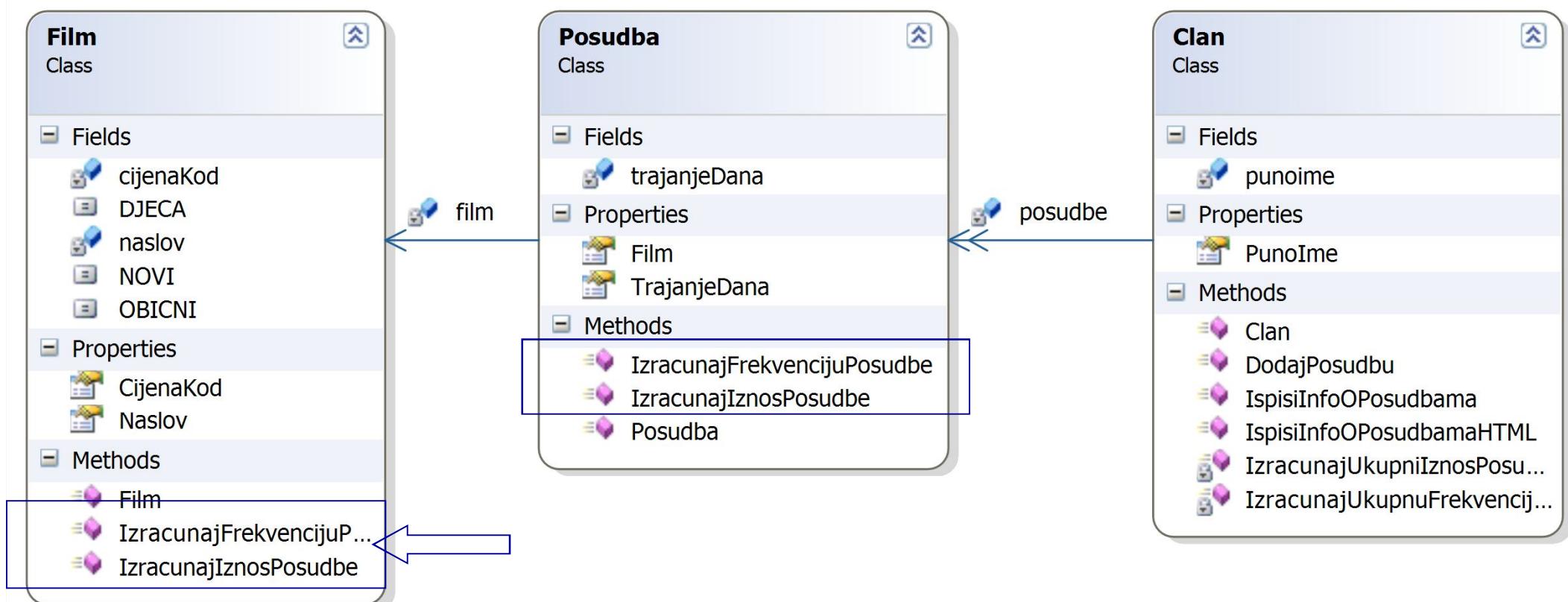
```
public double IzracunajIznosPosudbe(int trajanjeDana)
{
    double rezultat = 0;

    switch (this.CijenaKod)
    {
        case Film.OBICNI:
            rezultat += 2;
            if (trajanjeDana > 2)
                rezultat += (trajanjeDana - 2) * 1.5;
            break;
        case Film.NOVI:
        ...
    }
}
```

Prije kraja ...

□ Primjer: Izrada \ Refaktoriranje – R7

- slično *IzracunajIznosPosudbe*, preseljena je i *IzracunajFrekvencijuPosudbe*



Replace Conditional with Polymorphism

□ Preostaje riješiti *switch* dio u *Film.IzracunajIznosPosudbe*

- Postoji nekoliko tipova filmova koji na isti upit (izračunaj iznos posudbe, izračunaj frekvenciju posuđivanja i kod cijene) "odgovaraju" na različiti način
- Računica se obavlja temeljem atributa *cijenaKod*

□ Problem: što ako se pojavi novi tip filma

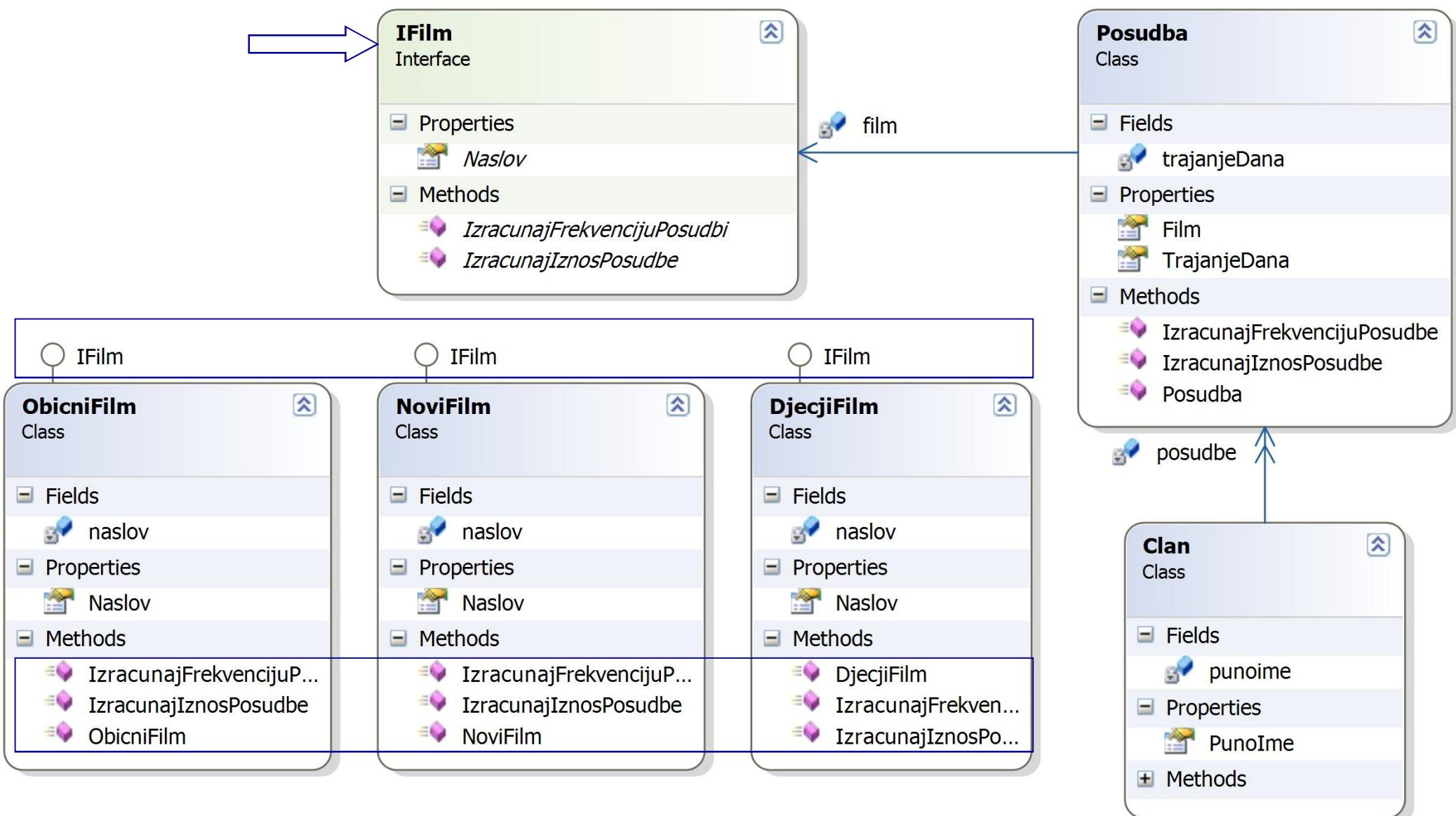
- Tada bi na svim mjestima gdje se obavlja provjera tipa cijene (u ovom kratkom primjeru samo na jednom mjestu) trebalo dodati novi case uvjet

□ Rješenje: nasljedivanje i polimorfizam

- za vrijeme izvođenja će se odrediti o kojem se tipu filma radi, te pozivati odgovarajuća metoda specijalizacije
- korištenjem postupka *Replace Conditional with Polymorphism*

Model s ugradenim polimorfizmom

□ Primjer: Izrada \ Refaktoriranje – R8



Sučelje i izvedeni razred

□ Primjer: Izrada \ Refaktoriranje – R8, sučelje *IFilm*

```
interface IFilm
{
    string Naslov {
        get;
    }
    double IzracunajIznosPosudbe(int trajanjeDana);
    int IzracunajFrekvencijuPosudbi(int trajanjeDana);
```

□ Primjer: Izrada \ Refaktoriranje – R8, realizacija

```
class ObicniFilm : IFilm {
...
    public double IzracunajIznosPosudbe(int trajanjeDana) {
        double rezultat = 2;
        if (trajanjeDana > 2)
            rezultat += (trajanjeDana - 2) * 1.5;
...
    public int IzracunajFrekvencijuPosudbi(int trajanjeDana)
    {
        return 1;
    }
```

Kada primijeniti refaktoriranje

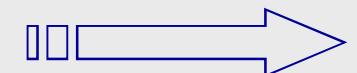
- “uvijek” – kad god se naprave neke promjene na programskom kodu koje “kvare” strukturu koda ili kada se promjene očekuju, pa se želimo pripremiti
- “***the rule of three***” - kad 1. put nešto radimo, onda to samo napravimo
 - 2. put, u sličnoj situaciji, bojimo se dupliciranja, ali ga napravimo
 - 3. put u trećoj sličnoj situaciji refaktoriramo
- **kada se dodaje nova funkcionalnost** – prije nego se nova funkcija doda, analiziramo dio koda na koji utječe. Kriteriji za refaktoriranje:
 - teško razumljiv kod
 - dodavanje nove funkcije kojim se narušava strukturu koda
- **kada treba ispraviti neki bug** – sama činjenica da bug treba otkriti znači da je kod problematičan i da je potrebno refaktoriranje
- **kada se obavlja pregled koda** – za vrijeme pregleda koda obično se pojave nove, bolje ideje, koje se primjenjuju tako da se obavi refaktoriranje
- “***bad code smells***” – duplicirani kod, duge metode, veliki razredi, metode s mnogo parametara, *divergent change*, *shotgun surgery*, *feature envy*, *primitive obsession*, *switch statements*, *parallel inheritance hierarchies*, *lazy class*, *temporary field*, *message chains*, *middle man*, ... (u dodatku predavanja)

Provjera ispravnosti

Testiranje

Provjera ispravnosti

- **Testiranje programa, provjeravanje programa, ispitivanje programa**
 - provjera programa izvođenjem, uz uporabu ispitnih podataka te analizom rezultata obrade
 - cilj testiranja je otkrivanje pogrešaka odnosno nedostataka unutar programa
 - uspješnost testa razmjerna je broju pronađenih pogrešaka
- **Stupnjevi, stadiji, faze**
 - testiranje jedinica, integracijsko testiranje, test sustava i test prihvatljivosti
- **Verifikacija - ovjera ispravnosti**
 - dokazivanje da je faza dobro provedena ili da je proizvod dobro napravljen, tj. da odgovara specifikaciji zahtjeva (slučajevima korištenja)
- **Validacija - potvrda valjanosti**
 - kojom se utvrđuje da je napravljen pravi proizvod, koji odgovara namjeni te da je prihvatljiv korisniku



Ključni pojmovi

- **Test – provjerava je li neki aspekt softvera ispravan**
 - pr. test da radi login, test da utrošak memorije ne premašuje 500Mb
- **Pogreška (error) – propust programera, npr. radi nerazumijevanja**
 - dovodi do jednog ili više kvarova
 - razlikujemo u odnosu na "pogrešku" koja znači neželjeno stanje, tj. kvar
- **Kvar (fault), defekt (defect), neformalno bug - neispravan dio koda**
 - npr. pogrešna pretpostavka da se polje indeksira od 1 umjesto 0 izaziva kvar pristupa elementu polja
- **Zastoj u radu (failure) – stanje izazvano jednim ili više kvarova**
 - npr. prestanak rada sustava zbog "buffer overrun" kvara
- **Ispravak (Fix) – stanje popravka**

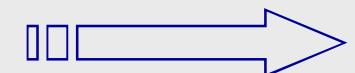
Problem testiranja objektno orijentiranih sustava

□ Problem: kako provjeriti da sustav zadovoljava potrebe korisnika

- rješenje: testiranjem poslovnih procesa
 - poslovni proces je raspodijeljen u skupu međudjelujućih razreda i sadržan u postupcima tih razreda
 - jedini način da se sazna učinak poslovnog procesa na sustav je zagledavanje u promjene stanja u sustavu
 - učahurivanje međutim sakriva podatke i obradu iza izloženih sučelja !

□ Drugi problem: što je osnovna "jedinica" za testiranje

- paket, razred ili metoda ?
- u tradicionalnim pristupima proces je sadržan u funkciji
- u OO sustavima zasebno testiranje pojedinačnih metoda nema puno smisla
 - uslijed nasljeđivanja i višeobličja postoje različita ponašanja, a bugovi nadređenog razreda propagiraju u neposredno i posredno izvedene
 - pri dinamičkom povezivanju ne zna se unaprijed koja će implementacija razreda biti izvršena



Plan testiranja

- Provjera počinje planom testiranja - plan definira niz testova**
 - plan treba napraviti na početku razvoja i stalno ažurirati
- Primjer: plan jedinične provjere razreda**

Class Test Plan		Page ____ of ____
Class Name:	Version Number:	CRC Card ID:
Tester:	Date Designed :	Date Conducted :
Class Objective:		
Associated Contract IDs:		
Associated Use Case IDs:		
Associated Superclass(es):		
Testing Objectives:		
Walkthrough Test Requirements:		
Invariant-Based Test Requirements:		
State-Based Test Requirements:		
Contract-Based Test Requirements:		

Plan testiranja (2)

□ Primjer: plan provjere invarijanti razreda

- navodi se izvorna i nova vrijednost atributa, događaj koji izaziva promjenu te rezultat tj. posljedica promjene. Evidentira se uspješnost testa (Pass/Fail)

Class Invariant Test Specification			Page ____ of ____		
Class Name: _____	Version Number: _____	CRC Card ID: _____			
Tester: _____	Date Designed : _____	Date Conducted : _____			
Testing Objectives:					
Test Cases					
Invariant Description	Original Attribute Value	Event	New Attribute Value	Expected Result	Result P/F
Attribute Name:					
1) _____	_____	_____	_____	_____	_____
2) _____	_____	_____	_____	_____	_____
3) _____	_____	_____	_____	_____	_____
Attribute Name:					
1) _____	_____	_____	_____	_____	_____

Testiranje jedinica

- **Testiranje jedinica (unit testing), pojedinačno testiranje**
 - najmanja jedinica mjere je razred !
 - testovi primjenjivi na nadređeni razred primjenjivi su na iz njega izvedene razrede, u dijelovima koji nisu preopterećeni nasljeđivanjem
 - posebnosti (višeobliče) – provjeriti u zasebnom kontekstu
 - » vrste testiranja
- **Funkcionalno (black-box testing)**
 - provjera se što cjelina radi, to jest da li zadovoljava zahtjeve
 - probni slučajevi izvode se iz specifikacija
 - provodi osoblje proizvođača ili korisnici
 - osnovica plana testiranja: CRC kartice, dijagrami razreda, ugovori
- **Strukturalno (white-box, clear box testing)**
 - provjera kako cjelina radi
 - probni slučajevi izvode se uvidom u programski kôd (inspekcija koda)
 - provode programeri
 - osnovica plana testiranja: specifikacije metoda

Integracijsko testiranje

□ Integracijska provjera (integration testing)

- jedinica je komponenta !
- razine: razredi koji tvore logičku cjelinu, sloj, paket, knjižnica
- ispitivanje provodi tim (analitičara i programera) za testiranje
 - » vrste testiranja

□ Testiranje korisničkog sučelja (User Interface Testing)

- provjerava se svaka funkcija sučelja
- osnovica: dizajn sučelja
- testiranje se radi prolaskom kroz svaku stavku izbornika sučelja

□ Testiranje slučajeva korištenja (Use-Case Testing)

- provjerava se svaki slučaj korištenja
- osnovica: slučajevi korištenja
- testiranje se radi prolaskom kroz svaki slučaj korištenja.
- često se kombinira s testom korisničkog sučelja jer UC ne testira sva sučelja

Integracijsko testiranje (nastavak)

□ Testiranje interakcije (Interaction Testing)

- testiranje svakog procesa korak po korak
- osnovica: dijagrami razreda, dijagrami slijeda, dijagrami komunikacije
- cijeli sustav započinje kao skup okrajaka
 - razredi se dodaju pojedinačno i rezultati se uspoređuju s očekivanim
 - nakon što neki razred prođe testove, dodaje se sljedeći i ponavljaju testovi
 - postupak se provodi za svaki paket
 - kad svi paketi prođu sve testove, proces se ponavlja za integriranje paketa

□ Testiranje sučelja sustava (System Interface Testing)

- testiranje razmjene podataka s drugim sustavima
- osnovica: dijagrami slučajeva korištenja
- prijenosi podataka između sustava često automatizirani
 - korisnici ih izravno ne nadziru – dodatna pažnja na provjeru / ispravnost

Testiranje sustava

□ Provjera sustava (System Testing)

- provjera rada sustava kao cjeline, kojom se osigurava da svi nezavisno razvijeni aplikacijski programi rade ispravno te sukladno specifikacijama
 - » vrste testiranja

□ Testiranje zahtjeva (Requirements Testing)

- testiranje jesu li zadovoljeni izvorni poslovni zahtjevi
- osnovica: dizajn sustava, testovi komponenti i integracijski testovi
- osigurava da promjene tijekom integracije nisu dovele do novih pogrešaka
- testeri se pretvaraju da su nekompetentni korisnici i izvode neprikladne radnje da testiraju robustnost (npr. dodavanje praznih ili duplih zapisu)

□ Testiranje uporabivosti (Usability Testing)

- testiranje prikladnosti sustava za korištenje
- osnovica: dizajn sučelja i slučajevi korištenja
- analitičar koji razumije korisnika i poznaje (dobar) dizajn sučelja

Testiranje sustava (nastavak)

□ Testiranje sigurnosti (Security Testing)

- testiranje mogućnosti oporavka i neautoriziranog pristupa
- osnovica: dizajn infrastrukture
- analitičar infrastrukture, u ekstremnim slučajevima zasebna tvrtka

□ Testiranje performansi (Performance Testing)

- ispitivanje sposobnosti izvođenja pod velikim opterećenjem
 - stress testing - velik broj interakcija (simulacijom pristupa)
 - load testing – velika količina podataka (npr. generatorima podataka)
- osnovica: prijedlog sustava, dizajn infrastrukture

□ Testiranje dokumentacije (Documentation Testing)

- testiranje ispravnosti dokumentacije
- osnovica: sustav pomoći, postupci, priručnici

Test prihvatljivosti

□ Provjera prihvatljivosti (Acceptance Testing)

- dokazivanje da proizvod zadovoljava zahtjeve i uvjete preuzimanja
- iscrpan i konačan test nad stvarnim podacima

□ Alfa-testiranje (Alpha Testing) - verifikacijsko

- probna uporaba koju provode korisnici kod izvođača
- simulacija stvarnog okruženja
- traženje pogrešaka i propusta

□ Beta-testiranje (Beta Testing) – validacijsko

- provode korisnici kod sebe, bez nazočnosti izvođača
- provjera u stvarnim uvjetima
 - performanse sustava
 - vršna opterećenja
 - provjera upotrebljivosti i lakoće uporabe
 - radne procedure
 - izrada rezervnih kopija i oporavak sustava

□ Nadzorni test (Audit Test) – provodi se opcionalno

- potvrda da je sustav gotov, ispravan i spremан за primjenu
- provode nezavisne tvrtke ili odjeli za osiguranje kvalitete

Neki primjeri

- **Provjera u kojoj sudjeluju poznati korisnici (alfa, beta)**
 - Provjeru obavlja ogledna skupina krajnjih korisnika koja, koristeći napravljena rješenja, nastoji obaviti svoje svakodnevne poslove.
 - Po želji krajnji korisnik dodatno iznosi svoja zapažanja ili prijedloge.
 - Primjedbe se prikupljaju dnevno a pogreške uklanjaju po mogućnosti istog dana.
 - Prikupljeni dodatni zahtjevi se procjenjuju te se izrađuje lista prioriteta ugradnje.
 - Nerealni i preveliki zahtjevi se odbacuju ili se planira njihova naknadna ugradnja.
- **Plan testiranja**
 - identifikator programa ili dijela obrade (npr. naziv opcije izbornika ili zaslona)
 - naziv funkcije (npr. unos ili izmjena)
 - vrstu poduzete akcije (npr. potvrda pohrane ili prekid obrade)
 - identifikator ili opis podatka koji se želi obraditi
 - ponašanje programa (npr. neregularni završetak rada, neispravni podaci, pogrešan prikaz podataka), po potrebi očekivani rezultat
- **Primjer:**  **Izrada\ ObrazacZaTest**
- **Primjeri:**  **Izrada \ VrsteTestova.xls, formulari specifikacija i testova**

Razvoj vođen testiranjem

□ Razvoj vođen testiranjem (Test Driven Development)

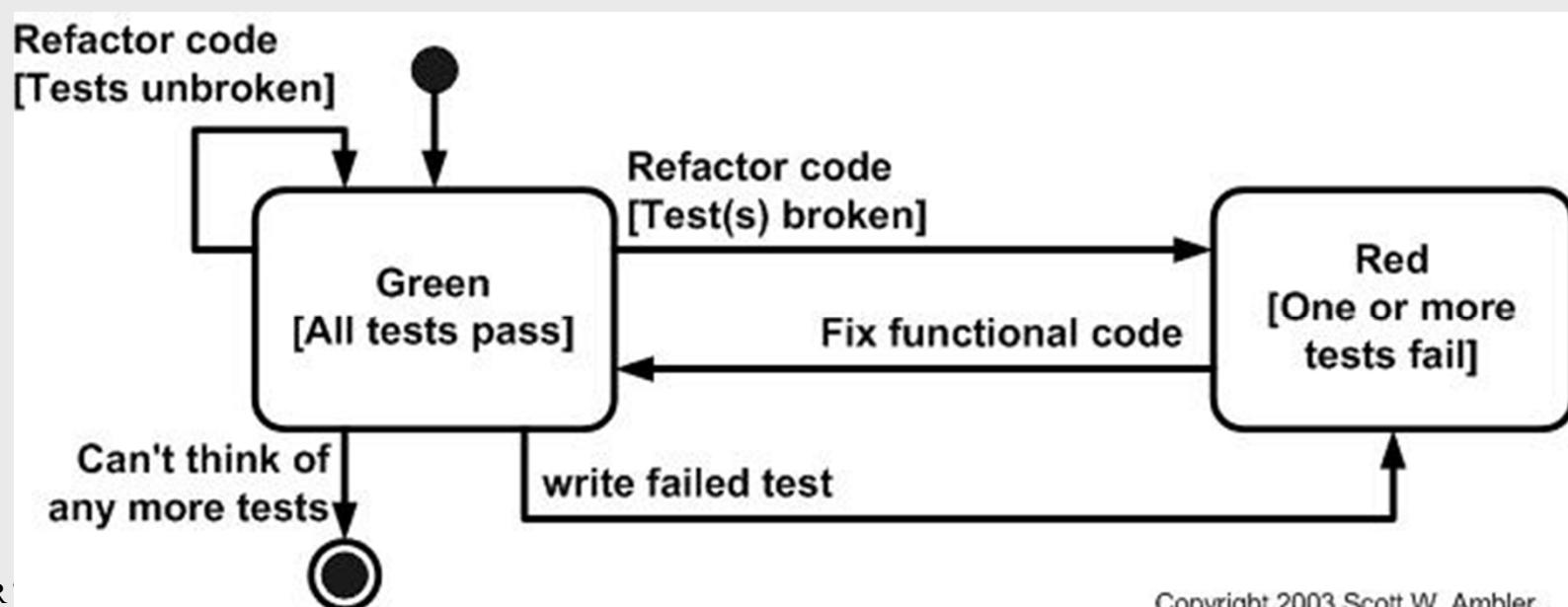
- Testovi se pišu prije koda, tj. ne kodira se nešto za što ne postoji test
- U svakoj iteraciji po obavljanju testa provodi se refaktoriranje

□ Automatizacija testiranja

- alati za automatsko testiranje - rezultate testa uspoređuju s očekivanim rezultatima
- *Unit (jUnit, nUnit, csUnit, xUnit, cppUnit, pyUnit, phpUnit, ...) – open source

□ Regresijsko testiranje (regression testing), retesting

- provjera kojom se dokazuje da softver nije nazadovao (regressed)
- pokretanje svih testova (starih i nanovo dodanih) pri promjeni softvera



Alati za jedinično testiranje

- <http://junit.org> – Beck, Gamma
- <http://www.nunit.org/> - Beck, prvotno po uzoru na jUnit
- <http://www.csunit.org/> Agile Utilities NZ, zapušten (zadnja objava 2009)
- <http://xunit.codeplex.com/> – „the original inventor of nUnit”
- ...

□ Ručna instalacija

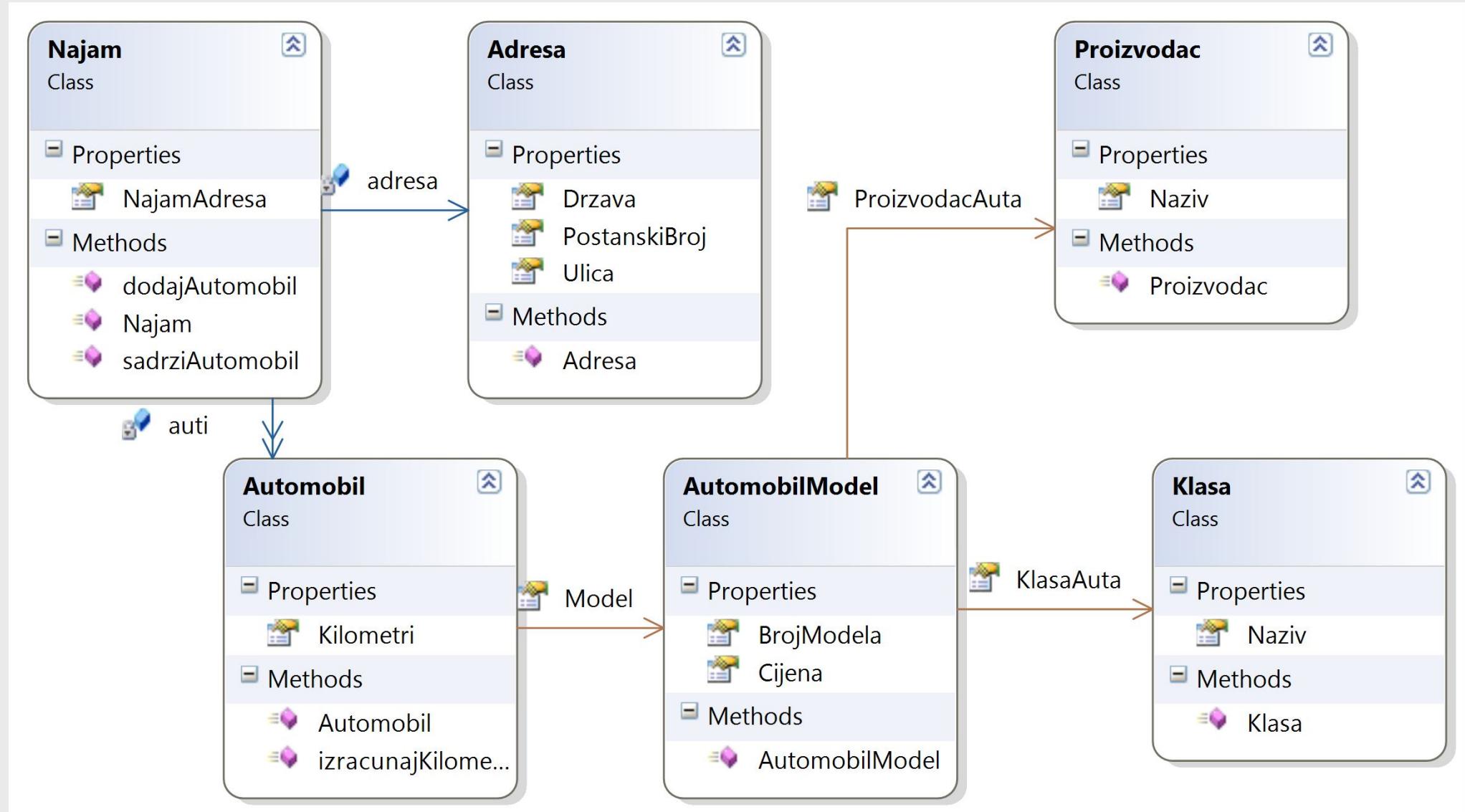
- Preuzimanje instalacije

□ Kreiranje testa

- Stvaranje i dodavanje projekta tipa „Class Library” u Solution
- dodavanje referenci u projekt za test na
 - projekt koji treba testirati
 - dinamičku knjižnicu za testiranje (npr. Nunit.Framework)
- stvaranje jedne ili više klasa za testiranje

Primjer: projekt koji treba testirati

□ Primjer: Izrada \Testiranje – RentACar



Primjer: razred i postupci za testiranje

□ Primjer: Izsada \Testiranje – TestiranjeJedinica.AutomobilTest.cs

- Kreira se razred kojem se pridružuje atribut [TestFixture]
- Metode koje pokreću testove označavaju se atributom [Test]
- Test se provodi klasom Assert okvira za testiranje
 - u ovom primjeru NUnit.Framework.Assert , u narednom bude
 - Microsoft.VisualStudio.TestTools.UnitTesting.Assert

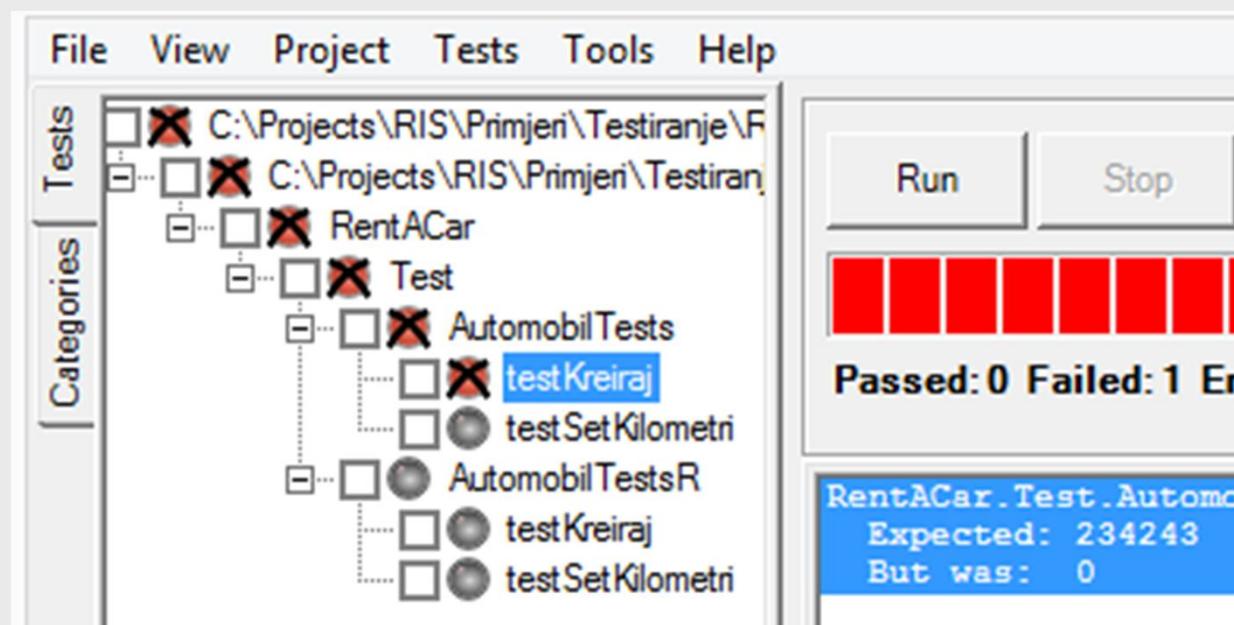
[TestFixture]

```
public class AutomobilTests {  
    [Test]  
    public void testKreiraj() {  
        // 1.korak: kreiranje objekata (arrange)  
        AutomobilModel automobilModel = new AutomobilModel(...);  
        // 2.korak: upravljanje objektima (act)  
        int kilometri = 234243;  
        Automobil automobil = new Automobil(automobilModel, kilometri);  
        // 3.korak: assert (assert)  
        Assert.AreEqual(kilometri, automobil.Kilometri);  
    }  
}
```

Izrada i pokretanje testa

□ Primjer:

- Windows - Run ... nUnit ili neki drugi (npr. csUnit)
- Kreiranje “projekta” testiranja
- Dodavanje knjižnica s testovima: Project \ Add Assembly
- Tests : Run All, Run Selected, Run Failed
 - Run All predstavlja regresijski test
- File: Reload Tests



Optimizacija testa

□ Primjer: Izrada \Testiranje – ... AutomobilTestR.cs

- Obje metode `testKreiraj` i `testSetKilometri` zahtijevaju kreiranje objekta `Automobil`
- Refaktoriranjem testa zajednički dio koda izdvojen u zasebnu metodu `setUp`
- Atribut `[SetUp]` označava metodu koja će se izvesti prije svakog testa
- Atribut `[TearDown]` označava metodu koja će se izvesti nakon svakog testa

[TestFixture]

```
public class AutomobilTestsR
{
    [SetUp]
    public void SetUp()
    {
        ...
    }

    [Test]
    public void testKreiraj()
    {
        ...
        Assert...
    }
}
```

Testiranje u razvojnoj okolini Visual Studio

- Solution \ Add Project tipa ***Unit Test Project***
 - Namespace Microsoft.VisualStudio.TestTools.UnitTesting
- Test \ Windows \ **Test Explorer (VS2012) ili Test List Editor (VS2010)**
 - *Run Failed, Run Not Run, Run Passed, Repeat Last Run*
- Primjer:  Izrada \Testiranje – RentACar.RegresijskoTestiranje
 - Koriste se slični atributi kao i kod *Unit

```
[TestClass()]
public class AutomobilTest
{
    ...
    [TestInitialize()]
    public void MyTestInitialize()
    {
        ...
    }
    [TestMethod()]
    public void KilometriTest()
    {
        ...
        Assert...
    }
}
```

Integracija vanjskih alata u okolinu VS 2012

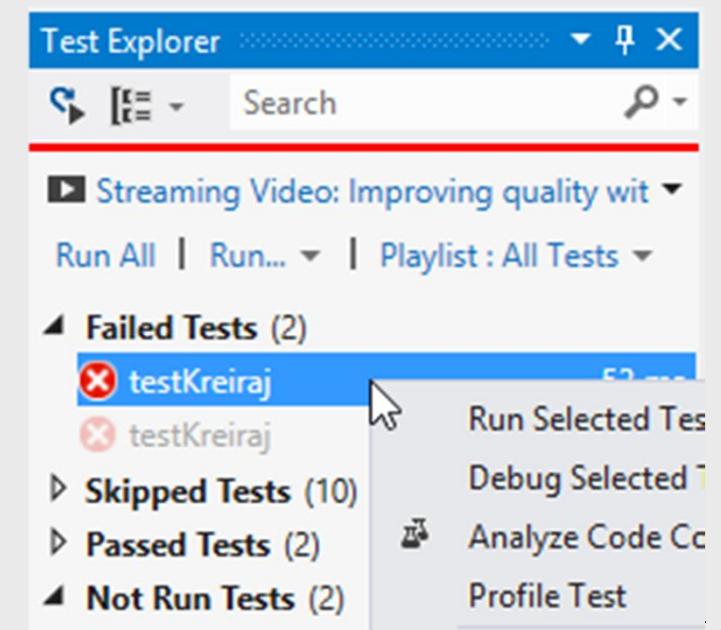
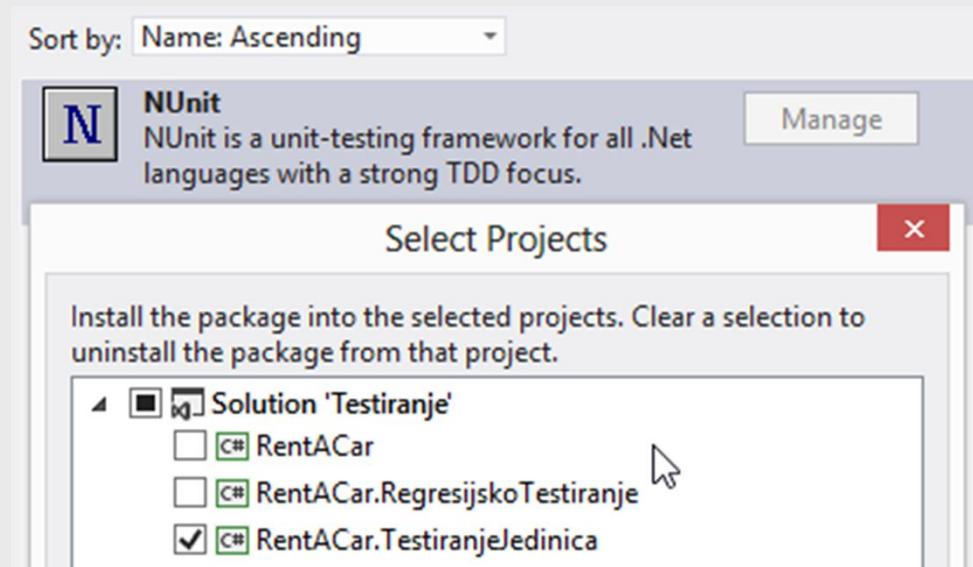
□ Tools \ Extensions and Updates ...

- online – search – install
- pr. NUnit Test Adapter for Visual Studio
- Integracija testa u razvojnu okolinu
- Nakon kompilacije VS prepoznaje koje testove treba dodati u *Test Explorer*



□ Solution \ Manage NuGet packages for Solution

- Kopira potrebne knjižnice i doda referencu u odabrani projekt
- Npr. C:\....\Testiranje\packages\NUnit.2.6.2\lib\nunit.framework.dll



Ostali primjeri

□ Primjeri alata

- Primjer:  PrimjeriLabosa
 - LP2-Nunit-AndreaKnez.pdf
 - LP2-PHPUnit-MartinVrklijan.pdf
 - LP2-PyUnit-MislavStipetic.pdf
 - LP2-csUnit-VukojevicMarin.pdf

□ Tablica usporedbe atributa i asserta

- Primjer:  Izrada \ NUnit 2.6 vs MSTest 2010_2012 vs xUnit.net

□ Dodatni primjeri (preuzeti s nUnit)

- Primjer:  Izrada \ Testiranje \ NUnit csharp
- Dodatno, postoji i primjer kako Nunit testira sam sebe

Microsoft Fake Framework

❑ Fake – krivotvorina

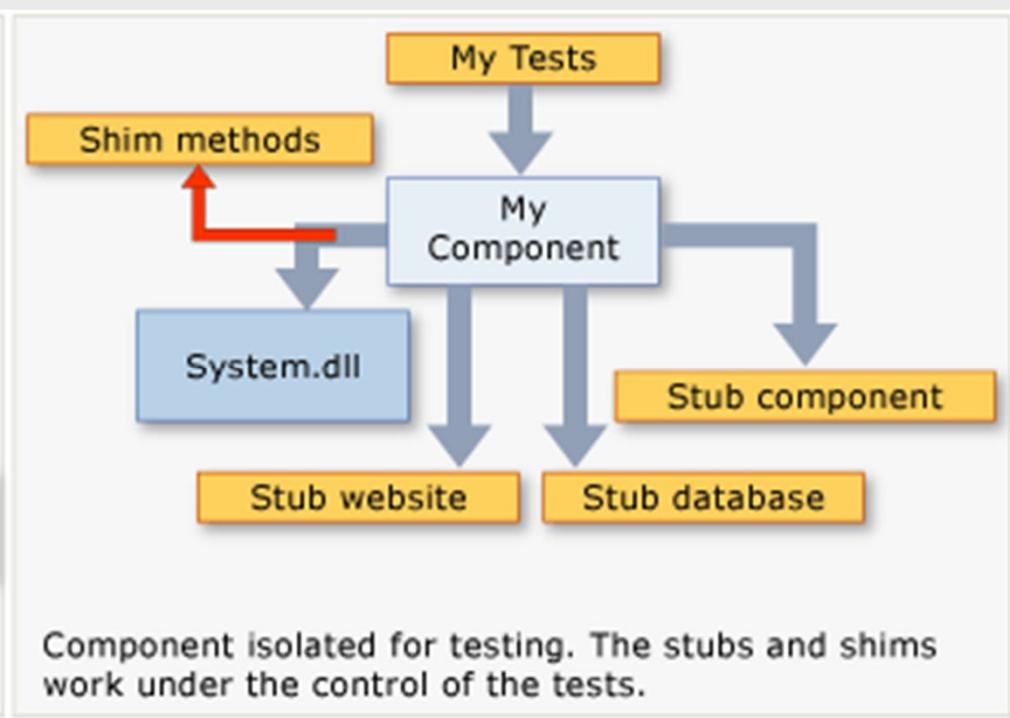
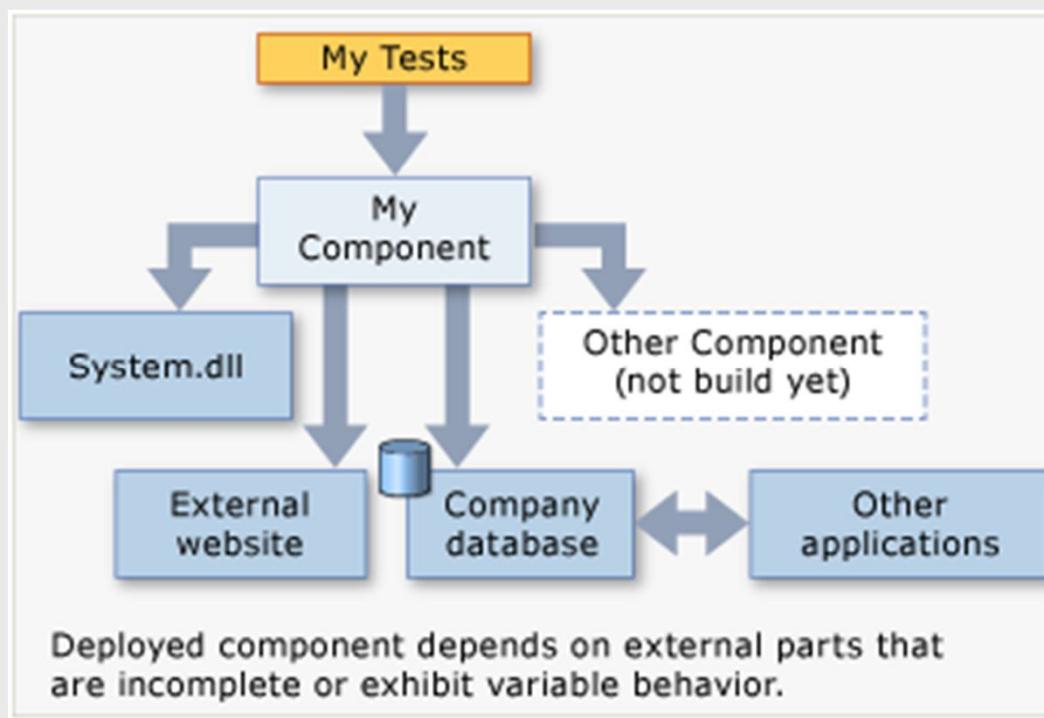
- omogućuje izolaciju koda (testa) zamjenom dijelova koda

❑ Stub – okrajak, čik

- Mali substitut koji implementira isto sučelje

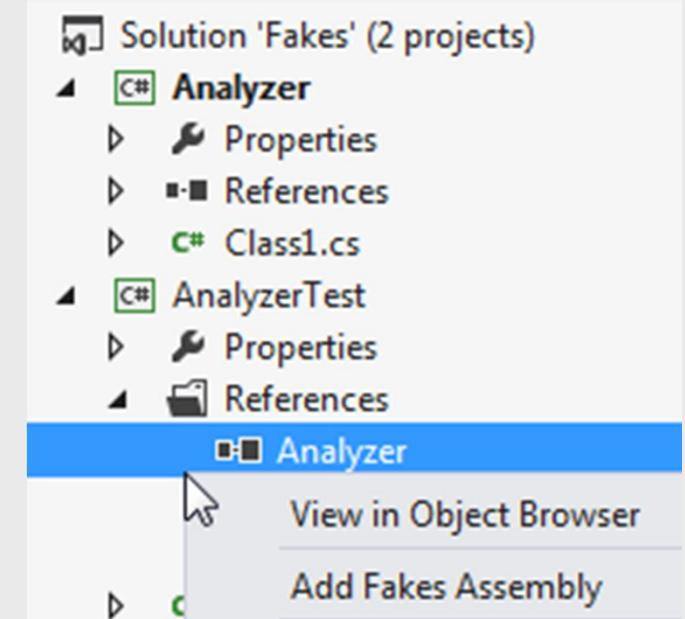
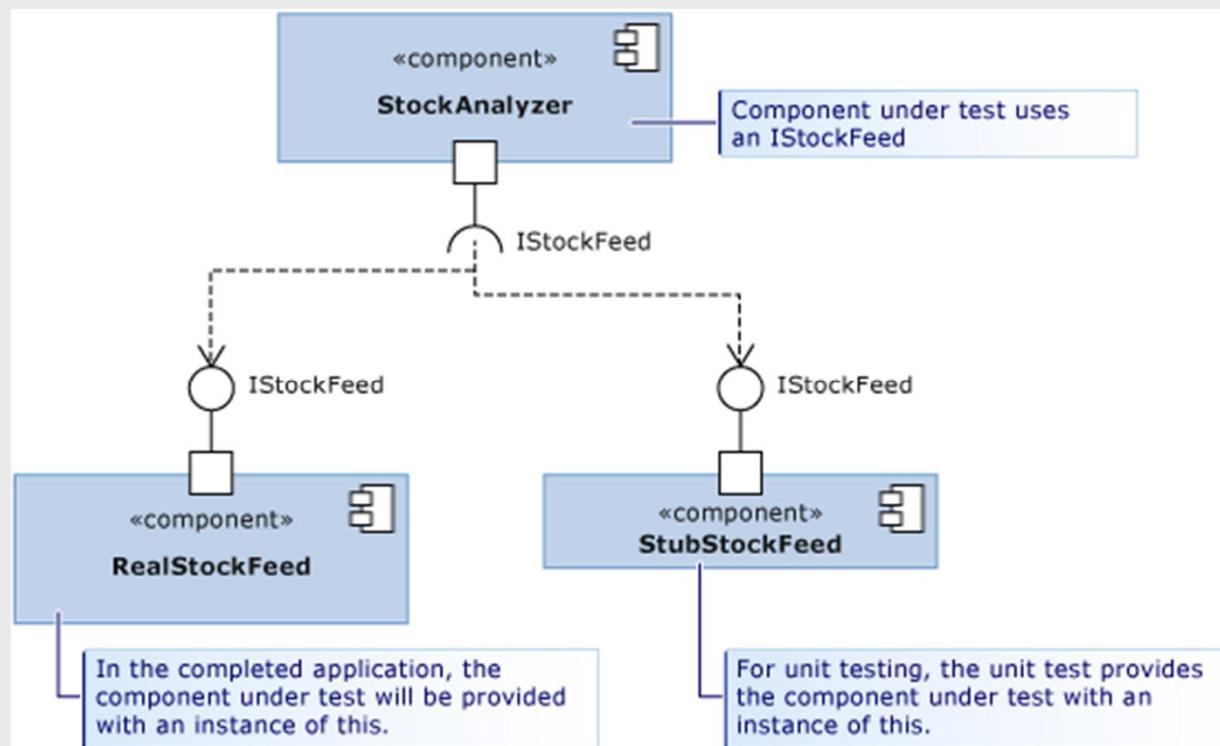
❑ Shim – čep, podloška

- Mijenja kompilirani kod u pogonu tako da se umjesto određenog poziva neke metode pokrene kod osiguran testom



Anti-primjer

- Isolating Code under Test with Microsoft Fakes
 - <http://msdn.microsoft.com/en-us/library/hh549175.aspx>
- privatne klase - nevidljivi testovi
- .fakes Diagnostic="true"
- ograničenja
- Add Fakes Assembly of System



Otklanjanje pogrešaka (debugiranje)

□ Debugiranje (debugging)

- Posljedica testiranja, ali je nezavisna aktivnost, tj. nije testiranje
- Može biti jedan od najviše iscrpljujućih dijelova razvoja
- Otkrivanje pogreške napornije (dugotrajnije!) od ispravljanja
- Slično liječenju osobe - vidimo simptome i pokušavamo pronaći uzrok

□ Simptom

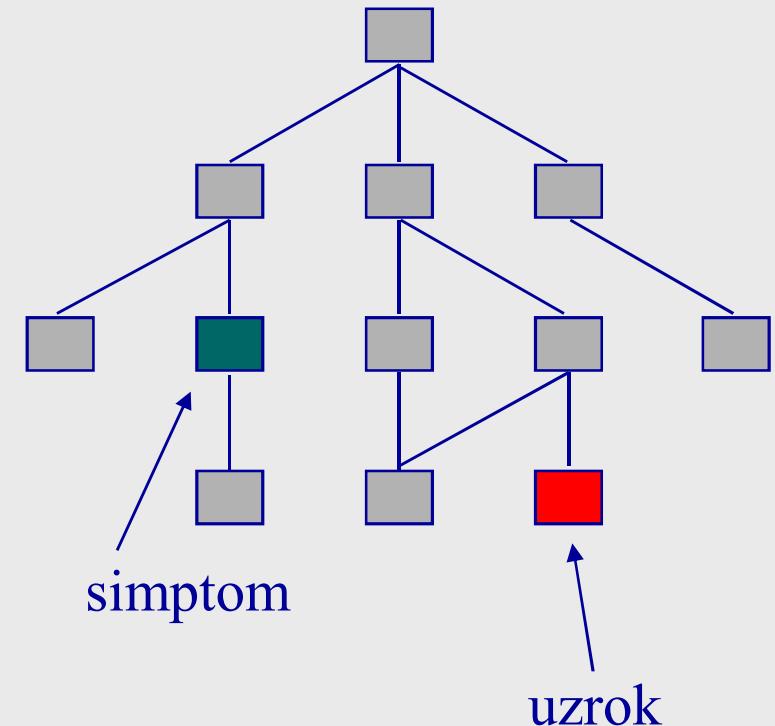
- može biti povremen
- može nestati kada se ispravi neki drugi problem

□ Uzrok

- može biti pogreška sustava ili prevoditelja
- može biti ljudska pogreška koju je teško otkriti
- mogu biti pretpostavke u koje svi vjeruju

□ Simptomi i uzroci mogu biti prostorno odvojeni

□ Pritisak da se isprave pogreške često dovodi do novih !



Tehnike otklanjanja pogrešaka

Gruba sila – jaki oslonac na računalo

- ispisi memorije
- praćenje izvođenja koda

Vraćanje unatrag (backtracking) – unatražno praćenje

- početi od mesta gdje se problem javlja i unatraške tražiti uzrok
- upotrebljivo za manje programe

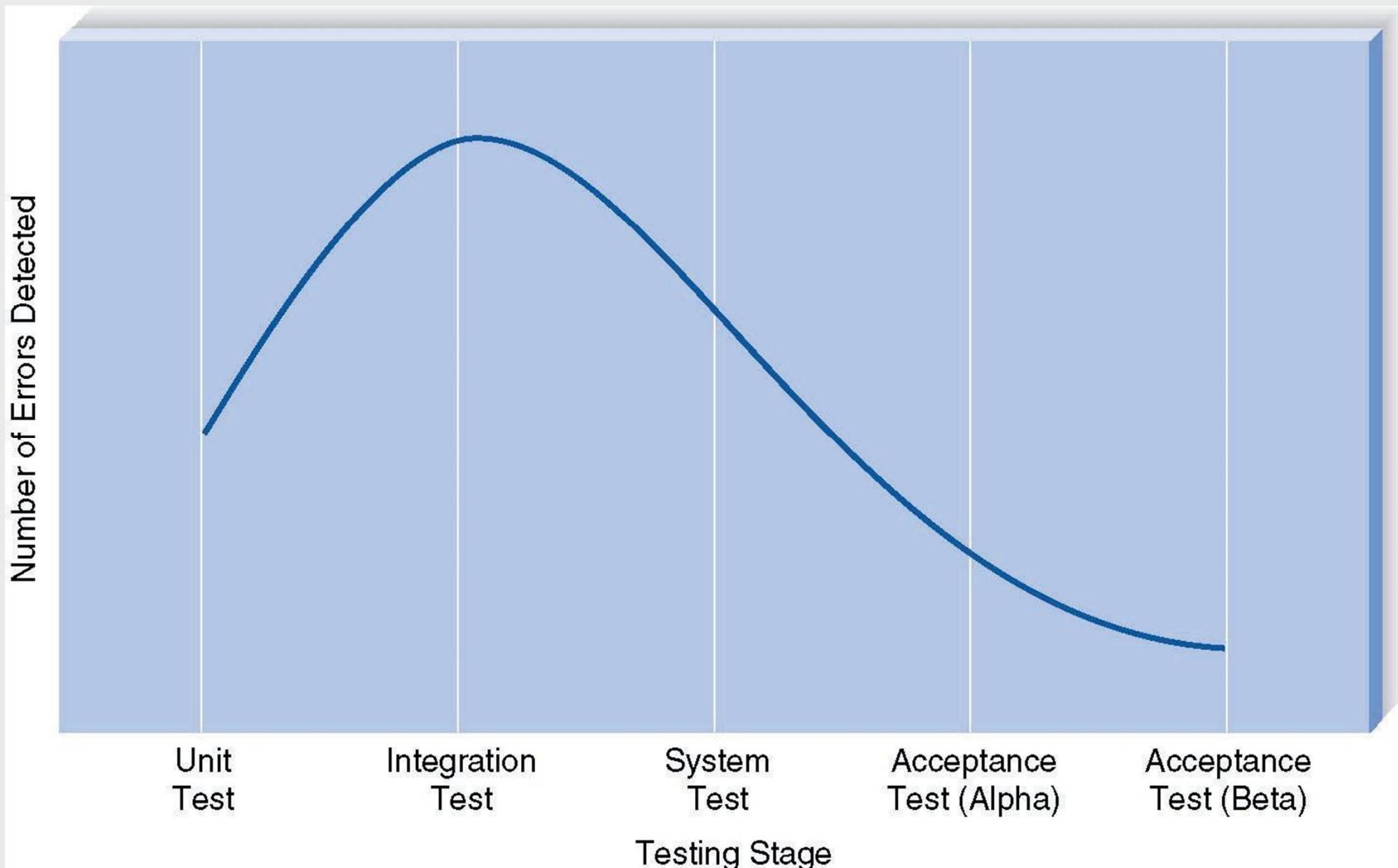
Eliminacija uzroka

- izrada skupa ulaznih podataka koji potvrđuju ili opovrgavaju određenu teoriju
- teško, no često jedino što je moguće !

Preporuke za otklanjanje

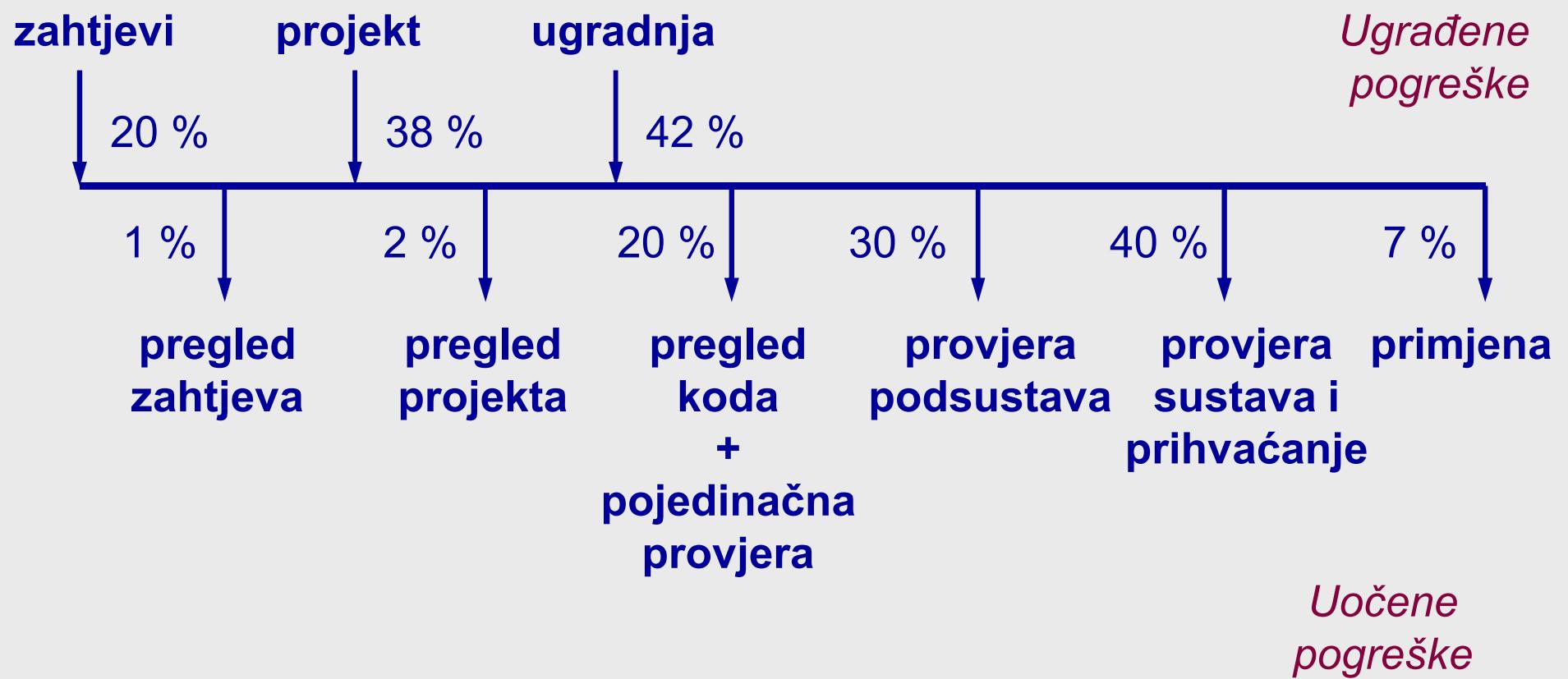
- Alati za debugiranje - mogu pomoći stjecanju boljeg pregleda situacije
- Razgovor s kolegama - možda drugi primijete ono što autor ne vidi
- Izolacija i usredotočenost – promisliti problem, neka rješenja nađu se kad ste udaljeni od računala
- Regresijsko testiranje - nakon što je pogreška ispravljena, provjera da nisu napravljene nove !

Raspodjela broja otkrivenih pogrešaka



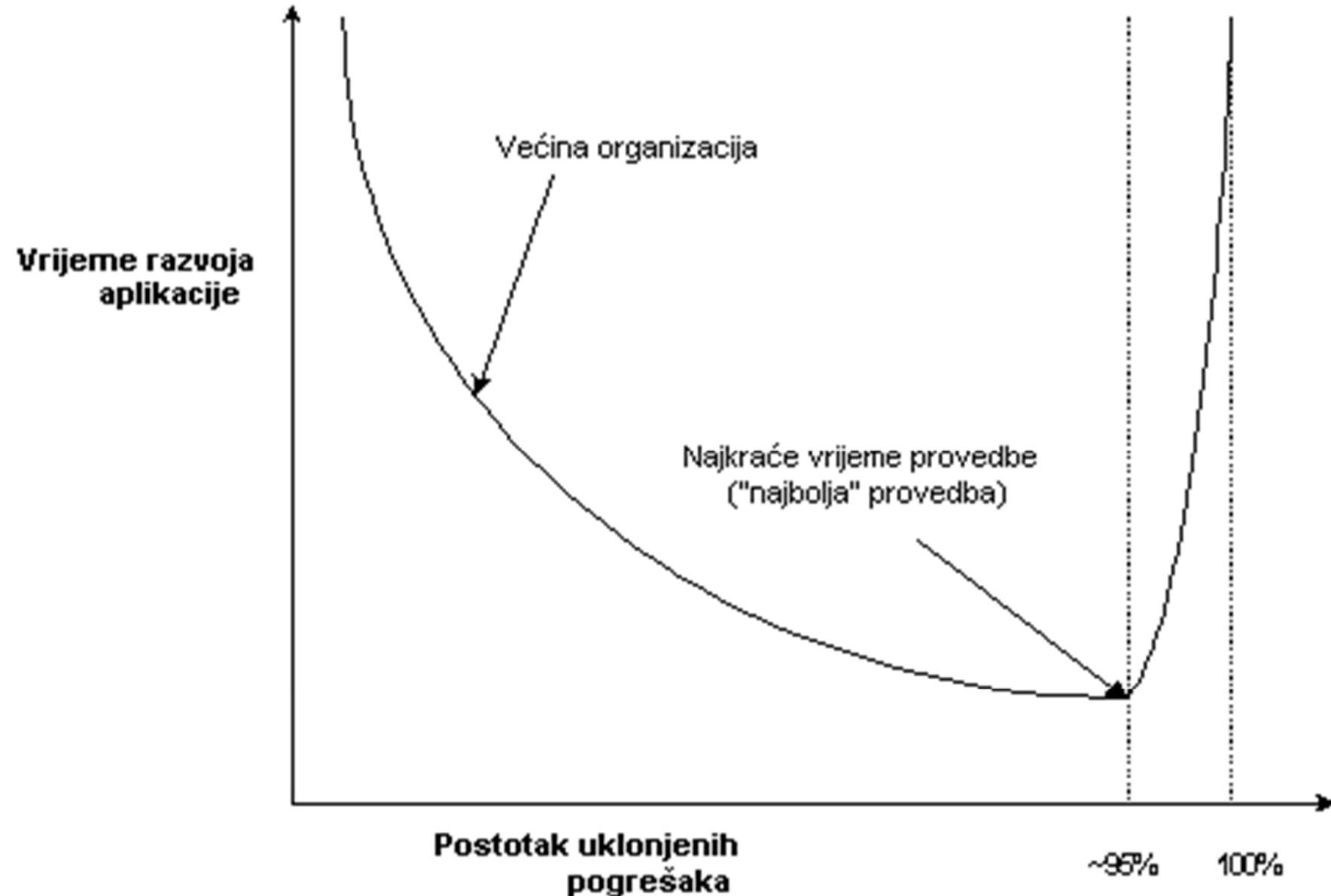
Slika: A.Dennis, B.H.Wixom, D.Tegarden:
Systems Analysis and Design with UML,
John Wiley & Sons

Statistika pogrešaka: pojava i ispravak



Vrijeme razvoja i postotak uklonjenih pogrešaka

- Odnos trajanja razvoja i postotka uklonjenih pogrešaka [Jones 1991].



Vrijeme razvoja i postotak uklonjenih pogrešaka

- **Područje oko točke s 95% otklonjenih pogrešaka**
 - najkraći razvojni ciklus, najmanji napor i najveće zadovoljstvo korisnika
- **Nakon što bude otklonjeno 95% pogrešaka**
 - daljnji napor otklanjanja postane prevelik
 - smislen u sustavima bitnim za životne funkcije (npr. svemirske letjelice)
- **Aplikacije koje su isporučene pod vremenskim pritiskom imaju do 4 puta više pogrešaka u odnosu na druge [Jones 1994].**
- **Klasična zabluda - testiranje je luksuz pred rok završetka**
 - zaobilazna rješenja umjesto odbacivanja problematičnih rutina i pisanja čistog koda ispočetka
 - posljedica - razbacivanje vremenom radi ispravljanja programa

Moduli s najvećim postotkom pogrešaka

□ Moduli s neproporcionalno velikim postotkom pogrešaka

- najčešće veći i lošije strukturirani
- svakako zahtjevniji i skuplji

□ Pravilo 80:20

- Uobičajeno 20% modula sadrži 80% svih pogrešaka [Boehm 1987b].
- IBM projekt IMS - 57% pogrešaka u 7% modula [Jones 1991].

□ Ponovno projektirati najbugovitije

- npr. ako se na svakih 1000 redaka u pojavljuje 10 pogrešaka

Izrada dokumentacije

Sistemska dokumentacija

Sistemska dokumentacija

- dokumentira razvoj i proizvode pojedinih faza
- namijenjena tehničkom osoblju
- potrebna za razumijevanje, izradu i održavanje sustava
- glavnina (pisane) dokumentacije nastaje tijekom analize i projektiranja (!?)
- programska dokumentacija dijelom se može generirati

upravljanje projektom i konfiguracijom sustava

- plan razvoja, specifikacija zahtjeva i dizajna, ...

programska dokumentacija

- izvorni kôd, opis baze podataka, probni podaci i rezultati provjere, dnevnik promjena, programski priručnici

upute za rukovanje i održavanje (installation, operations manual)

- opis instalacijske procedure
- opis procedura za pokretanje/zaustavljanje (startup/shutdown)
- opis izrade rezervnih kopija i vraćanja podataka (backup, restore)
- opis postupka ponovnog pokretanja i oporavka (restart, recovery)

Korisnička dokumentacija

□ Korisnička dokumentacija

- pomoći korisnicima pri korištenju aplikacija
- mora biti prilagođena korisnicima različitog iskustva
- korisnički priručnik, upute za uporabu (user manual)
- materijali za poduku, upute za vježbu (training guide, tutorial)
- dinamička pomoć (online help)

□ Vrste korisničke dokumentacije s obzirom na strukturu

- referentni priručnik (reference document), najčešće sustav pomoći
 - kad korisnik treba naučiti kako obaviti neku funkciju
 - obično se čita nakon što je intuitivan pokušaj obrade bio neuspješan, stoga treba biti prikladno/pažljivo napisan
- priručnik procedure (procedure manual)
 - opisuje kako obaviti poslovne zadatke koji se sastoje od više funkcija ili koraka, npr. ispis mjesečnog izvješća, zaprimanje narudžbe
- vodič, lekcija (tutorial)
 - podučava kako koristiti glavne komponente sustava
 - elementi dokumentacije dulji i oblikovani tako da se čitaju u slijedu

Planiranje dokumentiranja

- **Dokumentiranje se često ostavlja za kraj izrade – rizično !**
- **Vremenski zahtjevan postupak izrade dokumentacije**
 - dizajn dokumenata, pisanje teksta, uređivanje teksta, testiranje dokumenata
 - 3h po stranici pisanog teksta
 - 2h po zaslonu dinamičke pomoći
- **Planiranje izrade dokumentacije**
 - dokumentiranje treba ugraditi u plan projekta
 - početak izrade okvirno nakon što je definirano korisničko sučelje i napravljene programske specifikacije
 - kraj izrade planirati za neposredno nakon pojedinačnog testiranja
 - time se smanjuje volumen izmjena a ostavlja vrijeme za moguće promjene

Preporuke za izradu dokumentacije

□ Prilikom izrade treba

- izbjegavati ponavljanje i neodređenost
- koristiti standardizirane dokumente
- prije objavljivanja provjeriti da odgovara namjeni

□ Između ostalog, dobra dokumentacija mora:

- biti napisana s gledišta čitatelja, a ne pisca
 - konzistentnost pojmove, jednostavnost izraza, kratka poglavlja
- biti dobro ilustririrana (slikama zaslona i njihovog redoslijeda)
- opisivati postupak rada, a ne samo sadržaj zaslona
 - npr. redoslijed popunjavanja šifrarnika, određivanje lozinki i prava pristupa, radne procedure
- bilježiti načela (argumente odluka)
- biti ažurna, tj. odgovarati stvarnom stanju programa
- uključivati rječnik korištenih izraza

Primjeri

□ Dokumentacija po IEEE standardu

- Plan validacije i verifikacije softvera - Software Validation and Verification Plan (SVVP)
- Plan osiguranja kvalitete softvera - Software Quality Assurance Plan (SQAP)
- Plan upravljanja konfiguracijom softvera - Software Configuration Management Plan (SCMP)
- Plan upravljanja softverskim projektom - Software Project Management Plan (SPMP)
- Specifikacija zahtjeva na softver - Software Requirements Specification (SRS)
- Specifikacija dizajna softvera – Software Design Document (SDD)
- Dokumentacija o provjeri softvera - Software Test Documentation (STD)

□ Primjeri: Predlošci

- neki od gore navedenih

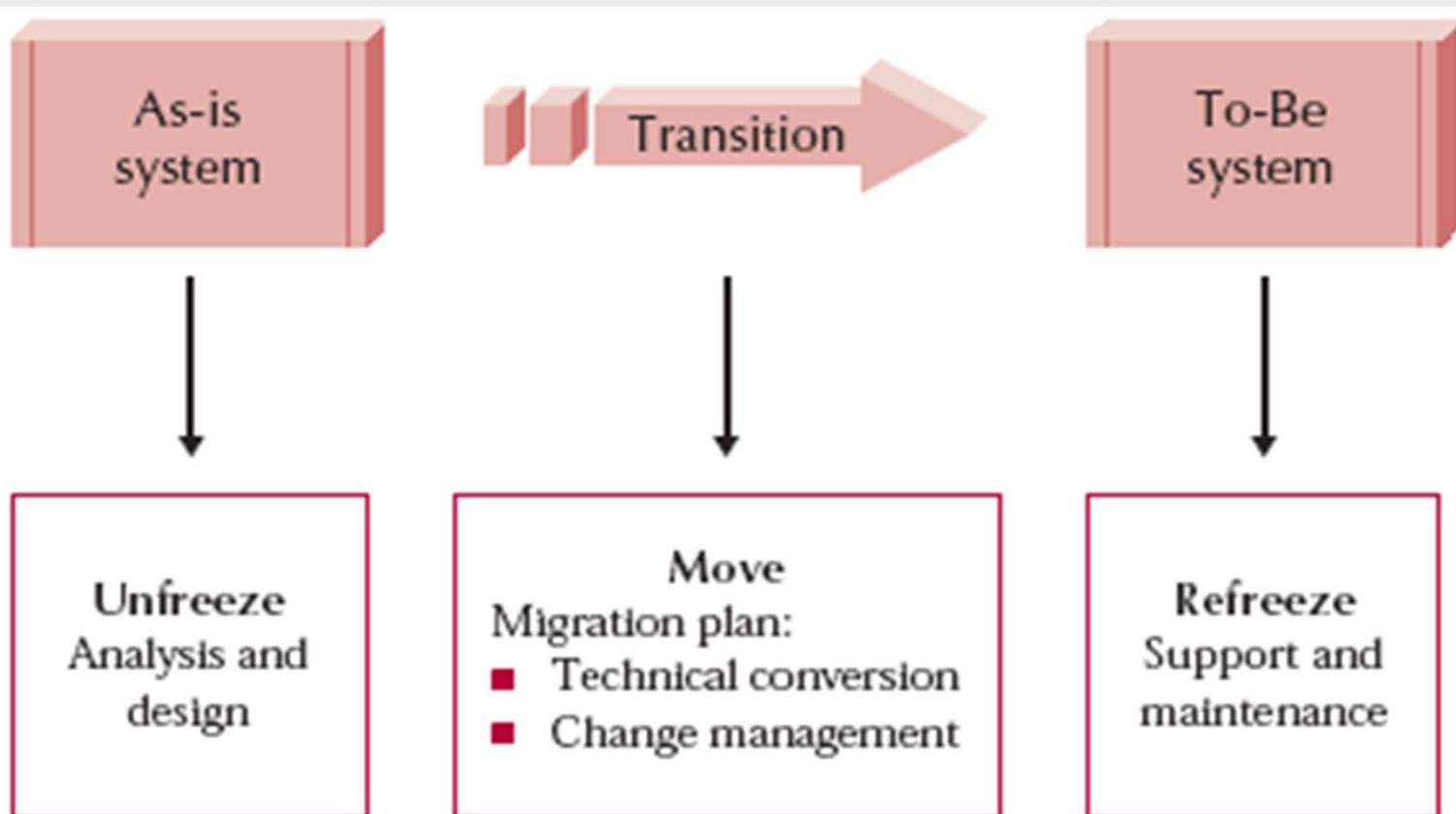
□ Primjeri: Izrada \ Prilozi.zip

□ Primjer: HVJP-upute, CCA-Upute

Primjena

Uvođenje u primjenu

- Uvođenje u primjenu podrazumijeva promjenu u sustavu
- Upravljanje promjenama smatra se najtežim zadatkom te uključuje
 - Odmrzavanje (Unfreezing) – napuštanje starih navika i normi
 - Prijelaz (Moving) – prijelaz sa starog na novi sustav
 - Zamrzavanje (Refreezing) – usvajanje i uhodavanje novog načina rada
- Plan migracije uključuje tehničke ali i organizacijske aspekte



Slika: A.Dennis, B.H.Wixom, D.Tegarden:
Systems Analysis and Design with UML,
John Wiley & Sons

Tehnička konverzija

□ Konverzija sustava

- tehnički proces u kojem novi sustav zamjenjuje stari sustav

□ Glavni koraci – najčešće slijedni na pojedinoj lokaciji

- instalacija hardvera
- instalacija softvera
- konverzija podataka – najsloženija uslijed promjene strukture podataka
 - inicijalni unos podataka, npr. novih šifrarnika
 - prijenos postojećih podataka uz konverziju, najčešće matičnih podataka
 - prijenos zbirnih stanja poslovna transakcija, za tzv. "prometne" podatke
- odgovarajući plan testiranja

□ Konverzija se provodi u 3 "dimenzije"

- stil konverzije (conversion style) – način uvođenja
- lokacija konverzije (conversion location) - gdje
- moduli konverzije (conversion modules) – što, koji dijelovi

Način (stil) konverzije

- **Izravno uvođenje (direct conversion, abrupt cutover, cold turkey, big bang)**
 - početak rada novog sustava uz istovremeni prestanak rada starog sustava
 - u poslovnim sustavima provodi se na određeni dan, uobičajeno datum završetka poslovnog razdoblja, po mogućnosti na kraju tjedna
 - mogući problemi: pojava pogrešaka koje nisu bile uočene tijekom testiranja, nepredviđeno preopterećenje opreme u punom pogonu
 - nedostatak: neposredna izloženost korisnika pogreškama sustava
- **Paralelno uvođenje (parallel conversion)**
 - 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
 - bitno manje rizičan postupak u odnosu na izravno uvođenje
 - nedostatak: potreba za dvostrukom obradom istih podataka, u starom i u novom sustavu → otpor korisnika
- **Primjeri: Foto marketing, D.O.O, Učilište**

Lokacija konverzije

Lokacija konverzije

- dijelovi organizacije smješteni na različitim zemljopisnim lokacijama
- ponekad se zasebnim lokacijama smatraju različite org. cjeline u istom kompleksu (npr. prodaja, otprema, nabava)

Probno uvođenje (pilot conversion)

- izravno/paralelno uvođenje sustava na jednoj lokaciji
- nakon uspješnog pilota, uvodi se na ostalim lokacijama
- prednost: dodatna razina provjere prije širenja, ograničenje problema na probu
- nedostaci: dulje trajanje, razdoblje u kojem dijelovi sustava koriste različite verzije

Postupno uvođenje, fazno uvođenje (phased conversion)

- uvođenje slijedno po grupama lokacija
- karakteristike slične probnom, ali zahtijeva manje ljudi

Istovremeno uvođenje (simultaneous conversion)

- istovremeno uvođenje na svim lokacijama – najčešće izravno
- uklanja heterogenost, ali zahtijeva više ljudi

Primjeri: Ministarstvo, Primarna Zdravstvena Zaštita

Modularnost konverzije

□ Uvođenje cijelog sustava (whole system conversion)

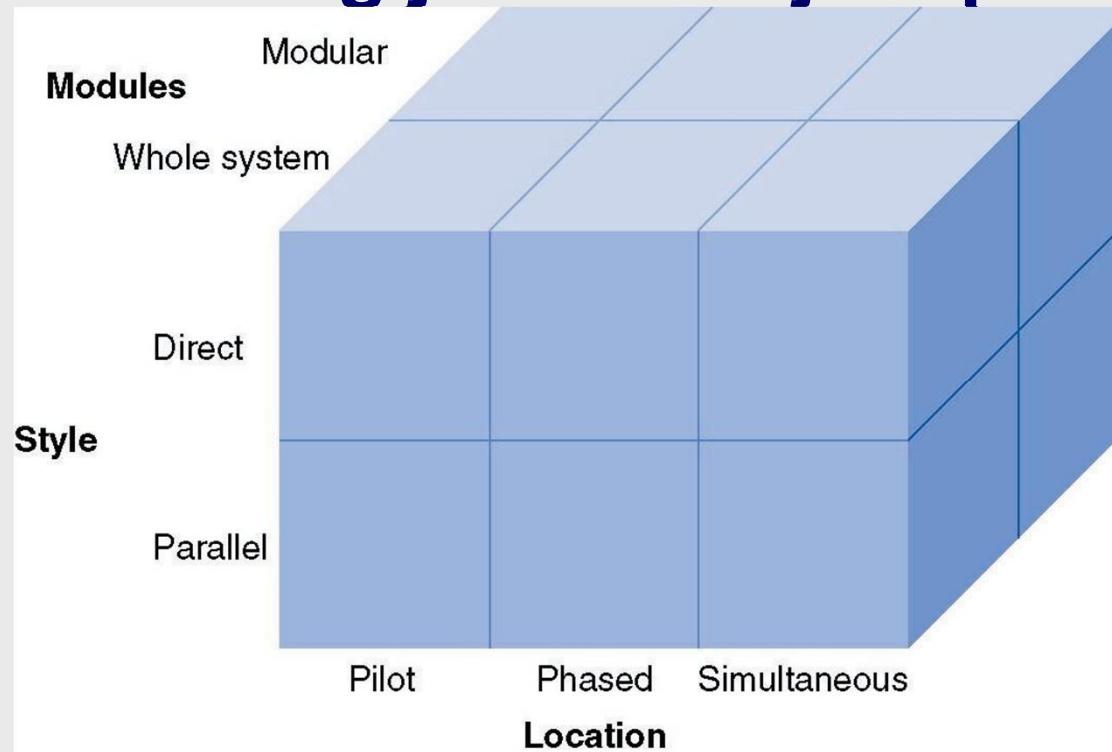
- čitav sustav instalira se odjednom – najčešći način
- u slučaju velikih sustava (npr. ERP), može biti naporno za korisnike

□ Modularno uvođenje (modular conversion, staged conversion)

- postupna zamjena starog sustava novim, uvođenjem po dijelovima
- izvedivo samo ako je moguć istovremeni rad oba (nekompletna) sustava
- problemi: potreba za spojnim programima, tj. premošćivanjem
 - svaki modul treba moći raditi i sa starim i s novim sustavom, ili
 - novi sustav mora moći učahuriti funkcionalnost starog
- prednost: postupni prijelaz, lakša poduka korisnika
- nedostatak: dulje trajanje

□ Primjer: Učilište

Odabir strategije uvođenja u primjenu



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

Upravljanje promjenama

□ Upravljanje promjenama (Change Management)

- podrška procesa prilagodbe korisnika na novi sustav

□ Ključne uloge

- sponzor promjena (sponsor of the change) – osoba koja želi promjenu
 - najčešće poslovnjak koji je pokrenuo zahtjev za novim sustavom ili viši rukovoditelj organizacijske cjeline u koju se uvodi sustav
 - presudno je da bude aktivan jer upravlja onima koji usvajaju sustav
- agent promjena (change agent) – osoba/osobe koje vode promjenu
 - obično netko izvan organizacijske jedinice na koju se promjena odnosi
- usvojitelj promjena (adopter) – osoba/osobe na koje se promjena odnosi
 - ljudi koji će u konačnici koristiti novi sustav

□ Otpor promjenama

- što je dobro za organizaciju, ne mora biti dobro za pojedinca
- promjena radne procedure (drukčiji posao ili više posla) za istu plaću
- promjena zahtijeva napor prilagodbe

Poduka

□ Priprema, obuka (training)

- usvojitelji će prihvati sustav (u)koliko ih se osposobi

□ Sadržaj poduke

- kako obaviti svoj posao (a ne kako koristiti aplikacije) !
- što korisnik treba/može napraviti (a ne što sustav može)

□ Poduka krajnjih korisnika može uključivati:

- opću informatičku kulturu (npr. uporaba osobnih računala)
- funkcije i način upotrebe - korištenje aplikacija
- posebna znanja za obavljanje posla (npr. optimizacija, CAD, GIS)

□ Poduka tehničkog osoblja može uključivati:

- operacijski sustav i uslužne programe
- administriranje baze podataka
- programske jezike, razvojne alate i alate za projektiranje

Način poduke

□ Redoslijed poduke (preporuka)

- poduka tehničkog osoblja za održavanje i potporu korisnika, da ubrza primjenu
- zatim (niže) rukovodstvo, da podupre poduku i primjenu ostalih korisnika
- slijedi poduka (krajnjih) korisnika, prilagođena poslovnim procesima

□ Postupci i tehnike poduke

- tečajevi
- probni rad fazi provjere rada sustava
- kvalitetni sustav interaktivne pomoći
- prikladna dokumentacija
- potpora tijekom primjene

□ Izvođači poduke mogu biti

- unutarnji izvođači - npr. odjel informatike ili grupa za to osposobljenih djelatnika
- vanjski izvođači – konzultanti, specijalizirane ustanove ili visokoobrazovne ustanove

□ Primjeri: HŽ, HŠ, IPISVU

Održavanje

Sistemska potpora i održavanje sustava

□ Post-implementacija, post-produkcija – nakon uvođenja, varijante

- sistemska potpora (system support)
- održavanje sustava (system maintenance)
- ocjena projekta (project assessment)

□ Sistemska potpora

- nakon uvođenja sustav preuzima "operativna grupa" – služba informacijske potpore ili oformljeni centar kompetencija
- pomoći korisnicima korištenju sustava (on-demand training)
- pomoći korisnicima "kad nešto zapne" (hardver, mreža, virusi, ...)
- računalna podrška (online support) – web stranice koje sadrže dokumentaciju, odgovore na često postavljana pitanja (FAQ, ...)
- odjel pomoći (help desk)

□ Primjeri: Sveučilište * 2

Odjel pomoći

□ Višerazinska organizacija podrške

- Prva razina podrške (1st level support) – odgovara na široki raspon zahtjeva
 - očekuje se da razriješi 80% problema
 - inače sastavlja zapisnik problema (problem report) i prosljeđuje 2. razini
- Druga razina podrške (2nd level support) – složenija stručna pomoć
 - mogu biti timovi po struci: npr. desktop tim, mrežni tim
- Ukoliko se ne pronađe rješenje nego se ispostavi da postoji bug
 - zapisnik problema postaje zahtjev za promjenom (change request) koji se prosljeđuje odjelu održavanja

□ Sustav za praćenje problema (issue tracking system, trouble ticket system, incident ticket system)

- sadrži bazu znanja o problemima, rješenjima, korisnicima, ...
- omogućuje praćenje problema korisnika
- sprema kartone (ticket), jednoznačno označene (unique reference number)
 - karton evidentira problem i intervencije

Održavanje sustava

□ Održavanje (IEEE, 1993):

- Modifikacija programskog proizvoda nakon isporuke da bi se ispravili kvarovi, popravile performanse ili druga svojstva ili da bi se proizvod prilagodio promjenama okruženja

□ Održavanje je trajna aktivnost koja započinje odmah po uvođenju

- Bez obzira na to koliko je sustav dobar, kvarovi su neizbjegli!

□ Zahtjev za promjenom (change request)

- manja verzija zahtjeva na sustav (system request) s početka ž.ciklusa

□ Najčešći razlozi (po prioritetu)

- prijava kvara
- zahtjev za poboljšanje, npr. ugradnja nove funkcije
- zahtjevi vanjskih sustava s kojima se očekuje integracija, npr. ISVU-IPISVU
- nove inačice sistemskog softvera, npr. baze podataka ili OS
- zahtjevi višeg rukovodstva, obično radi promjena strategije organizacije

Vrste održavanja

□ 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)

□ Primjeri: Primjena \ MaintIns

Materijali

- M.M. Lehman, Laws of Software Evolution Revisited, Lecture Notes in Computer Science, London, 1996.
 -  Izrada \ Lehman-2
- <http://www.refactoring.com>
- Unit Testing and Generating Source Code for Unit Test ... VS2005 TS
 - <http://msdn.microsoft.com/en-us/library/ms364064.aspx>
- Team Development with Visual Studio Team Foundation Server
 - <http://tfsguide.codeplex.com/>
- Alati za jedinično testiranje
 - <http://junit.org>
 - <http://www.nunit.org/>
 - <http://www.csunit.org/>
 - <http://xunit.codeplex.com/>

Reference

□ Literatura

- A.Dennis, B.H.Wixom, D.Tegarden: Systems Analysis and Design with UML, John Wiley & Sons, 2005
- M. Fowler. Refactoring: Improving the Design of Existing Code, Addison-Wesley Professional, 1999.
- Mike O'Docherty: Object-Oriented Analysis And Design Understanding System Development With Uml 2.0, John Wiley and Sons, 2005.
- McConnell: Code Complete: A Practical Handbook of Software Construction, Microsoft Press; 2nd edition, 2004. <http://www.cc2e.com/>

□ Standardi

- IEEE Std 1008-1987 (R1993), Standard for Software Unit Testing
- IEEE Std 829-1998, Standard for Software Test Documentation
- IEEE Std 730-2002, Standard for Software Quality Assurance Plans
- IEEE Std 1063-2001, Standard for Software User Documentation

Metodologije razvoja

2012/13.10

Metodologije

□ Metodologija (methodology) = metoda + idejni pristup

- *Kolekcija procedura, tehnika, alata i dokumentacijskih pomagala, potkrijepljenih filozofijom, koji potpomažu izgradnju IS [Avison & Fitzgerald]*
- *Skup načela izrade IS, koji se u određenoj situaciji svodi na metodu jedinstveno prikladnu toj situaciji [Checkland]*
- razrađen "plan bitke" ili "kuharica" za postizanje željenog rezultata

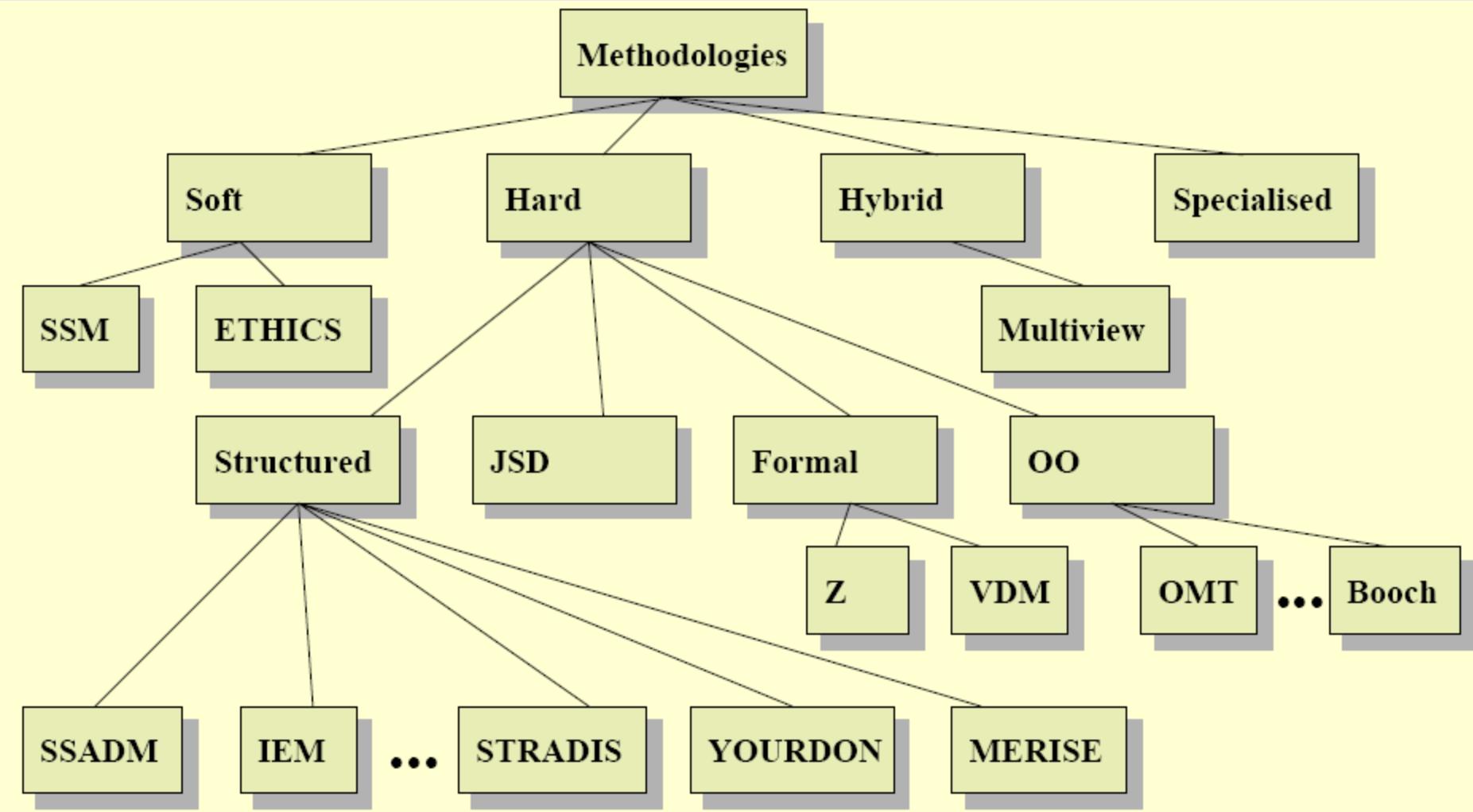
□ Komponente metodologije (Avison & Fitzgerald, 2003)

- definirane faze – varijanta životnog ciklusa
- procedure, zadatci, pravila, tehnike i upute za pojedine faze
- dokumentacija – formati, predlošci i organizacija dokumenata
- alati - preporuke (vodič) uporabe tehnika i pomagala
- upute o upravljanju i organizaciji projekta – upravljanje, nadzor, poduka, ...

□ Cilj

- standardizirani razvojni proces
- bolji i kvalitetniji proizvod

Taksonomija metodologija



Glavne vrste metodologija

□ Čvrste, tvrde (hard), teške (heavy)

- teške ili tvrde - zbog opsežnosti, složenosti i velike količine dokumentacije
- pozadina potječe od tradicionalnog životnog ciklusa
- prvotne, strukturirane su zasnovane na funkcionalnoj dekompoziciji
 - SSADM (Structured Systems Analysis and Design Method), BSP (Business System Planning), CASE*Method, IEM (Information Engineering Methodology, Martin), SADT, STRADIS, ...
- današnje naglašavaju objektno orijentirani pristup
 - RUP – nastala fuzijom ranijih objektnih (Booch, OMT, Objectory)
 - derivati – Open UP, Agile UP, dX (neki spadaju u agilne)

□ Meke (soft), lagane (light), agilne (agile)

- naglasak na socološkim i ljudskim aspektima
- bave se slabo definiranim, neizrazitim (fuzzy edged) stvarnim situacijama
- prvotne meke metodologije - SSM, ETHICS
- metode za brzi razvoj aplikacija (Rapid Application Development Methods) – DSDM
- moderne agilne metode - XP, Crystal, FDD, Scrum

□ Hibridne metodologije (hybrid methodologies)

- kombinacija tvrdih i mekih

Karakteristike metodologija

- **Kuhanje po kuharici (receptu) ne znači da će jelo biti dobro!**
 - Preporučene aktivnosti ne moraju uvijek biti prikladne, primjenjive ili potrebne
 - Korisnik je nepredvidljiv i "prevrtljiv" - zahtjevi se mogu mijenjati u vremenu
 - Inzistiranje na propisanim procedurama vodi u zanemarivanje stvarnih problema
 - Posljedica: formalno dobro napravljen, a neuspješan sustav
- **Nedostaci metodologija (općenito)**
 - Nedostatak standarda – velik broj varijanti postupaka i notacija
 - (Ne)odmjerenost – (pre)komplicirane ili (pre)jednostavne
 - Pokrivenost životnog ciklusa – rijetke podupiru sve faze životnog ciklusa
 - Podržanost alatima – nepotpuna jer proizvođači alata nastoje približno podržati što veći broj notacija
- **Dodaci ili alternative metodologijama**
 - zdrav razum
 - najbolje dokazano u praksi - "Best practices"
 - prečice do rješenja temeljem sličnih iskustava - "Rules of thumb"

Tradicionalne naspram agilnih metodologija

	Proces	Prakse	Struktura projektne ekipe
Tradicionalni pristup	„Težak“: plan strogo definiran na početku – planski usmjeren (eng. plan driven); linearan i predvidiv	Nisu definirane. Gradi se postepeno, a isporučuje samo jednom.	Distribuirane ili kolocirane, uglavnom velike ekipe sa strogo definiranim ulogama
Moderni agilni pristup	„Lagan“: nema strogo definiranog plana, planiranje se odvija kroz cikluse, proces: nelinearan i prilagodljiv, inkrementalan	Sedam osnovnih: samoorganizirajuće ekipe, česta isporuka, planiranje učenja snažna komunikacija, testiranje svega, mjerjenje vrijednosti i „čišćenje puta“	Manje, kolocirane ekipe

Tradicionalne naspram agilnih metodologija (2)

	Dokumentacija	Tipovi programske podrške	Alati
Tradicionalni pristup	Dobro dokumentirane. Najprije dokumentirati, a onda razvijati prema definiranim dokumentima.	Bilo koji tip, ali s različitim rizikom. Inženjerski ili znanstveni. Veliki ili mali projekti.	Različiti alati za svaku fazu s kasnijom integracijom.
Moderni agilni pristup	Malo ili bez dokumentacije. Usmjereno na taktičko znanje – dijeljenje znanja između članova ekipe.	Poslovni sustavi, s promjenjivim zahtjevima. Manji projekti.	Integrirana razvojna okruženja.

Odabir metodoloigije

□ Ne postoji, univerzalni ili najbolji pristup

- Samo jedna metodologija se ne može primijeniti na sve projekte
- Treba identificirati posebnosti projekta na kojem će se raditi i tek onda odabrati prikladnu metodologiju.

□ Selekcija metodologije

- DESMET - metoda za evaluaciju metodologija i alata

□ Integracija metoda

- Prevladava mišljenje da su potrebna oba pristupa (tradicionalni i agilni)
- Kombiniranje postupaka i tehnika za pojedine faze

Strukturirane metodologije

SSADM

□ Structured Systems Analysis and Design Method

- najpoznatija i najčešće korištena
- podatkovno-vodjena ("data-driven") metodologija

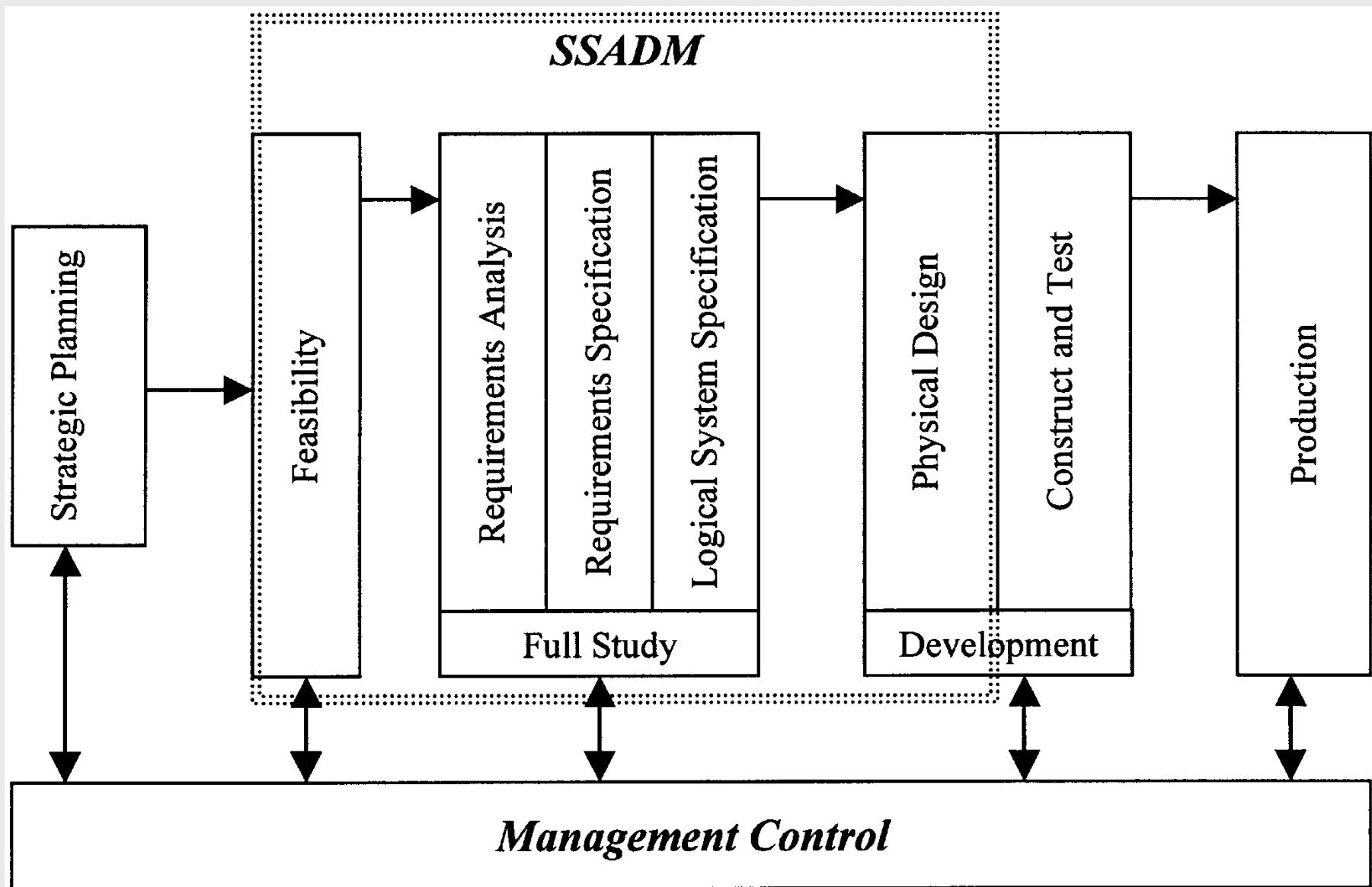
□ Faze – moduli, dalje razloženi u stadije (stages)

- Studija izvodljivosti - izvedivost
- Analiza zahtijeva - analiza postojeće okoline, opcije poslovnog sustava
- Specifikacija zahtijeva - definicija zahtijeva
- Specifikacije logičkog sustava - alternative tehničkog sustava, logički dizajn (strukturnim kartama)
- Fizički dizajn – specifikacija i raspodjela opreme i potpore

□ Glavne tehnike

- logičko oblikovanje podataka - Logical Data Structure (LDS), slično ERD
- oblikovanje toka podataka - Data Flow Modelling, DFD dijagrami
- oblikovanje događaja entiteta - Entity Life Histories (ELH)

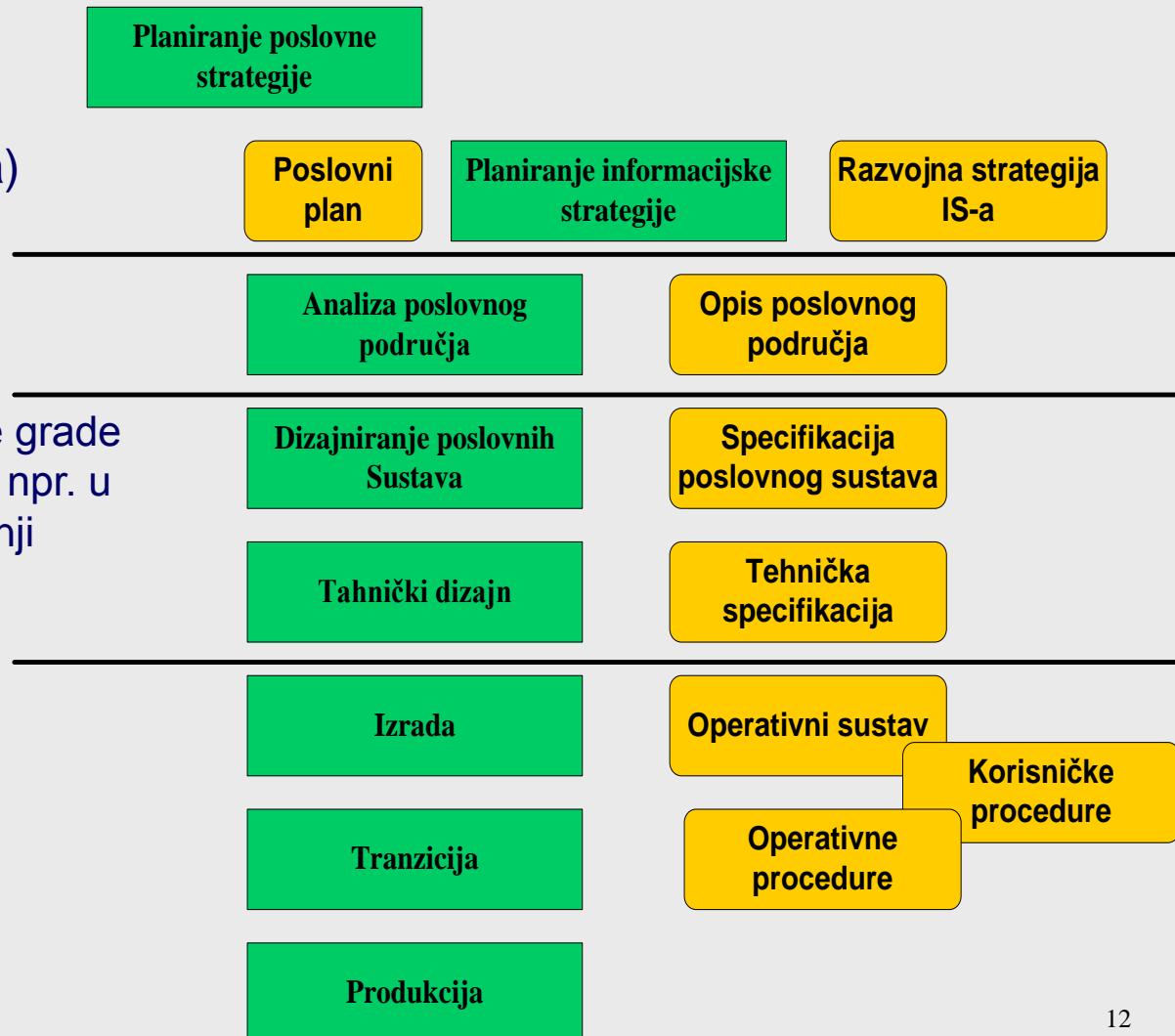
Životni ciklus prema SSADM



Metodologija informacijskog inženjerstva

- Information Engineering Methodology (IEM) – James Martin
- Razvojni okvir (“framework”) s 4 stadija

- Planiranje (strategija)
- Analiza
- Dizajn (projekt sustava)
- Konstrukcija (izrada)



□ Osnovno načelo

- IS treba graditi kao što se grade drugi “unikatni” proizvodi, npr. u graditeljstvu ili brodogradnji

Životni ciklus IE projekta

□ Ključne faze IE

- Planiranje strategije IS - Information Strategy Planning (ISP)
 - promatranje poslovanja kao cjeline s ciljem definiranja općeg, sveobuhvatnog plana i arhitekture za slijedni razvoj inf. (pod)sustava
 - izdvajanje poslovnih područja i pridjeljivanje prioriteta
 - poslovno područje – skup poslovnih procesa koji se protežu organizacijom a moraju biti visoko integrirani da bi se ostvarila strategija ili misija
- Analiza poslovnih područja - Business Area Analysis (BAA)
 - proučavanje poslovnih područja i definiranje poslovnih zahtjeva za organizirani i integrirani skup inf. (pod)sustava i aplikacija
 - definiranje aplikacija
 - izdvajanje aplikacija i definiranje prioriteta temeljem analize PP
 - aplikacije - projekti u kojima se primjenjuju drugi postupci analize i dizajna

□ Podatkovno usmjerena paradigma

- Budući da je informacija proizvod podataka, podaci moraju biti planirani prvi!
- Zatim se rade modeli procesa slično onima u strukturiranoj analizi

Ujedinjeni razvojni proces

RUP i derivati

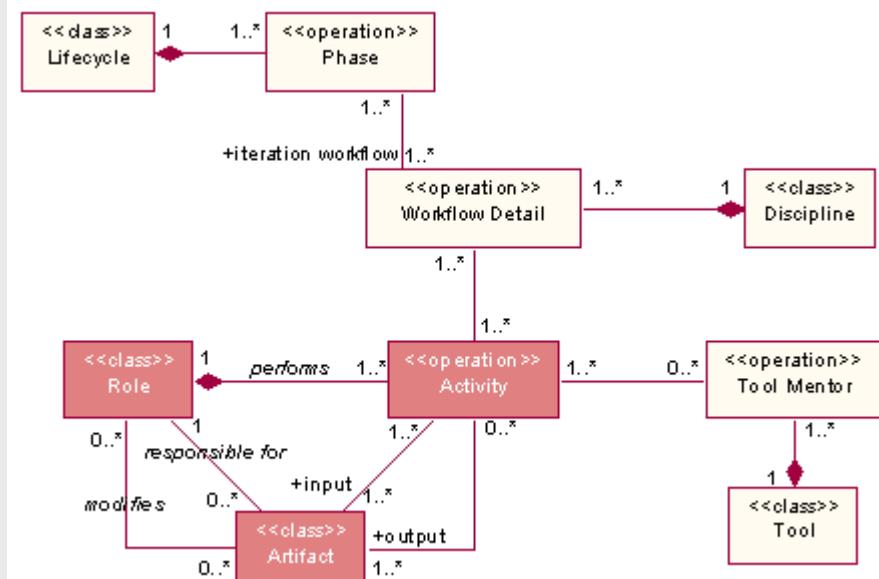
Ujedinjeni razvojni proces softvera

□ Unified software development process (UDP)

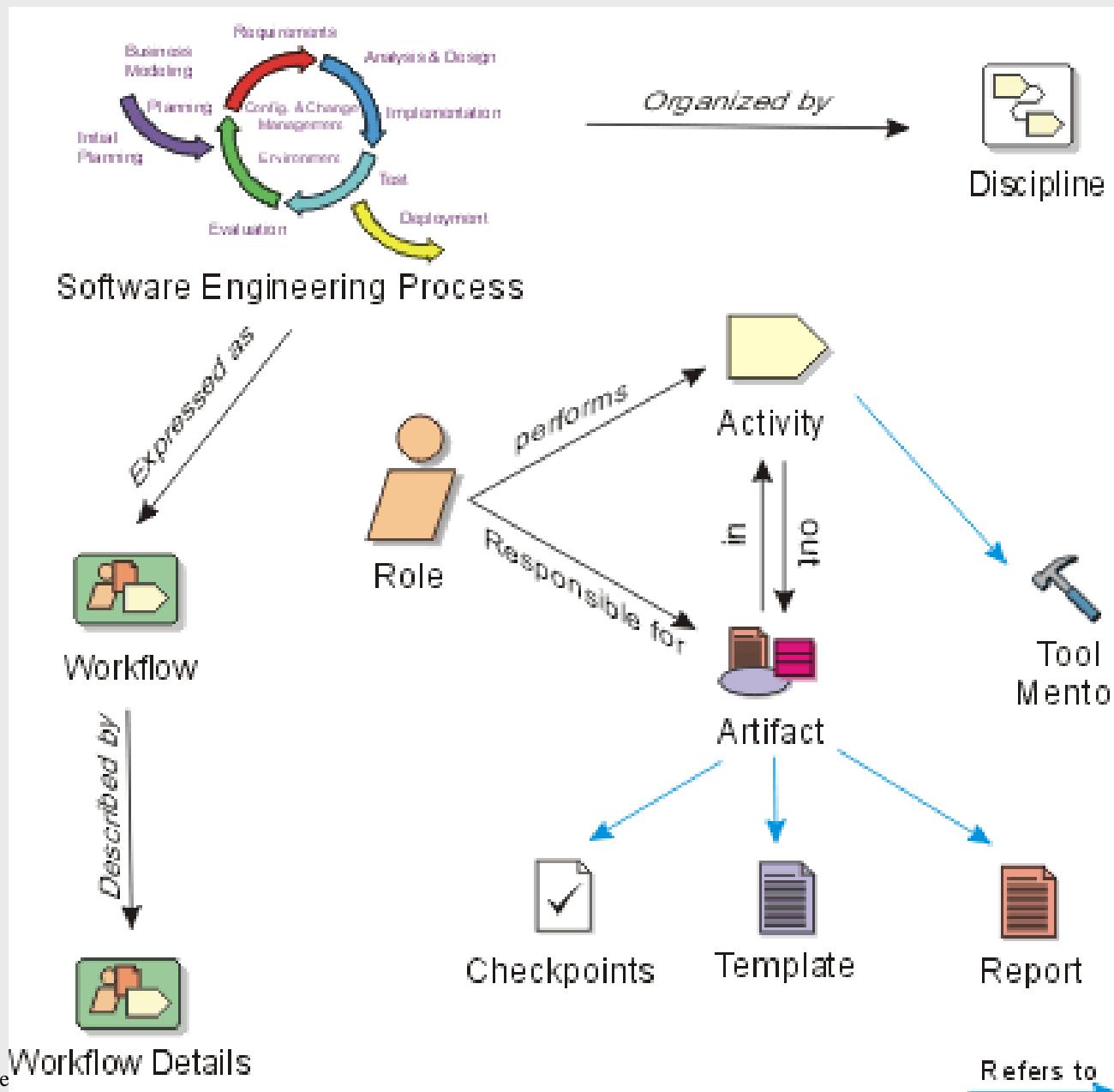
- okruženje za razvoj softvera temeljeno na iterativnom i inkrementalnom procesu
- najpoznatija komercijalna inačica – IBM Rational Unified Process (RUP)

□ RUP definiraju tri komponente

- skup filozofija, ključnih praksi (core practices, best practices) te načela (process essentials) za uspješan razvoj programske podrške
- metodološki okvir izgrađen od višekratno iskoristivih metoda i procesnih blokova
- jezik za definiranje procesa
 - process meta-model (ova slika)
 - elementi za definiranje procesa (sljedeća slika)



Struktura procesa – osnovni elementi



Glavne karakteristike URP-a

□ Produktivnost ekipe

- baza znanja (web) – upute, predlošci, korištenje alata
- osigurava da svi članovi dijele zajednički jezik, proces i pristup razvoju

□ Vizualno modeliranje

- naglašava razvoj i održavanje modela umjesto papirnate dokumentacije

□ Oslonac na UML

- tzv. industrijski standard

□ Podržanost alatima

- izrada i održavanje raznih artefakata (pretežno modela), vizualno modeliranje, programiranje, testiranje itd.
- alati pomažu razvoj ali i procese upravljanja izmjenama i konfiguracijom

□ Iterativan i inkrementalan razvoj

□ Elastičnost i prilagodljivost

- prema veličini ekipa ili tipu projekta
- RUP sadrži niz "predložaka" razvojnih procesa (roadmaps) za različite modele razvoja i tipove projekata

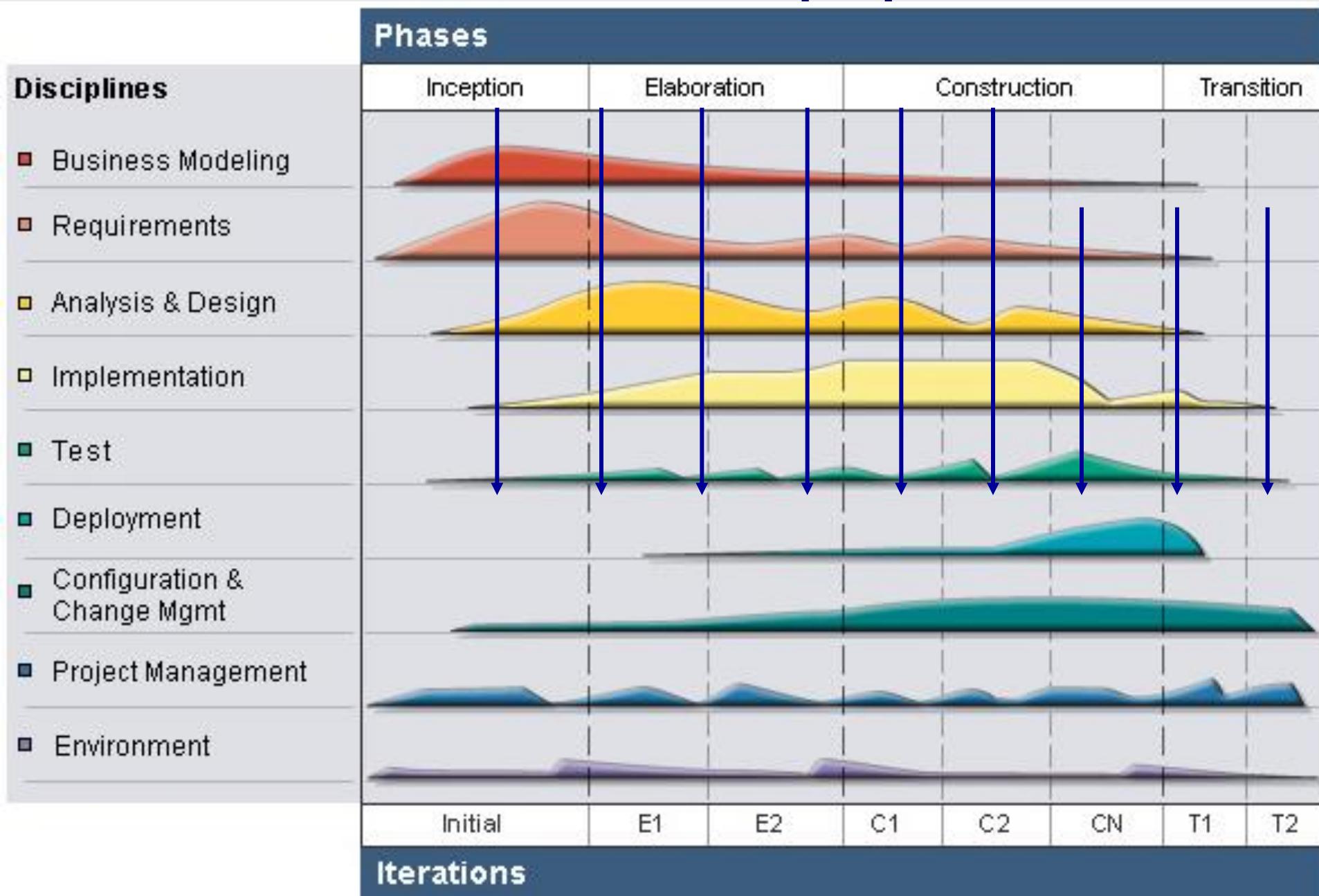
Načela (Process essentials)

- **Vizija: Definirati viziju**
 - artefakt Vizije – dokument zahtjeva visoke razine i ograničenja u oblikovanju
- **Plan: Upravljati prema planu**
 - dokumentiranje *Planom razvoja softvera (Software Development Plan)*
- **Rizici: Ublažiti rizike i pratiti probleme**
 - *Registar rizika (Risk List)* s prioritetima i s planom razrješenja
- **Poslovni slučaj: Istražiti poslovni slučaj**
 - Poslovni slučaj opravdava ulaganje u projekt i povrat investicije (ROI)
- **Arhitektura: Oblikovati arhitekturu zasnovanu na komponentama**
 - arhitektura kao struktura komponenti koje komuniciraju putem sučelja
 - *Dokument arhitekture softvera (Software Architecture Document)*

Načela (Process essentials) - nastavak

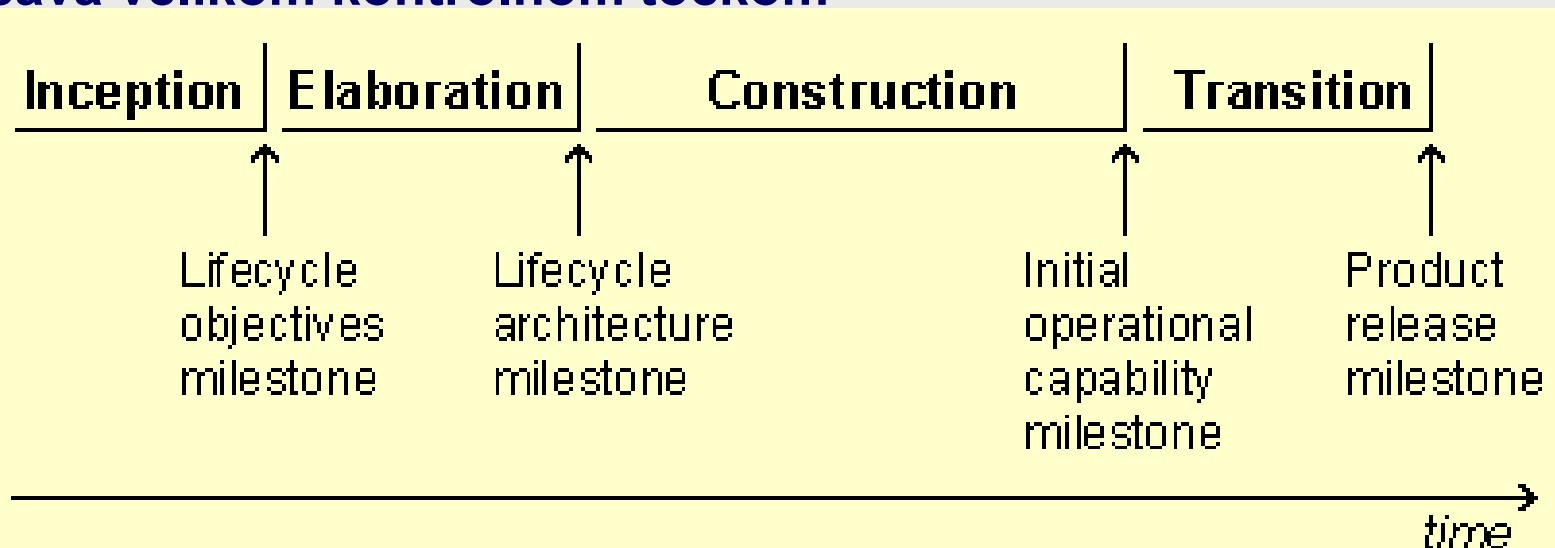
- **Prototip:** Inkrementalno graditi i testirati proizvod
 - iterativan i inkrementalan pristup - što ranije otkrivanje i razrješenje problema
- **Evaluacija:** Redovito procjenjivati rezultate
 - izvješća o statusu – identifikacija, komuniciranje i razrješenje problema
 - procjena na kraju svake iteracije – ocjena rezultata i prijedlog promjena
- **Zahtjevi za promjenama:** Upravljati izmjenama i kontrolirati izmjene
 - zahtjevi moraju biti predloženi i obrađeni kroz konzistentan proces
 - bilježenje odluka te osiguravanje da svi članovi tima razumiju promjene
- **Korisnička podrška:** Isporučiti iskoristiv proizvod
 - proizvod je više od programske podrške – korisnička i sistemska dokumentacija
- **Proces:** Osigurati proces koji odgovara projektu
 - proces razvoja mora odgovarati tipu rješenja koje se izrađuje
 - nakon što je proces odabran, treba ga shvatiti fleksibilno i stalno prilagođavati

Životni ciklus RUP projekta



Faze, discipline i kontrolne točke

- **Horizontalna os predstavlja vrijeme i aspekte životnog ciklusa**
 - cikluse, faze, iteracije i kontrolne točke
- **Vertikalna os predstavlja discipline - logički grupirane aktivnosti**
 - u ranijim fazama naglasak je na poslovnom modeliranju i zahtjevima
 - u kasnijima na implementaciji, testiranju i ugradnji
 - te upravljanju izmjenama i konfiguracijom
 - disciplina upravljanja projektom ujednačenog je intenziteta.
- **Osnovicu životnog ciklusa čine četiri slijedne faze od kojih svaka završava velikom kontrolnom točkom**



Glavne discipline

- **Poslovno modeliranje (Business Modeling)**
 - Identificira poslovni kontekst sustava i oblik organizacije
 - Definiraju se ciljevi i okvirna funkcionalnost, poslovna pravila i sl.
- **Requirements (Zahtjevi)**
 - Definira kako saznati i prikupiti želje te ih pretvoriti u skup zahtjeva
- **Analiza i dizajn (Analysis & Design)**
 - Definira pretvorbu zahtjeva u dizajn
 - Analiza usmjerena na logički pogled i funkcionalne zahtjeve
 - Dizajn usmjerjen na fizički pogled i nefunkcionalne zahtjeve
- **Implementacija (Implementation)**
 - Kako razviti, organizirati, testirati i integrirati komponente
- **Provjera (Test)**
 - Kako testirati i procijeniti kvalitetu rješenja
- **Uvođenje u primjenu (Deployment)**
 - Aktivnosti potrebne da sustav bude dostupan krajnjim korisnicima

Podupiruće discipline

- **Upravljanje konfiguracijom i promjenama (Configuration & Change Management)**
 - kako kontrolirati i sinkronizirati evoluciju skupa komponenti i isporuka
- **Upravljanje projektom (Project Management)**
 - planiranje projekta, upravljanje rizicima, praćenje napretka i metriku
- **Okolina (Environment)**
 - organizira dijelove metodologije, procese i alate kao okruženje timu
- **Životni ciklus iteracije : mini-vodopad (Mini-Waterfall)**
 - usitnjeni standardni životni ciklus razvitička
 - zasnovan i vođen na slučajevima korištenja

Glavne faze razvoja - Počinjanje

□ Faza incepcije (Inception phase) - Počinjanje

- Formuliranje opsega projekta
 - opis problemskog konteksta te najvažnijih zahtjeva i ograničenja
 - **prikupljanje najvažnijih zahtjeva (10% detaljno)**
 - preporuča se istaknuti i kritične scenarije korištenja (UC scenariji)
- Inicijalna procjena ukupnog troška, vremena i rizika
- Planiranje i priprema poslovnog slučaja
- Izrada prijedloga arhitekture
 - demonstrirati izvedivost simulacijom, inicijalnim prototipom i sl.
- Priprema okruženja za projekt
 - Procjena projekta i organizacije, odabir alata, razvojnih okruženja i procesa

Glavne faze razvoja - Elaboracija

□ Faza elaboracije (Elaboration phase) - Razrada

- Definiranje, validacija i zacrtavanje arhitekture
- Osiguranje da su arhitektura, zahtjevi i planovi stabilni, a rizici ublaženi
 - tako da se može pouzdano odrediti trošak i završetak projekta
- Prikupljanje detaljnih zahtjeva (80%)
- Ažuriranje vizije projekta
 - razumijevanjem kritičnih UC koji su ujedno i nositelji najvećih rizika
- Izrada plana iteracija za fazu konstrukcije
- Dorada razvojnog procesa i uspostava razvojnog okruženja
 - uključujući proces, alate i podršku za automatizaciju
- Dorada arhitekture i odabir komponenti
 - procjena potencijalnih komponenti - cijena i trajanje naredne faze

Glavne faze razvoja - Konstrukcija

□ Faza konstrukcije (Construction phase) - Izgradnja

- Upravljanje resursima, kontrola projekta i optimizacija procesa
 - paralelni razvoj nekoliko razvojnih timova s ciljem ubrzanja razvoja
- Završetak iterativnog i inkrementalnog razvoja konačnog proizvoda
 - provjera prihvatljivosti
 - podrazumijeva dovršetak analize, dizajna, razvoja i testiranja
- Procjena razvijenih isporuka naspram definirane *Vizije projekta*
- Provjera da li su programska podrška, lokacije i korisnici spremni za beta isporuku

Glavne faze razvoja - Tranzicija

□ Faza tranzicije (Transition phase) - Prijelaz

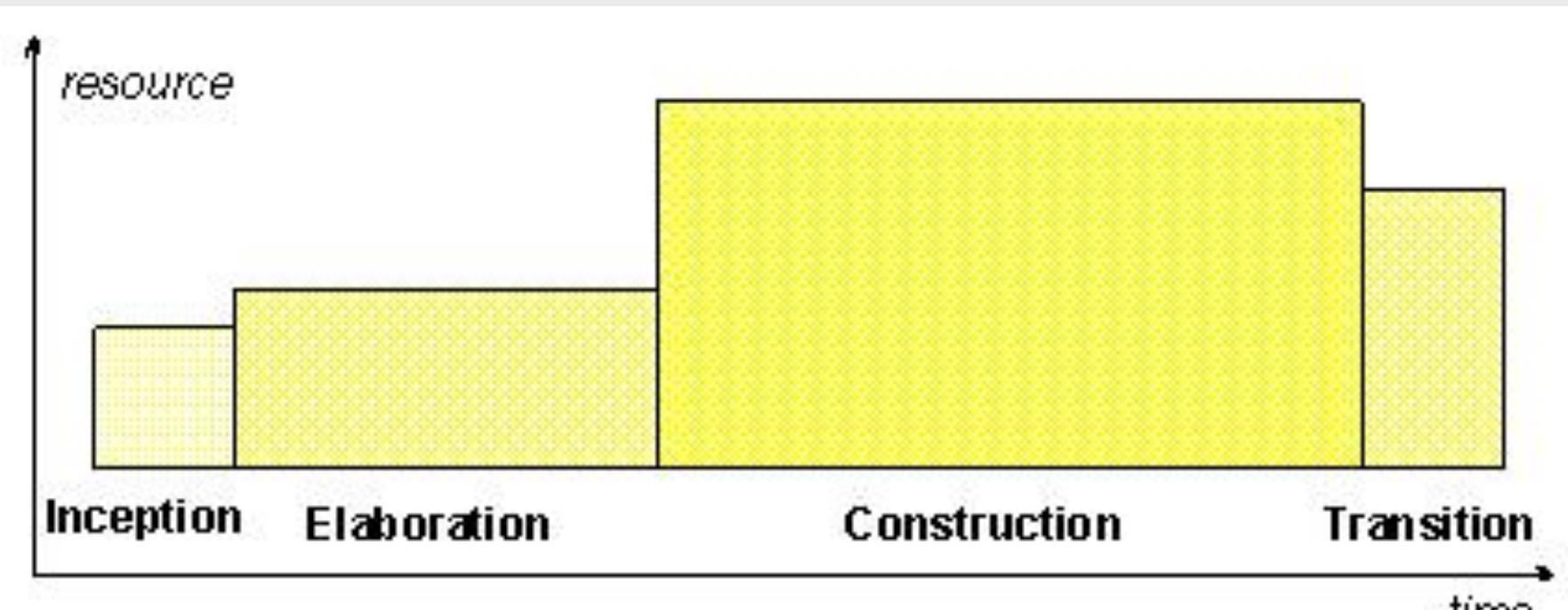
- Izvršavanje planova uvođenja u primjenu
- Dovršavanje korisničke dokumentacije i uputa
- Poduka krajnjih korisnika i održavatelja
- Testiranje programskog rješenja na lokaciji isporuke
- Izrada isporuke (release) konačnog programskog rješenja
- Prikupljanje povratne informacije od krajnjih korisnika
- Fino podešavanje rješenja (popravak manjih pogrešaka, poboljšanje performansi) na temelju povratne informacije
- Omogućavanje proizvoda dostupnim svim krajnjim korisnicima

Procjena napora i trajanja pojedine faze

□ Broj i trajanje iteracija, općenito

- za projekte do 18 mjeseci, otprilike 3 do 6 iteracija

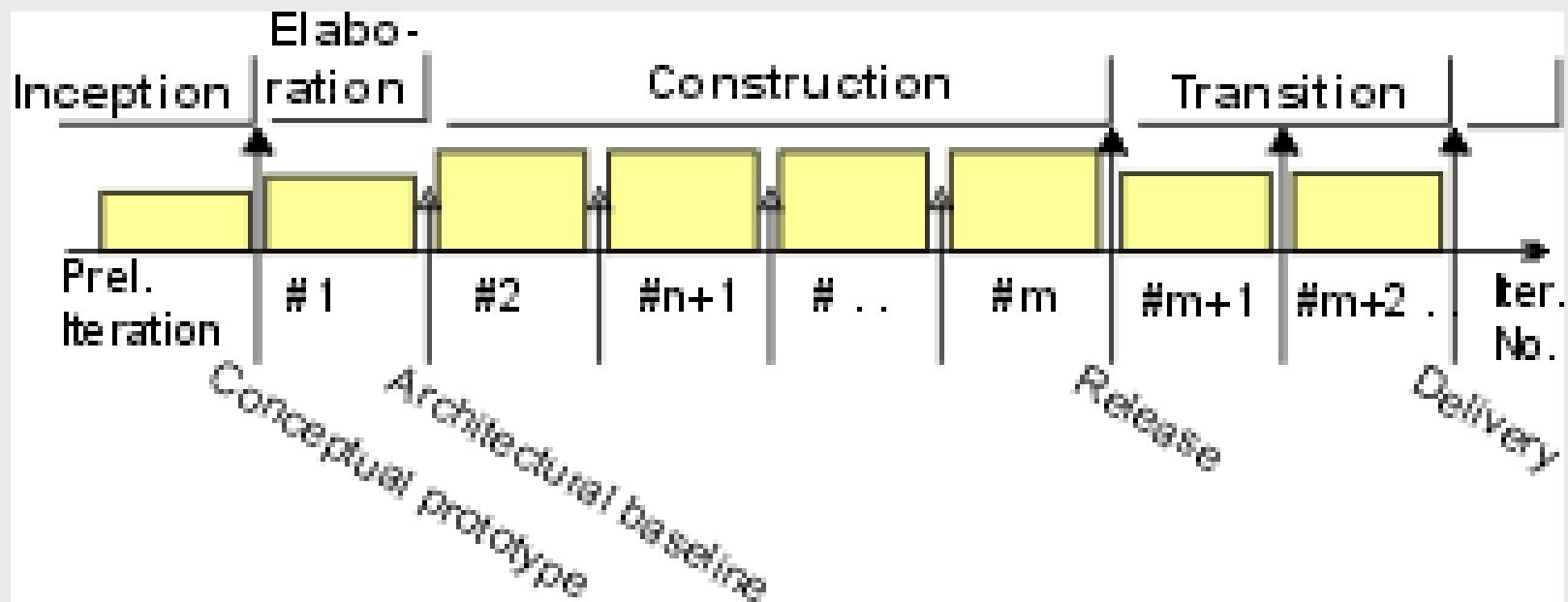
	Počinjanje (Inception)	Elaboracija (Elaboration)	Konstrukcija (Construction)	Prijelaz (Transition)
uložen napor (u odnosu na ukupan)	5%	20%	65%	10%
trajanje (u odnosu na ukupno)	10%	30%	50%	10%



Strategije iterativne provedbe RUP projekta (1)

□ Inkrementalna strategija

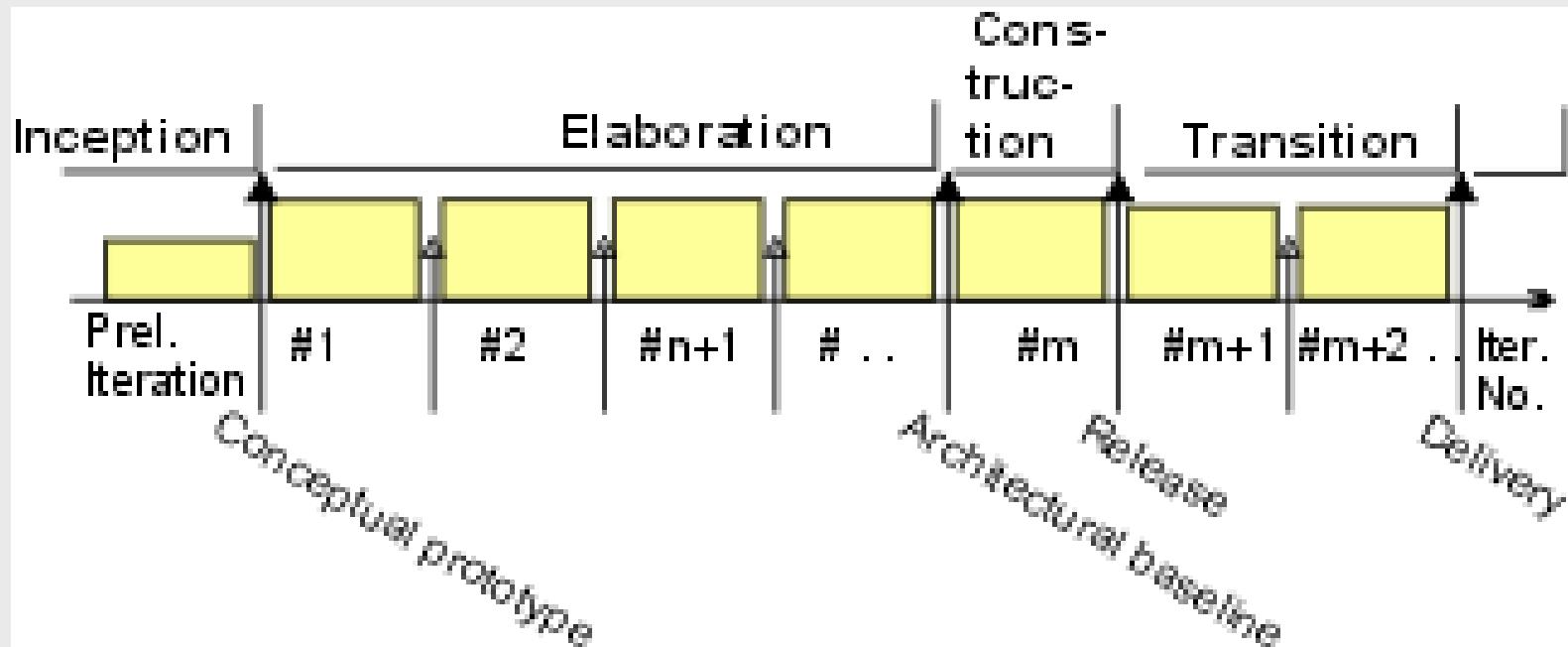
- poznata domena problema
- vrlo dobro poznati rizici
- iskusan projektni tim



Strategije iterativne provedbe RUP projekta (2)

□ Evolucijska strategija

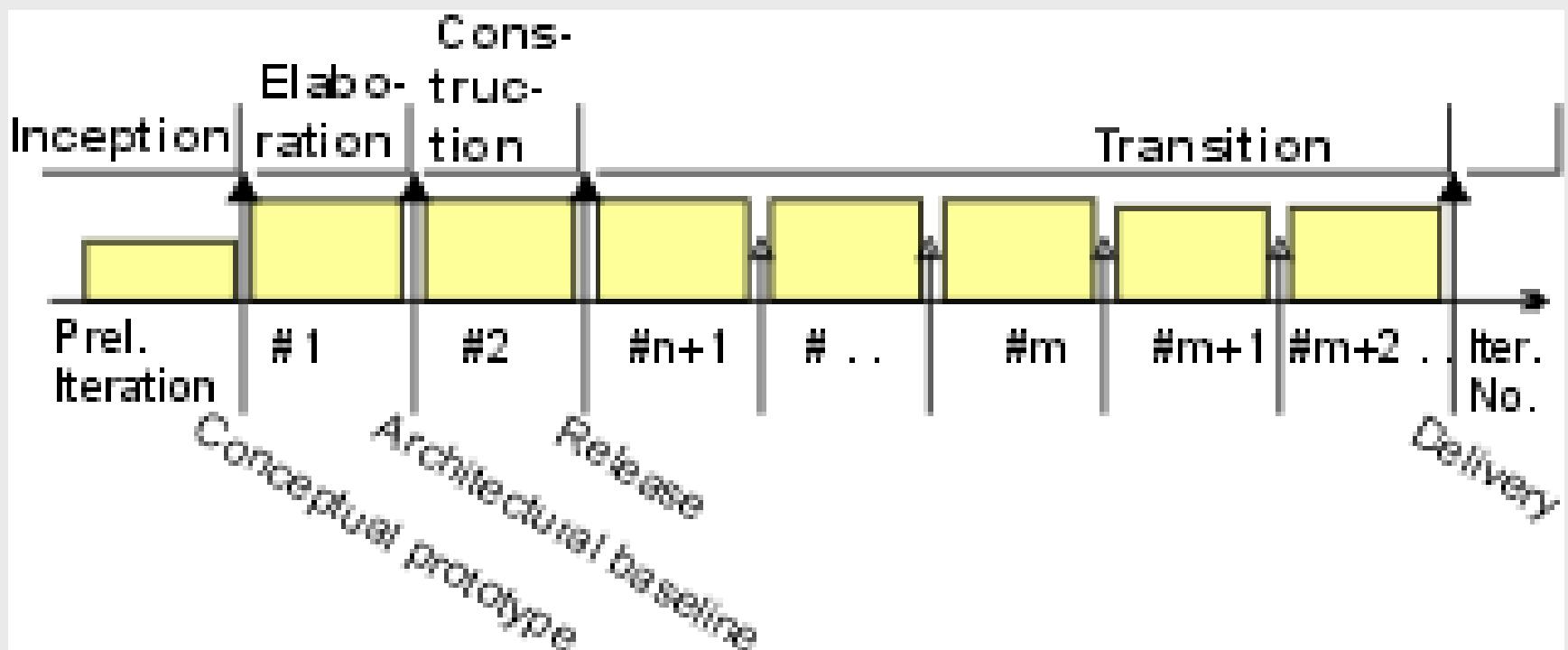
- nova ili nepoznata domena problema
- neiskusan projektni tim



Strategije iterativne provedbe RUP projekta (3)

□ Strategija inkrementalne isporuke

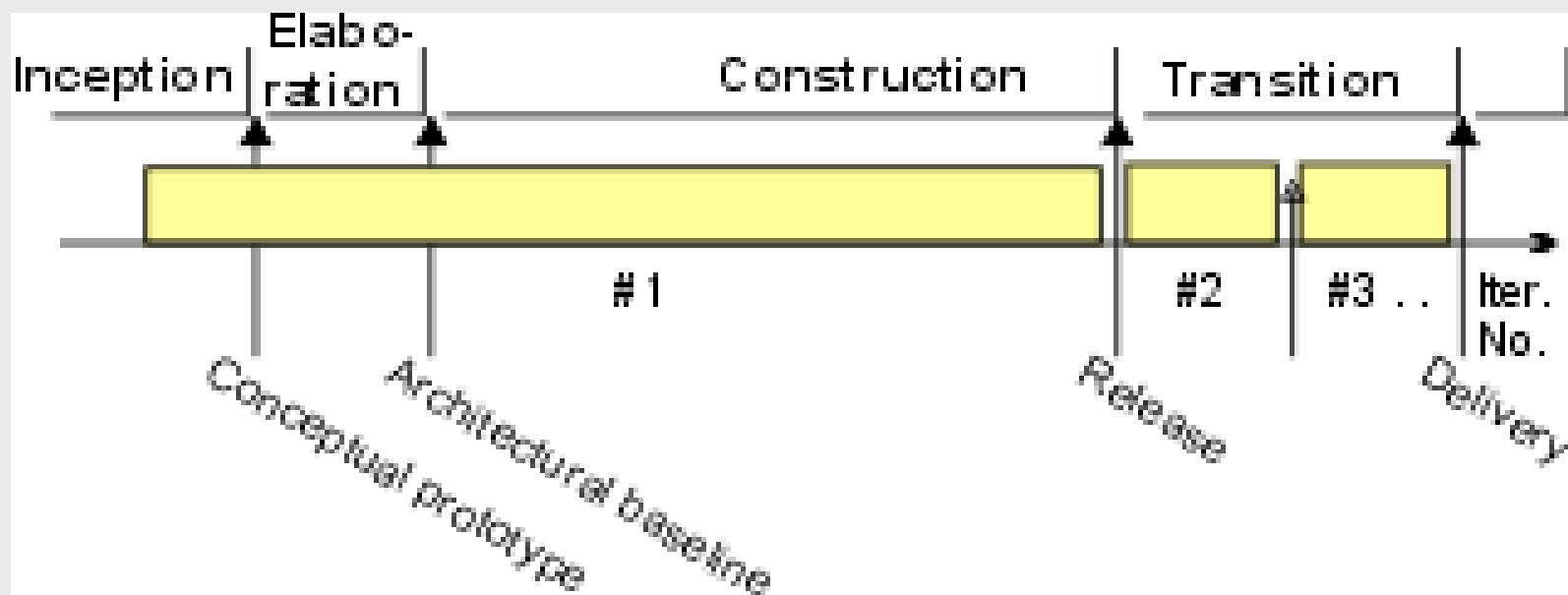
- poznata domena problema
- arhitektura i zahtjevi mogu biti stabilizirani rano u razvojnom ciklusu
- malo novog i nepoznatog u projektu
- iskusan projektni tim
- inkrementalne isporuke su vrlo važne i imaju visoku vrijednost za korisnika



Strategije iterativne provedbe RUP projekta (4)

□ Strategija “Velikog oblikovanja” (vodopadni model)

- dodaje se inkrement dobro definirane funkcionalnosti na vrlo stabilan proizvod
- nova funkcionalnost je vrlo dobro definirana i shvaćena
- tim posjeduje iskustvo kako u domeni tako i sa postojećim proizvodom



RUP kao programski proizvod

- Za sve faze dostupne su detaljne upute koje se sastoje od:
 - sažetka i svrhe faze, ključnih ciljeva i aktivnosti
 - grafičkog prikaza procesa i aktivnosti unutar faze u vidu strukturne raščlambe posla (Work Breakdown Structure)
 - za svaku aktivnost dostupni su koraci izvođenja, uloge koje ju obavljaju, ulazne i izlazne informacije, primjeri i predlošci za vezane isporuke
 - detaljnu raspodjelu uloga po aktivnostima faze
 - popis poželjnih isporuka po aktivnostima faze, uključujući detaljne upute o svrsi, sadržaju i obliku svake isporuke

- Primjeri:
 - IBM Rational Method Composer
 - RUP(R) Configuration for Microsoft(R) .NET Developers

Inačice RUP-a

- *RUP – Rational Unified Process* – IBM (Rational) razvojni proces
- *EUP – Enterprise Unified Process* – proširenje na „poduzeće“
- *OUM – Oracle Unified Method* – Oracleov
- *BUP – Basic Unified Process* – pojednostavljenje, prethodnik OpenUP
- *OpenUP – Open Unified Process* – agilni, Eclipse Process Framework
- *AUP – Agile Unified Process* – agilna inačica Scotta W. Amblera (IBM)
- *EssUP – Essential Unified Process* – pojednostavljenja inačica Ivara Jacobsona temeljena na tzv. „praksama“ (*practice*)

□ Faze procesa

Faze	RUP	EUP	OpenUP	Agile UP
Početak (<i>Inception</i>)	X	X	X	X
Razrada (<i>Elaboration</i>)	X	X	X	X
Izgradnja (<i>Construction</i>)	X	X	X	X
Prijelaz (<i>Transition</i>)	X	X	X	X
Producija (<i>Production</i>)		X		
Umirovljenje (<i>Retirement</i>)		X		

Discipline procesa

Discipline	RUP	EUP	Open UP	Agile UP
Poslovno modeliranje	X	X		X
Zahtjevi	X	X	X	
Analiza i dizajn	X	X	X	
Implementacija	X	X	X	X
Testiranje	X	X	X	X
Postavljanje	X	X		X
Upravljanje konfiguracijom i promjenama	X	X		X
Upravljanje projektom	X	X	X	X
Okruženje	X	X		X
Operacije i podrška		X		
Poslovno modeliranje poduzeća		X		
Upravljanje portfeljem		X		
Arhitektura poduzeća		X		
Strateško ponovno korištenje		X		
Upravljanje kadrovima		X		
Administracija poduzeća		X		
Proces poboljšanja softvera		X		

- Open UP: Analiza i dizajn zove se Arhitektura, a implementacija Razvoj

Agilni postupci razvoja

Proglas agilnosti

- **agilitas (lat.) - svojstvo brzine, okretnosti, hitrosti, lakoće, radinosti**
- **Manifest agilnosti (objava, proglas)**
 - Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas
 - skijalište Snowbird, Utah, 2001.
- *Pojedinci i interakcije* važniji od procesa i alata
- *Softver koji radi* važniji od sveobuhvatne dokumentacije
- *Suradnja s naručiteljem/korisnikom* važnija od pregovora o ugovoru
- *Odziv na promjenu* važniji od praćenja plana

Načela (principi) agilnosti

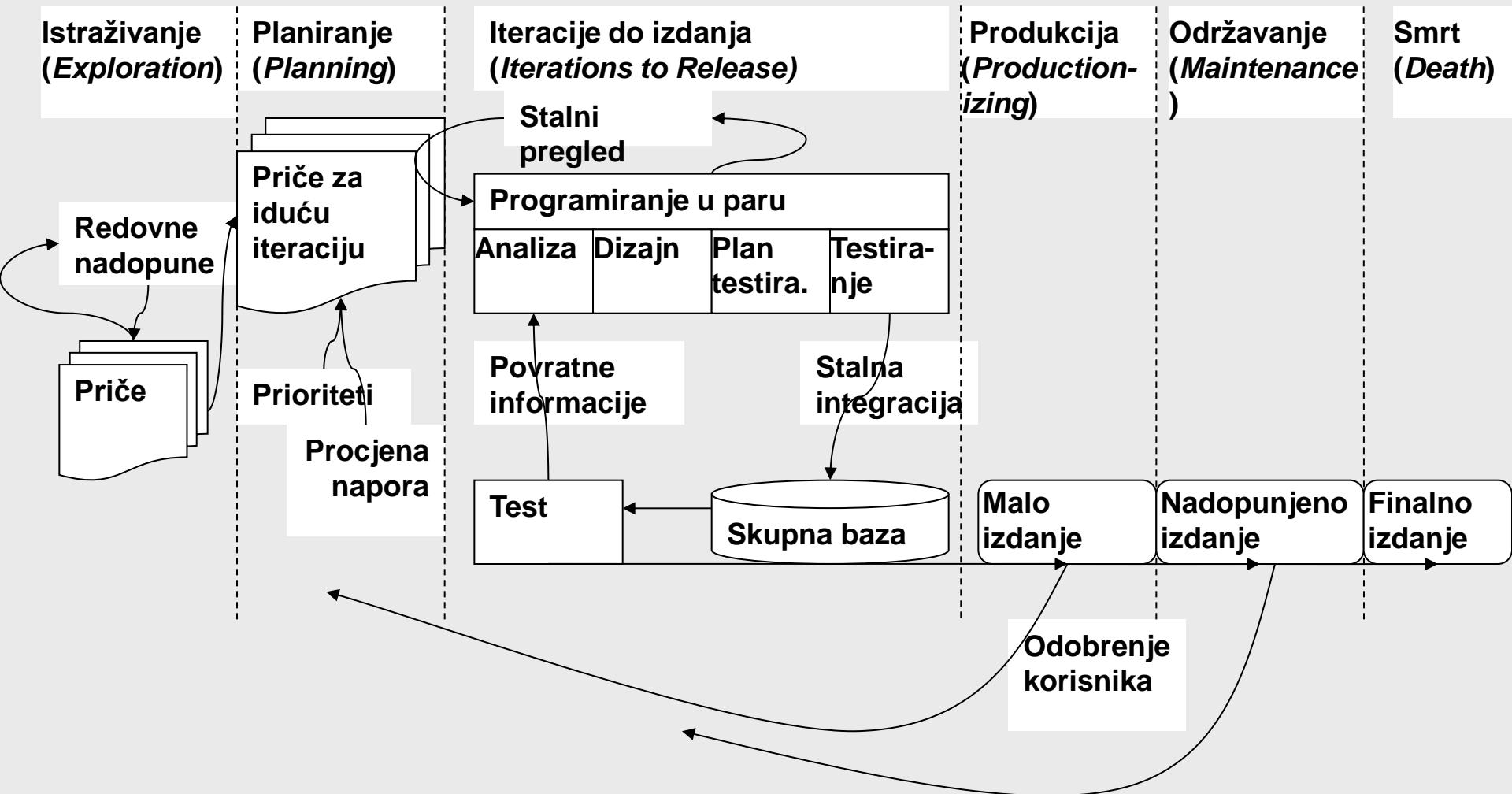
- Zadovoljstvo korisnika ranim i kontinuiranim isporukama softvera**
- Promjene zahtjeva se željno prihvaćaju, čak i u kasnoj fazi razvoja**
- Česta i što ranija isporuka softvera koji radi – 2x/mj do 1x/nekoliko mj**
- Česta (dnevna) suradnja „poslovnjaka“ i razvojnika**
- Motiviranje pojedinaca za rad u projektu – okruženje, povjerenje**
- Usmena komunikacija - najbolja metoda za razmjenu informacija**
- Glavna mjera napretka - softver koji radi**
- Održivi razvoj - sponzori, tim i korisnici - održavati stalni tempo**
- Kontinuirana pažnja na tehničku izvrsnost i dobar dizajn**
- Jednostavnost je nužna**
- Najbolje arhitekture, zahtjevi i dizajn - iz samoorganiziranih timova**
- Tim spoznaje kako postati efektivniji, a zatim se često prilagođava**

Agilne metode

Naziv metode	Autor	Karakteristika
Ekstremno programiranje (XP - eXtreme Programming)	Kent Beck	tehnički orijentirana
Scrum (Scrum)	Ken Schwaber	upravljanje procesom
Kristalne metode (Crystal methods)	Alistair Cockburn	organizacijski orijentirane; zasebna metoda za određenu svrhu
Razvoj vođen mogućnostima (FDD - Feature Driven Development)	Jeff De Luca, Stephen Palmer	orientirana na oblikovanje i implementaciju
Metoda za dinamički razvoj sustava (DSDM – Dynamic Systems Development Method)	DSDM konzorcij	prilagodba funkcionalnosti sustava vremenu i resursima
Prilagodljivi razvoj programske podrške (ASD – Adaptive Software Development)	James A. Highsmith III	prilagođavanje u svrhu ostvarenja ciljeva
Razvoj programske podrške otvorenim kodom (OSS – Open Source Software Development)	Richard Stallman	dostupan izvorni kod; Na granici tradicionalne i agilne
Agilno modeliranje (AM - Agile Modeling)	Scott W. Ambler	modeliranje; definiranje zahtijeva; faza analize i dizajna
Vizljasti razvoj (LD – Lean Development)	Ron Masticelli, Mary Poppendieck	postizanje ušteda po uzoru na (auto) industriju

Ekstremno programiranje (eXtreme Programming)

□ Životni ciklus



Faze ekstremnog programiranja (1)

□ Istraživanje

- Korisnici bilježe svoje priče (⁉)
- Svaka kartica sadrži jednu mogućnost (feature) programa.
- Projektni tim se upoznaje s alatima, tehnologijom i postupcima projekta
- Radi se prototip sustava za testiranje tehnologije i varijanti arhitekture
- trajanje: nekoliko tjedana do nekoliko mjeseci

□ Planiranje

- Postavlja prioritete na korisničke priče (tj. svojstva programskog rješenja)
- Planira se doseg prvog malog izdanja i vrijeme za pojedinu karticu
- Zatim se određuje cjelokupni vremenski raspored
- Rok za izdavanje prvog malog izdanja obično je unutar dva mjeseca
- trajanje: nekoliko dana

Faze ekstremnog programiranja (2)

□ Iteracije do izdanja

- Uključuje nekoliko iteracija sustava prije prvog izdanja
- Vremenski raspored iz faze planiranja se razlaže u više iteracija
- Prva iteracija stvara verziju koja obuhvaća cijelu arhitekturu ciljanog sustava
- Klijent određuje kartice koje će se koristiti pri svakoj narednoj iteraciji
- Testovi prihvatljivosti izvode na kraju svake iteracije
- Na kraju posljednje iteracije, sustav je spreman za produkciju
- trajanje pojedine iteracije: jedan do četiri tjedna

□ Producija

- Dodatno testiranje i provjera performansi sustava prije isporuke klijentu
- Razrješenje primjedbi te odlučivanje da li će se riješiti u tekućem izdanju
- Zakašnjele nove ideje i prijedlozi se dokumentiraju a implementacija odgađa
- trajanje pojedine iteracije: tri dana do najviše tjedan dana

Faze ekstremnog programiranja (3)

□ Održavanje

- Nakon što je prvo izdanje pušteno u produkciju
- treba istovremeno održavati softver u primjeni i proizvoditi nove verzije
- Zbog toga se brzina implementacije smanjuje
- Održavanje može zahtijevati nove članove tima i promjenu strukture tima

□ Faza smrti je blizu kada klijent nema više novih kartica s pričama

- Podrazumijeva se da sustav zadovoljava sve zahtjeve
- Vrijeme da se konačno napiše sva korisnička dokumentacija budući da više nema promjena na arhitekture, dizajna i programskog koda sustava
- Smrt može nastupiti i kada sustav ne ispunjava sva korisnička očekivanja, ili ako postane preskup za daljnji razvoj

Temeljne vrijednosti (core values)

Komunikacija (communication)

- XP zahtjeva komunikaciju u svim fazama projekta
- komunikacija članova razvojnog tima, komunikacija članova s voditeljima projekta, komunikacija naručitelja s izvođačima (razvojnicima i voditeljima)

Jednostavnost (simplicity)

- dijelovi softvera i cjelokupna arhitektura moraju u svakoj fazi biti jednostavnii
- ostvaruje se kontinuiranim refaktoriranjem i minimizacijom dokumentacije

Povratne informacije (feedback)

- XP nalaže kontinuirane povratne informacije od svih sudionika projekta
- povratne informacije onemogućavaju nerazumijevanje među sudionicima

Hrabrost (courage)

- sposobnost provedbe teških odluka (npr. odbacivanja dijelova koda kada je to nužno ili poduzimanje velikih promjena u kasnoj fazi projekta)
- također, podrazumijeva međusobnu iskrenost svih članova projektnog tima

Uvažavanje (respect)

- razvojnici nikad neće napraviti promjene koje bi onesposobile aktualnu verziju ili vodile neispravnom testiranju ili usporile napredak ostalih

Načela (principles)

- Poveznica između apstraktnih temeljnih vrijednosti i prakse

- **Ljudskost (Humanity)** – kompromis interesa ljudi i organizacije
- **Ekonomičnost (Economics)** – dokaz vrijednosti za naručitelja
- **Obostrana korist (Mutual benefit)** – zadovoljstvo naručitelja
- **Samo-sličnost (Self-similarity)** – višestruko upotrebljiv kod
- **Unaprjeđenje (Improvement)** – trajno poboljšanje u iteracijama
- **Raznolikost (Diversity)** – komplementarni pojedinci
- **Promišljanje (Reflection)** – učenje na vlastitim pogreškama
- **Tijek aktivnosti (Flow)** – kontinuirani proces, ne slijed aktivnosti
- **Prilika (Opportunity)** – problemi kao mogućnost za napredak
- **Redundancija (Redundancy)** – raznolika rješenja jamstvo napretka
- **Neuspjeh (Failure)** – bolje pogriješiti nego zastati
- **Kvaliteta (Quality)** – jamstvo stabilnosti, ne perfekcija
- **Mali koraci (Baby steps)** – stalni, vidljivi napredak
- **Prihvaćena odgovornost (Accepted Responsibility) - osoblja**

Osnovne XP prakse (Primary practices)

□ Priče (korisničke priče) - Stories (User Stories)

- kratki opis funkcionalnosti

□ Tjedni ciklus (Weekly Cycle)

- razvoj u tjednim ciklusima, svaki tjedan započinje sastankom izbora priča

□ Kvartalni ciklus (Quarterly Cycle)

- grublje, na dulje staze, razvoj se planira kvartalno

□ Rezerva (Slack)

- izbjegavanje obećanja koja ne mogu biti ispunjena

□ Smještaj ekipe (Sit Together)

- smještaj ekipe na isto mjesto da se poveća komunikacija

Osnovne XP prakse (2)

❑ Cjelovitost i zajedništvo ekipe (Whole Team)

- ekipa sastavljena od članova potrebne stručnosti
- osjećaj pripadnosti i spremnosti na pomoć ostalim članovima

❑ Informativno radno okruženje (Informative Workspace)

- pružanje osjećaja pripadnosti i predanosti projektu (posteri itd.)

❑ Energičan rad (Energised Work)

- programeri moraju biti odmoreni (svježi) da bi bili produktivni
- smanjenje prekovremenog rada

❑ Programiranje u paru (Pair Programming)

- Programeri rade u parovima, tako da jedan piše kôd, a drugi prati pisanje i revidira kôd pazeci da kod bude jasan i razumljiv

Osnovne XP prakse (3)

□ Inkrementalni dizajn (Incremental Design)

- dizajn kao kontinuirani proces koji se u malim koracima odvija tijekom čitavog razvoja
- Refaktoriranje – uklanjanje složenosti i zalihosti programskog koda
- Jednostavan dizajn – što je moguće jednostavniji (KISS)

□ Test prije programiranja (Test-First Programming)

- Testovi trebaju biti napisani prije kodiranja

□ Destminutna gradnja (Ten-Minute Build)

- sustav se mora moći kompilirati i testirati unutar 10 minuta
- da bi mogao postići odgovarajuću povratnu informaciju (feedback)

□ Kontinuirana integracija (Continuous Integration)

- Sustav treba integrirati svaka 2 sata ili nakon većih promjena

Scrum

- jednostavni upravljački okvir za inkrementalni razvoj
 - nije definiran proces, ne bavi se tehnikalijama
- iteracija = sprint
- rezultat sprinta – potencijalno isporučivi inkrement proizvoda (shippable)

□ Uloge

- Ekipa (Scrum Development Team)
 - jedna ili više ekipa od po 7 plus/minus 2 člana
 - svestrani članovi (cross-functional)
 - samoorganizirajuća ekipa (self-organizing)
- Vlasnik proizvoda (Product Owner)
 - zadužen za plan, prioritete, troškove i povrat investicije
- Majstor (Scrum Master)
 - brine o procesu, koordinira, ali ne donosi poslovne ni tehničke odluke
- Klijent
 - održava *Product Backlog*
- Uprava (Management)
 - odlučuje o dosegu, promjenama, standardima, ...

Životni ciklus Scruma

□ Prije igre (pre-game)

- podfaze: Planiranje i Dizajn/Arhitektura
- izrađuje se radna lista proizvoda (Product Backlog - PB)
 - u PB se konstantno zapisuju zahtjevi, procjene napora i prioriteti

□ Razvoj / "igra" (development / game)

- razvoj iterativnim ciklusima, takozvanim sprintovima
- sprintovi - okvirno jednakog trajanja, 30 dana (prema knjizi)
 - tjedan do tri u praksi
- sprint ima sve faze klasičnog ciklusa (RUP ima mini-vodopad !)
 - zahtjeve, analizu, dizajn, evoluciju, test i isporuku
- tri do osam sprintova dok sustav ne bude spreman za distribuciju

□ Poslije igre (post-game)

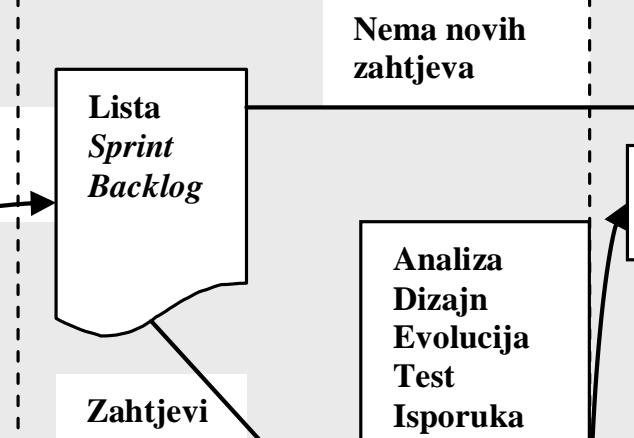
- priprema sustav za izdanje kroz integraciju, testiranje i druge aktivnosti

Scrum proces

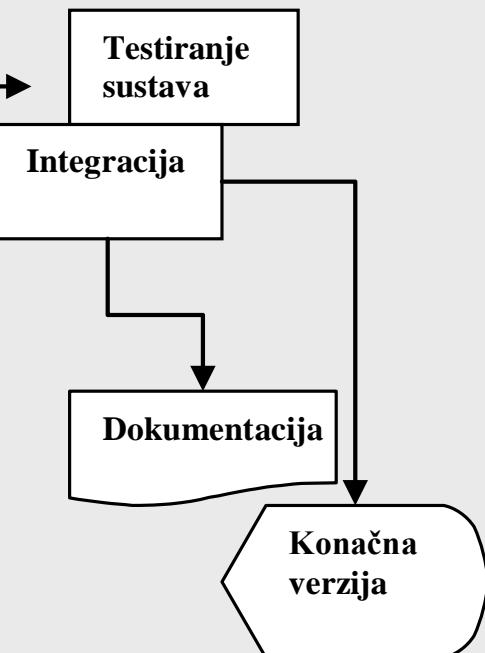
Prije igre



Razvoj (Igra)



Poslije igre



Scrum artefakti

□ ***Product Backlog***

- neizvršen rad, preostali posao
- lista poželjne funkcionalnosti
- vidljiva svim dionicima
- svatko može dodati elemente

□ ***Product Backlog Item - element***

- definira "ŠTO", najčešće kao korisnička priča
- ima kriterij prihvatljivosti, definiciju "dovršenosti"
- sadrži više zadataka
- poslovnu vrijednost odredi Vlasnik
- napor procijeni Ekipa

□ ***Sprint Backlog***

- popis zadataka i statusa
- ažuriran tokom sprinta
- vidljiv Ekipi

□ ***Zadatak sprinta (Sprint Task)***

- "KAKO" za PBI "ŠTO"
- dan posla ili manje
- preostali napor procjenjuje se dnevno u satima

Scrum sastanci

□ Planiranje sprinta

- 1. na početku iteracije – koji PB elementi idu u sprint
- 2. tim dekomponira PB elemente u listu zadataka
- 30d sprint planira se max 8 sati

□ Dnevni Scrum

- 15min, članovi tima međusobno
- "standup meeting" – dojam užurbanosti

□ Pregled sprinta

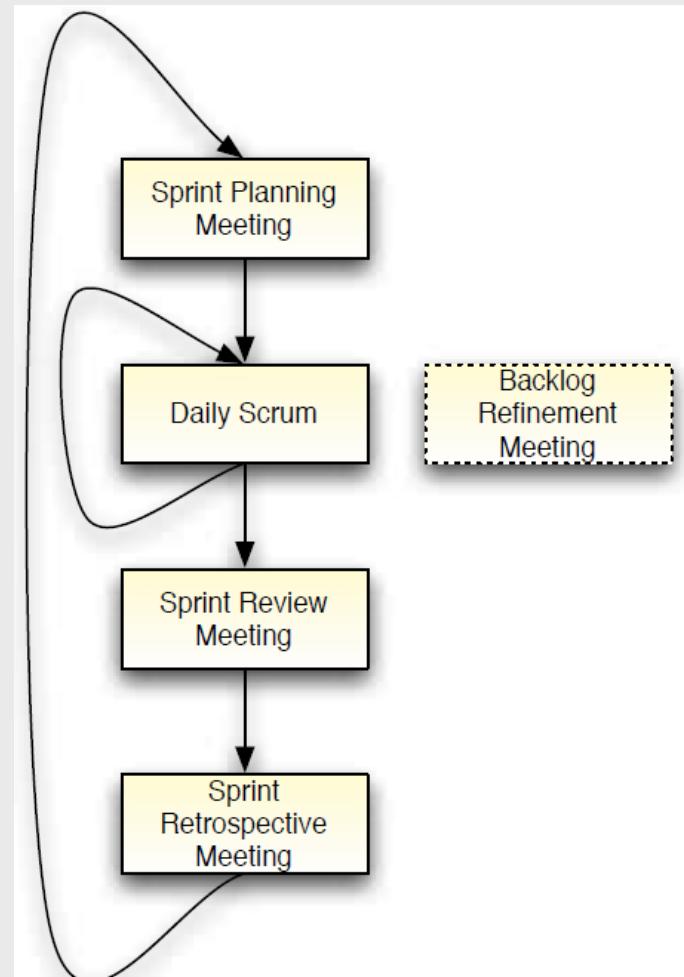
- demonstracija inkrementa na kraju sprinta
- Vlasnik deklarira "dovršeno"
- ostalo ide u naredni sprint

□ Retrospektiva sprinta

- samoanaliza procesa

□ Pročišćavanje preostalog posla

- podjela, procjena, prioriteti ...



Materijali i internet adrese

□ Primjeri metoda: Metode \ *

□ Internet adrese

- Adaptable Process Model (+ Document Templates)
 - <http://www.rspa.com/apm>
- Construx Software Development Best Practices
 - <http://www.construx.com/> (checklists, templates, ...)
- <http://www.extremeprogramming.org/>
- <http://www.agilemodeling.com/>
- <http://www.agilealliance.com>
- <http://www.realsoftwaredevelopment.com/the-complete-list-of-software-development-frameworks-processes-methods-or-philosophies/>
- <http://www.noop.nl/2008/07/the-definitive-list-of-software-development-methodologies.html>

Reference

- IBM Rational Unified Process web site,
<http://www-01.ibm.com/software/awdtools/rup>, [2009-01-27]
- Leslee Probasco, The Ten Essentials of RUP – the Essence of an Effective Development Process, Rational Software White Paper, TP177, 9/00,
<ftp://ftp.software.ibm.com/software/rational/web/whitepapers/2003/TP177.pdf>,
[2009-01-20]
- Manifesto for Agile Software Development,
<http://www.agilemanifesto.org/>, [2009-01-20]
- Don Wells, Extreme Programming: A gentle introduction, 2006,
<http://www.extremeprogramming.org> , [2009-01-16]
- Michele Marchesi, The New XP,
<http://www.snip.gob.ni/Xdc/Agile/TheNewXP.pdf>, [2009-01-19]
- OpenUP Wiki,
<http://epf.eclipse.org/wikis/openup/>, [2009-01-27]
- Scott Ambler, The Agile Unified Process,
<http://www.ambysoft.com/unifiedprocess/agileUP.html>, [2009-01-27]
- Gary Pollice, Using the Rational Unified Process for Small Projects: Expanding Upon eXtreme Programming, Rational Software White Paper, TP 183, 3/01,
<ftp://ftp.software.ibm.com/software/rational/web/whitepapers/2003/tp183.pdf>,
[2009-01-10]

Literatura

- Jacobson, Booch, Rumbaugh: The Unified Software Development Process. Addison Wesley, 1999
- Agile software development methods, review and analysis, Pekka Abrahamsson, Outi Salo & Jussi Ronkainen, 2002
- Juhani Iivari, Jari Maansaari: The usage of systems development methods: are we stuck to old practices?, Information and Software Technology, 1998, Linnanmaa, Finland, 1998, pp. 501-510
- Alistair Cockburn: Agile Software Development, Addison Wesley, Boston, USA, 2000
- XP123 - Exploring Extreme Programming, available at <http://www.xp123.com/>, Dostupno: 10.06.2008

CASE

CASE

□ CASE

- eng. Computer Aided Software Engineering
- eng. Computer Aided System Engineering

□ CAISE

- Computer Aided Information System Engineering

□ Računalom podržano programsko/informacijsko inženjerstvo

- programski sustav za pomoć pri izgradnji IS i razvoju programske opreme
- uporaba alata koji podupiru neku od faza životnog ciklusa
- automatizacija programske opreme (Software Automation)

CASE pomagala

- **CASE pomagala podupiru (ne sva i ne uvijek!)**
 - jednu ili više metoda informacijskog/programskog inženjerstva
 - njima svojstvenu ili neku od standardnih „metodologija” (notacija)
 - mogućnost definiranja vlastite metodologije (meta-case)
- **Često se radi o skupu integriranih alata kojim se podupire:**
 - jedna od faza životnog ciklusa (toolkit)
 - životni ciklus (workbench)
 - cjelokupni proces (environments), uključujući potporne aktivnosti
- **Vrste pomagala**
 - Gornji CASE (Upper CASE, Front-End CASE) – rane faze
 - Donji CASE (Lower CASE, Back-End CASE) – izrada, ugradnja
 - Integrirani CASE (ICASE - Integrated CASE) – cijeli ciklus

Preoblikovanje programske podrške

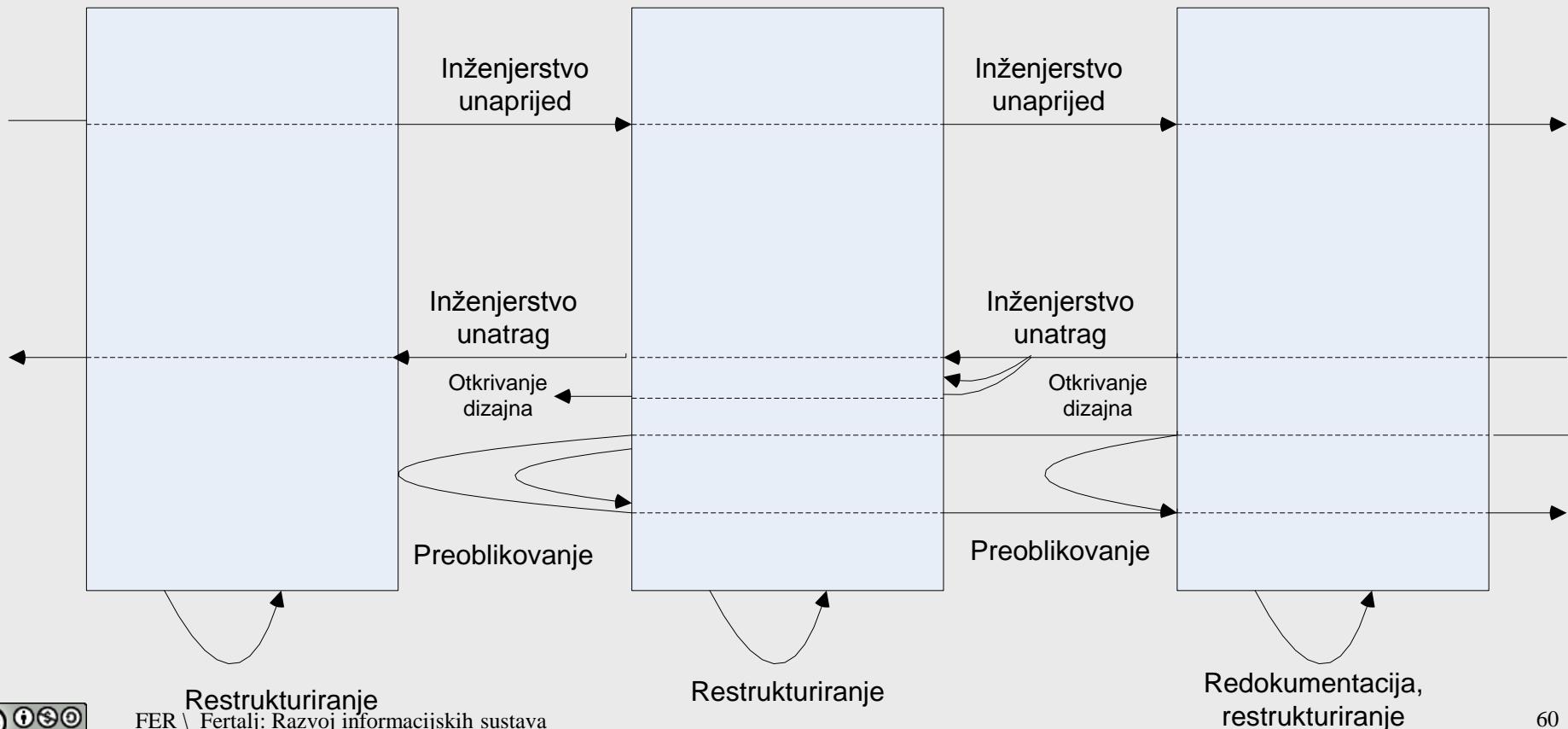
□ Preoblikovanje programske podrške (software reengineering)

- skup ispitivanja i izmjena postojećeg sustava koji rezultiraju preustrojem
- sustav čine svi programi, baze podataka, podaci i razvojna dokumentacija
- moguće je mijenjati čitav sustav ili samo jedan manji dio

ZAHTEVI

PROJEKTIRANJE

IZVOĐENJE



Inženjerstvo prema naprijed i unatrag

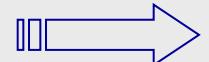
- **Inženjerstvo prema naprijed (forward engineering)**
 - tradicionalni postupak - s viših razina apstrakcije prelazimo na projektiranje i izvođenje
 - oznaka „naprijed“ - samo da se napravi razlika prema inženjerstvu unatrag
- **Inženjerstvo unatrag (reverse engineering) - obrnuto, reverzno, povratno**
 - kontekst
 - izvorni kod je dostupan, ali je dokumentacija nepotpuna, neažurna ili je nema
 - izvorni kod nije dostupan i cilj je doći do njega
 - glavne aktivnosti
 - prepoznavanje dijelova sustava i njihove povezanosti, s ciljem da se identificiraju komponente sustava i dobije potpuna specifikacija
 - oblikovanje sustava na višim razinama apstrakcije temeljem nižih, npr. izrada dijagrama temeljem izvornog koda programa
 - ne podrazumijeva izmjene u postojećem sustavu. nego proučavanje i razumijevanje
 - ispitivanje postojeće implementacije sustava,
 - ponovno otkrivanje dizajna sustava,
 - dešifriranje zahtjeva ugrađenih u sustav

Područja obrnutog inženjerstva

□ Redokumentacija (redocumentation)

- naknadno dokumentiranje postojeće programske podrške
- generiranje temeljem komentara u izvornom kodu ili rječnika BP

□ Metrika (software metric, system metric)



- mjerenje veličine, složenosti i uvezanosti programskog koda
- ocjena kvalitete, prikladnosti za održavanje, utjecaja promjena nekog dijela koda na ostatak sustava, prepoznavanje ponovno upotrebljivih dijelova

□ Restrukturiranje (restructuring)

- pretvorba iz jednog oblika u drugi na istoj razini apstrakcije, uz očuvano ponašanje
- formatiranje izvornog koda - najjednostavniji oblik (uvlačenje, kapitalizacija, ...)
- preusmjeravanje (retargeting) - prevođenje na novu konfiguraciju ili platformu
- refaktoriranje – tehniku OO programiranja

□ Objektifikacija (objectification)

- preoblikovanje proceduralnog programa u funkcionalno ekvivalentni OO program

□ Inženjerstvo višestruke iskoristivosti (reuse engineering)

- analiza prikladnih dijelova, izmjena dijelova s ciljem odvajanja, izrada funkcionalnih specifikacija za izdvojene dijelove

Mjerenje programske potpore

- **Softverska metrika (software metric)**
 - mjera nekog svojstva ili komada programske potpore ili njezinih specifikacija

- **Mjerenja programske potpore (software measurements)**
 - *Code coverage* – pokrivenost programskog koda testovima
 - *Cohesion* – mjera prianjanja, povezanosti odgovornosti unutar modula
 - *Coupling* – kopčanje, zavisnost o drugim modulima
 - *Cyclomatic complexity* – ciklomatska složenost programa, brojanjem nezavisnih puteva programskog toka
 - *Function point analysis* – funkcija točke kao mjera količine funkcionalnosti
 - ...
 - *Source lines of code (Program Length)*
 - *Bugs per line of code*
 - ...
 - *Program load time*
 - *Execution Time*

Inženjerstvo u krug (round-trip engineering)

- Kombinacija inženjerstva unaprijed i unatrag

- Skraćeni prikaz



Reference

- <http://portal.zpr.fer.hr/ris/Datoteke/PrimjeriLabosa>
- **Alati općenito**
 - <http://www-01.ibm.com/software/rational/offering/design/?ca=rhp>
 - http://www.enterprise-architecture.info/EA_Tools.htm
 - <http://www.softdevtools.com/>
- **Upravljanje zahtjevima**
 - http://www.jiludwig.com/Requirements_Management_Tools.html
- **Modeliranje, projektiranje**
 - <http://www.eclipse.org/>
 - <http://www.objecteering.com/>
 - <http://staruml.sourceforge.net/en/>
 - <http://www.sparxsystems.com>
 - <http://www.sybase.com/products/modelingdevelopment/powerdesigner>
 - http://developer.tibco.com/business_studio/
 - <http://www.visual-paradigm.com>
 - <http://www.visible.com/>

Reference

□ Ekipni razvoj

- <http://www.codeplex.com/TFSGuide>
- <http://subversion.apache.org>
- <http://mercurial.selenic.com/wiki/Mercurial>

□ Generatori koda, aplikacija

- CodeCharge <http://www.yessoftware.com/>
- Iron Speed <http://www.ironspeed.com/>
- LLBLGen <http://www.llblgen.com/defaultgeneric.aspx>
- MyGeneration <http://www.mygenerationsoftware.com>
- Sculpture toolkit <http://www.modelingsoft.com>

Reference

□ Preoblikovanje, refaktoriranje

- ReSharper <http://www.jetbrains.com/resharper/>
 - dodatak RGreatEx <http://www.brothersoft.com/rgreatex-resource-refactoring-tool-69903.html>
- CodeIt.Right <http://submain.com/download/codeit.right/>

□ Potpora kodiranju, analiza koda, metrika

- ANTS Profiler <http://www.red-gate.com/products/dotnet-development/ants-performance-profiler/>
- CodeSmart <http://www.axtools.com/>
- NDepend <http://www.ndepend.com/>

□ Testiranje

- <http://www.csunit.org/>, <http://www.nunit.org/>, ...
- <http://www.devcurry.com/2010/07/10-free-tools-to-loadstress-test-your.html>

Sustavi za upravljanje poduzećem

2012/13.11

Sustavi za upravljanje poduzećem

□ ERP (Enterprise Resource Planning)

- "planiranje resursa poduzeća" ?
- integrirani informacijski sustav za upravljanje poslovanjem
- jedna baza podataka, jedna aplikacija i jedno unificirano sučelje kroz cijelu poslovnu organizaciju [Tadjer]
- jedinstveni informacijski sustav koji zamjenjuje odvojene sustave pojedinih organizacijskih jedinica, te zadovoljava sve potrebe upravljanja poslovanjem [Koch, Slater & Baatz]

□ "Pravi" ERP podržava barem 3 od sljedeća 4 segmenta poslovanja

- financijsko poslovanje (accounting),
- proizvodnja (manufacturing),
- robno-materijalno poslovanje (material management/distribution),
- upravljanje ljudskim resursima i plaće (HR management, payroll).

□ Vlasnički, naslijedeni sustav (legacy system)

- Sagrađen po mjeri

Poslovna područja (moduli), pr. SAP ERP

□ Financijsko računovodstvo

- Upravljanje glavnom knjigom.
- Bilanca i Račun dobiti i gubitka (financijski izvještaji).
- Analitika kupaca.
- Analitika dobavljača.
- Računovodstvo osnovnih sredstava.
- Računovodstvo zaliha.
- Porezno računovodstvo

□ Upravljačko računovodstvo (Kontroling)

- Računovodstvo troškovnih centara.
- Računovodstvo profitnih centara.
- Računovodstvo internih naloga.
- Računovodstvo profitabilnosti.

□ Prodaja i distribucija

- Upravljanje nalozima za prodaju.
- Upravljanje ugovorima.
- Fakturiranje.

□ Nabava, skladištenje i logistika

- Zahtjevnice.
- Obrada narudžbenica.
- Evidencija primki materijala.
- Ovjera faktura (likvidacija).
- Upravljanje ugovorima.
- Upravljanje zalihamama i skladištem, uključujući inventuru i fizički promet robe.

□ Upravljanje ljudskim kapitalom

- Kadrovska evidencija.
- Obračun plaća.

Poslovna područja (moduli), nastavak

□ Proizvodnja

- Planiranje proizvodnje.
- Izvršenje proizvodnje.
- Proizvodnja po narudžbi.
- Procesna proizvodnja.

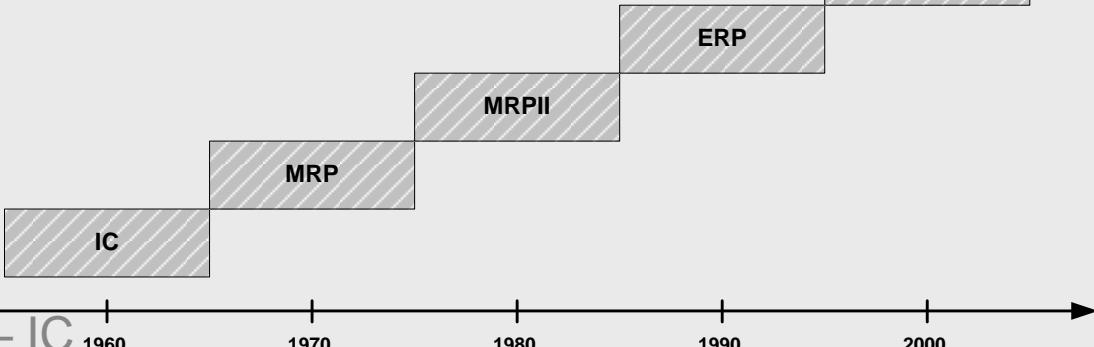
□ Upravljanje kvalitetom

- Ispitivanje kvalitete.

□ Izvještaji

- Finansijsko i upravljačko izvještavanje.
- Finansijsko planiranje i budžetiranje.
- Upravljanje profitabilnošću.
- Upravljanje režijskim troškovima.
- Analitika nabave.
- Analitika upravljanja zalihamama i skladištem.
- Prodajna analitika.

Razvoj i trendovi



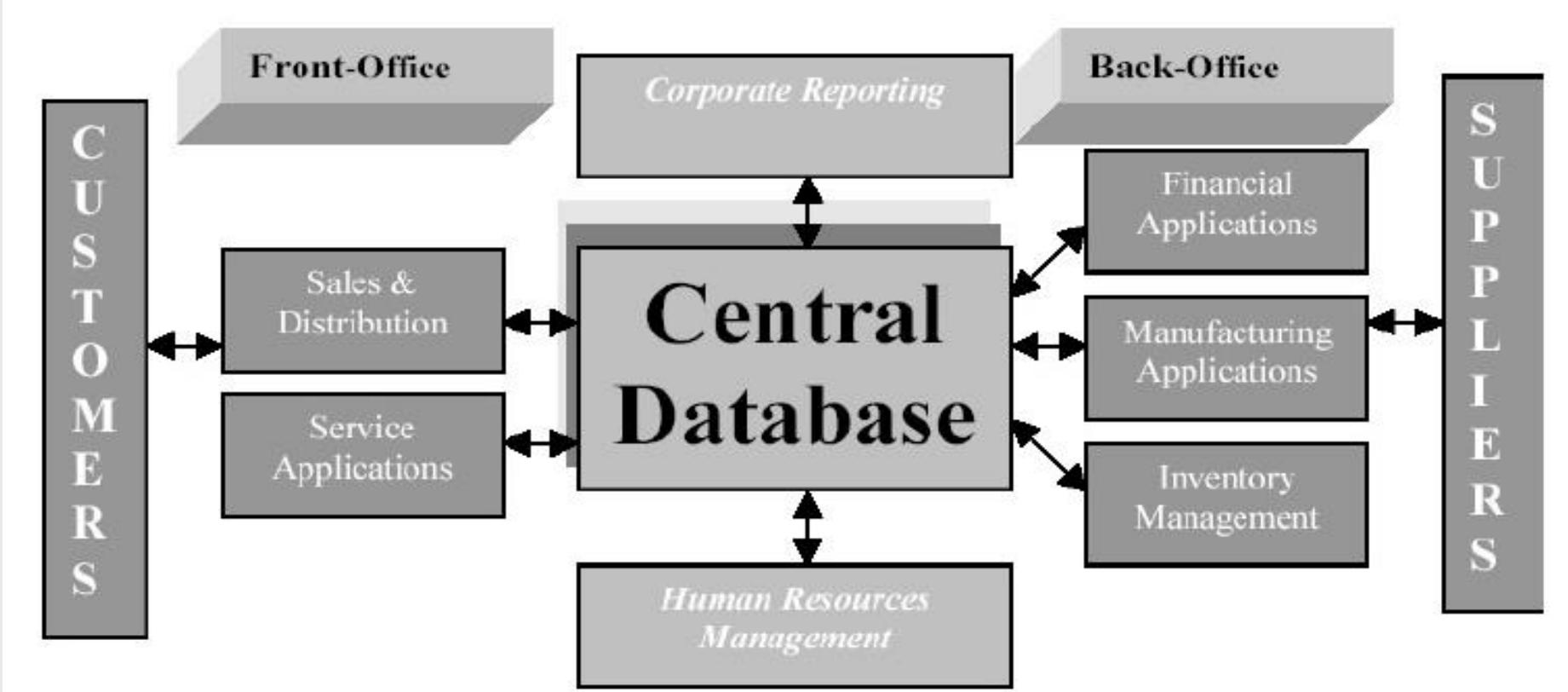
□ Razvoj

- Inventory Control System – IC 1960.
- Material Resource planning – MRP
- Manufacturing Resources Planning – MRP II
- Enterprise Resource Planning – ERP
- Extended Enterprise Resource Planning – Extended ERP

□ Stanje : Extended ERP

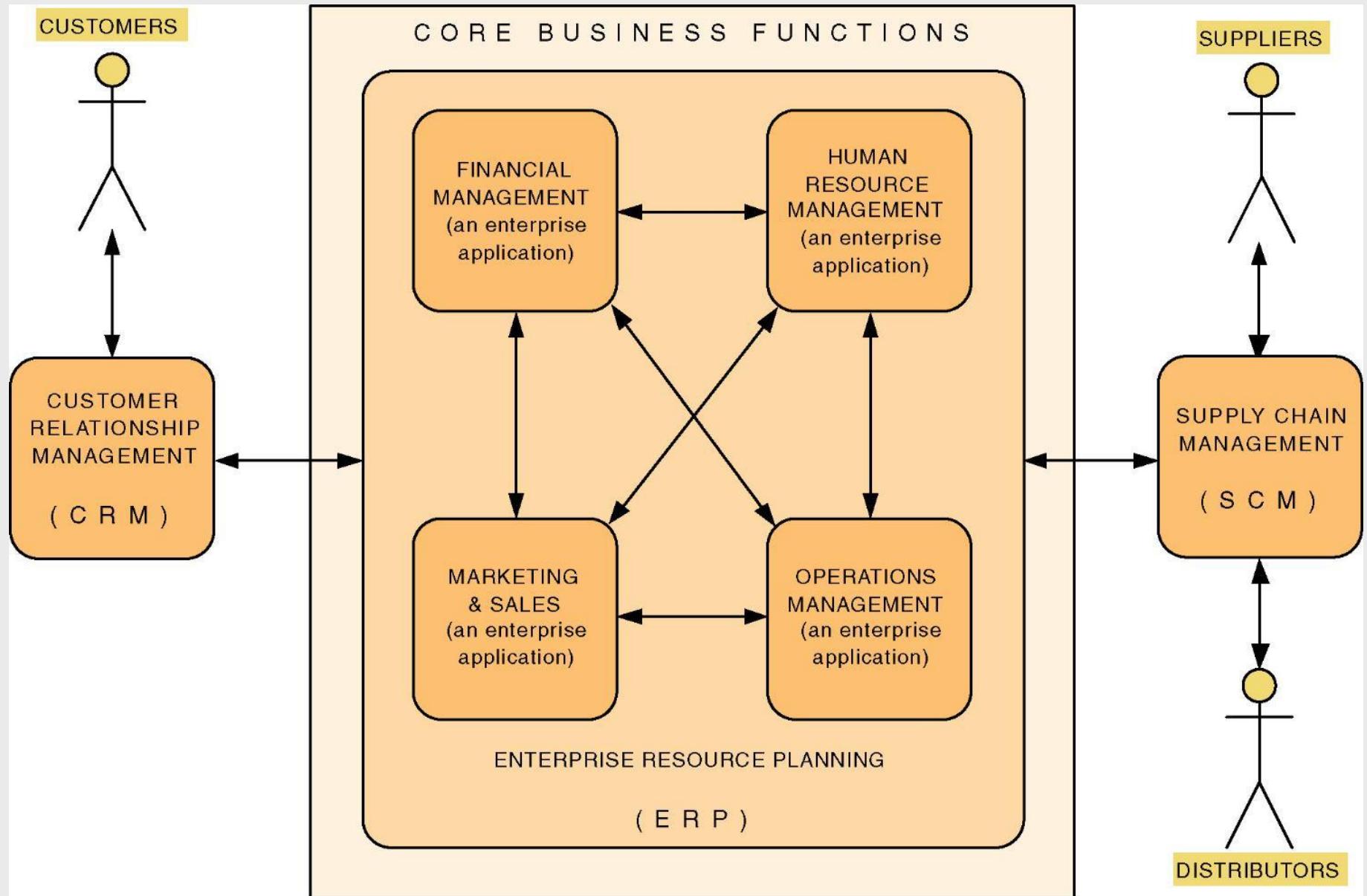
- orijentacija na Internet, vanjski moduli, e-bussines rješenja
- napredno poslovno planiranje (*APS – Adwanced Planning and Scheduling*),
- automatizirana prodaja (*SFA – Sales Force Automation*),
- poslovna inteligencija (*BI – Business Inteligence*),
- upravljanje odnosom s kupcima (*CRM – Customer Relationship Management*)
- upravljanje lancem nabave (*SCM - Supply Chain Management*).

ERP arhitektura



*Enterprise Resource Planning – Global Opportunities & Challenges;
Liaquat Hossain, Jon David Patrick, M.A. Rashid (2002)*

ERP i proširenja



Upravljanje odnosom s kupcima

□ Customer Relationship Management (CRM)

- posljedica poslovne filozofije usmjerene na što bolje upoznavanje kupaca, kako bi se što bolje zadovoljile njihove potrebe i želje
- cilj je saznati što je više moguće o kupcu da bi se znalo što u danom trenutku kupac treba, po kojoj cijeni i u koje vrijeme
- u vrijeme ručne obrade – dodjela referenata najvažnijim kupcima

□ ERP sustavi omogućuju automatizirano prikupljanje informacija

- što je kupac prošli put kupio te kakvo je stanje na skladištu, a na temelju toga pokrenuti marketinšku akciju ili ponuditi popust
- pr. Amazon: "kupac neke knjige je gledao ili kupio još i ..."
- pr. banke: posebne ponude za kredite redovnim platišama anuiteta

□ Podmoduli

- marketing, prodaja, podrška klijentima itd.

Upravljanje lancem nabave

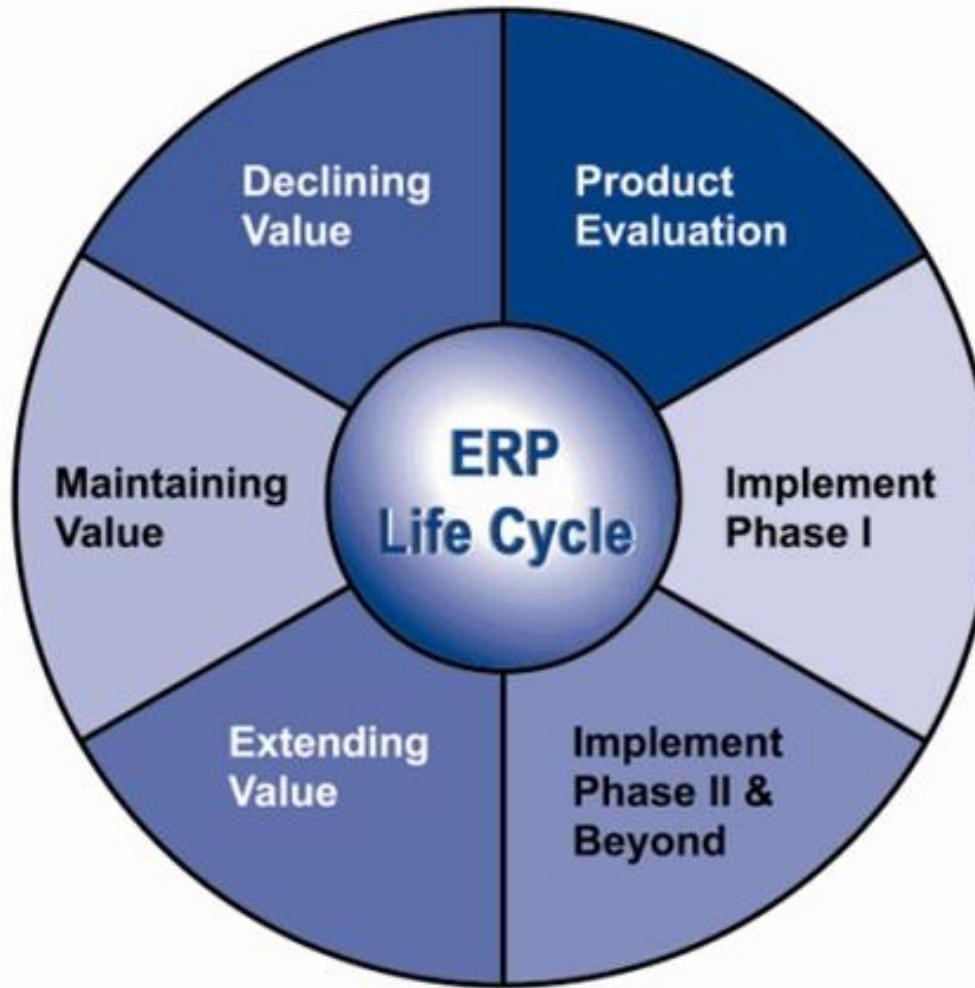
□ Supply Chain Management (SCM)

- podrška procesima planiranja i provedbe protoka materijala i proizvoda
- planiranje potražnje za gotovim proizvodima
- odabir najboljeg/najboljih dobavljača
- zaprimanje i skladištenje materijala te njihova uporaba u proizvodnom procesu
- planiranje distribucije krajnjih proizvoda – ponude proizvoda kupcima na određeno mjesto u određeno vrijeme po najnižoj mogućoj cijeni

□ Internet i SCM

- Kada poduzeća za komunikaciju koriste Internet tada postoji mogućnost povezivanja materijalnih tokova poduzeća
- kada bi programska rješenja u pojedinim poduzećima bila međusobno kompatibilna tada bi postojala mogućnost da se sve narudžbe i plaćanje provode elektroničkim putem (npr. e-račun)

Implementacija ERP sustava općenito



The ERP Life Cycle: From Birth to Death and Birth Again
by Andy Klee, President, Klee Associates, Inc.

Implementacija ERP sustava općenito (2)

□ Evaluacija (product evaluation) ili javna nabava

- odrede se najvažniji zahtjevi te se
- prezentacija dobavljača ili vlastita procjena prikladnog rješenja
 - izbor se suzi na 2-3 proizvođača koje se onda detaljnije analizira
- alternativa, javna nabava – natječaj uz odabir najpovoljnije ponude
- s odabranim dobavljačem se potpisuje ugovor

□ Implementacija – faza 1

- implementaciju obavlja dobavljač odabranog ERP i njegovi konzultanti
- mogu se angažirati nezavisni konzultanti (jeftinije, fleksibilnije) !?!
- implementira se dio po dio sustava ili sve dijelove odjednom (tzv. big-bang)

□ Implementacija – faza 2

- otprilike 6 mjeseci nakon završetka faze 1, kad se sve "slegne"
- implementiraju se preostali dijelovi i/ili dorađuju postojeći
- loša prva faza ili nezadovoljstvo implementatorom - trenutak za zamjenu

Implementacija ERP sustava

□ Proširenja (extanding values)

- dodatne funkcionalnosti, pr. CRM, BI, SCM koje nisu bile prvotno predviđene
- obično duga i zahtjevna faza (poput faza implementacije)

□ Održavanje (maintaining values)

- inkrementalna proširenja kako bi se zadovoljili novi zahtjevi i potrebe
- preporučljivo uz procjenu koristi i povrata investicije
- često faza u kojoj dolazi do odluke o zamjeni sustavom drugog proizvođača

□ Opadanje vrijednosti (declining value, decline-in-value)

- promjena poslovanja s vremenom - održavanje sve teže izvedivo
- na tržištu postoje moderniji sustavi koji bi mogli zamijeniti postojeći sustav
- kreće se u novi ciklus implementacije ERP sustava

□ Opisana implementacija ERP sustava može trajati 10 i više godina

Koristi uvođenja

- Pouzdani pristup podacima**
 - zajednička BP, konzistentni i ispravni podaci, napredni izvještaji
- Uklanjanje redundancije podataka i operacija**
 - središnja BP i modularnost transakcija
- Brže ispunjavanje zahtjeva**
 - zahtjevi korisnika, prikupljanje i prezentacija podataka nadređenima
- Smanjenje troškova**
 - kroz uštedu na vremenu i bolju kontrolu kroz cijelu organizaciju
- Jednostavna prilagodba (ovisno o dobavljaču !)**
 - prilagodba poslovnim procesima i promjenama poslovnih pravila
- Unaprijeđena skalabilnost**
 - zbog strukturiranog i modularnog dizajna
- Olakšano održavanje**
 - ugovaranje dugoročne podrške - savjetodavna i/ili praktičnu pomoć
- Globalni doseg**
 - proširivanje modulima poput CRM i SCM
- E-poslovanje**
 - poslovanjem putem Interneta

Problemi uvođenja

□ Vremenski dugotrajan proces uvođenja sustava

- definiranje potreba, ulaza i izlaza u informacijski sustav
- identificiranje potrebnih komponenti – obično manji podskup cijelog sustava - otežano pronalaženje željene strukture i funkcija
- oblikovanje i prilagodba zaslonskih maski
- ugradnja složenih izvješća - standardno ugrađeni samo jednostavni ispisi

□ Trošak

- veliki početni troškovi (rješenje + licence, jači hardver)
- neočekivani skriveni troškovi (npr. trošak prilagodbe kad "nešto zapne")
- konzultant/dan cca natprosječna mjesecna plaća

□ Ovisnost o proizvođaču

- sva buduća unapređenja sustava ovise o volji proizvođača
- teška (ponekad nemoguća) prilagodba, naročito manjim korisnicima

Problemi uvođenja (nastavak)

- **Mnoštvo mogućnosti i složenost softvera**
 - velike BP i softver (pr. SAP cca 20k tablica, neke po stotinjak stupaca)
 - nepotrebno opterećenje kod korisnika koji koriste samo dio sustava
- **Potreba za prilagodbom postojećih poslovnih procesa**
 - sustavi su projektirani kao veliki skup unaprijed određenih funkcija
 - korisnik za kojeg u skupu ponuđenih parcijalnih rješenja ne postoji odgovarajuće rješenje mora svoju organizacijsku strukturu ili način poslovanja prilagoditi sustavu
- **Potreba za kvalificiranim osobljem**
 - specijalizirani stručnjaci za uvođenje ali i održavanje
 - administriranje zahtijeva poznavanje velikog broja parametara i postupaka
 - jedno rješenje: centar kompetencije korisnika (customer competence center)
- **Nedostatna uključenost i sposobljenost krajnjih korisnika**
 - potrebno određeno vrijeme za privikavanje na novi sustav
 - otpor promjeni poslovnih navika

Ostalo

- ERP sustav ne mora nužno unaprijediti poslovanje !**
- Da bi do unapređenja uopće došlo potrebno je:**
 - ERP sustav uklopiti u okvire postojećeg načina funkciranja
 - proces uvođenja i konfiguracije sustava provesti na način da odgovara postojećoj poslovnoj kulturi, strategiji i strukturi organizacije
- Samo uvođenje ERP sustava neće donijeti predviđenu korist, ako nije praćeno promjenom ponašanja korisnika !**
- Mala i srednja poduzeća (Small and Medium Enterprises - SME)**
 - ERP sustavi nisu namijenjeni samo velikim poslovnim organizacijama
 - velikih je zapravo (kod nas) malo, a tržište zasićeno
 - za SME nudi se mogućnost implementacije samo određenih modula
 - implementiraju se samo oni dijelovi sustava koji su potrebni organizaciji
 - moguće zahvaljujući komponentnoj organiziranosti ERP sustava
- Cijena ovisna o tržištu**

Materijali, resursi i reference

□ Mapa \Datoteke\ERP

□ Reference

- Tadjer, R. (1998). Enterprise resource planning. *Internetweek*, Manhasset, April 13
- Koch, C., Slater, D., Baatz, E. (1999), "The ABCs of ERP", CIO Magazine, December 22
 - <http://teaching.fec.anu.edu.au/INFS3024/Lecture%20Notes/The%20ABCs%20of%20ERP%20-%20Enterprise%20-%20CIOb.pdf>

□ Projekti vrednovanja ERP sustava

- <http://www.zpr.fer.hr/zpr/hr-hr/projekti/erp.aspx>
- <http://www.zpr.fer.hr/zpr/hr-hr/projekti/ipes.aspx>
- <http://www.technologyevaluation.com/>

Neki ERP sustavi (abecedno)

- 4D Wand. <http://www.4d-software.com>
- Datalab PANTHEON. <http://www.datalab.hr>
- ECSAT. <http://www.ecsat.hr/epos.asp>
- IN2 proizvodi. <http://www.in2.hr/in2-proizvodi>, Oracle eBS
- INFODOM. <http://www.infodom.hr/>, Oracle eBS
- ININ. <http://www.inin.hr/>
- Jupiter. <http://www.jupiter-software.com>
- Konto. <http://www.konto.hr>
- MS Dynamics NAV. <http://www.microsoft.com/croatia/dynamics/nav/>
- N-LAB. <http://www.n-lab4b.com/hr>
- Omega Imperios. <http://omega-software.hr/main.aspx?id=12>
- Oracle E-Business Suite (eBS). → IN2, INFODOM
- PIS. <http://www.pis.eu.com/>
- Point 2000. <http://www.point.hr>
- TIS Znalac. <http://www.tis.hr>
- Times. <http://www.times.hr>
- SAP. <http://www.sap.com/croatia>, www.b4b.hr, ...
- SPA-ERP. <http://www.rest-art.hr>
- StepOne. <http://www.infokom.hr>

Upravljanje projektom

Projekt

□ Projekt

- *Projekt je vremenski određeno nastojanje da se proizvede jedinstven proizvod, usluga ili rezultat. [PMBOK, www.pmi.org]*

□ Vremenska određenost

- Svaki projekt mora imati jasno određen početak i kraj
- Projekt završava u trenutku kada su ciljevi projekta dostignuti ili kada se zaključi da ciljevi projekta ne mogu ili neće biti dostignuti.

□ Jedinstvenost

- rad na nečemu novom ili različitom od rezultata sličnih projekata.

Rezultati projekta

- **Proizvod ili artefakt - kvantitativno odrediv**
 - krajnji proizvod ili sastavna komponenta
 - uobičajeno materijal ili roba
- **Sposobnost obavljanja usluge - korisni rad bez opipljivog proizvoda ili rezultata**
 - npr. poslovne funkcije potpore proizvodnje ili distribucije
- **Rezultat, u vidu ishoda ili dokumenta**
 - ishodi - integrirani sustav, revidirani proces, restrukturirana organizacija ili podučeno osoblje
 - dokumenti - pravilnici, planovi, studije, definirane procedure, specifikacije, izvješća

Slični pothvati

□ Operacije

- Projekti su vremenski ograničeni i jedinstveni
- Namjera projekta je postići zadane ciljeve i završiti
- Operacije su neprekidne i mogu se ponavljati
- Svrha operacije - podupiranje i održanje poslovanja, čak i kada se ciljevi promijene

□ Programi

- program - grupa projekta organiziranih da priskrbe korist koja ne bi bila moguća da se radi o zasebnim projektima.
- Programi mogu uključivati i grupu akcija koje se ciklički ponavljaju, npr.:
 - izrada godišnjeg plana proizvodnje, nastavni plan i program, nabava

□ Podprojekti

- Projekti se često dijele na podprojekte koji su upravlјiviji, npr.:
 - provedba jedne faze životnog ciklusa, primjerice dizajn Web stranica
 - izgradnja podsustava, pr. CRM (Customer Relationship Management)
- često se dodjeljuju zasebnoj funkcionalnoj jedinici ili vanjskoj organizaciji

Upravljanje projektom

□ Upravljanje, rukovođenje projektom (Project management)

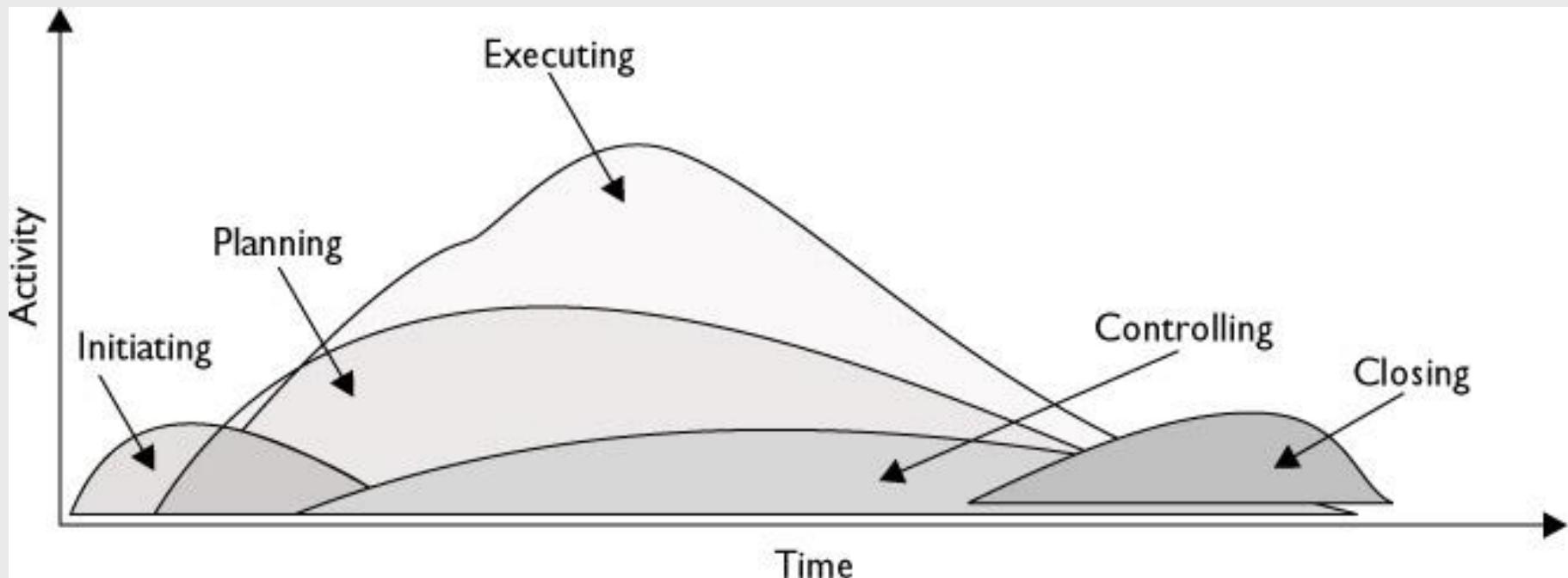
- primjena znanja, vještina, alata i tehnika u projektnim aktivnostima da bi se ispunili projektni zahtjevi [PMI]

□ “plan, staff, organize, schedule, direct, control”

- Proces organiziranja, planiranja, upravljanja i nadzora razvoja sustava kojim će se postići prava funkcionalnost, na vrijeme i uz minimalne troškove.
- plan – Koje aktivnosti i u kojem vremenskom roku treba obaviti?
- sredstva/resursi – Koji su kadrovi (osoblje) i oprema potrebni?
- organizacija – Koji je odnos pojedinih resursa?
- vremenski raspored – Koji je redoslijed aktivnosti?
- upravljanje – Kako usmjeriti i motivirati izvođače (ekipu)?
- kontrola, nadzor – Poštuje li se plan?

Procesi projekta

- započinjanje – definiranje i odobravanje projekta ili faze
- planiranje – istančavanje svrhe, planiranje smjera i akcija za postizanje cilja
- izvršavanje – koordinira ljudske i druge resurse u svrhu provedbe
- nadzor i kontrola – praćenje napretka i korektivne akcije
- zatvaranje – formalizira prihvaćanje rezultata i završava fazu / projekt



Područja upravljanja projektom

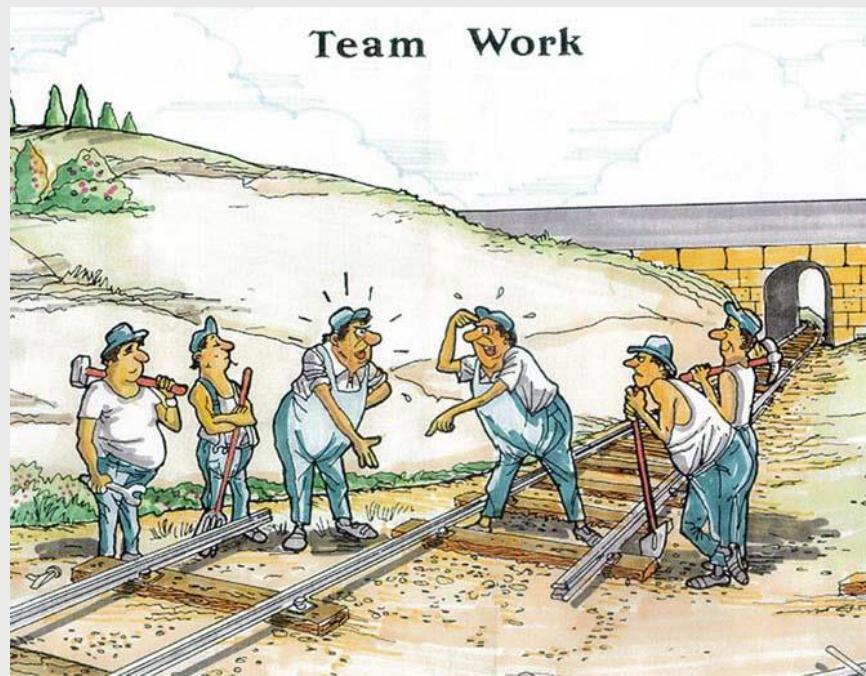
- Koordinacija projekta (Project Integration Management)
- Upravljanje dosegom projekta (Project Scope Management)
- Upravljanje vremenskim rasporedom projekta (Project Time Management)
- Upravljanje troškovima projekta (Project Cost Management)
- Upravljanje kvalitetom projekta (Project Quality Management)
- Upravljanje ljudskim resursima projekta (Project Human Resource Mgt.)
- Upravljanje razmjenom informacija u projektu (Project Communications Mgt.)
- Upravljanje rizicima projekta (Project Risk Management)
- Upravljanje nabavom za potrebe projekta (Project Procurement Management)

Organizacija projekta

Organizacija i ekipni rad

□ Ekipni rad (teamwork)

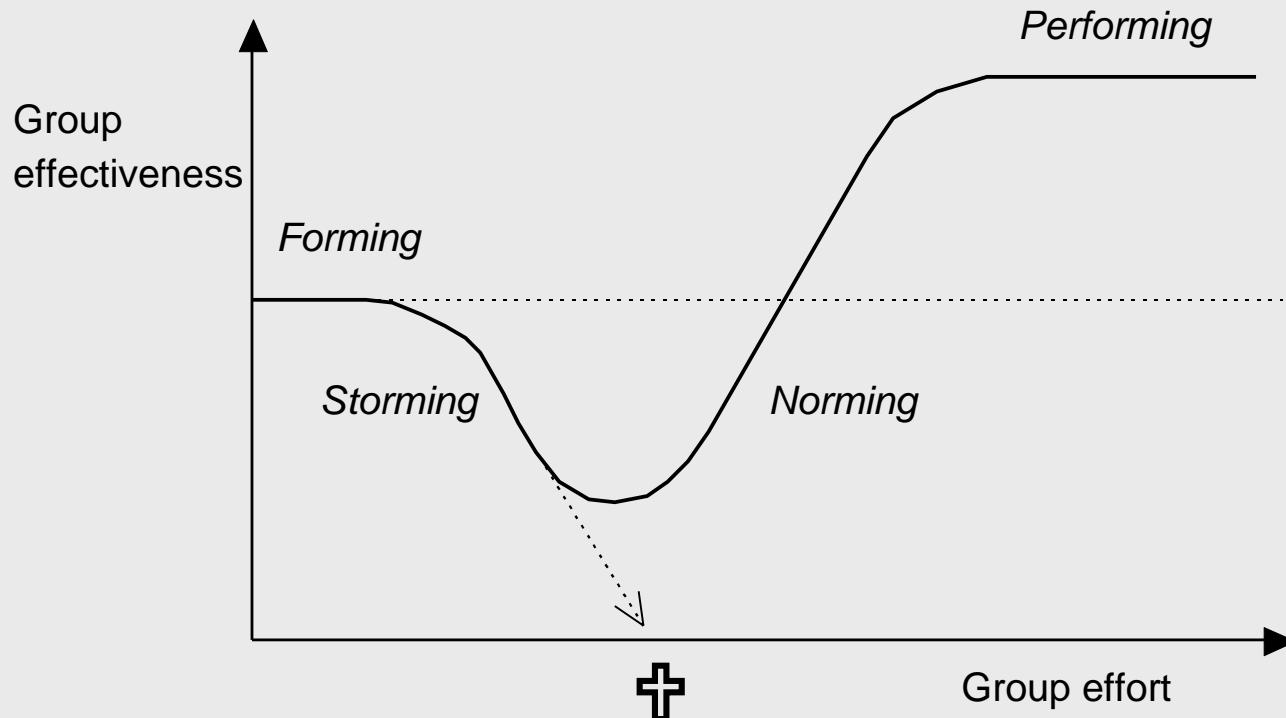
- Ekipa, tim - grupa međuzavisnih pojedinaca koji zajedno rade na ostvarivanju zajedničkog cilja te dijele odgovornost za završetak posla
- svaki član ima specifičnu ulogu (ili uloge) i odgovornost(i)
- Prednosti:
 - kvalitetnije donošenje odluka – "ne treba nam soliti pamet"
 - motivacija članova – "zajedno smo jači"
 - inovativnost – "dvoje zna više nego jedan, ..."
- Sinergija
 - $2+2=5$?



Razvoj ekipe

□ Razvoj ekipe (Forming, Storming, Norming, Performing)

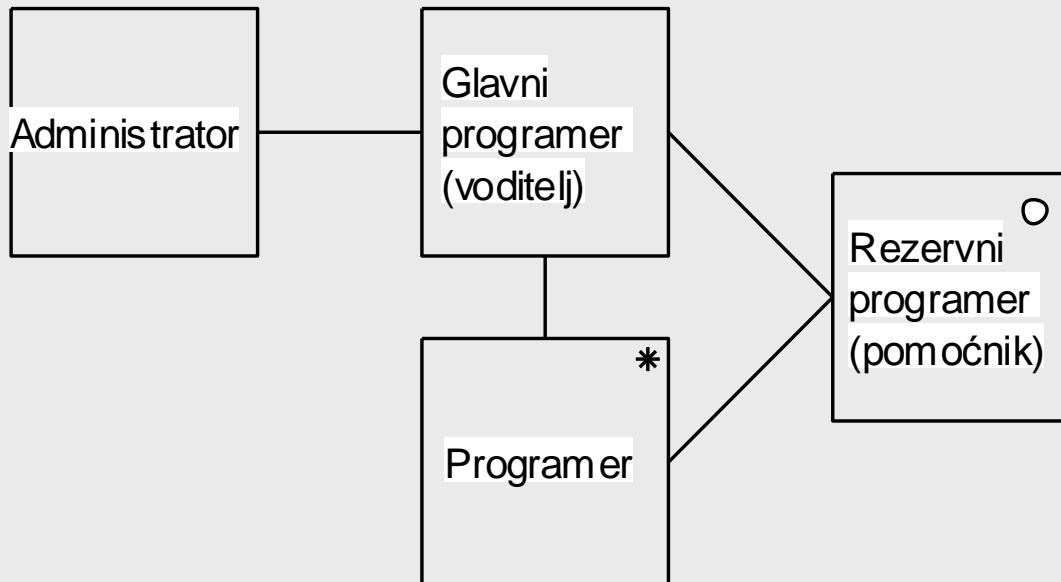
- Formiranje – ljubaznost, nesklonost iznošenju stavova, prepuštanje vođenju
- "Jurišanje" – nesloga, sukob osobnosti, grupašenje/strančarstvo, pomanjkanje kvalitetne komunikacije, nesposobnost dogovaranja
- Normiranje – uviđanje dobrih strana zajedničkog rada, uvažavanje
- Predstavljanje, djelovanje – povezivanje u učinkovitu operativnu grupu



Ekipa s glavnim programerom

□ Ekipa s glavnim programerom (Chief Programmer Team) - klasična

- glavni programer (chief programmer) utjelovljuje znanje i odlučivanje ekipе
 - mora istovremeno biti dobar (vrhunski) programer i voditelj
 - u poboljšanoj (revidiranoj) organizaciji ima ulogu voditelja ekipе
- rezervni programer (backup programmer)
 - služi kao zamjena za nekog od mlađih (junior) programera
 - u poboljšanoj (revidiranoj) organizaciji ima ulogu pomoćnika voditelja



4GL ekipa

□ Moderna organizacija ekipe – (4GL ekipa, poslovna ekipa)

- voditelj projekta (project leader) – viši sistem analitičar
- suradnja s korisnikom (user liason) – poslovni analitičar
- konceptualno i logičko oblikovanje – sistem analitičar
- isporuka sustava/aplikacija – poslovni analitičar
- nabava i pogon opreme – sistem inženjer za računala
- mrežni servisi – sistem inženjer za komunikacije
- programsko inženjerstvo – programer-analitičar
- izrada dokumentacije – urednik/pisac (editor / technical writer)
- potporno osoblje: administrativni koordinator, tehničari, činovnici

□ Neke organizacije koriste gornju podjelu za opis radnih mesta!

XP ekipa

- Razvojnik (developer)
 - kodiranje, pisanje testova, restrukturiranje programskog koda
- Klijent, korisnik
 - piše korisničke priče i testove prihvatljivosti
 - određuje redoslijed i prioritete implementacije
- Tester
 - pomoći korisniku pri izradi testova prihvatljivosti
 - izvođenje testova, objava rezultata testiranja, održavanje alata za testiranje
- „Pratilac“ (tracker)
 - prati procjene ekipe (npr. utrošenog vremena) ocjenjuje njihovu točnost
 - prati napredak svake iteracije i procjenjuje uspješnost, predlaže promjene
- Trener
 - osoba zadužena za XP proces u cjelini, vodi ekipu kroz proces
- Savjetnik
 - vanjski član, koji ima specifično tehničko znanje potrebno projektu
- Upravitelj (Veliki šef)
 - donosi odluke, komunicira s članovima, prepoznaće teškoće, uvodi rješenja

RUP ekipa (1.dio)

□ Pogled po širini i pogled po dubini

Discipline	Breadth role	Depth role
Business Modeling	Business Process Analyst Discovers all business use cases.	Business Designer Details a single set of business use cases.
Requirements	Systems Analyst Discovers all requirement use cases.	Requirements Specifier Details a single set of requirement use cases.
Analysis and Design	Software Architect Decides on technologies for the whole solution.	Designer Details the analysis and design for a single set of use cases.
Implementation	Integrator Owns the build plan that shows what classes will integrate with one another.	Implementer Codes a single set of classes or a single set of class operations.
Test	Test Manager Ensures that testing is complete and conducted for the right motivators.	Test Designer Implements automated portions of the test design for the iteration.
	Test Analyst Selects what to test based on the motivators.	Tester Runs a specific test.
	Test Designer Decides what tests should be automated vs. manual and creates automations.	

RUP ekipa (2.dio)

Deployment	Deployment Manager	Tech Writer, Course Developer, Graphic Artist
	Oversees deployment for all deployment units.	Create detailed materials to ensure a successful deployment.
Project Management	Project Manager	Project Manager
	Creates the business case and a coarse-grained plan; makes go / no go decisions.	Plans, tracks, and manages risk for a single iteration. <i>(Note that this discipline has only one role. Assigning the depth view to a project coordinator can provide relief for overburdened project managers.)</i>
Environment	Process Engineer	Tool Specialist
	Owns the process for the project.	Creates guidelines for using a specific tool.
Configuration and Change Mgt	Configuration Manager	Configuration Manager
	Sets up the CM environment, policies, and plan.	Creates a deployment unit, reports on configuration status, performs audits, and so forth.
	Change Control Manager	Change Control Manager
	Establishes a change control process.	Reviews and manages change requests.
		(Again, note that breadth and depth roles are assigned to the same people in this discipline; assistant or associate managers in the depth roles would be helpful.)
A. Crain: Understanding RUP Roles		

- <http://www.ibm.com/developerworks/rational/library/apr05/crain/#N1007A>

Prilagodba strukture tima

□ Elastični model ekipe

- upravitelj ekipe – upravljanje osobljem (plaće, režije)
- voditelj ekipe – upravljanje razvojem (organizacija posla)
- projektant (analitičar-programer) – analiza, oblikovanje i izvedba
- programer (programer aplikacija) – kodiranje, testiranje
- administrator baze podataka – administriranje baze podataka
- sistem inženjer(i) – održavanje mreže i računala

□ Sastav ekipe odgovara poslovima koje treba obaviti.

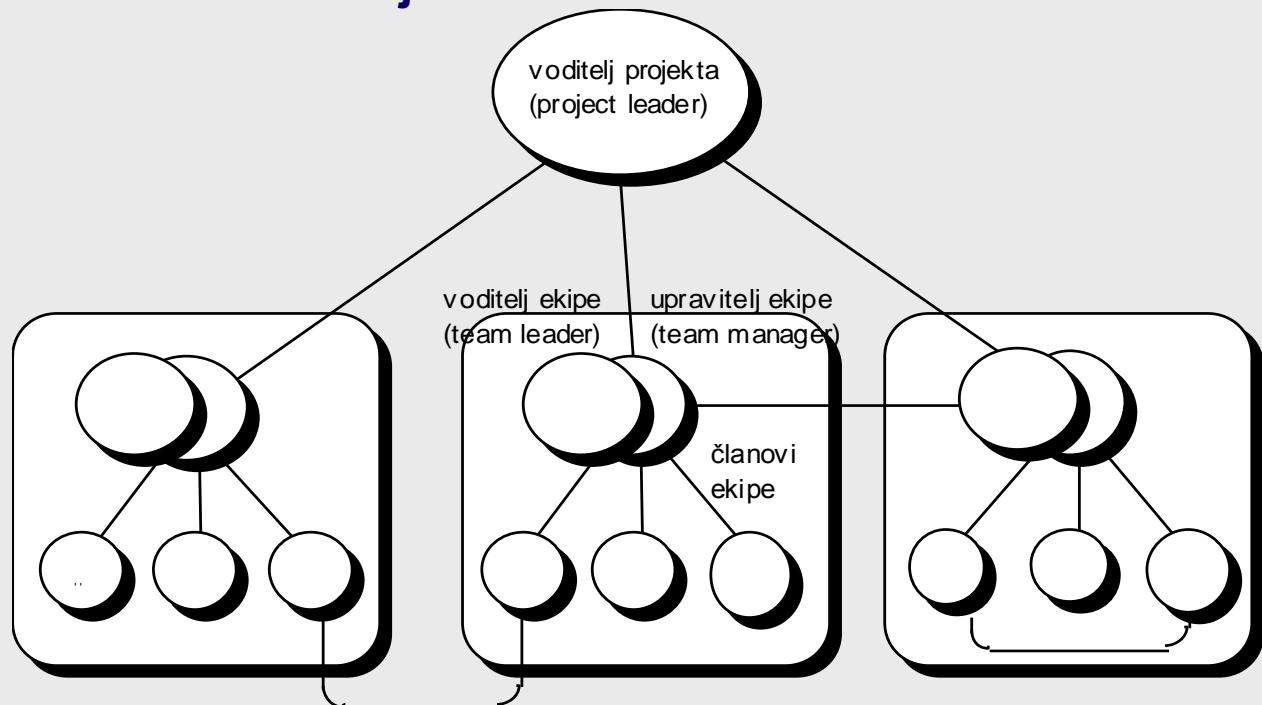
- Raspodjela uloga konkretnim članovima, kao i broj članova pojedine kategorije ovise o konkretnom projektu i raspoloživosti djelatnika.
- Na primjer:
 - ulogu upravitelja ekipe i voditelja ekipe može imati ista osoba
 - ekipa može imati više programera
 - uloga administratora BP i sistem inženjera može se dodijeliti istoj osobi

□ Primjer: Ministarstvo

Organizacija velikih projekata

- **Upravitelj ili voditelj projekta, (project manager, project leader)**
 - upravlja projektom
 - posao obavlja više ekipa
 - nadređen voditeljima / upraviteljima ekipa
- **Ekipa može imati i dva voditelja**

- **Upravitelj ekipe (team manager)**
 - planiranje, upravljanje i nadzor, rukovođenje ostalim članovima ekipe
- **Voditelj ekipe (team leader)**
 - tehnički aspekte aktivnosti koje se odnose na izradu i/ili uvođenje aplikacija/podsustava IS



Karakteristike dobrih ekipa

- **Zajednička, inspirirajuća vizija ili cilj**
 - Cilj koji se postavlja mora biti inspirirajući, a posao izazovan (!?)
- **Snažan identitet tima i osjećaj pripadnosti timu**
 - Uspjeh - samopouzdanje – povećani angažman – produktivnost – elitizam
- **Struktura orijentirana na povećanje produktivnosti (zarade)**
 - Jasne uloge, dobra komunikacija, odluke na temelju činjenica, objektivan sustav nagrađivanja
- **Kompetentni članovi**
 - Tehničke kompetencije (metodologije, platforme, programski jezici...), angažman i doprinos projektu, komunikacijske sposobnosti
- **Predanost timu i organizaciji**
 - poticaji: vizija, izazov, identitet tima ili karizmatični vođa
 - predanost je moguća samo ako je osoba rasterećena problema u radnoj okolini (radna atmosfera, financijska stabilnost)

Karakteristike dobrih ekipa (nastavak)

- **Međusobno povjerenje i međuzavisnost članova tima**
 - iskrenost, otvorenost, dosljednost i uvažavanje
- **Učinkovita komunikacija**
 - tehnološka i verbalna
- **Osnazivanje (*empowerment*) - osjećaj autonomije i ovlaštenja**
 - mogućnost poduzimanja akcija koje ekipa smatra potrebnim
- **Mali broj članova**
 - pravilo 7 ± 2 – četiri osobe nisu dovoljne za stvaranje grupnog identiteta, deset i više je teško koordinirati
- **Uživanje u poslu**
 - članovi dobrih timova vole biti produktivni, a ako rade posao koji vole, napravit će još više posla
 - može se povećati uz (ograničenu) zabavu i humor

Karakteristike dobrog voditelja

- Voditelj mora zastupati interes organizacije i interes članova tima**
- Voditelj mora biti podržan u provođenju svojih ovlasti jer inače gubi autoritet**
- Mogućnost nagrađivanja i kažnjavanja članova tima**
 - direktno ili indirektno preko viših razina upravljanja
- Objektivnost**
 - dobra procjena količine i kvalitete posla, shodno tome vrednovanje članova
 - jednaka mjerila za sve članove tima
- Tehnička kompetencija**
 - dovoljno znanje – razumijevanje problema i procjena napora
- Preuzimanje odgovornosti**
 - autoritet i odgovornost za donošenje “najbolje”, makar krive, odluke
- Brzo i efikasno rješavanje konflikata**
 - npr. rješavanje problema problematičnog člana

Vrednovanje, nagrađivanje

□ Državne institucije

- rangiranje službenika i namještenika uz fiksnu plaću po razredima
- primjer: administrativni referent ima koeficijent 1, dipl.inž. 1.45, ...
- državne tvrtke - nemogućnost stimulacije (ili podjele honorara)

□ Poduzeća koja dozvoljavaju stimulaciju

- plaće prema sistematizaciji radnih mjesta i učinku
- dodatna stimulacija u ingerenciji neposrednog rukovoditelja
- sindromi: svima jednako, svaki mjesec nekom drugom

□ Tvrtke u privatnom vlasništvu

- primjer vrednovanja: 1-12 kKN

□ Dohodak od nesamostalnog rada (honorari)

- izlazni račun 123 KN - PDV = 100 KN
- 100 KN - 10% FER - 5% FOND - 2% REŽIJE = 83 KN
- 83 KN = 15 % IMT + 85 % izvođači
- pri isplati se odbija porez i pritez na dohodak građana, a nekome i PDV

Raspodjela uspjeha

□ Kriteriji za procjenu doprinosa članova

- uloženi trud (vrijeme)
- rezultati rada (korisnost, kvaliteta proizvoda i zadovoljstvo korisnika)
- objektivna težina posla (vrsta, uvjeti, intenzitet i trajanje)
- pristup radu (zalaganje, samoinicijativa, kooperativnost, odgovornost)
- sposobnost (svestranost, samostalnost)

□ Stimuliranje, nagrađivanje i trajno usavršavanje

- Pretplata na stručne časopise, kupnja stručnih knjiga.
- Sudjelovanje na prezentacijama informatičke opreme i sajmovima, s ciljem praćenja stanja i trendova IT
- Sudjelovanje na stručnim konferencijama, s ciljem praćenja razvoja struke. Može se uvjetovati pisanjem stručnih radova vezanih uz projekte i probleme
- Stimulacija kroz formalnu izobrazbu (dodiplomski, poslijediplomski studij) kao uvjet ostanka u tvrtki u određenom razdoblju
- Pohađanje stručnih tečajeva, radionica ili tečajeva iz engleskog jezika

Elementi upravljanja projektom

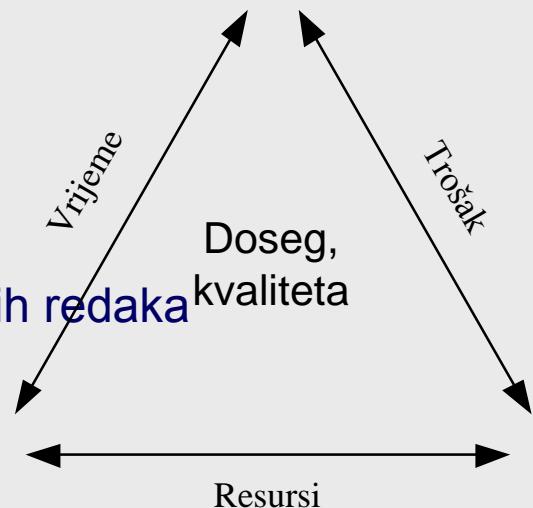
Planiranje projekta

□ Planiranje projekta (planning)

- odrediti doseg, rokove, trošak i ostale resurse
- identificirati pokroviteljstvo (sponsorship) kao jamstvo provedbe
- izabrati upravitelja/voditelja projekta
- odabratи alate za upravljanje projektom
- pokrenuti projekt

□ Elementi plana

- Veličina projekta: funkcione točke, broj programskih redaka
- Napor izrade: čovjek-mjeseci
- Vrijeme izrade: mjeseci



□ Izgradnja IS se obavlja kao neki drugi inženjerski poslovi, u planiranom vremenu i s planiranim resursima

- uravnotežiti zahtjeve na kvalitetu, doseg, vrijeme i trošak

Vremenski raspored projekta

□ Projekt kao definiran redoslijed aktivnosti

- aktivnost – komad posla, dio posla koji ima određen ulaz i izlaz
- zadatak – manja jedinica, korak aktivnosti

□ WBS (work breakdown structure) - struktorna raščlamba posla

- hijerarhija aktivnosti i zadataka te kontrolnih točki (*milestone* – prekretnica)
- projekt – podprojekti – faze – radni paketi – aktivnosti – zadaci
- radni paket : 8-80 sati

□ Planiranje vremena, izrada rasporeda (scheduling)

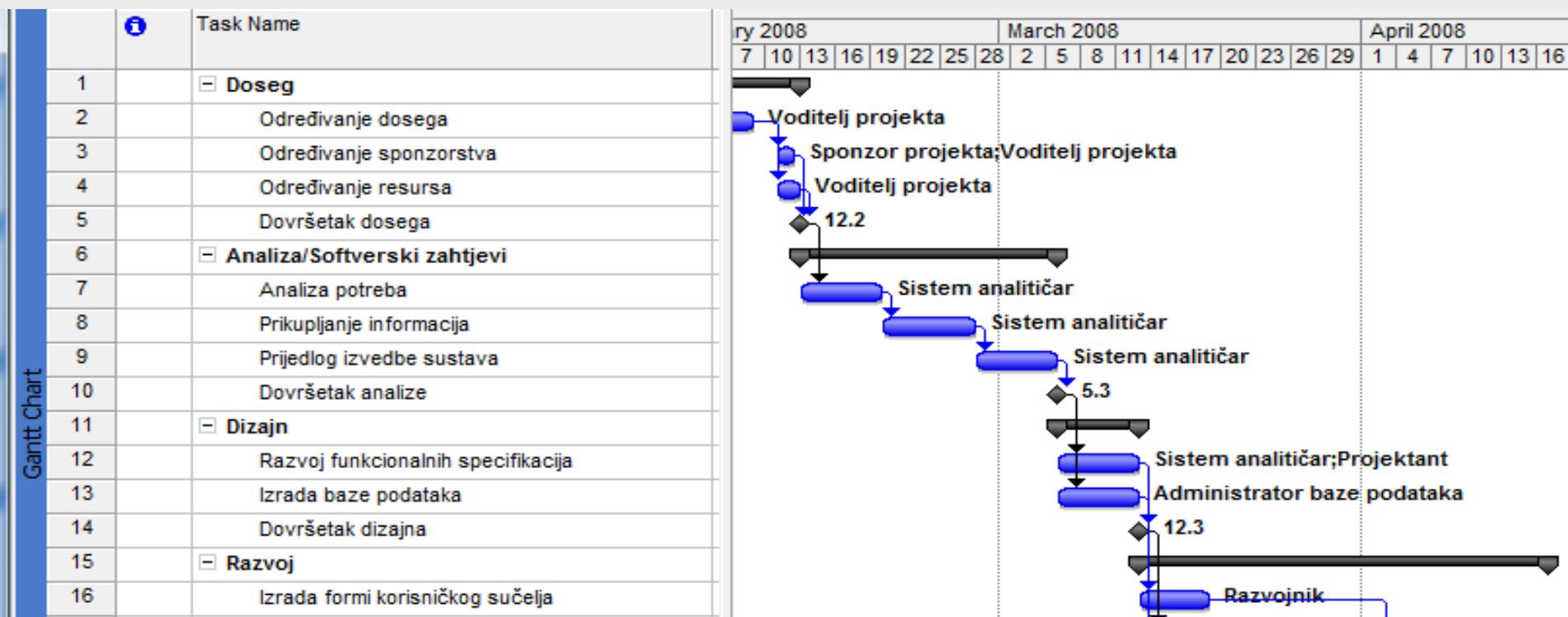
- određivanje aktivnosti
- procjena i pridjeljivanje sredstava potrebnih za pojedinu aktivnost
- procjena trajanja pojedine aktivnosti
- određivanje zavisnosti između aktivnosti
- pripisati/racionalizirati pripadne troškove
- iterativno razraditi plan

□ Revizija plana sukladno iskustvu/saznanjima → praćenje i kontrola

Primjer vremenskog rasporeda

□ Gantov dijagram (Gantt Chart)

- elementarni zadaci (taks)
- grupe zadataka (summary tasks)
- prekretnice ili miljokazi (milestones)
 - ključni događaj ili rok koji treba postići, trajanja 0

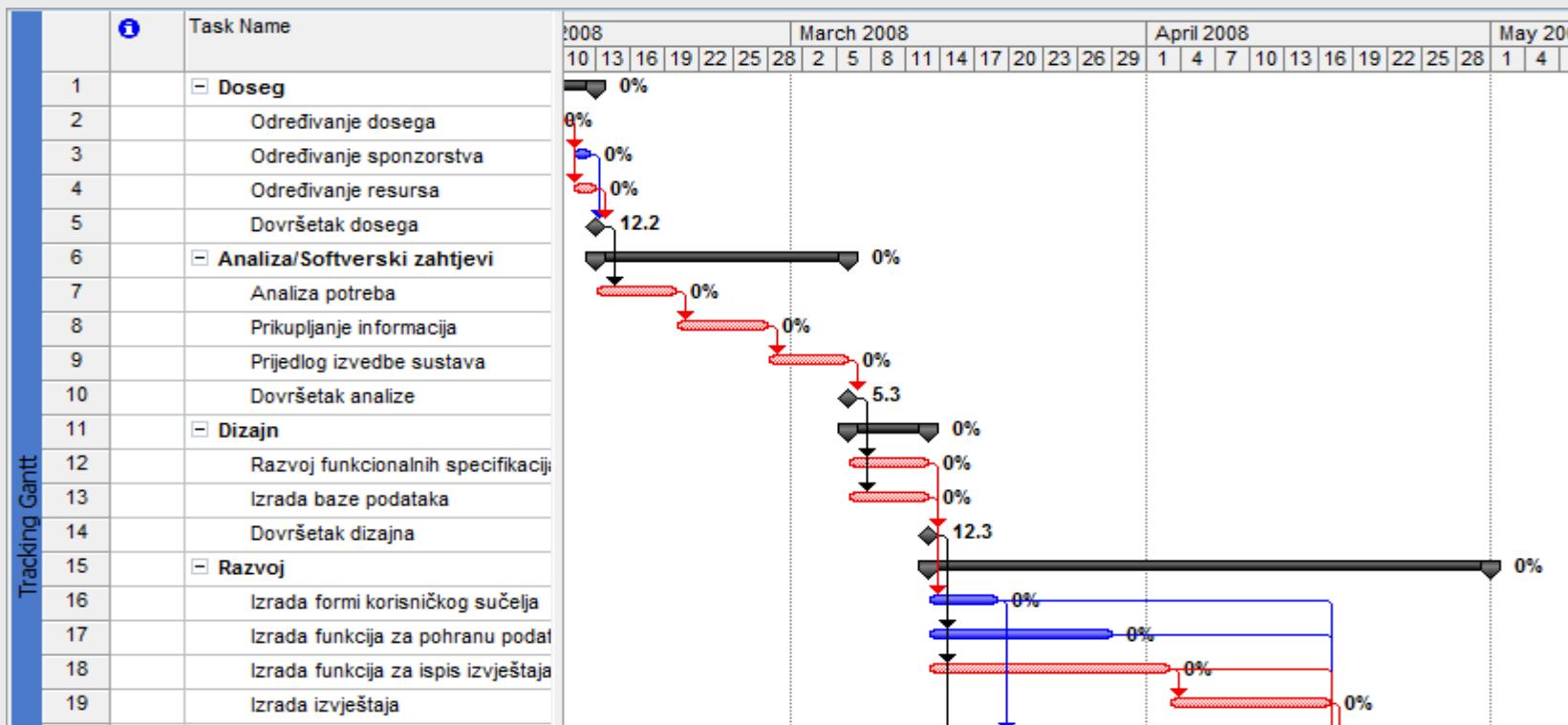


Kritični put

□ Kritični put

- niz zadataka koji moraju završiti na vrijeme da bi projekt završio na vrijeme
- svaki zadatak na kritičnom putu je kritični zadatak
- kašnjenje kritičnih zadataka uzrokuje kašnjenje projekta

□ Primjer kritičnog puta (View/Tracking Gantt)



Nadziranje i kontrola projekta

Nadzor i kontrola (monitor and control)

- određivanje postupka izvještavanja o napretku projekta
- praćenje napretka redovitim revizijama plana projekta
- preraspodjela sredstava i aktivnosti sukladno stanju i događajima
- ažuriranje vremenskog rasporeda

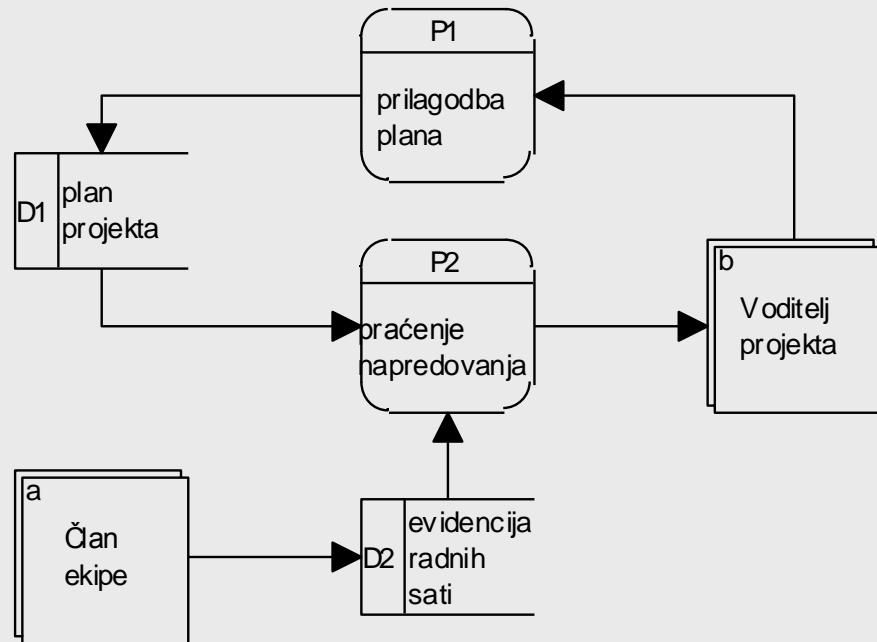
Postupci nadzora i kontrole

□ Nadzor

- Nadzire se posao, a ne zaposlenici
- Provjera dovršenosti posla i izvršenja plana
- postavljanje kontrolnih točki
- praćenje napretka - evidencija radnih sati, Tracking Gantt

□ Kontrola

- Ignoriranje – "nije važno" ili "riješit će se samo"
- Korektivne akcije – nagovaranje ili prisila izvođača na "dobru volju"
- Revidiranje plana – preraspodjela nositelja zadataka, ažuriranje rasporeda
- Pojačani nadzor – promjena učestalosti ili detalja nadzora

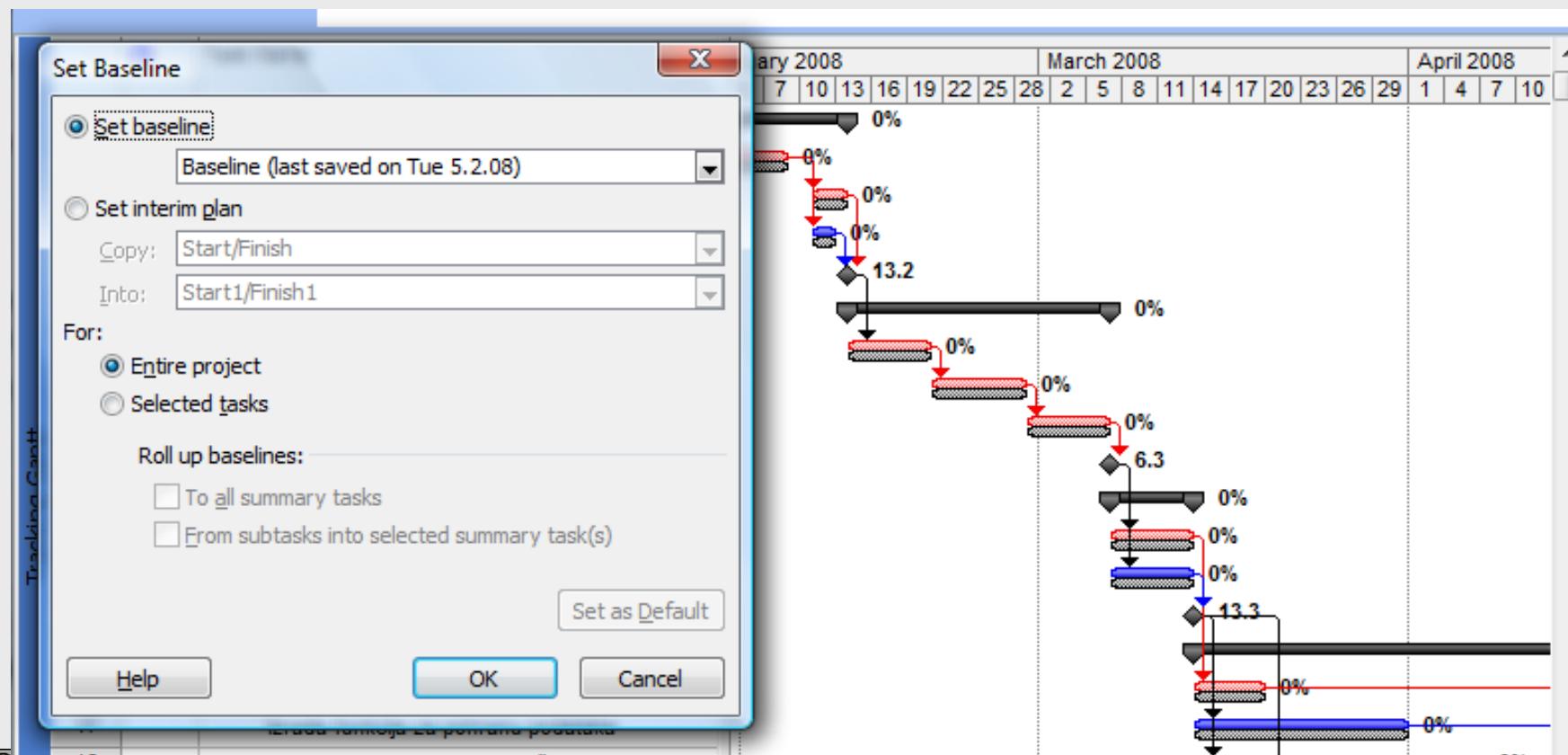


Zadatak	Izvršitelji	Planirani početak	Planirani završetak	Prioritet	Planirano trajanje	Stvarno trajanje	Status	% ispunjenja	Stvarni završetak	Kašnjenje	Komentar

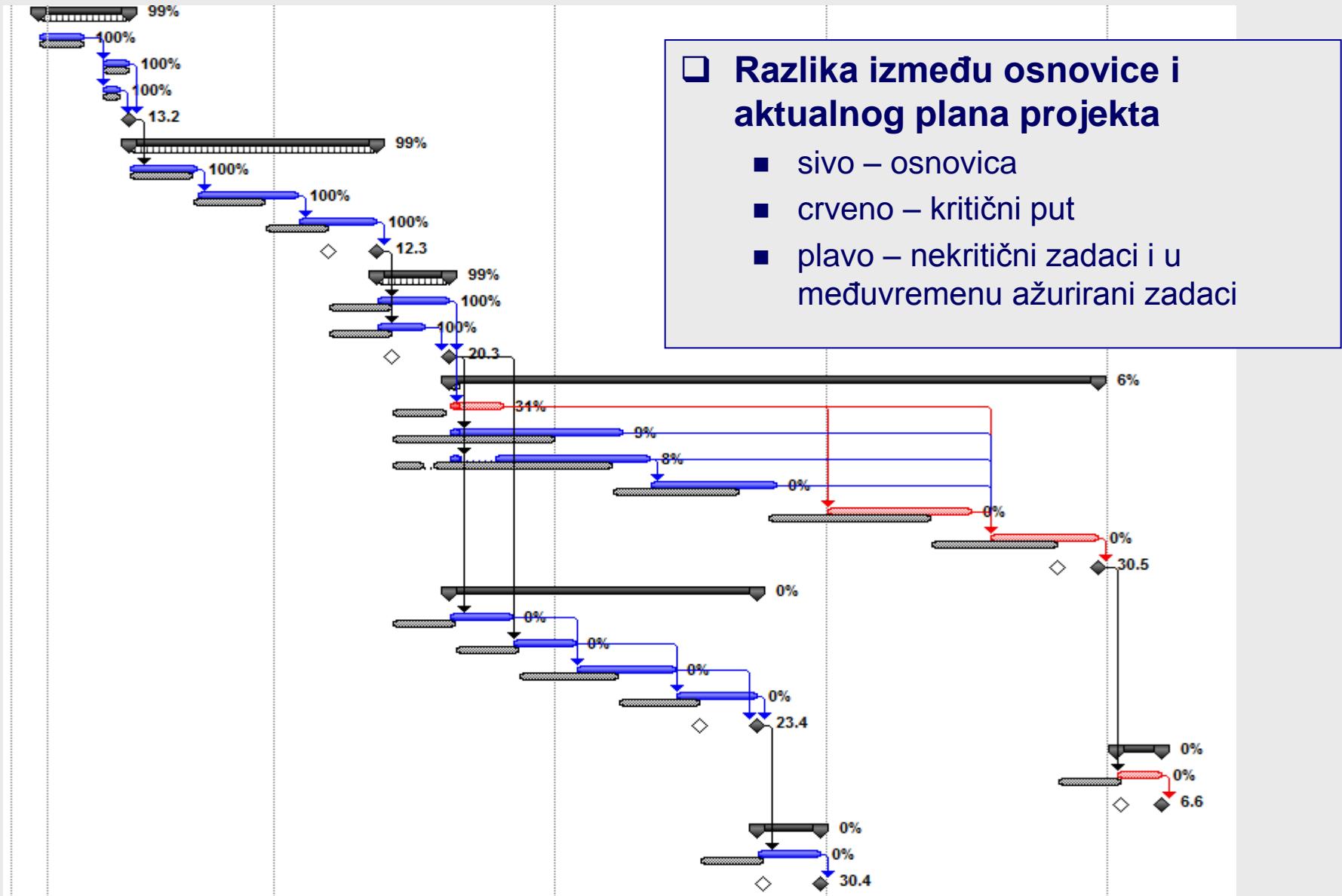
Praćenje napretka – spremanje osnovice

□ Osnovica (baseline) – plan u odnosu na koji se prati napredak

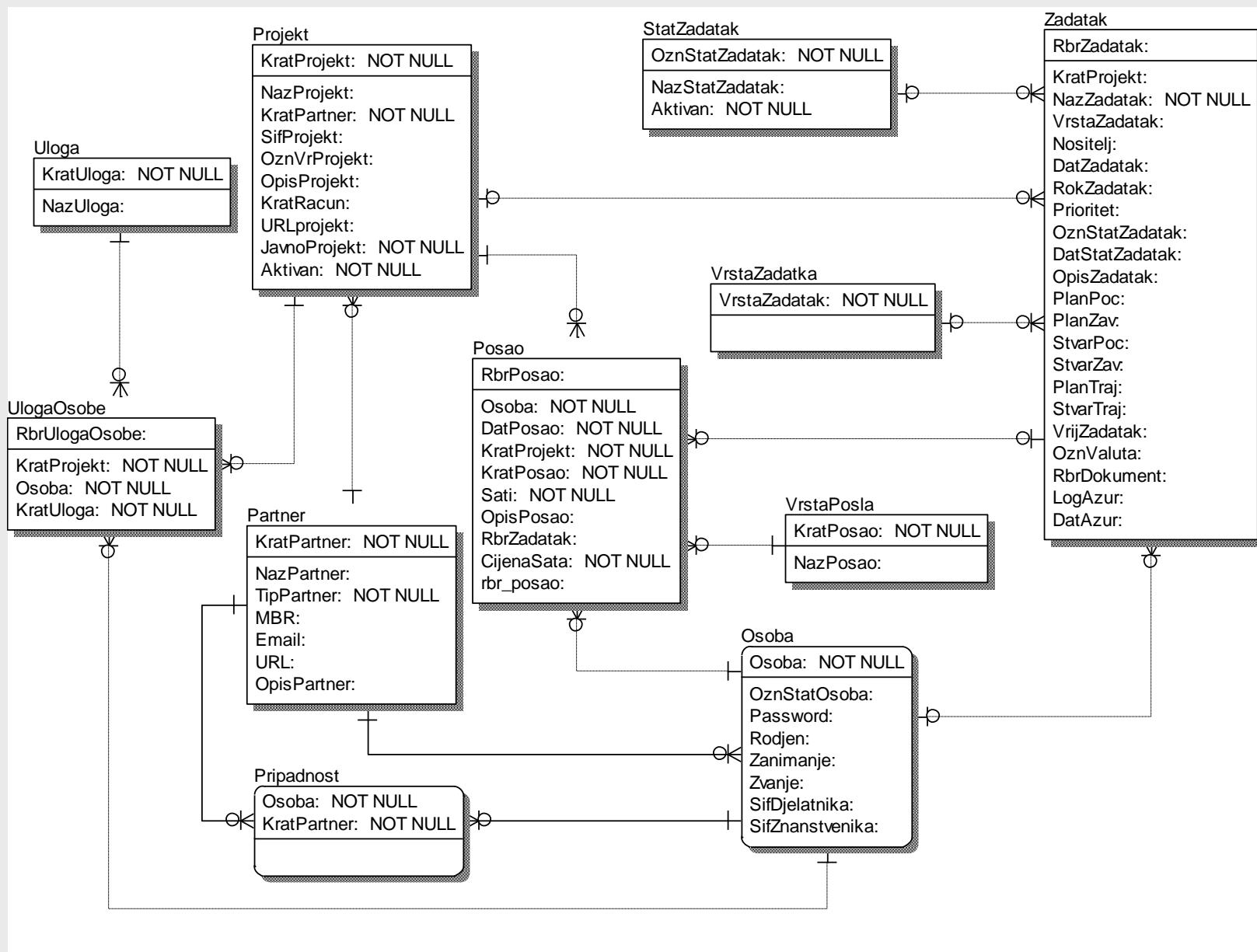
- skup važnih vrijednosti u projektnom planu, kao što je planirani datum početka, završni datum i trošak zadatka, resursa
- primjer (MSP) - plan snimljen u osnovicu plana projekta
- Tools/Tracking/Set Baseline...



Praćenje napretka u odnosu na osnovicu



Elementi projektne evidencije



Nadzor napretka i performansi, kolaboracija

Zadaci	Naslov	Modul	Prioritet	Sati	Status	Dodijeljeno	Stvoreno	Izmijenjeno	Krajnji rok
Broj = 112 Zbroj = 1.014									
Optimizacija formi		(1) Visok		Napreduje		18.12.2008 14:29	22.12.2008 15:16		
Bug - blagajnicki dnevnik	Blagajna	(1) Visok		Nezapočeto	Krešimir Fertalj	19.12.2008 12:53	19.12.2008 12:53		
Ne radi Return u pozicijama	Generalno	(1) Visok	0	Nezapočeto	Tomislav Rajnović	17.12.2008 11:30	17.12.2008 11:30		
Pivot table za izvjestaje	Plaće	(1) Visok		Nezapočeto	Ivan Benussi	8.12.2008 13:31	8.12.2008 13:31		
Povrat robe od kupca	Skladište	(1) Visok	24	Nezapočeto	Bruno Dević	5.9.2008 16:07	4.12.2008 16:35		
Ispis povratnice kupca	Skladište	(1) Visok	8	Nezapočeto	Bruno Dević	5.9.2008 16:07	4.12.2008 16:32		
Generiranje podataka		(1) Visok		Napreduje	Bruno Dević	20.10.2008 17:19	4.12.2008 16:31		
Login ekran		(1) Visok		Nezapočeto		2.12.2008 11:26	4.12.2008 0:53		
Integracija plaća - poslovne klase	Plaće	(1) Visok		Napreduje	Ivan Rendulić	4.11.2008 16:13	4.12.2008 0:52		
M3 kod odlaska skladišta u minus	Robno	(1) Visok	4	Napreduje	Davor Smojver	20.11.2008 10:14	1.12.2008 14:16		
Osigurati evidentiranje isporučenih u odnosu na naručene količine robe od dobavljača.	FIC.Robno	(1) Visok		Čekanje	Tomislav Jaklin	5.9.2008 16:33	28.11.2008 18:57		
Promet asortirana	Robno-Statistika	(1) Visok	4	Nezapočeto	Bruno Dević	5.9.2008 16:07	28.11.2008 18:55		
Osoba	DatPosla	KratProjekta	KratPosla	Sati	Opis				

Događaji

Trenutno nema nadolazećih događaja. Da biste dodali novi događaj, kliknite "Dodaj novi događaj".

Dodaj novi događaj

Obavijesti

Broj aktivnih taskova i potrebnog vremena je 112, a zadani su od Krešimir Fertalj

23.11. 117/1099

07.11. 122/1202

19.10. 135/1445

03.10. 139/1509

...

15.09. 151/1600 preciziranje FIC-a

05.09. 161/1484 gruba procjena

Završetak faze

od Krešimir Fertalj

Cilj je "odmah"

- * završiti Plaće (integracija GUI, a zatim API)
- * Osnovna sredstva (5 zadataka koje su uvećani)
- * ključne preostale rob-mat taskove "odmah"

Renta i Tomislav rade Plaće, Davor i Vlatko

Remote Debugging

od Vlatko Malović

U slučaju kada se bug može producirati samo u verziji aplikacije koju je razvila druga generacija developera sve radi, moguće je obaviti udaljeni debug na drugoj verziji aplikacije. Upute za udaljeni debug (Dokumenti/Razvoj (<http://bob.zpr.fer.hr/projekti/PROLOGIS/Debugging/20debugging.docx>)).

Dodajte novu najavu



Upravljanje komunikacijom

□ Konzultacije i razmjena informacija

- zajedničko dnevno druženje svih članova ekipe
- dinamičko rješavanje detalja konkretnih praktičnih problema
- dnevno po 15++ min., na početku radnog dana ili prije/poslije pauze
- po potrebi se mogu sazvati i izvan dogovorenih termina

□ Sastanci

- upoznavanje s trenutnim stanjem, realizacijom plana i dalnjim poslovima
- razmatranje problema, predlaganje načina i redoslijeda njihovog rješavanja
- kratki, tjedno, unaprijed dogovorenog dana, najbolje na početku tjedna
- kontrolni, jednom mjesечно ili na kraju faze
- po potrebi se mogu sazvati i izvan dogovorenih termina
- sastanci moraju biti pripremljeni - mjesto, vrijeme, teme, sudionici
- izbjegavati raspravu o općepoznatim ili nevažnim stvarima
- treba ih prekinuti u trenutku kada se pretvore u razgovor o temama koje se ne odnose na posao

Upravljanje vremenom (time management)

□ pravilno raspoređivanje poslova

- točnom procjenom što i kada treba napraviti
- podjelom posla u pod-poslove
- izradom izvješća o napredovanju i gotovosti

□ izbjegavanje obavljanja tuđih poslova (problem delegiranja)

- "mogu to i sam"
- "mogu to brže"
- "teže mi je objasniti, nego napraviti"

□ neprimjeren utrošak vremena (razbacivanje vremenom)

- druženje – telefonski razgovori, neproduktivna konverzacija
- započimanje – problem "prebacivanja" s jednog posla na drugi ili "promjene brzine" → pokušati s manje poslova i grupirati slične poslove
- ugodni poslovi → izbjegavati produljenje rada na poslovima "za gušt"
- neugodni poslovi → izbjegavati odgovlačenje neugodnih ili teških poslova

Dokumentiranje projekta

□ Povelja projekta (Project Charter)

- Dokument kojim pokretač projekta ili sponzor formalno odobrava postojanje projekta i kojim ovlašćuje voditelja za primjenu organizacijskih resursa u provedbi projekta.

□ Plan upravljanja softverskim projektom (Software Project Management Plan)

- IEEE Standard for Software Project Management Plans 1058-1998

□ Primjeri

- ProjectCharter
- ProjectPlan

1. Introduction

- 1.1 Project Overview
- 1.2 Project Deliverables
- 1.3 Evolution of the Software Project Management Plan
- 1.4 Reference Materials
- 1.5 Definitions and Acronyms

2. Project Organization

- 2.1 Process Model
- 2.2 Organizational Structure
- 2.3 Organizational Boundaries and Interfaces
- 2.4 Project Responsibilities

3. Managerial Process

- 3.1 Management Objectives and Priorities
- 3.2 Assumptions, Dependencies, and Constraints
- 3.3 Risk Management
- 3.4 Monitoring and Controlling Mechanisms
- 3.5 Staffing Plan

4. Technical Process

- 4.1 Methods, Tools, and Techniques
- 4.2 Software Documentation
- 4.3 Project Support Functions

5. Work Packages, Schedule, and Budget

- 5.1 Work Packages
- 5.2 Dependencies
- 5.3 Resource Requirements
- 5.4 Budget and Resource Allocation
- 5.5 Schedule

6. Additional Components

7. Index

8. Appendices

Metode za upravljanje projektima

- CPM (Critical Path Method)
 - vremensko raspoređivanje određivanjem kritičnog puta
- PERT (Program Evaluation and Review Technique)
 - vremensko raspoređivanje procjenom optimističkog/vjerojatnog/pesimističkog trajanja
- PRINCE (PRojects IN Controlled Environments)
 - strukturirana metoda za upravljanje projektom
 - definiranje organizacijske strukture projekta
 - definiranje strukture i sadržaja plana projekta
 - definira skup provjera i izvješća koji se koriste za nadzor provedbe
- COCOMO
 - model određivanja troškova softvera

Mitovi i legende

□ Zašto planirati ? "Gore ne može"

- Sve što može poći krivo, poći će krivo (Murphyjev zakon)
- Murphy je bio optimist ! (Grimmov korolar)
- Neuspješno planiranje = planiranje neuspjeha

□ Plan

- najbolja procjena, zasnovana na pretpostavkama i iskustvu → revizija plana

□ Paretovo načelo – pravilo 80/20 [V. Pareto, talijanski ekonomist, 1906]

- 20% problema izaziva 80% posla

□ Parkinsonov zakon [C.N. Parkinson, The Economist, 1955]

- rad se povećava tako da potroši čitavo planirano ili raspoloživo vrijeme

□ Brooksov zakon, Programerski paradoks [F.Brooks, 1975, 1982, 1995]

- *The Mythical Man-Month, No silver bullet*
- dodavanje osoblja u projekt koji kasni povećava kašnjenje

□ Fertaljev poučak

- za neki posao treba vremena upravo onoliko koliko je raspoloživo !

Reference

□ Udruge

- Project Management Institute (PMI)
 - <http://www.pmi.org/> , <http://www.pmi-croatia.hr/>
- International Project Management Association (IPMA)
 - <http://www.ipma.ch/>, <http://www.capm.hr/>, <http://www.vspu.hr/ipma/>

□ Materijali

- http://www.ebook3000.com/Software-Project-Management-in-Practice_22125.html
- <http://office.microsoft.com/en-us/help/HA011361531033.aspx>

□ Alati

- <http://www.smashingmagazine.com/2008/11/13/15-useful-project-management-tools/>