

Upravljanje protokom poslova

2013/14.08

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

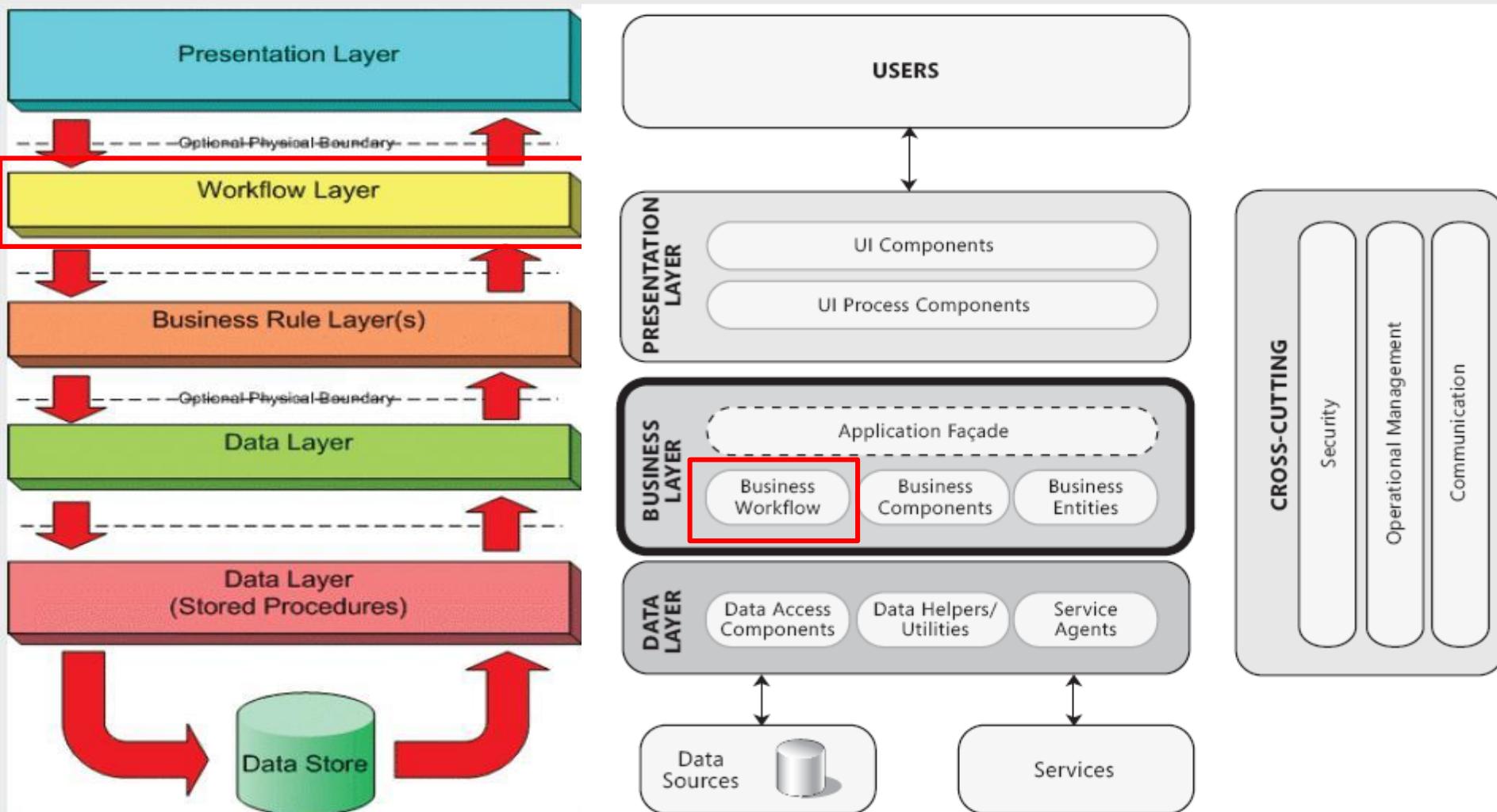
- Smanjen rizik razvoja – zajednički jezik analitičara i razvojnika
- Centralizirana implementacija – promjena poslovnog procesa ne zahtjeva značajne (drastične) promjene u softveru
- Brzi razvoj – slaganje procesa ubrzava razvoj i olakšava održavanje

□ Općenita prednost

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

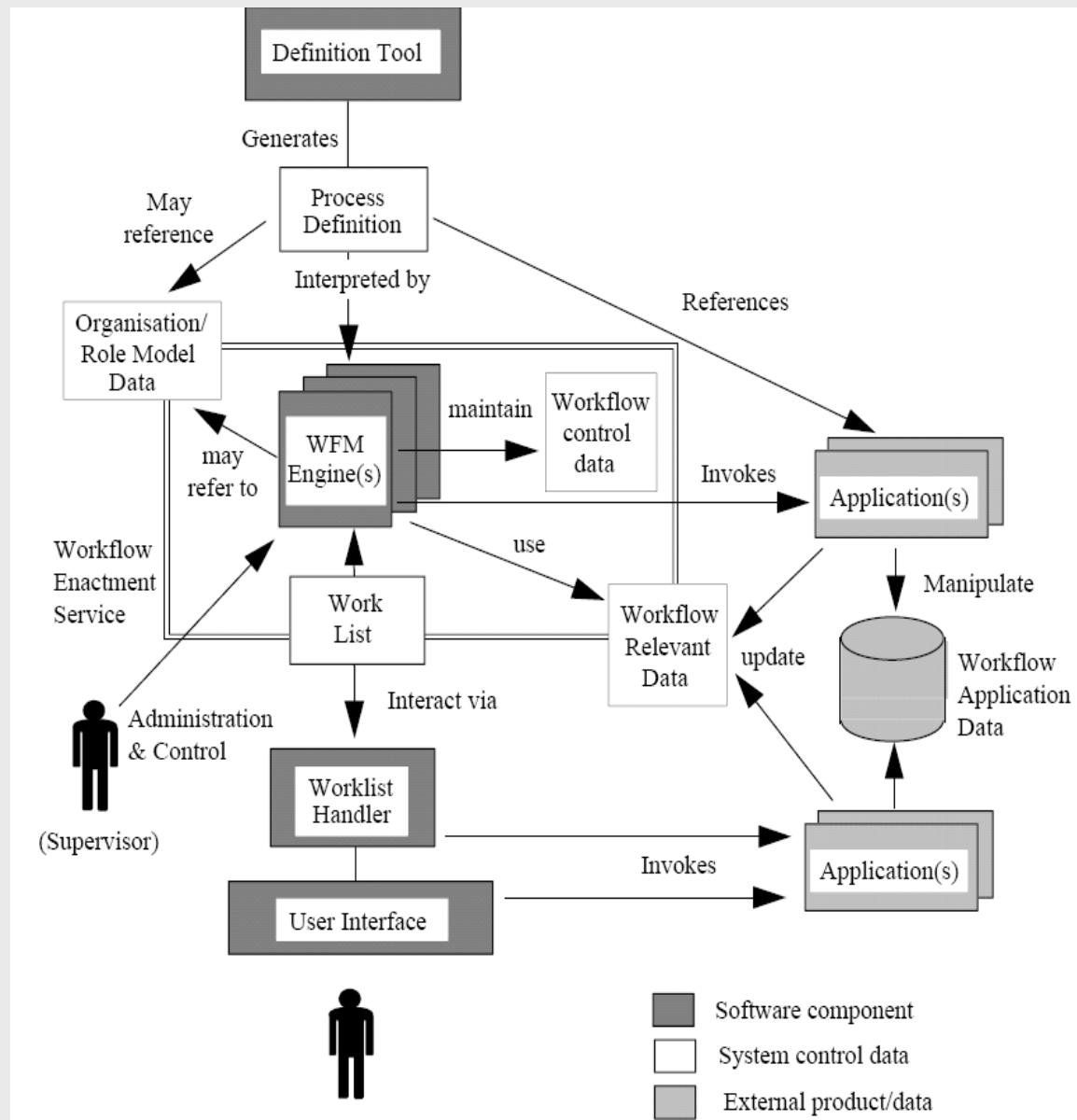
Protok poslova u višeslojnoj aplikaciji

- 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 sustava za ali se pozivaju od strane sustava

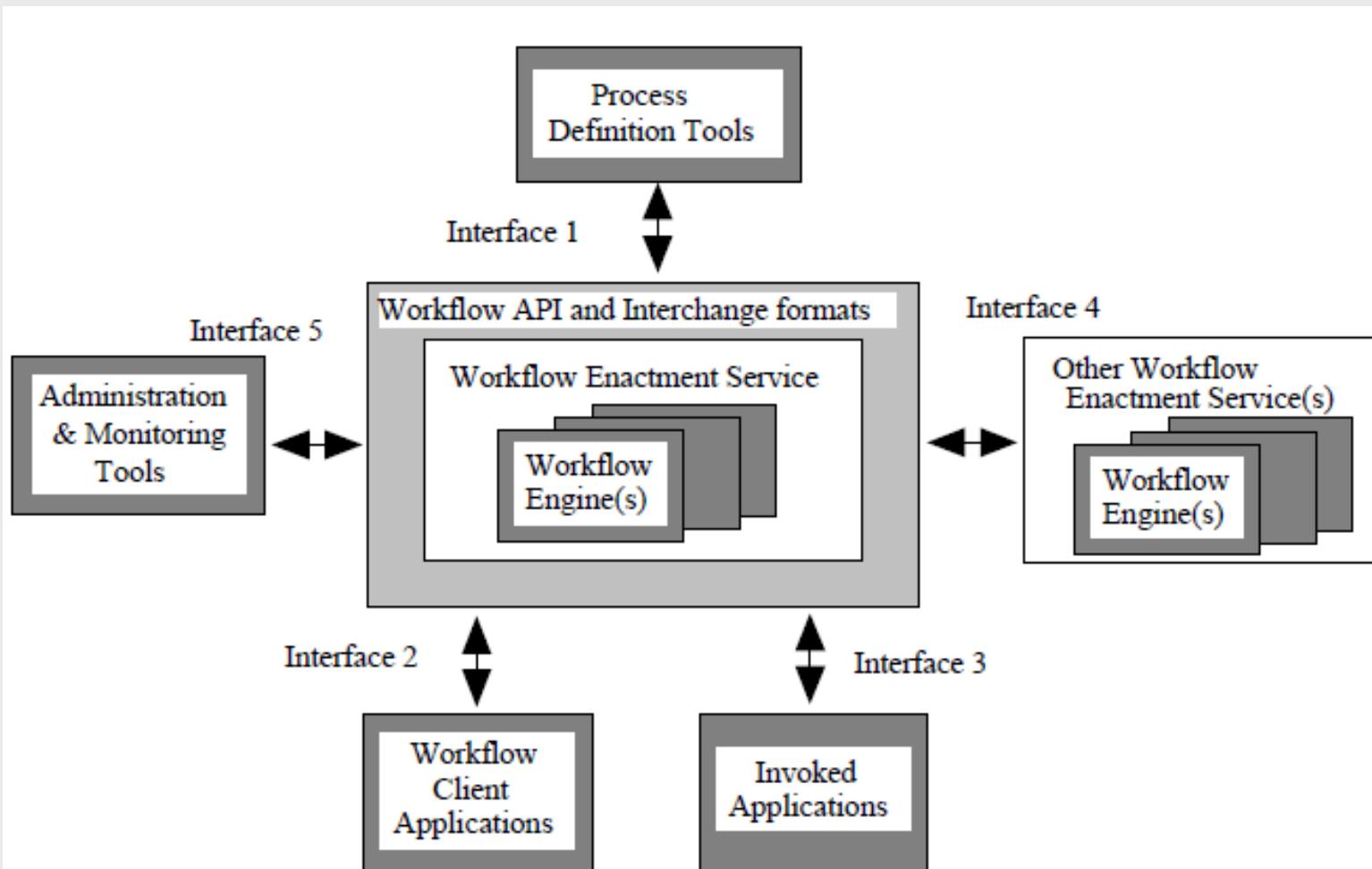


Referentni model

- različiti SUPP, nepostojanje jedinstvenog, zajedničkog standarda

□ Nastojanje:

- 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

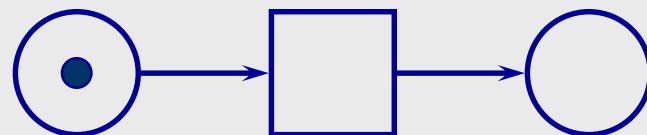
□ Složeniji obrasci (>40 obrazaca)

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

Slijed

□ Sequence

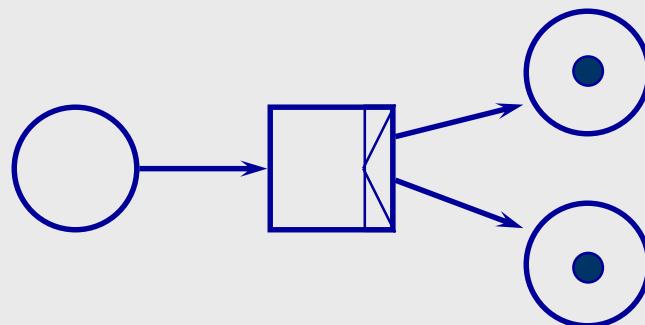
- prijelaz između dvije aktivnosti,
 - gdje se slijedeća aktivnost izvodi odmah nakon završetka prethodne
- ne postoje posebni uvjeti prijelaza,
 - početak slijedne aktivnosti ovisi samo o možebitnim vanjskim okolnostima,
 - pr. (ne)postojanje resursa koji mogu započeti aktivnost



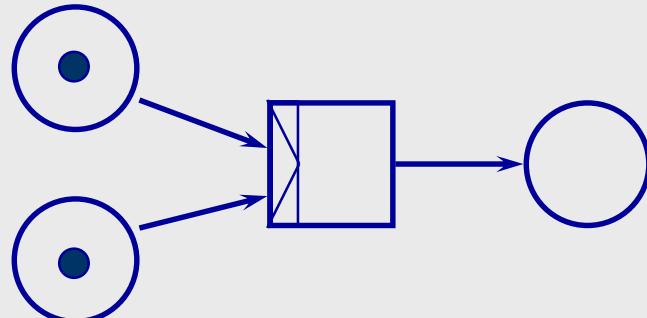
– Prema notaciji iz [Aalst2000] : prijelaz - pravokutnik, aktivnost - krug

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



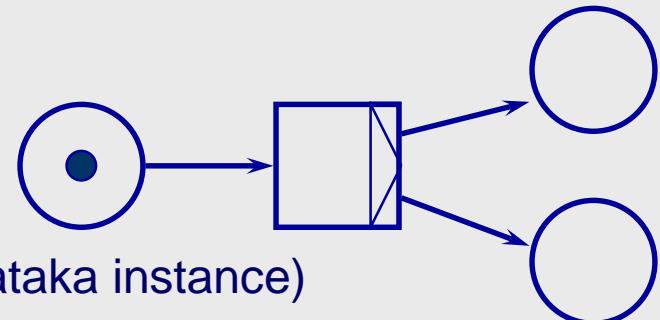
Odabir

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

- nakon završetka prethodne aktivnosti dolazi do odabira jedne od grana

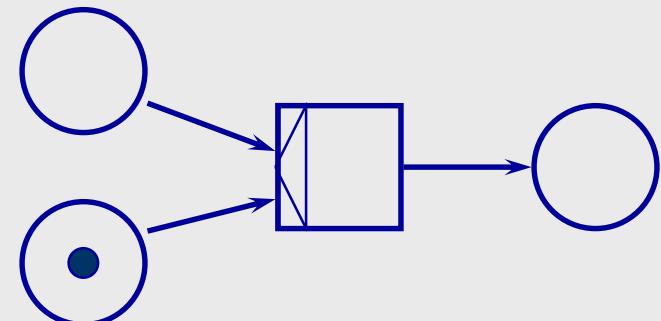
□ Različite varijante:

- 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)

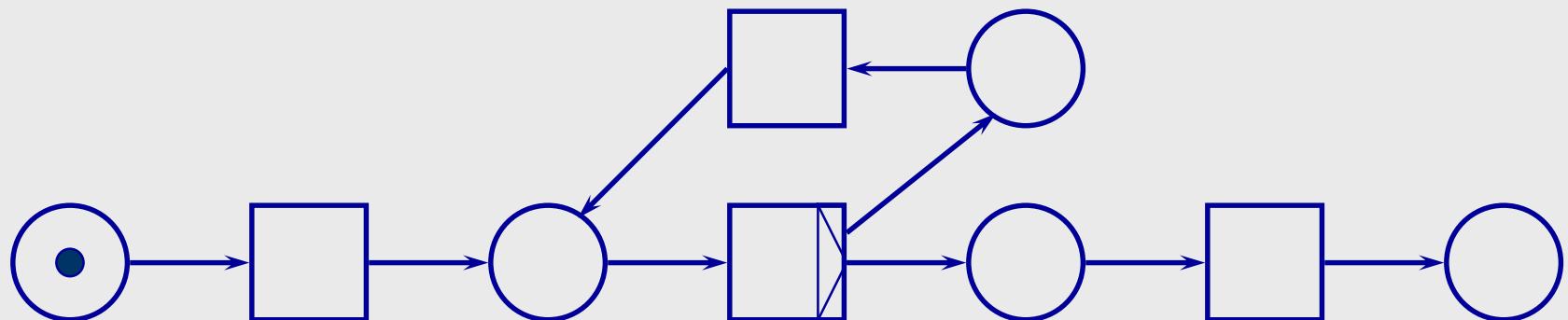
- složenija izvedba ako se radi o višestrukom odabiru



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

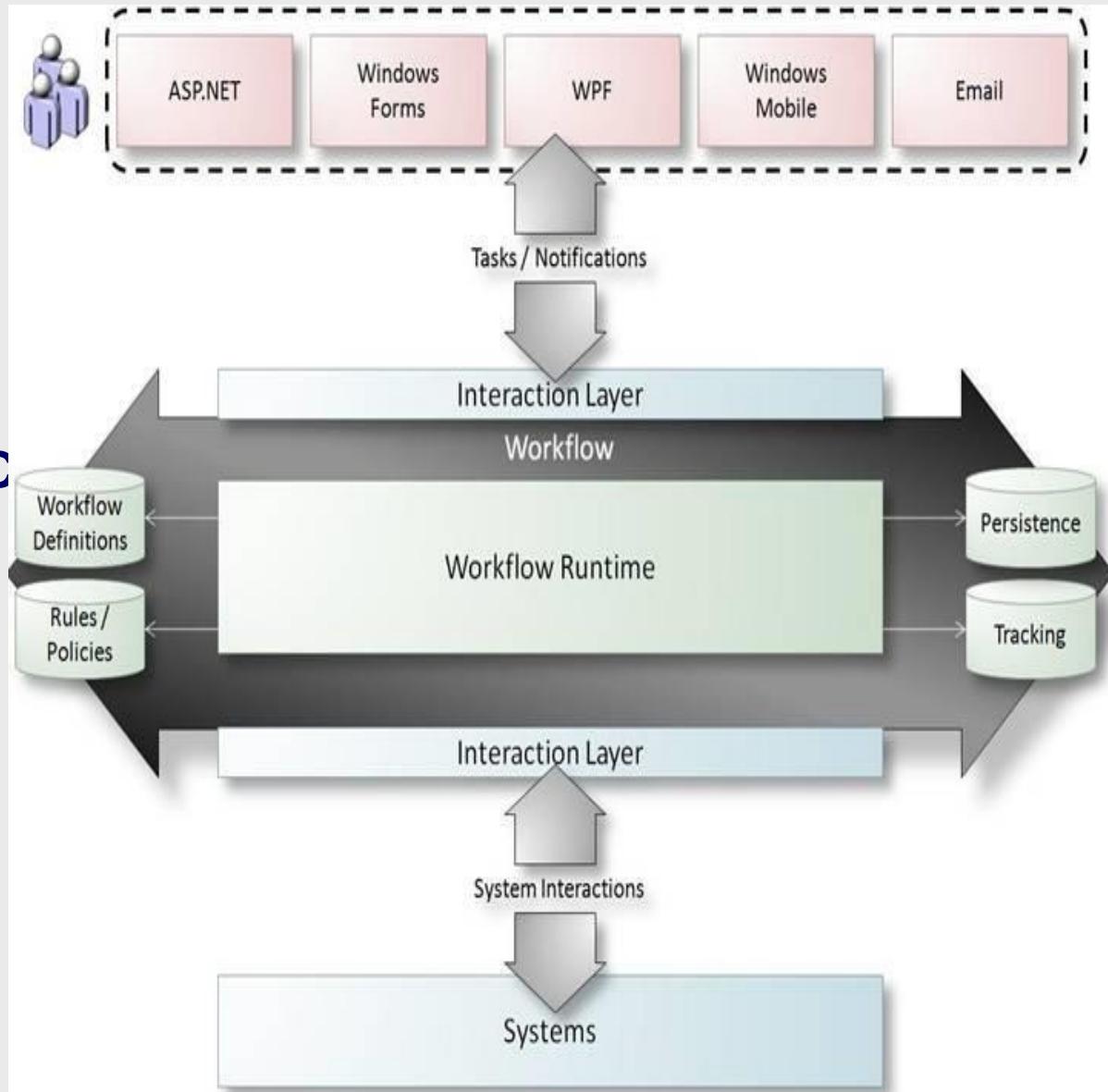


Programski primjeri

 WFprimjer.zip

Windows Workflow Foundation (WF)

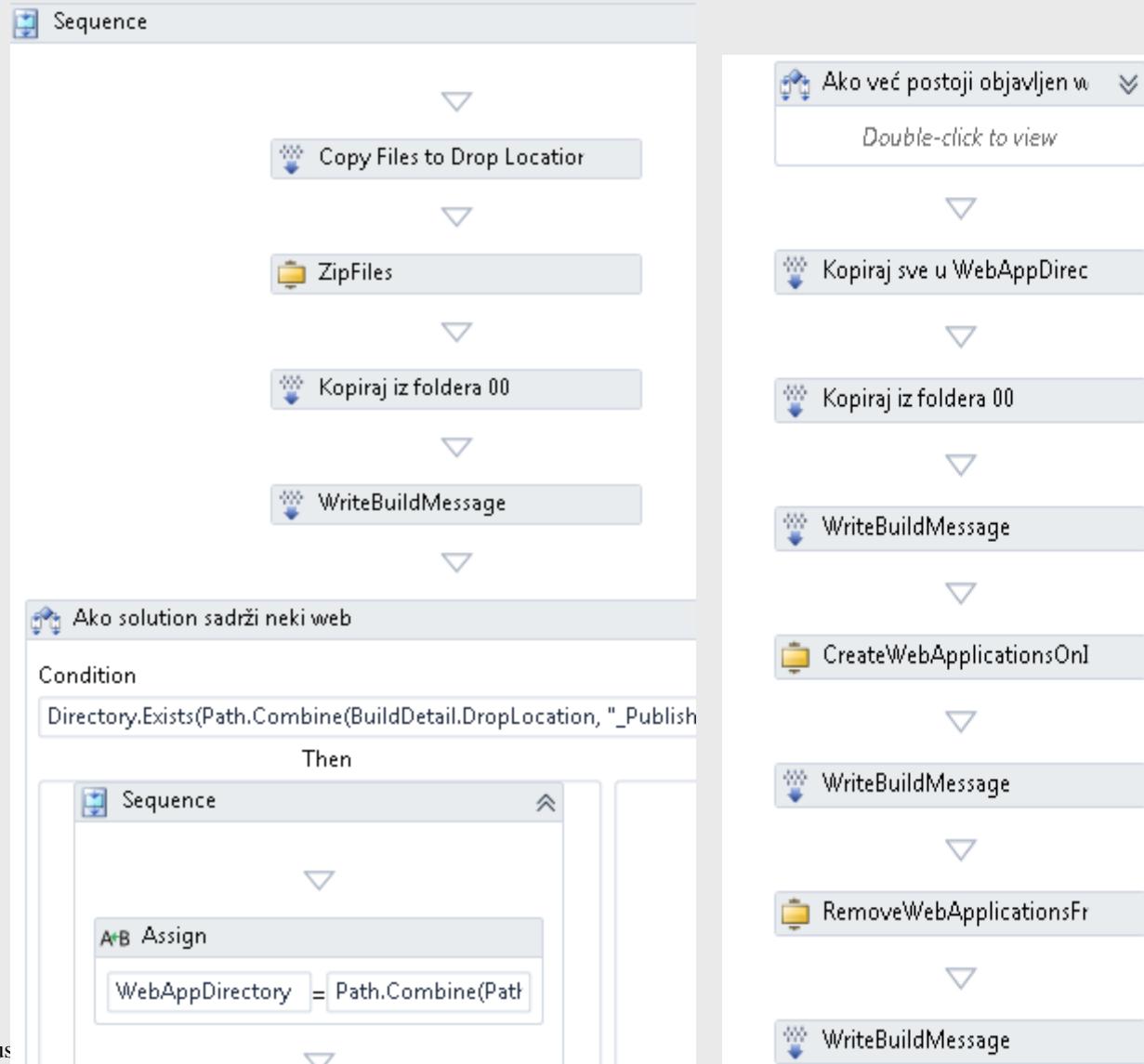
- Okvir za ugradnju protoka u aplikacije
 - API za *workflow*
 - Nije zasebna aplikacija ili server (kao npr. Biztalk server)
- Koristi XML za definiciju, ali nije WfMC kompatibilan
- Tipovi modela u WF-u
 - ControlFlow
 - Flowchart
 - StateMachine
- Ugrađene komponente za praćenje rada i "postojanost"



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

Dobrodošao Boris Milašinović Akcije

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.

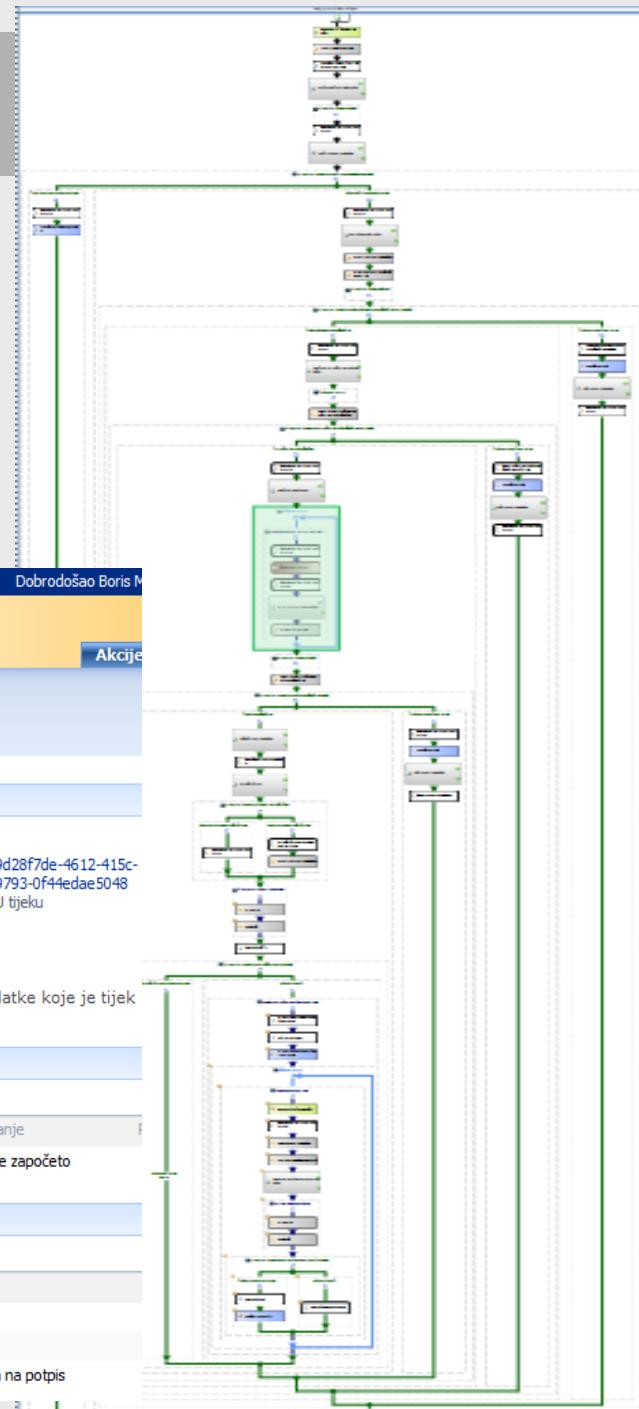
Dodatajeno	Naslov	Krajnji rok	Stanje
Krešimir Fertalj	Putni nalog za djelatnika: Boris Milašinović(Maribor, 12.07.2010.) Broj naloga:12437 Novo!		Nije započeto

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

cc BY NC SA FER \ Fertalj: Razvoj informacijskih sustava



Značajnije aktivnosti (*ControlFlow*, *Messaging*)

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*

Pick

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

Send i Receive

- poziv WCF servisa iz modela PP i izlaganje modela kao WCF servisa
- Obično idu u paru (*ReceiveAndSendReply* i *SendAndReceiveReply*)

Značajnije aktivnosti (*Primitives, FlowChart*)

□ **Assign**

- pridruživanje nove vrijednosti varijabli ili argumentu

□ **Delay**

- Zaustavlja izvršavanje (trenutne grane) na određeno vrijeme

□ **InvokeMethod**

- poziv javnog postupka objekta ili javnog statičkog postupka razreda

□ **WriteLine**

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

□ **Flowchart**

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

Značajnije aktivnosti (*ControlFlow\Parallel*)

□ Parallel

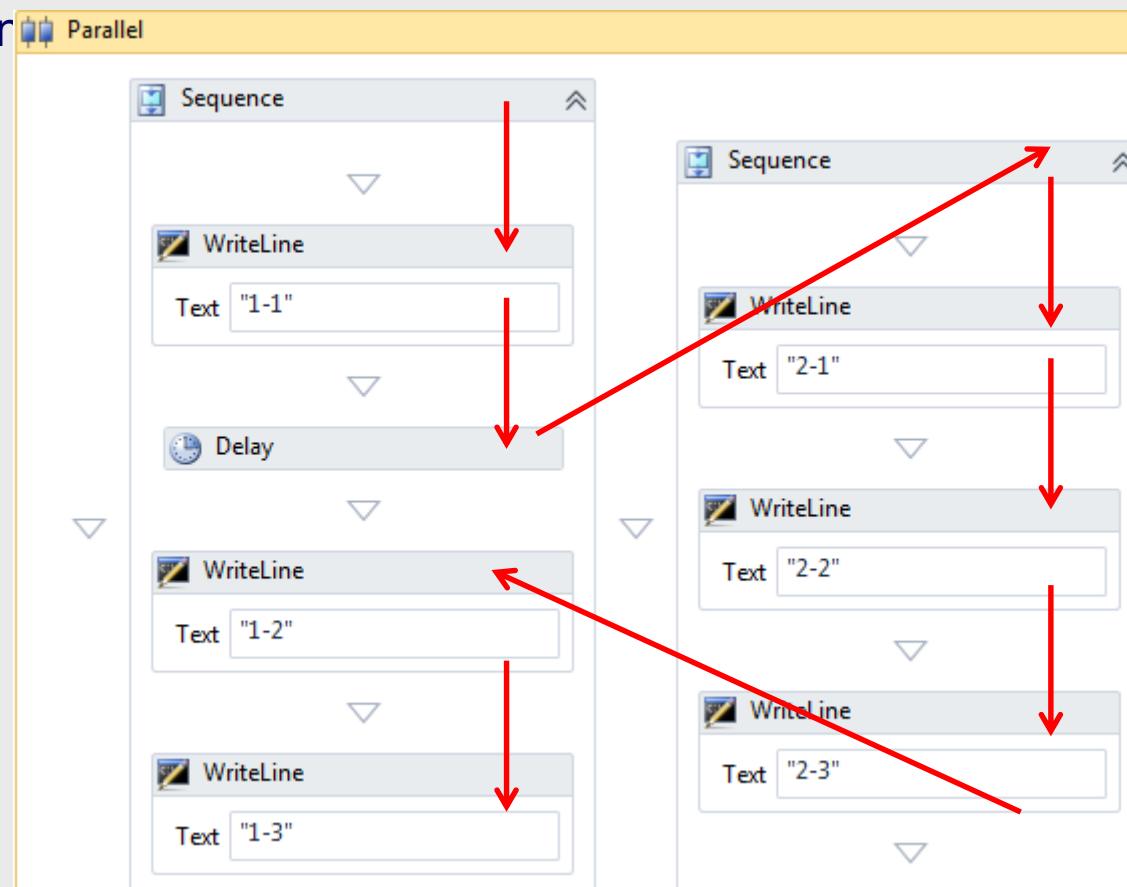
- aktivnosti koje se smiju istovremeno izvršavati

□ Pripremi se više sekvenci istovremeno za izvršavanje, 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

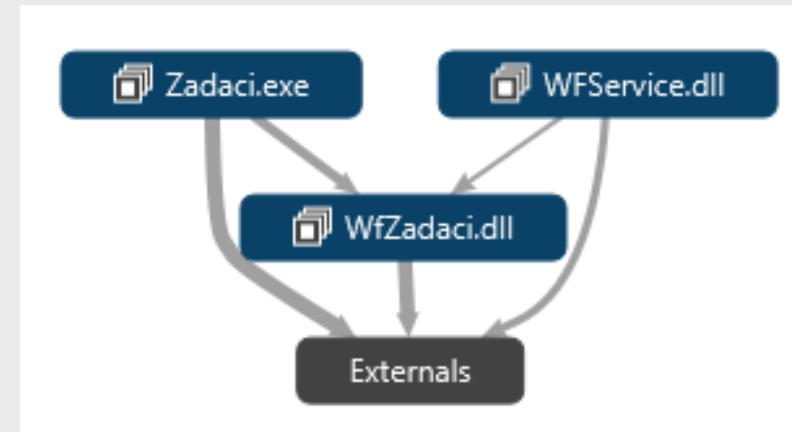
- Modeli protoka poslova i razredi za rad sa zadacima

❑ Primjer: WF \ WfService # WCF Workflow Service Application

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

❑ Tipovi datoteka

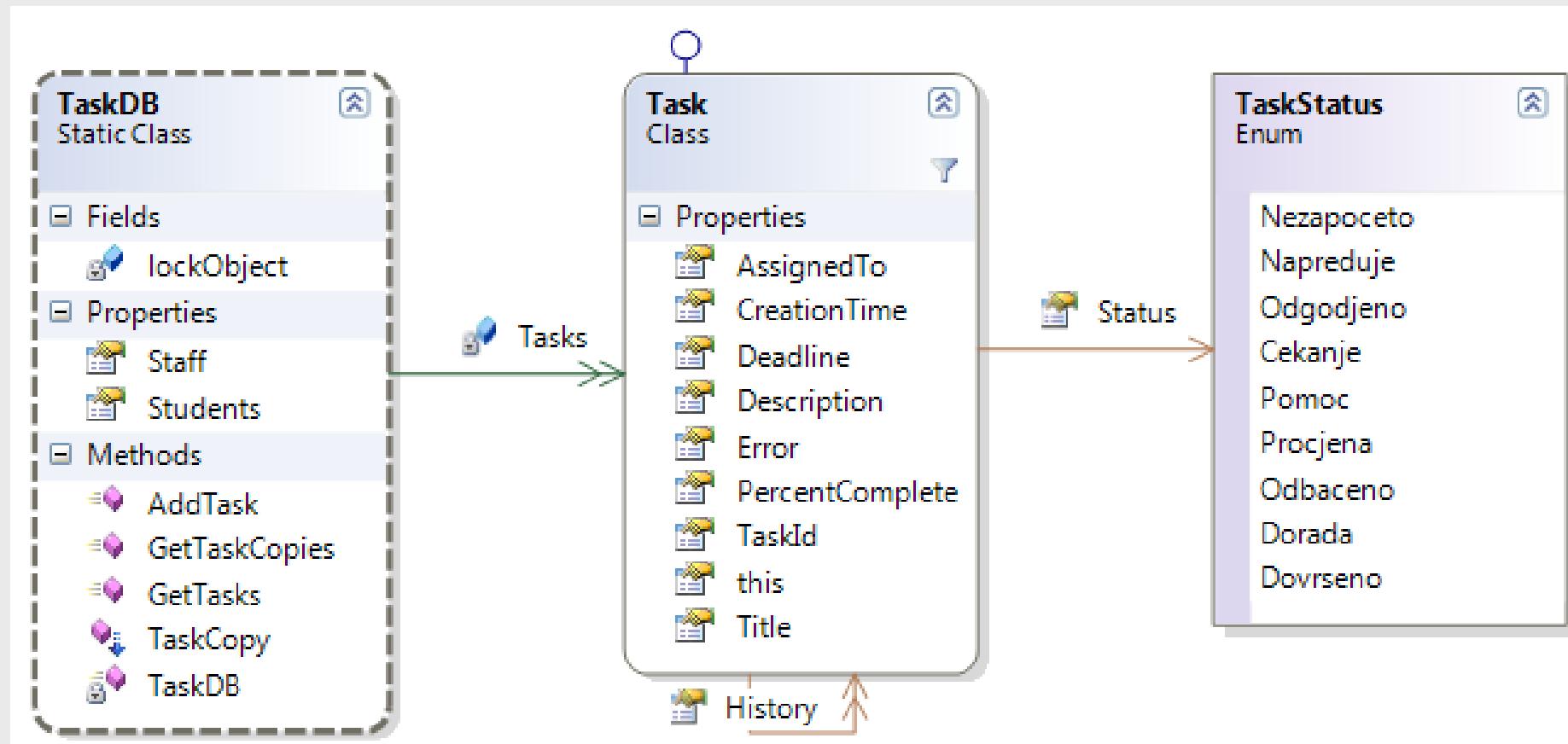
- *.XAML - activity
- *.XAMLX – workflow service



Organizacija zadatka

□ Primjer: WF \ WfZadaci

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



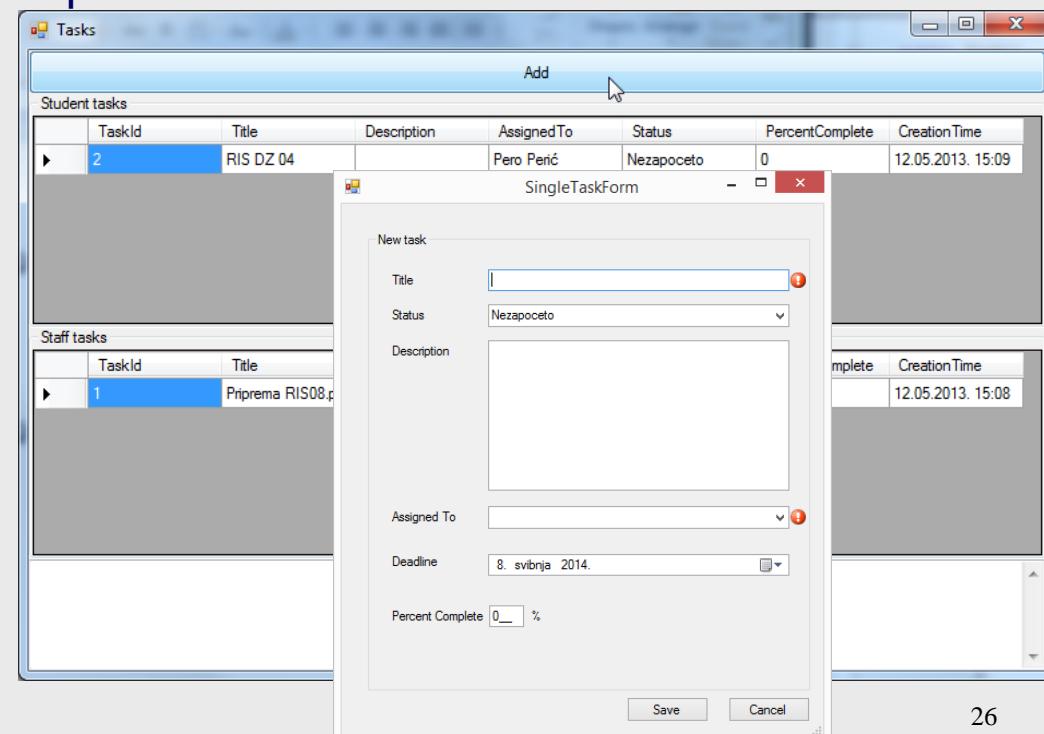
Korisničko sučelje: lista zadataka

□ Primjer: WF \ Zadaci \ Tasks

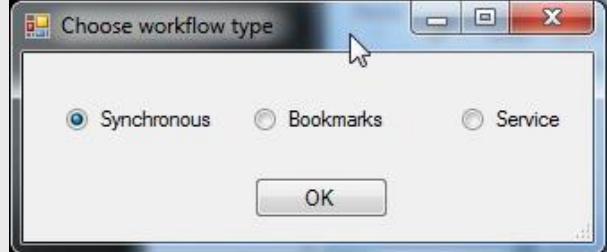
- Dvije liste zadataka (za studente i za djelatnika)
 - Dvostruki klik na ćeliju otvara formu za ažuriranje zadatka
 - Desni klik na zadatak omogućava prikaz povijesti promjena
- Tekstualni okvir za ispis poruka primljenih od instanci protoka poslova

□ Zatvaranjem forme za uređivanje zadatka,

- zadatak se šalje instanci protoka poslova
- novi nositelj i status se određuju u protoku poslova
- liste se osvježavaju
- Primjer: *Tasks.ShowEditForm*



Instanciranje protoka poslova



□ **Synchronous # RunSynchronousWorkflow**

- Pri dodavanju/izmjeni zadatka stvara se nova instanca protoka
- Sinkroni poziv - aplikacija čeka na dovršetak instance pa osvježava listu
- Ulaz – zadatak, povratna vrijednost - logička vrijednost „je ispravan” zadatak

□ **Bookmarks # RunBookmarkWorkflow**

- Asinkrono pozivanje protoka poslova
- Novi zadatak - nova instanca, promjena zadatka – aktiviranje postojeće
- Ako se zadatak nije promijenio određeno vrijeme (parametar zadan pri kreiranju nove instance) instanca protoka poslova šalje obavijest aplikaciji
- Komunikacija mehanizmom 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

- Grafički prikaz, Toolbox

- Ulazni i izlazni argumenti

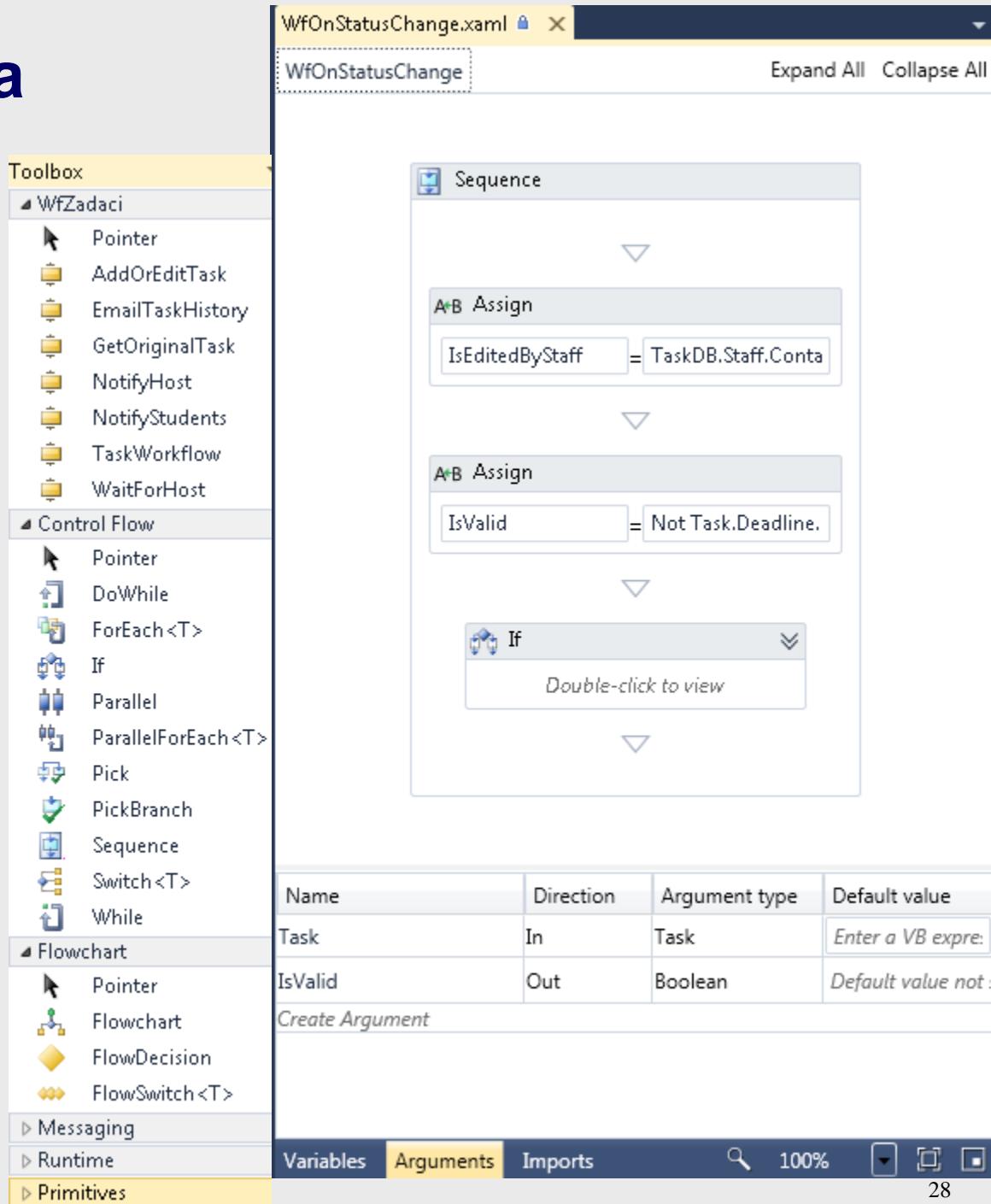
- Arguments.Create
Argument ...

- Interne varijable

- doseg – aktivnost
 - dijag: *Variables, Scope*

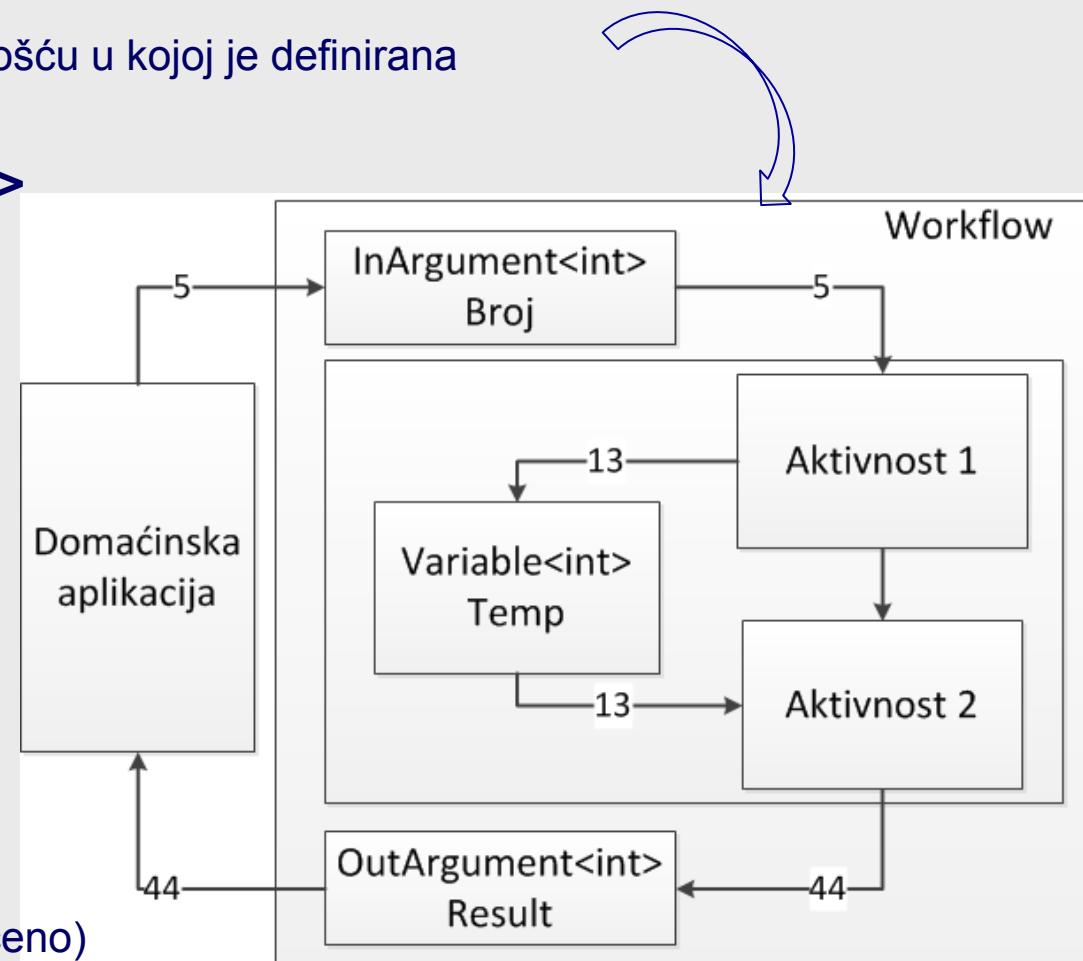
- Izrazi prijelaza i tipovi podataka u VB.Net, od VS 2012 i C#

- 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
- Ulagani 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 variabile - jedno spremište po tipu aktivnosti
 - Moraju se moći serijalizirati (osim ako drugačije nije naznačeno)



Pisanje vlastitih aktivnosti

- Nasljeđuje se jedan od apstraktnih razreda
 - *CodeActivity, AsyncCodeActivity, NativeActivity*
 - Varijante s povratnom vrijednošću **Activity<TResult>* tipa *TResult*

1. **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

2. **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)

3. NativeActivity, NativeActivity<TResult>

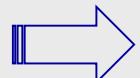
- za aktivnosti koje traju duže i koje trebaju složenije funkcionalnosti
 - pr. mehanizme koje nije moguće koristiti s prethodna dva tipa aktivnosti
- Implementira se postupak *Execute*
- Primjer korištenja na slajdu *Aktiviranje oznake*
 -  WF\ WfZadaci \ WaitForHost.cs

□ Za sva tri tipa aktivnosti prilikom izvršavanja

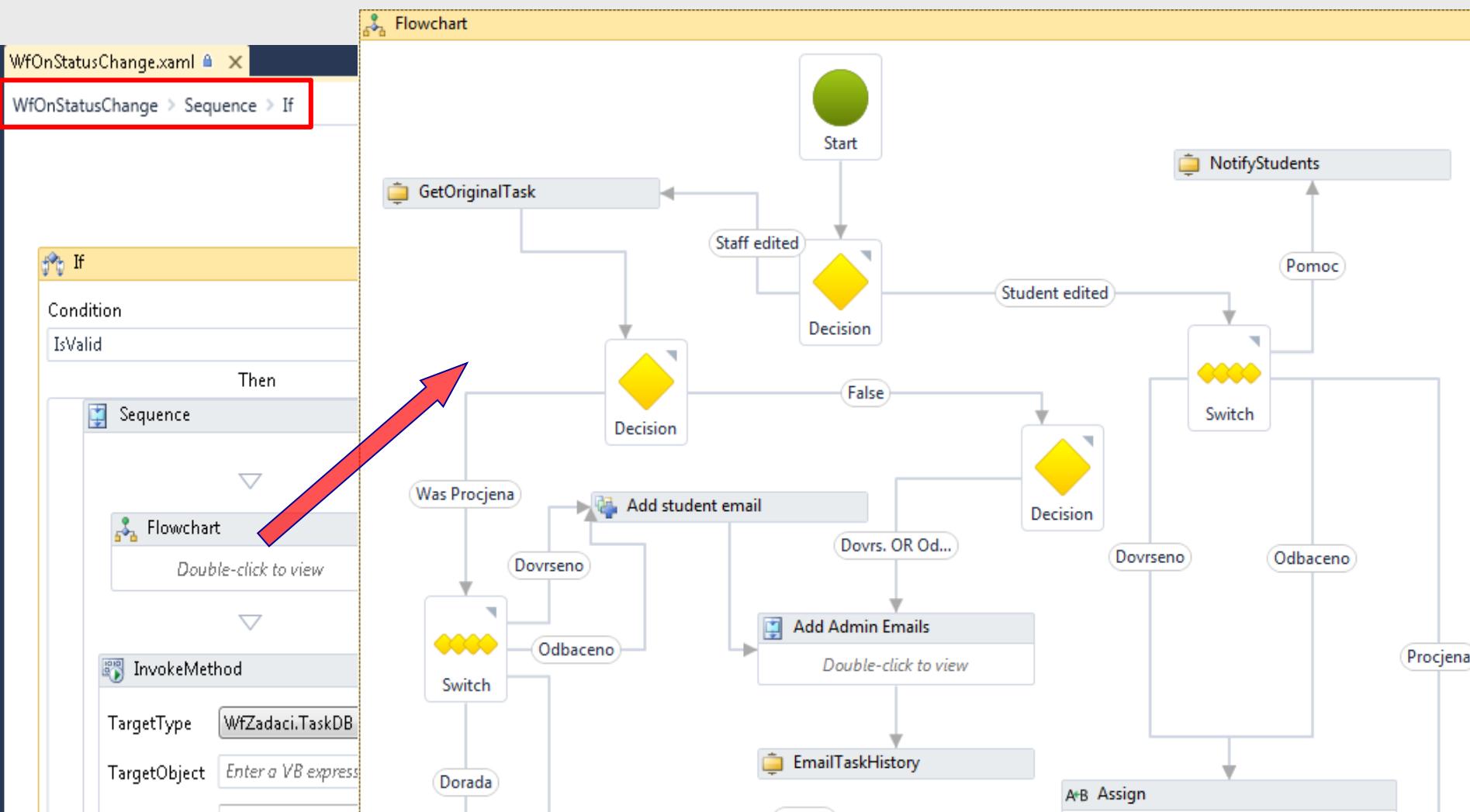
- dostupna je odgovarajuća implementacija razreda *ActivityContext*
- Omogućava dohvat ulaznih argumenata
 - NazivArgumenta.Get(context)
 - npr. *NotifyHost.cs*: *Task.Get(context);*

□ Korištenje vlastitih aktivnosti

- Toolbox pri dizajnu dijagrama protoka
- pr. *GetOriginalTask*, *NotifyStudents*



Osnovni protok poslova prilikom promjene zadatka



Protok završava kad se iz nekog stanja ne može dalje

Sinkroni poziv protoka poslova

□ Primjer: WF \ Zadaci \ Tasks.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

□ Protok završi obavljanjem *InvokeMethod* (DbTasks.AddTask)

- Pozivatelj čeka na dovršetak instance i osvježi podatke u listama

Asinhrono pozivanje (1)

□ Primjer: WF \ Zadaci \ Tasks.cs - RunBookmarkWorkflow

- protok bude instanciran pri stvaranju zadatka i izvršava se dok se zadatak ne dovrši
- Instanciramo *TaskWorkflow* koji koristi *WfOnStatusChange*
- pri tom se stvori *WorkflowApplication* i pridruži 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 \ Tasks.cs

- instanca protoka čuva se da bi se pozvala prilikom promjene 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

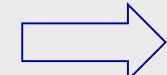
- Uparivanje zadatka i instance obavlja se naknadno primanjem identifikatora novostvorenog zadatka (*TaskId*) od instance (*OnTaskStarted*)
 - *workflows[taskId] = workflowsGuids[workflowInstanceId];*
- Konkretni primjer primanja poruke iz instance i slanja poruka instanci obrađen na slajdu *Implementacija 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* = *true* – protok može postati „besposlen”
 - 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** :

- `wfApp.ResumeBookmark(naziv oznake, objekt);`
- „nastavi od oznake koristeći zadani objekt”

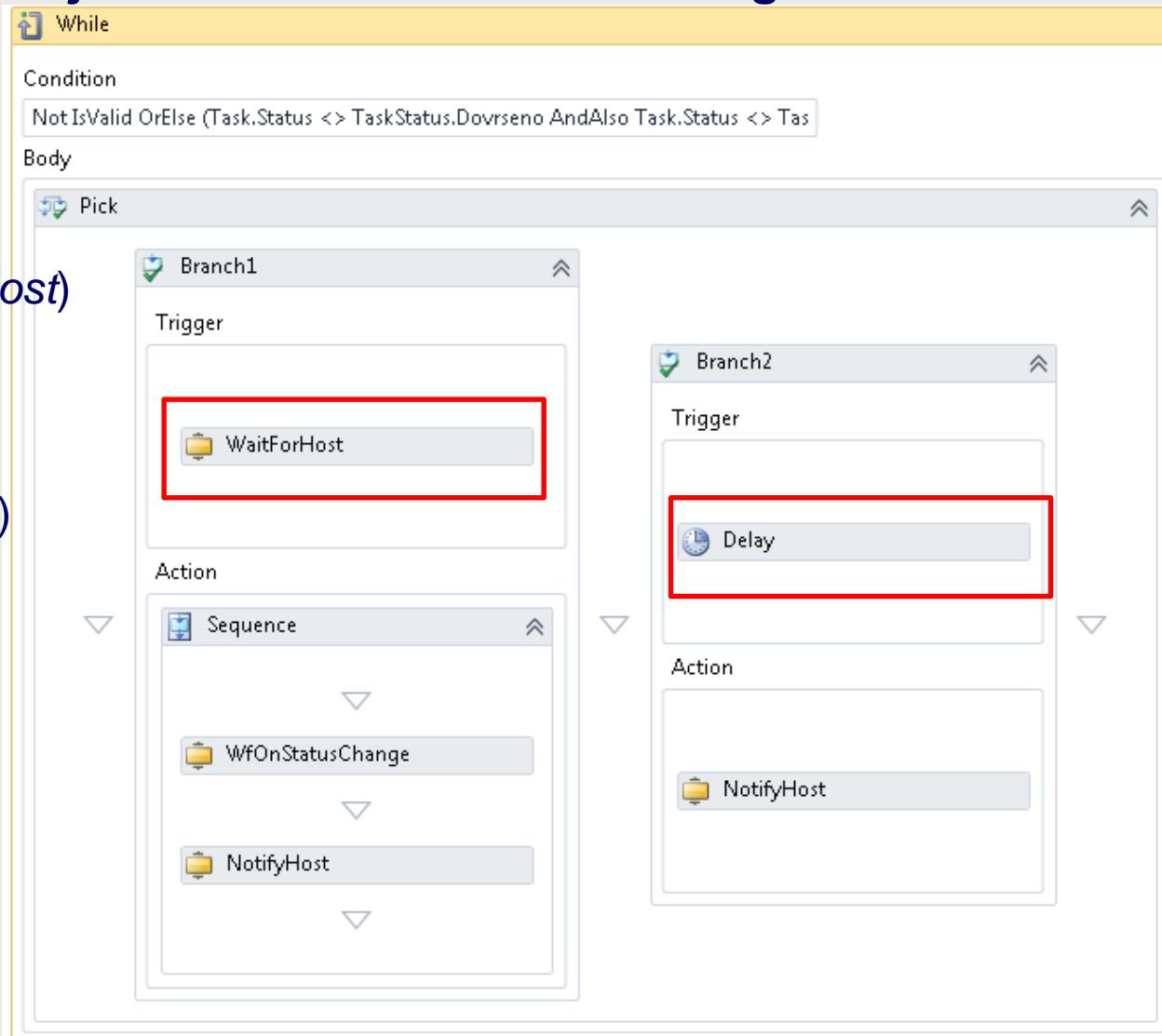
□ Primjer: WF \ Zadaci \ Tasks

- ←
- Instanca protoka prethodno je kreirana s *RunBookmarkWorkflow*, opisano u "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(„Oznaka”, 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

□ ... 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

```
...  
    public event TaskStartedDelegate TaskStarted;  
...  
    public void NotifyTaskStarted(string workflowInstanceId, int taskId) {  
        if (TaskStarted != null) {  
            ThreadPool.QueueUserWorkItem((state) =>  
                TaskStarted(workflowInstanceId, taskId), null);  
        }  
    }
```

□ Implementacija **NotifyTaskStarted** u posebnoj niti podiže **TaskStarted**

Aktiviranje proširenja

□ ... na koji se forma pretplati

- Primjer: WF \ Zadaci \ Tasks
- Pri stvaranju objekta *WorkflowApplication* definiraju se podržana proširenja

```
private void RunBookmarkWorkflow(Task t) {  
  
    var wfApp = new WorkflowApplication(workflow, inputs);  
HostEventNotifier extension = new HostEventNotifier();  
    extension.TaskStarted += OnTaskStarted;  
  
    ...  
wfApp.Extensions.Add(extension);  
    wfApp.Run();
```

□ ... i obavi osvježavanje liste te uparivanje zadatka i instance

```
void OnTaskStarted(string workflowInstanceId, int taskId) {  
  
    ...  
    workflows[taskId] = workflowsGuids[workflowInstanceId];  
    LoadData();
```

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);
        ...
    }
}
```

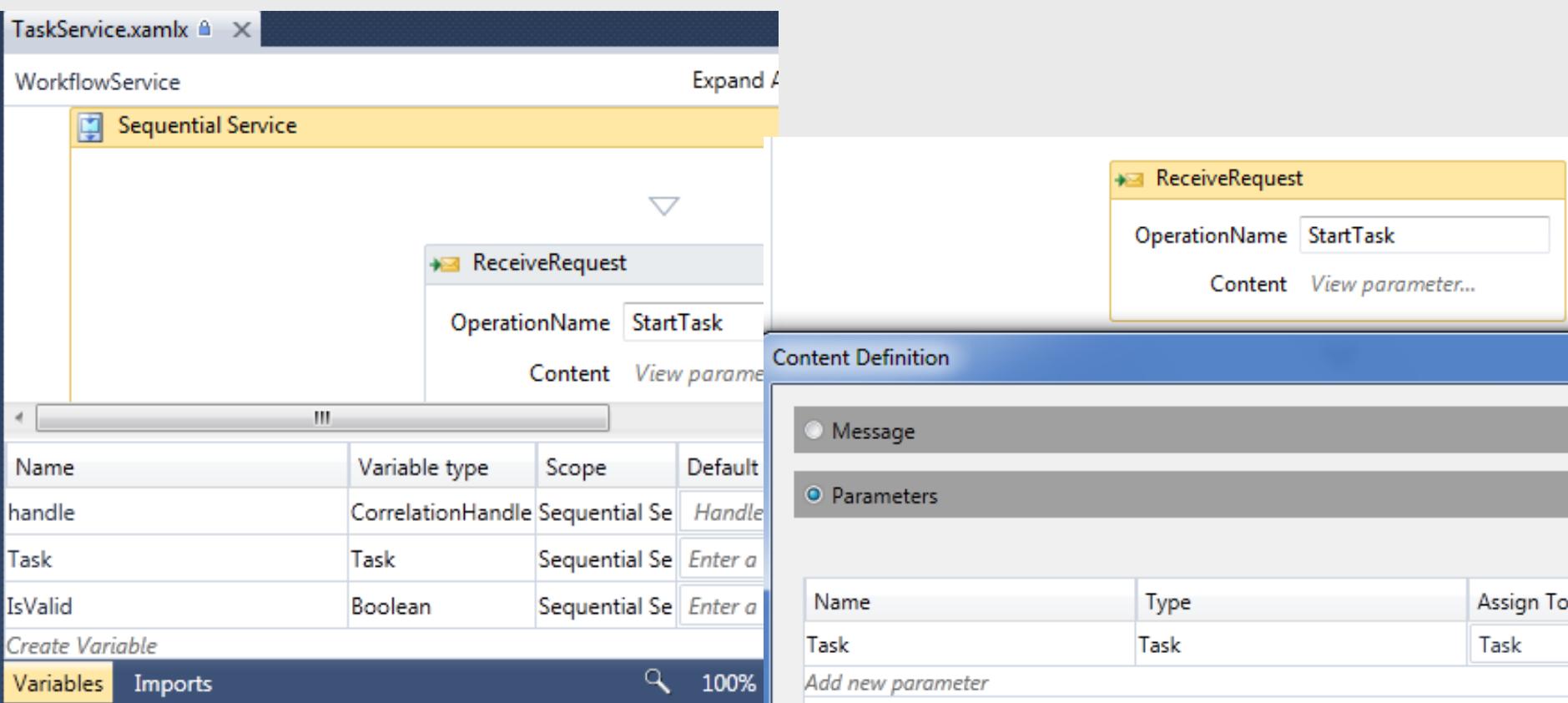
Workflow servisi

- **Primjer:**  **WF \ WFSERVICE \ TaskService.xamlx**
 - Ekstenzija xamlx – protok poslova izložen kao WCF servis
- **Prva aktivnost mora biti tipa *Receive*, npr. *ReceiveRequest***
 - 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

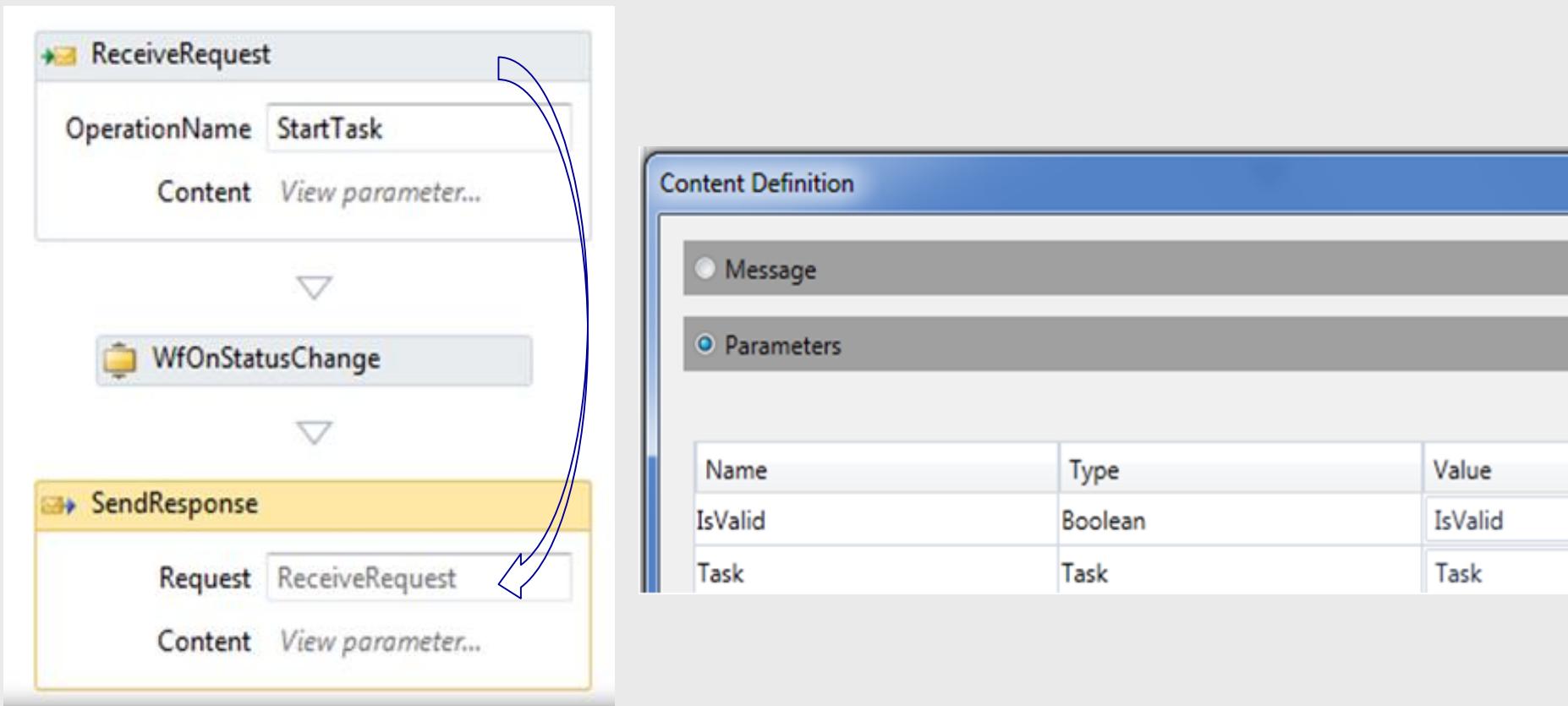
- Unutar protoka definirana varijabla *Task*
- Ulazni argument *Task* tipa *Task*, pri pozivu pridružen varijabli *Task*
 - pr. *ReceiveRequest – Content – View parameter...*



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

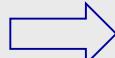
□ Primjer: WF \ Zadaci \ Tasks.cs

- Task je ulazno-izlazni argument pa postupak ima *ref* parametar
- 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

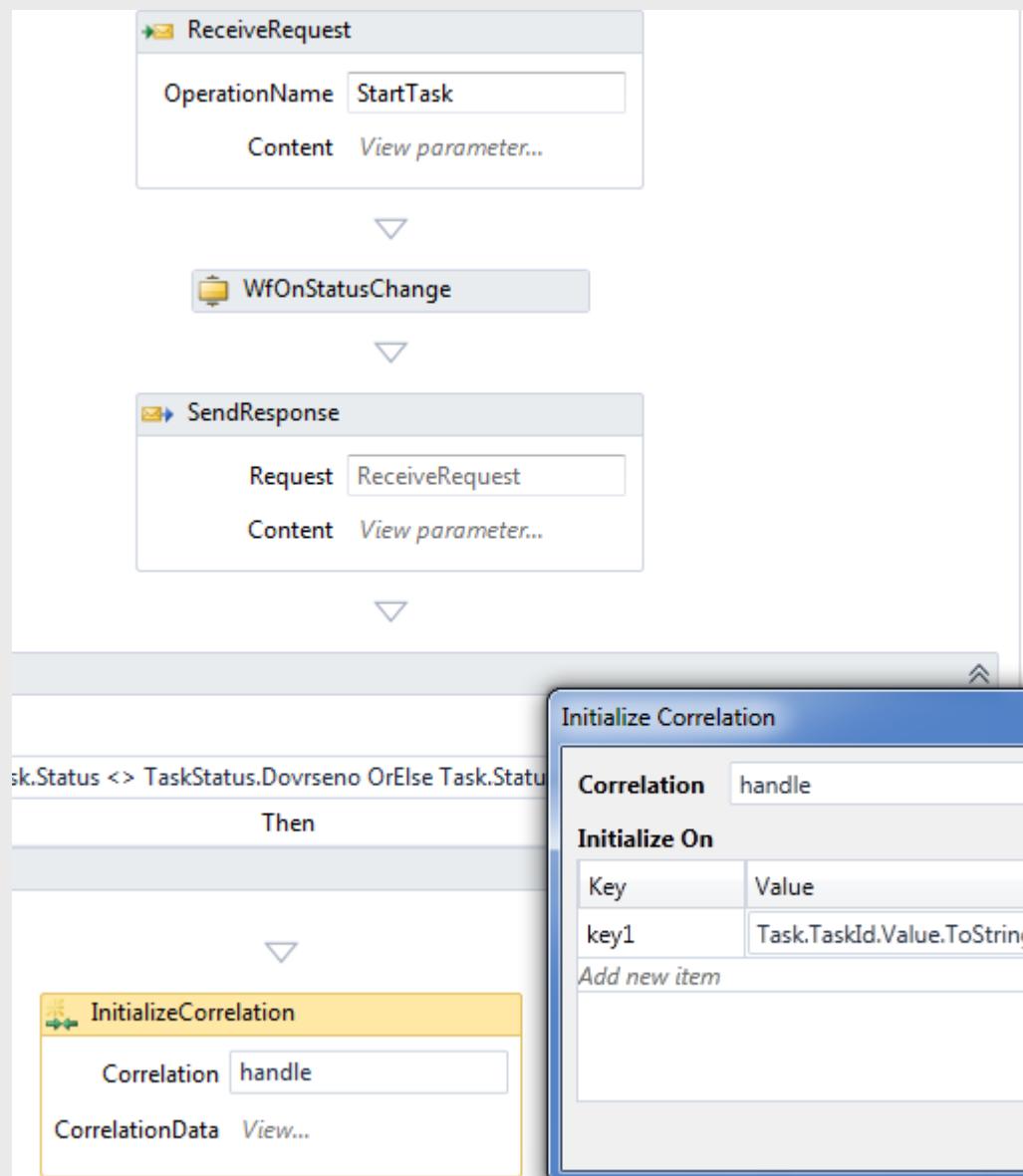
- Jedan **workflow** može sadržavati više aktivnosti **Receive**
 - Svaka od aktivnosti definira naziv postupka za koji je vezana
 - Barem jedna mora imati postavljeno *CanCreateInstance = True*
 - *Receive* aktivnosti vezane za postupke koji trebaju postojeću instancu moraju imati svojstvo *CanCreateInstance = False*
- Poziv postupka WCF servisa aktivira odgovarajuću **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 objekt novostvorenog zadatka (*Task*)



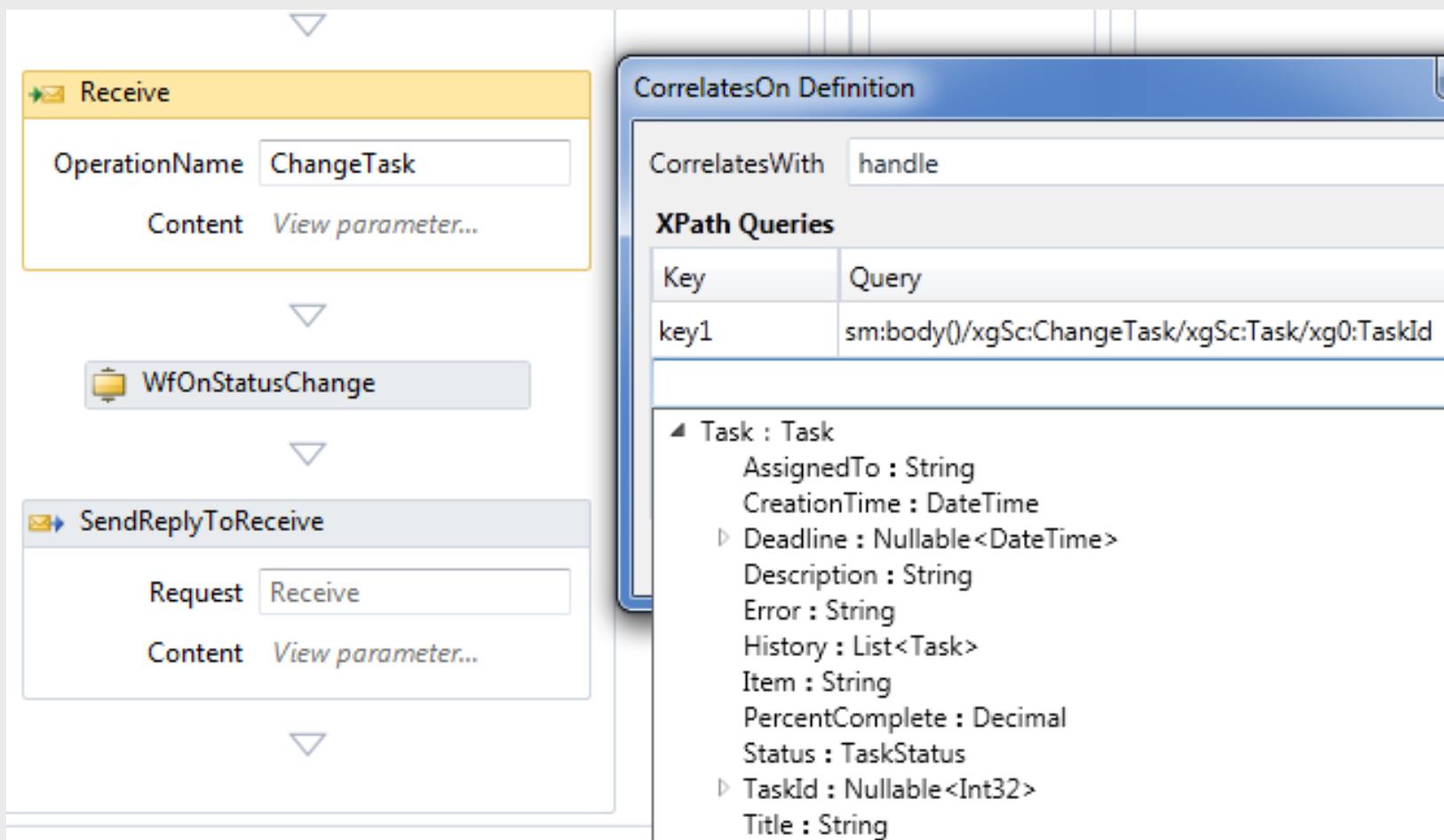
□ Slijedi inicijalizacija korelacije pomoću aktivnosti *InitializeCorrelation*

- 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* za uparivanje s 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 \ Tasks.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.
- Sharp, A.; McDermott, P. Tools for Process Improvement and Application Development, Second Edition. Boston, SAD: Artech House, 2008.

□ Windows Workflow Foundation

- <http://msdn.microsoft.com/en-us/netframework/aa663328>

□ WMC - Workflow Management Coalition

- <http://www.wfmc.org/>

□ 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/projekt.php?sif_proj=28

Izrada sustava

2013/14.09



Izrada, kodiranje, programiranje

□ 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

□ Programiranje

- proces pisanja (kodiranja), provjere (testiranja), ispravljanja pogrešaka (debugiranja) te održavanja izvornog programskega koda

□ Kodiranje (iako je sinonim za programiranje, činimo razliku)

- pretvorba detaljnog opisa programa u stvarni program, najčešće pisanje izvornog koda nekog formalnog programskega jezika
- ručno kodiranje
- automatsko kodiranje - generiranje programskega koda, sheme BP, ...

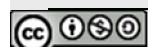
Ugovor (Contract Form)

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)

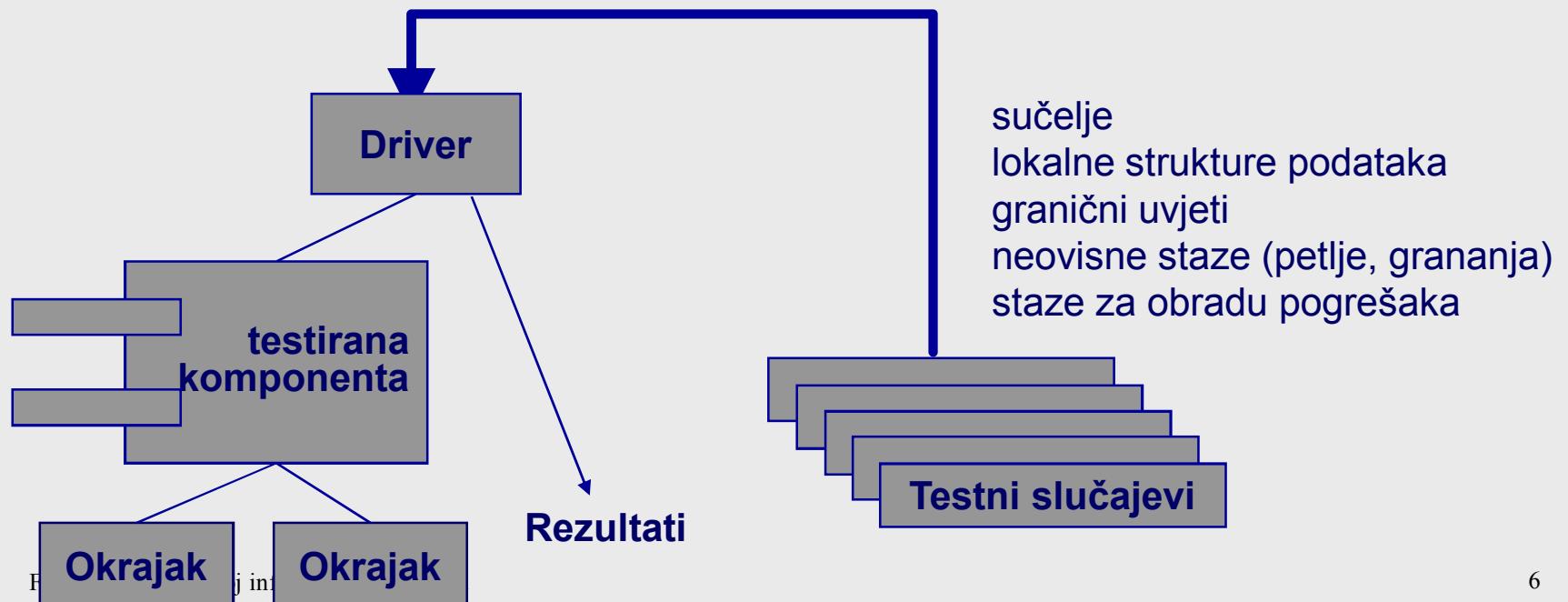
- dugotrajno *kodiranje*, a zatim niz ponavljanja oblika *provjera+ispravak*
- odgađa otkrivanje problema (pogrešaka u kodu i dizajnu)
- prosljeđuje probleme u primjeni i održavanje

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

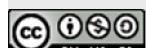
□ Što veće unutarnje prianjanje modula – kohezija (cohesion)

- potrebna visoka unutarnja povezanost elemenata
- svaki modul treba obavljati jednu i samo jednu funkciju
- postizanje ponovne upotrebljivosti u budućim programima

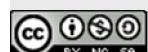
□ Što manja vanjska zavisnost modula – kopčanje (coupling)

- moduli trebaju biti minimalno međusobno zavisni
- minimizacija utjecaja promjene jednog modula na druge module

□ Iako se navedeni kriteriji pripisuju dizajnu, provode se pri izradi !



Refaktiranje



Osnovni pojmovi

□ Refaktoriranje (refactoring)

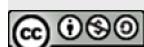
- promjena interne strukture programske podrške da bi ju se bolje razumjelo i lakše održavalo, uz očuvanje vanjskog ponašanja (Fowler 1999)
- jedna od osnovnih praksi agilnog (ekstremnog) programiranja
 - agresivno refaktoriranje - niz koraka *kodiraj+refaktoriraj*
- primjena – svi aspekti softvera (dokumentacija, kod, modeli, ...)

□ Prednosti refaktoriranja

- sprječava narušavanje strukture programskog koda
- povećanje razumljivosti i čitljivosti programskog koda
- olakšano otkrivanje bugova
- povećanje produktivnosti
- lakše i jeftinije održavanje

□ Nedostaci refaktoriranja

- pretjerana primjena smanjuje produktivnost
- neautomatizirano refaktoriranje - dugotrajno i mukotrpno
- nezreli i nepouzdani alati - 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

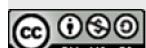
- Razred koji “radi puno toga” dijeli se na više razreda – *extract class*
- obrnuto – *inline class*

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*

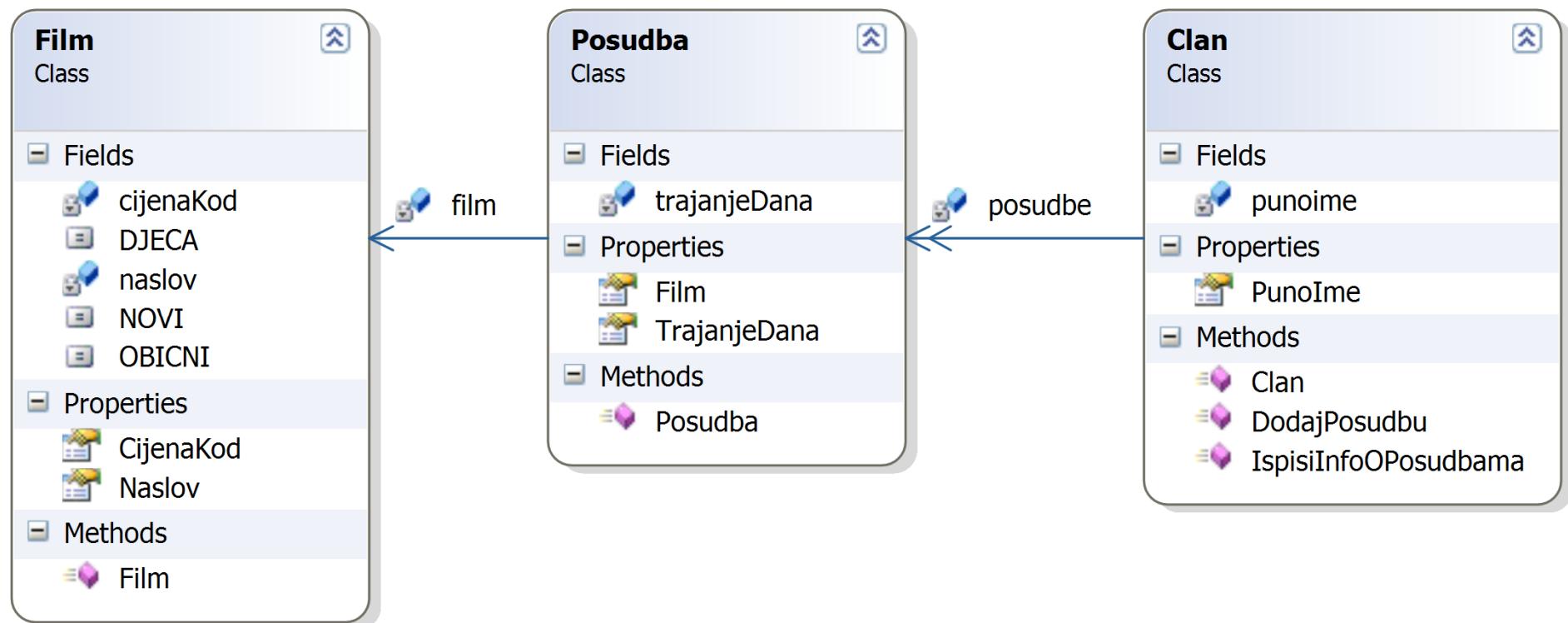
- Ispravno refaktoriranje prate odgovarajući testovi (o testiranju malo kasnije)

□ 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: 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;
            ...
        }
    }
}
```



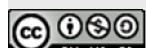
Postupak *Clan.IspisiInfoOPosudbama* (2)

```
...
    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 posuđivanja: "
    + frekvencijaPosudbiBodovi.ToString();

return ispis;
} // ispisi info o posudbama
```



Extract Method

- *IspisInfoOPosodbama* 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 – kompliciranje ionako prevelike metode
- Izolacija neutralnog, opće primjenjivog koda u novu metodu a zatim ...
 - koji bi to dio koda bio ?

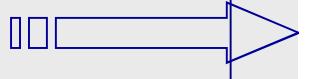
□ Ekstrakcija postupka

- 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

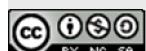
□ Primjer: Refaktoriranje \ R1

```
public string IspisiInfoOPosudbama ()  
{  
...  
  
foreach (Posudba p in this.posudbe)  
{  
    double trenutniIznos = 0;  
  
trenutniIznos = IzracunajIznosPosudbe (p) ;  
  
...  
}
```



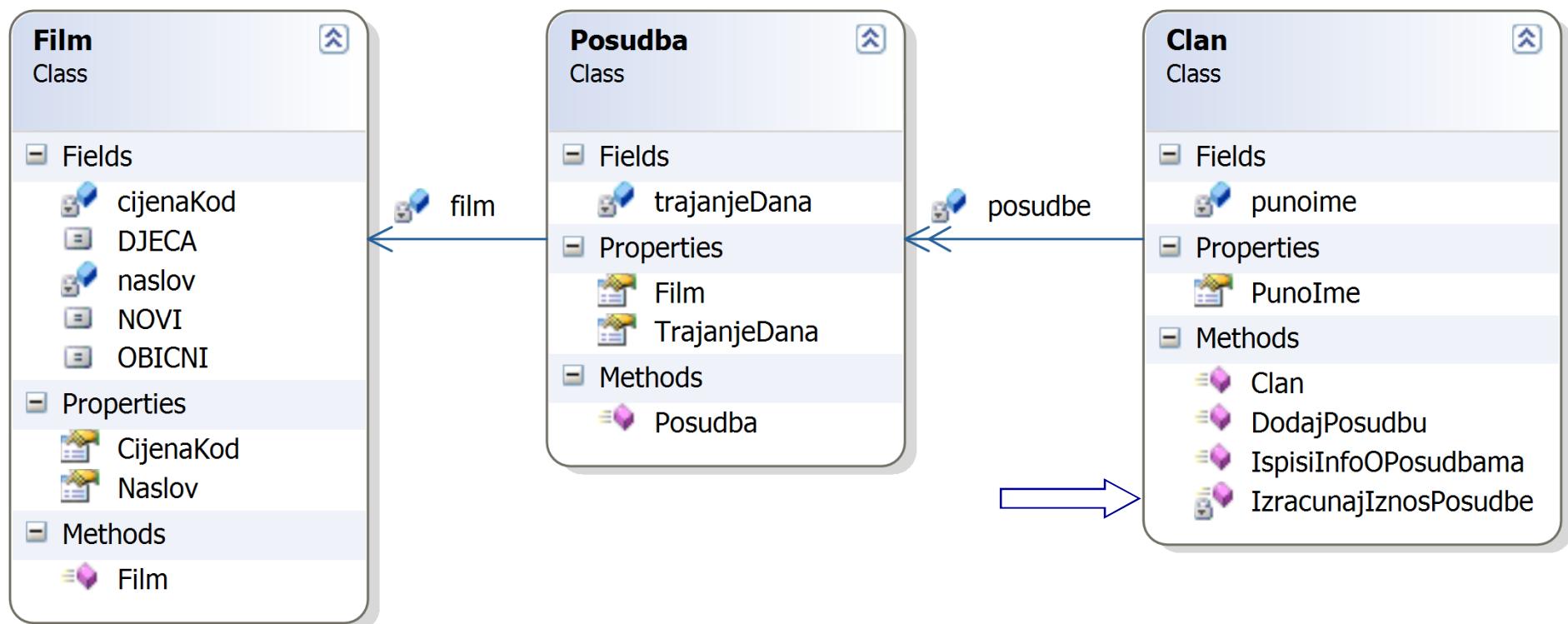
```
private double IzracunajIznosPosudbe(Posudba p)
{
    double trenutniIznos = 0;

    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

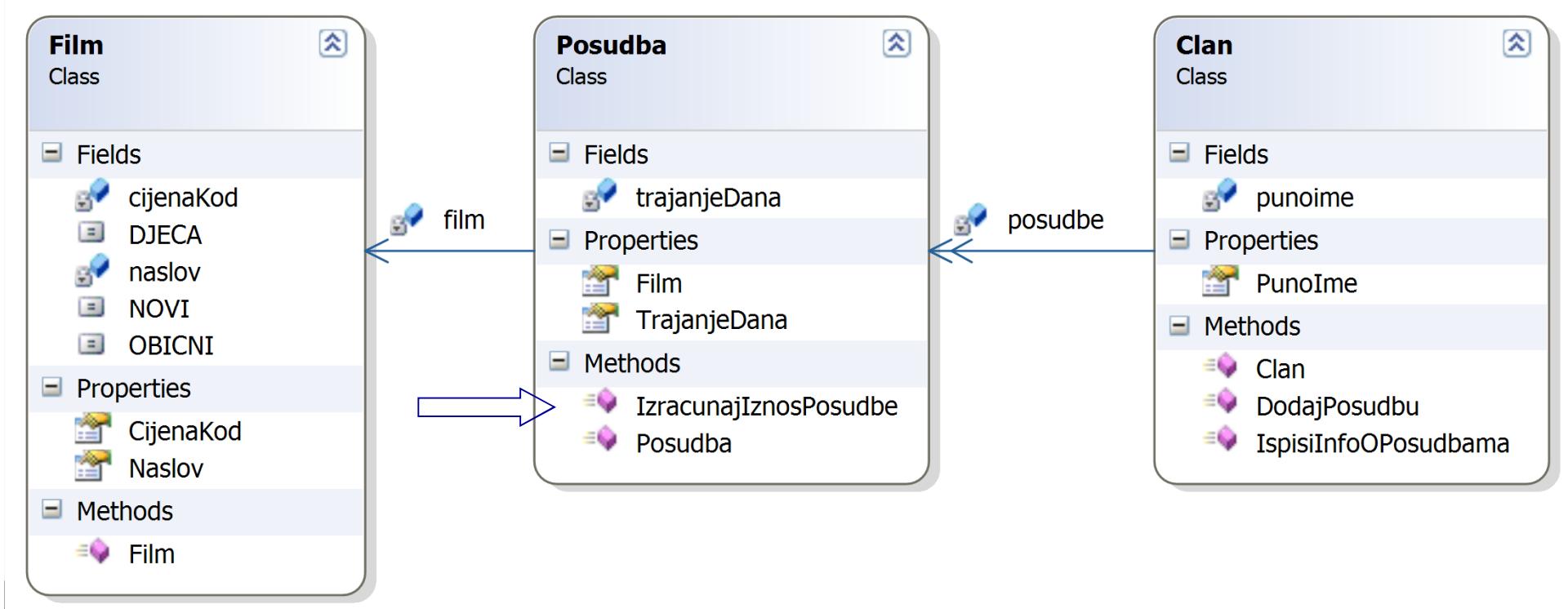
□ Primjer: Refaktoriranje \ R1



- **IzracunajIznosPosudbe** metoda klase *Clan* bavi se samo posudbama
 - Metoda se nalazi u krivom razredu te ju treba prebaciti u razred *Posudba*

Move Method

- Metoda *zracunajIznosPosudbe* 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: 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: 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: 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 potrajalao ?

- preporuča se optimizaciju odgoditi dok kod ne bude restrukturiran

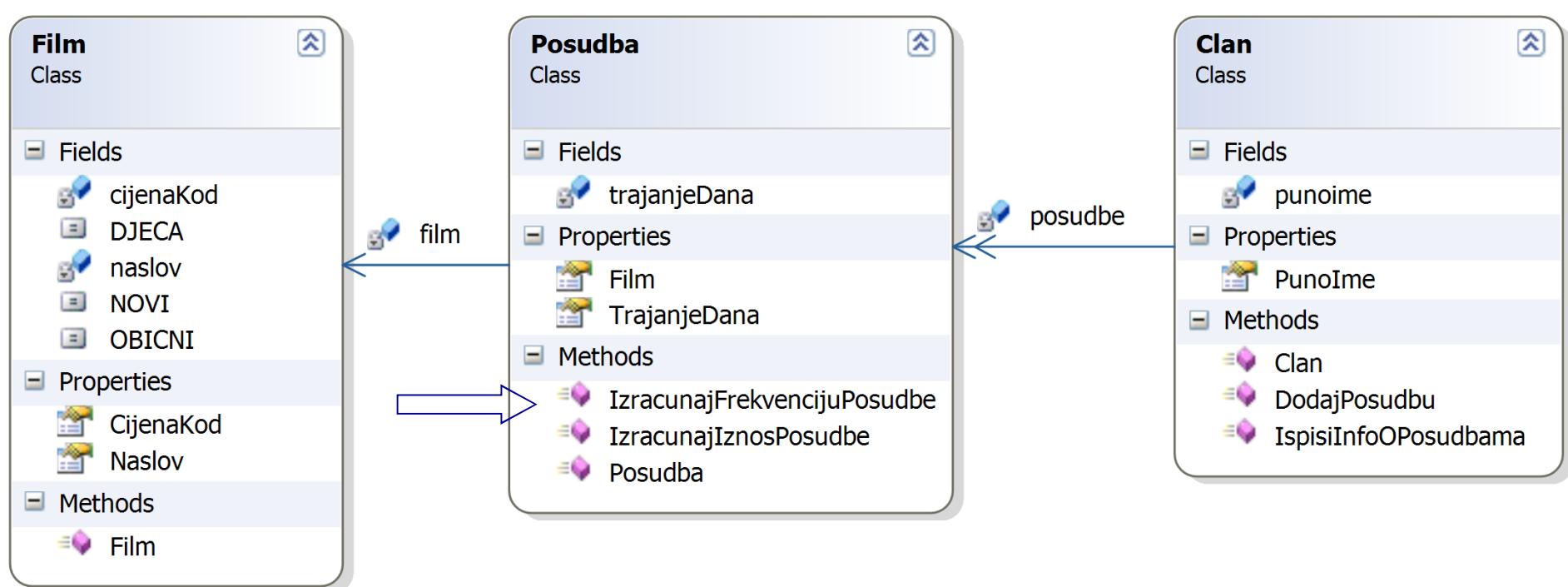


Ponavljanje *Extract Method* i *Move Method*

□ 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: Refaktoriranje \ R4



Nakon *Extract Method* i *Move Method*

- ... u metodi *IspisiInfoOPosudbama* uklanjamo varijable *ukupnilznos* i *frekvencijaPosudbiBodovi*.
 - primjenom *Replace Temp with Query*
- Primjer:  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 posuđivanja: "  
        + frekvencijaPosudbiBodovi.ToString();  
}
```



Nakon Replace Temp with Query – glavna metoda

□ Primjer: 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: 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;
}
```



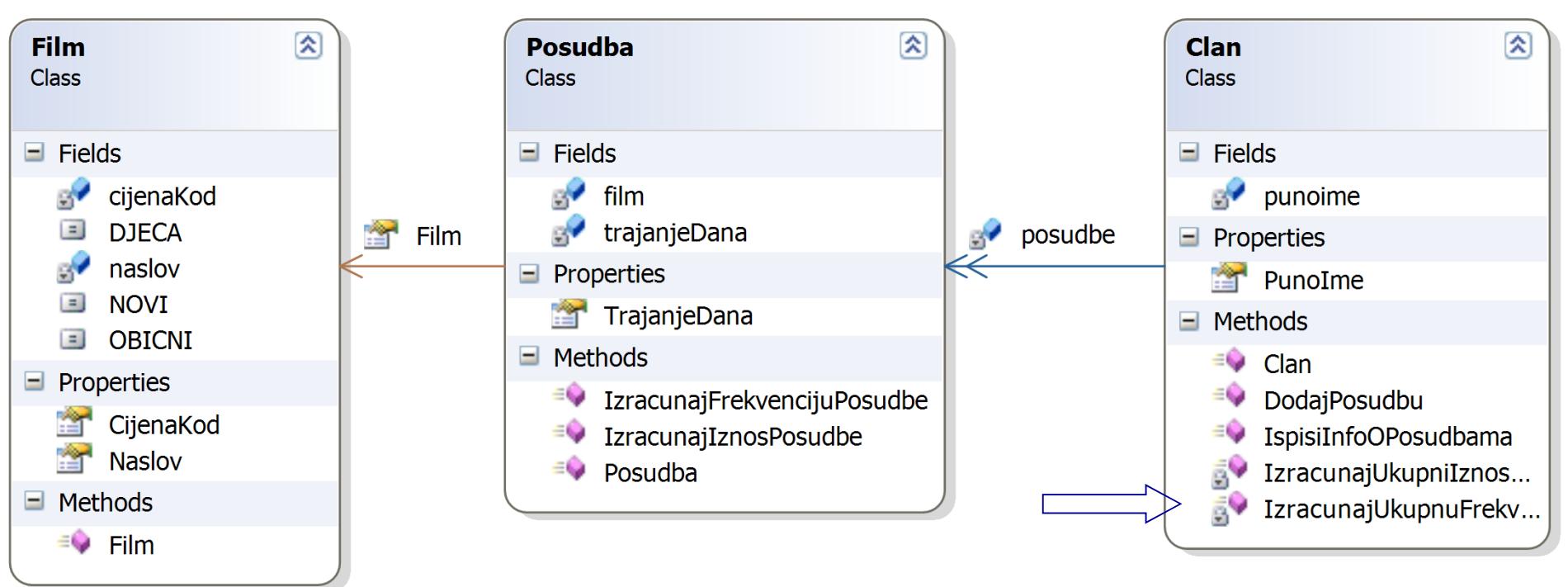
Komentar stanja

□ Primjer: 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 čemo 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:  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

Prije toga

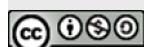
- *switch* dio u metodi *IzracunajIznosPosudbe* obavlja se na temelju atributa *CijenaKod* koji je atribut razreda *Film*

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: 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: Refaktoriranje \ R7, Razred **Posudba**

```
public double IzracunajIznosPosudbe ()  
{  
    return this.Film.IzracunajIznosPosudbe (this.TrajanjeDana);  
}
```

Primjer: Refaktoriranje \ R7, Razred **Film**

```
public double IzracunajIznosPosudbe (int trajanjeDana)  
{  
    double rezultat = 0;  
  
    switch (this.CijenaKod)  

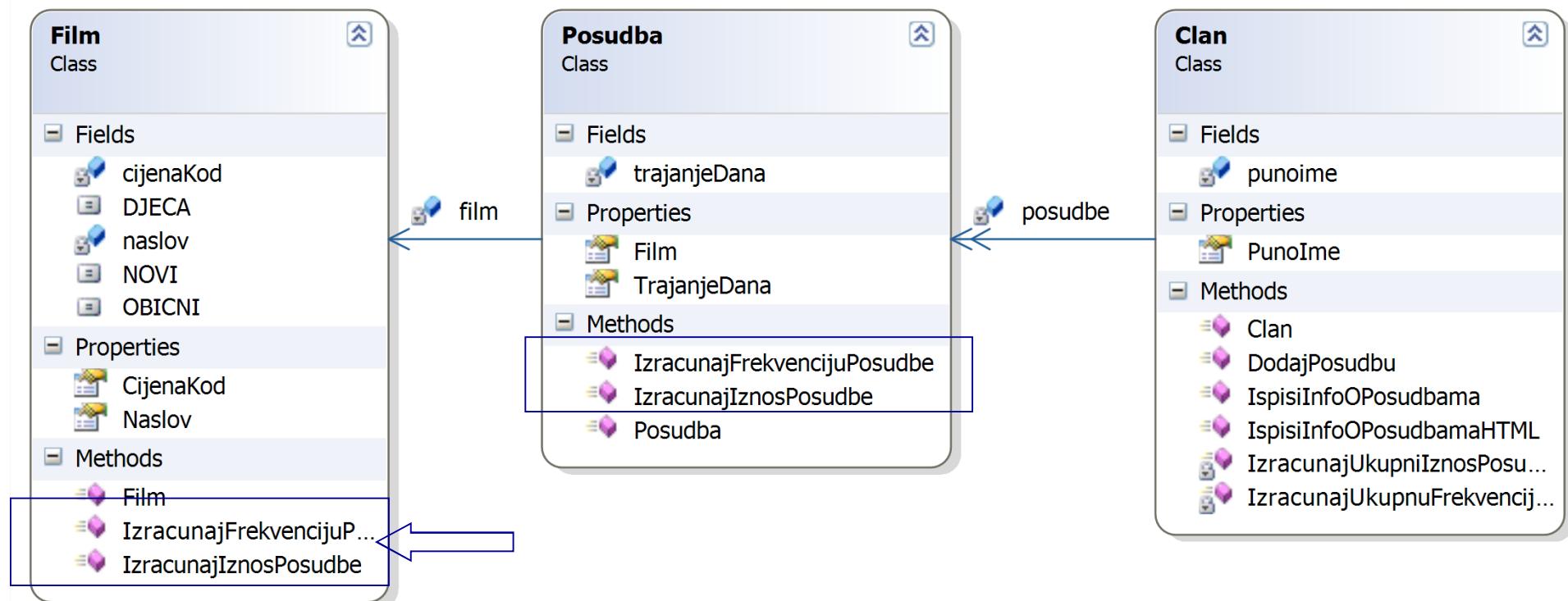
```



Prije kraja ...

□ Primjer: 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

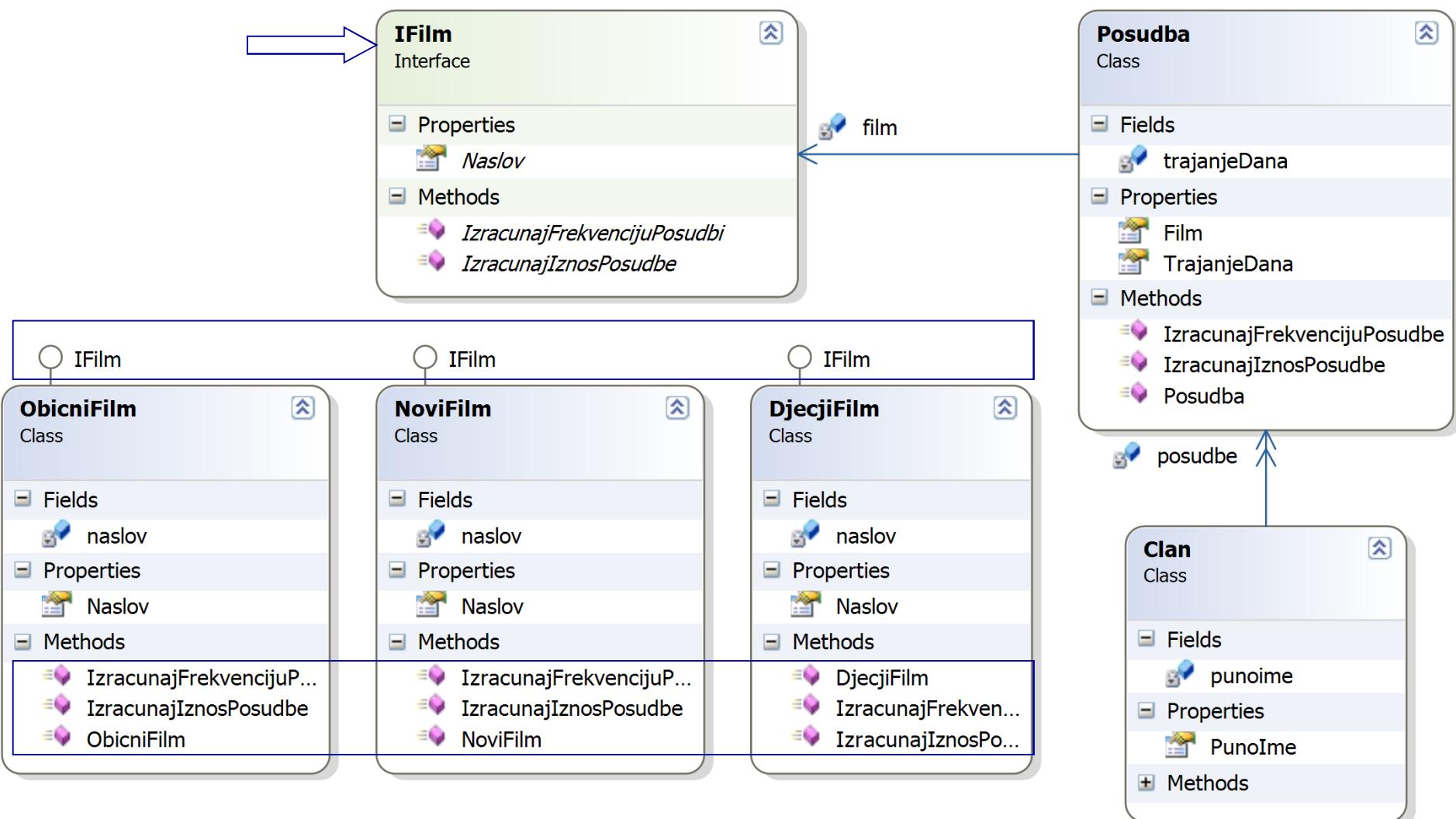
- Tada bi na svim mjestima gdje se obavlja provjera tipa cijene (u ovom kratkom primjeru samo na jednom mjestu) trebalo dodati novi case uvjet

Rješenje: nasljeđivanje 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 ugrađenim polimorfizmom

□ Primjer: Refaktoriranje \ R8



Sučelje i izvedeni razred

□ Primjer: Refaktoriranje \ R8, sučelje *IFilm*

```
interface IFilm
{
    string Naslov {
        get;
    }
    double IzracunajIznosPosudbe(int trajanjeDana);
    int IzracunajFrekvencijuPosudbi(int trajanjeDana);
```

□ Primjer: 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 primjeniti 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 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
- **nakon što se ispravi 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

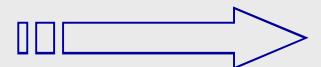


Provjera ispravnosti

□ 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 prepostavka 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
- **Ispравак (Fix) – stanje popravka**

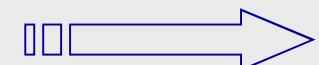
Problem testiranja objektno orijentiranih sustava

□ 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 (višeoblicje) – 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



Razvoj vođen testiranjem

□ Razvoj vođen testiranjem (Test Driven Development)

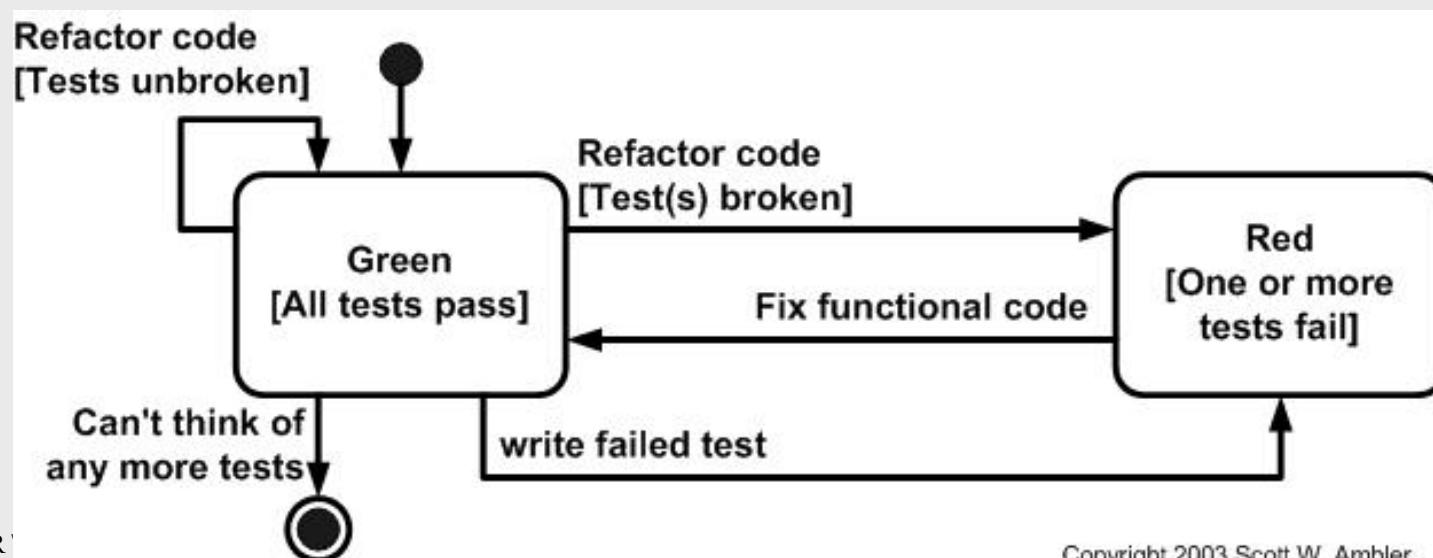
- Testovi se pišu prije koda, tj. ne kodira se nešto za što ne postoji test
- U svakoj iteraciji po obavljanju testa provodi se refaktoriranje

□ Automatizacija testiranja

- alati za automatsko testiranje - rezultate testa uspoređuju s očekivanim rezultatima
- *Unit (JUnit, nUnit, csUnit, xUnit, cppUnit, pyUnit, phpUnit, ...) – *open source*

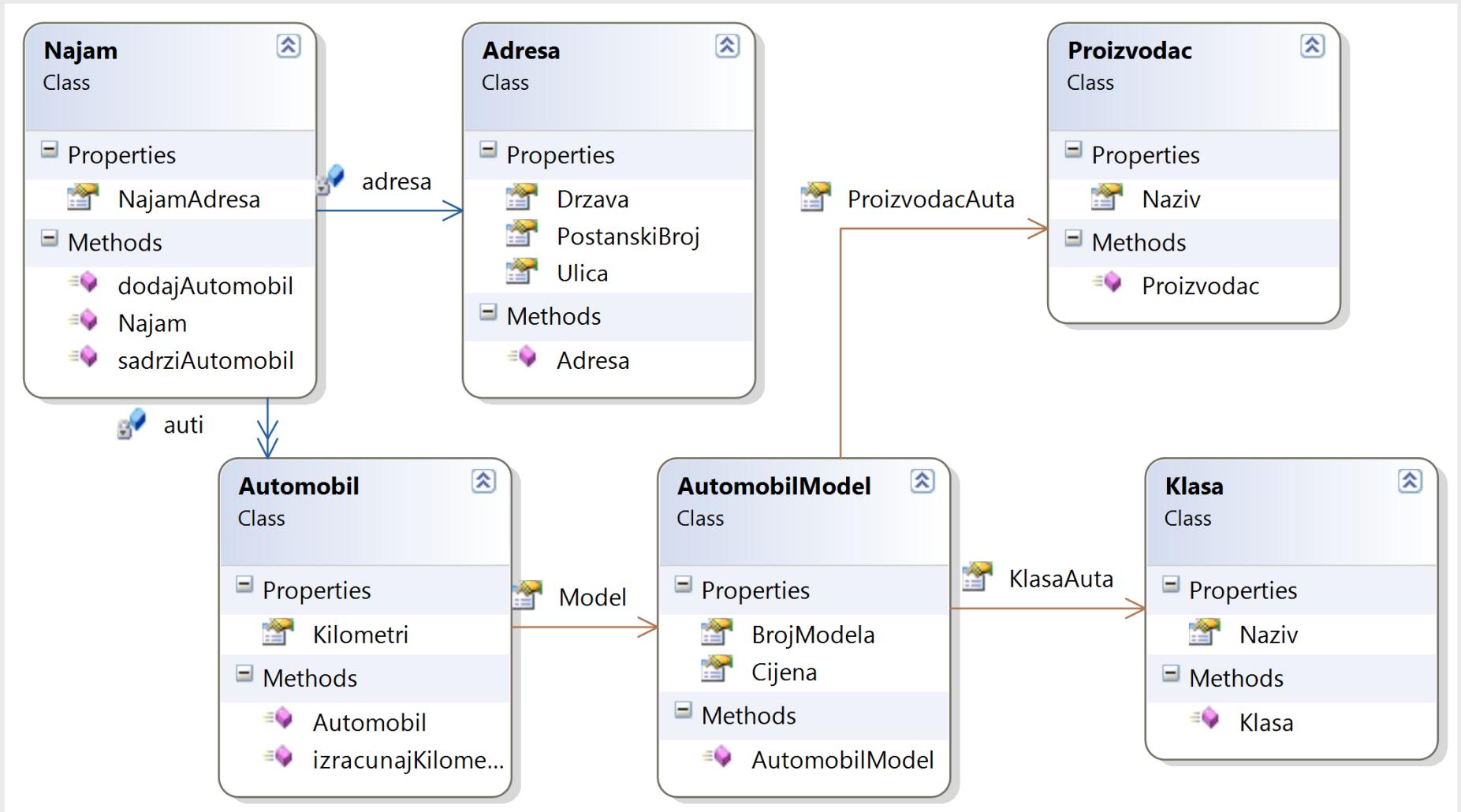
□ Regresijsko testiranje (regression testing), retesting

- provjera kojom se dokazuje da softver nije nazadovao (regressed)
- pokretanje svih testova (starih i nanovo dodanih) pri promjeni softvera



Primjer: projekt koji treba testirati

□ Primjer: Testiranje \ RentACar



Tehnika testiranja općenito

□ Napravi se projekt za test koji se komplira u DLL

- sadrži klasu testa s metodama testa, označene atributima refleksije
- u metodi testa, pokreću se metode klase `Assert` okvira za testiranje
- opcionalno, test se optimira izdvajanjem zajedničkih dijelova iz metoda

```
[klasa testa]
public class AutomobilTests {
    [inicijalizacija testa]
    public void testPostavi()
    { ... }
    [metoda testa]
    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);
    }
}
```



Testiranje u razvojnoj okolini Visual Studio

□ Solution \ Add Project tipa *Unit Test Project*

- Namespace Microsoft.VisualStudio.TestTools.UnitTesting

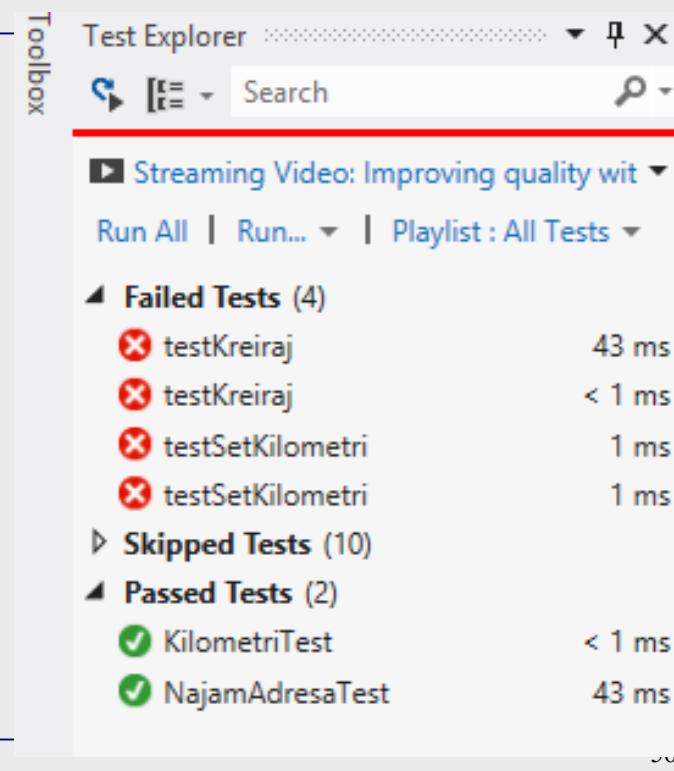
□ Test \ Windows \ Test Explorer (VS2012) ili Test List Editor (VS2010)

- Run Failed, Run Not Run, Run Passed, Repeat Last Run

□ Primjer: Testiranje \ RentACar.TestVS

- Također pogledati #region Additional

```
[TestClass()]
public class AutomobilTest
{
    ...
    [TestInitialize()]
    public void MyTestInitialize()
    {
        ...
    }
    [TestMethod()]
    public void KilometriTest()
    {
        ...
        Assert...
    }
}
```



Vanjski alati za jedinično testiranje

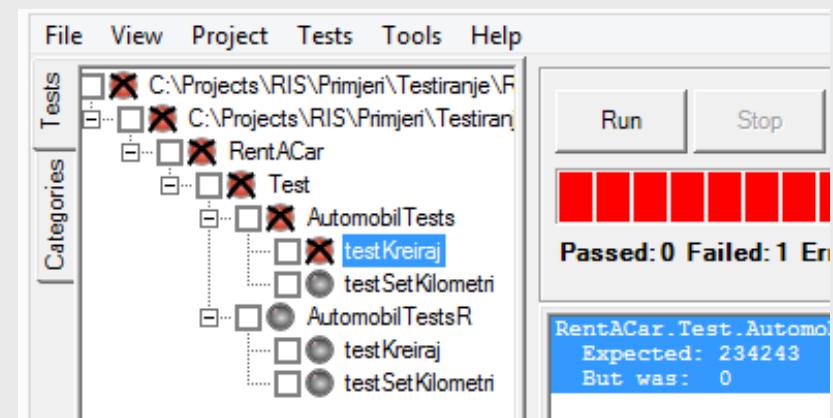
- <http://junit.org> – Beck, Gamma
- <http://www.nunit.org/> - Beck, prvotno po uzoru na jUnit
- <http://www.csunit.org/> Agile Utilities NZ, zapušten (zadnja objava 2009)
- <https://github.com/xunit/xunit> – „the original inventor of nUnit”
- ...

□ Kreiranje testa

- dodavanje projekta tipa „Class Library” u Solution
- dodavanje referenci u projekt za test na
 - projekt koji treba testirati
 - DLL za testiranje (npr. Nunit.Framework)
- stvaranje jedne ili više klase za testiranje
- kompilacija projekta za testiranje

□ GUI alata

- New Projekt
- Add Assembly (testa) + Run ...



Primjer: projekt testa vanjskim alatom

□ Primjer: Testiranje \ RentACar.TestNU

- AutomobilTest.cs - originalni test
- AutomobilTestR.cs – restrukturirani (optimirani) test
- Opcionalno, [TearDown] označi metodu koja se izvodi nakon svakog testa

```
[TestFixture]
```

```
public class AutomobilTestsR
{
```

```
[SetUp]
```

```
public void SetUp()
{ ... }
```

```
[Test]
```

```
public void testKreiraj()
{
    ...
    Assert...
```

Integracija vanjskih alata u okolinu VS

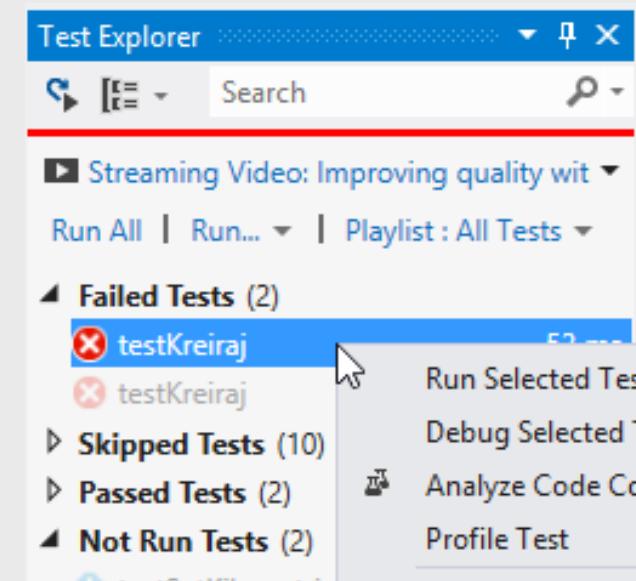
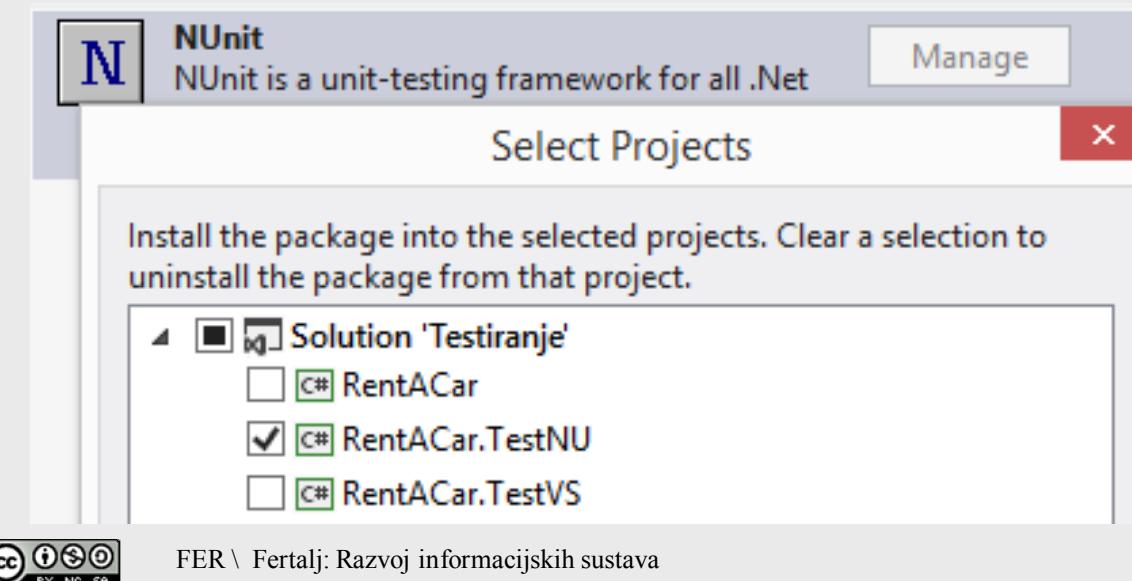
□ Tools \ Extensions and Updates ...

- online – search – install
- pr. NUnit Test Adapter for Visual Studio
- Integracija testa u razvojnu okolinu
- kompilacijom VS prepozna 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



Microsoft Fake Framework

□ Fake – krivotvorina

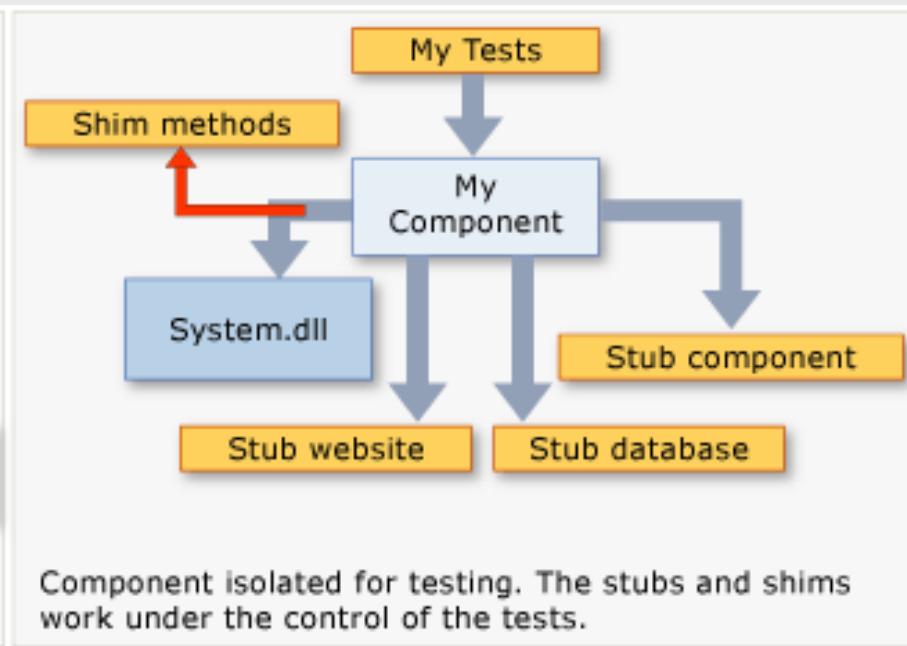
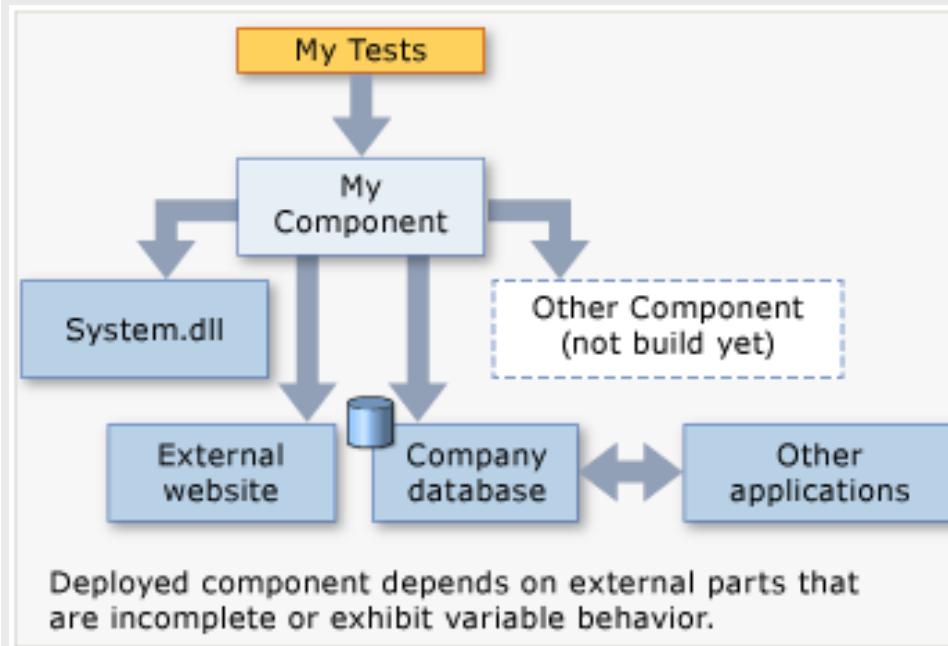
- omogućuje izolaciju koda (testa) zamjenom dijelova koda

□ 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

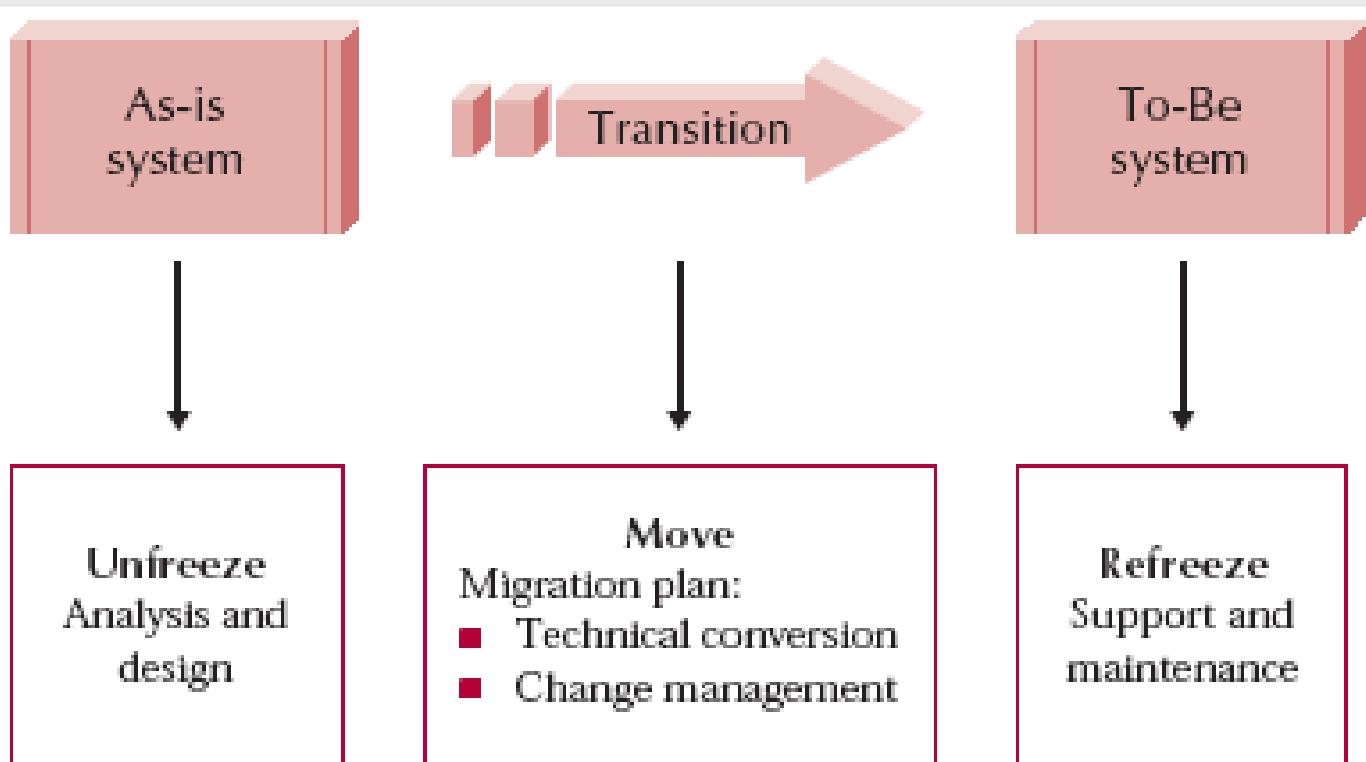


Primjena



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

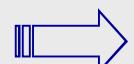
- tehnički proces u kojem novi sustav zamjenjuje stari sustav

□ 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, 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

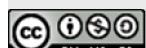
□ Postupno uvođenje, fazno uvođenje (phased conversion)

- 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

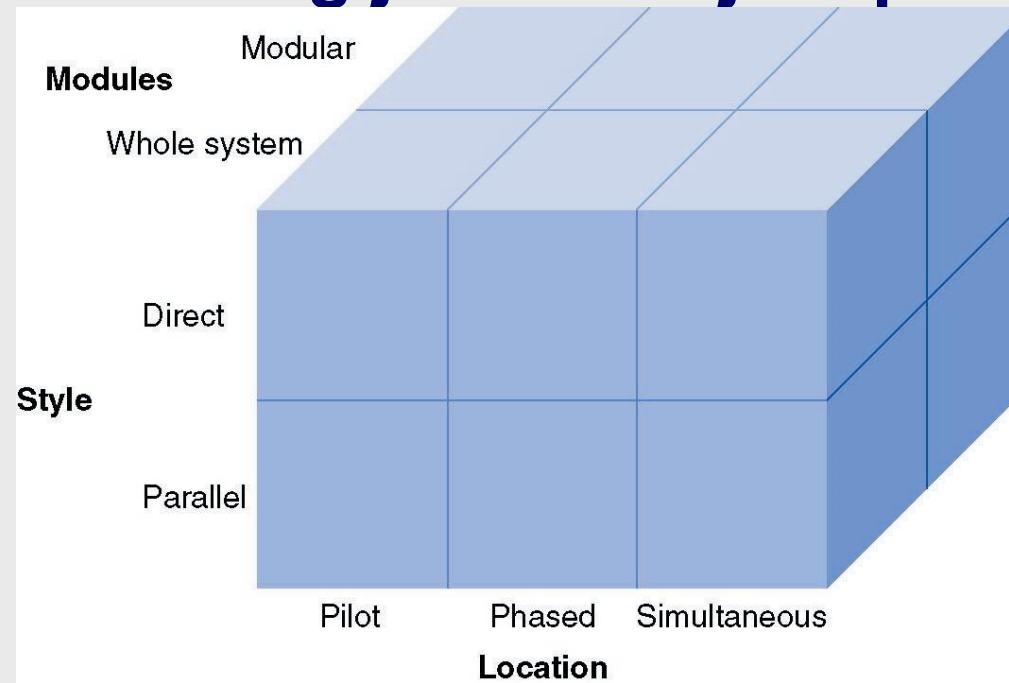
□ Modularno uvođenje (modular conversion, staged conversion)

- 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)

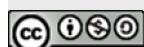
- podrška procesa prilagodbe korisnika na novi sustav

□ 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 prihvatiti sustav (u)koliko ih se osposobi

□ Sadržaj poduke

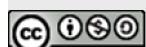
- kako obaviti svoj posao (a ne kako koristiti aplikacije) !
- što korisnik treba/može napraviti (a ne što sustav može)

□ 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

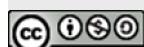
□ Postupci i tehnike poduke

- 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
- pomoć korisnicima korištenju sustava (on-demand training)
- pomoć 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 proslijeđuje 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 proslijeđuje 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

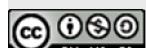
- Bez obzira na to koliko je sustav dobar, kvarovi su neizbjegni!

□ 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, mjesечно)

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)

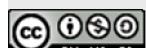
Perfektivno

- nadgradnja sustava da bi se riješili novi problemi
- ugradnja novih mogućnosti (features)



Materijali

- <http://www.refactoring.com>
- Dodatak – „bad code smells”
 - Dodatak:  \ Dodaci\ RIS09-dodatak.pdf
- Prilog - tablica usporedbe atributa i asserta
 - Prilog:  \ Izrada \ NUnit 2.6 vs MSTest 2010_2012 vs xUnit.net
- Izvorni kod (preuzeto s nUnit)
 - Primjer:  \ Testiranje \ NUnit csharp
 - Dodatno, postoji i primjer kako Nunit testira sam sebe
- 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/>



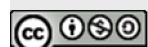
Reference

□ 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



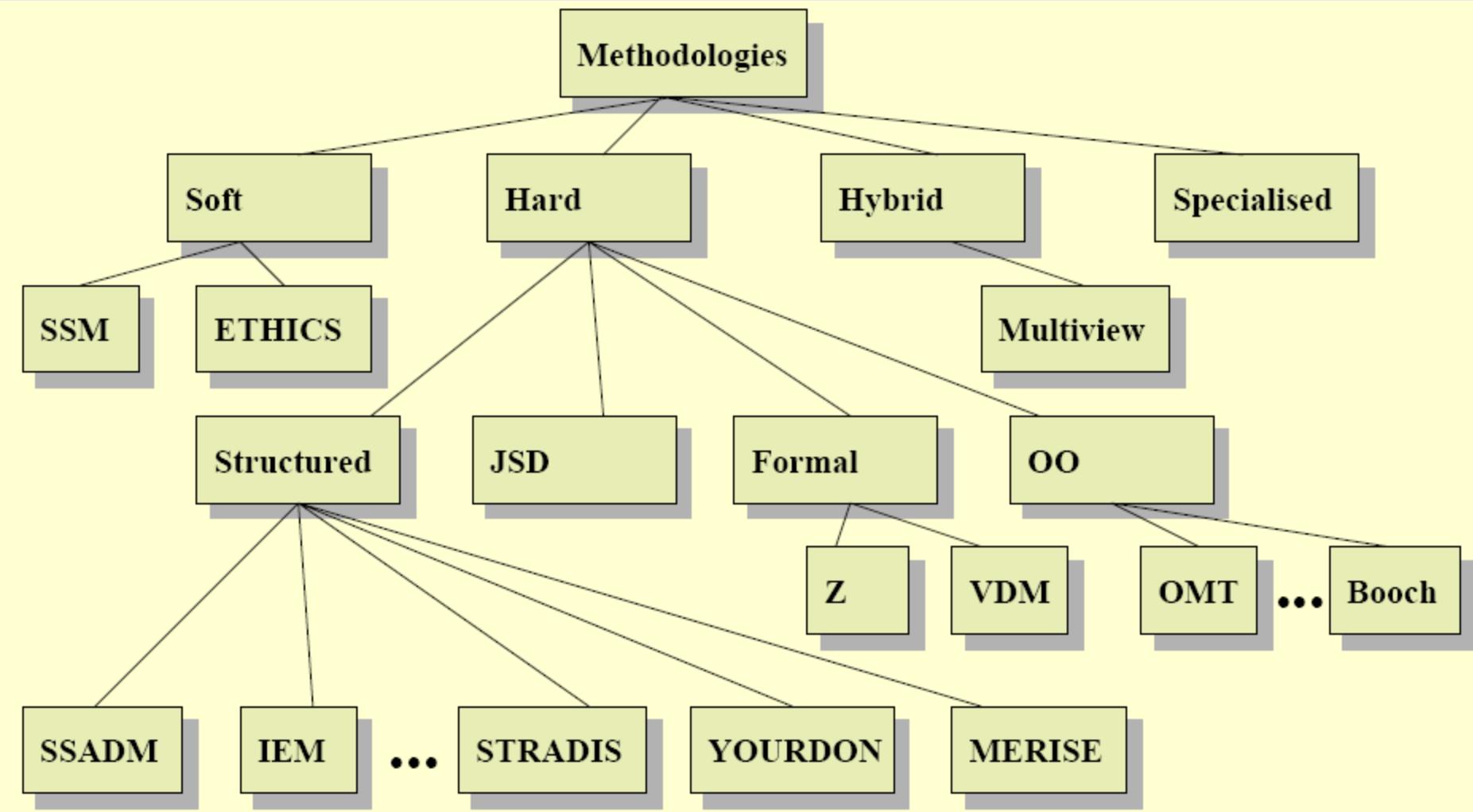
Metodologije razvoja

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 sociološkim 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) – pr. 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, naročito kada alati podržavaju više 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

□ Definiranje vlastite

- Prilagodbom postojeće i/ili kombiniranjem dijelova drugih

Strukturirane metodologije

SSADM

□ Structured Systems Analysis and Design Method

- najpoznatija i najčešće korištena
- podatkovno-vodjena ("data-driven") metodologija

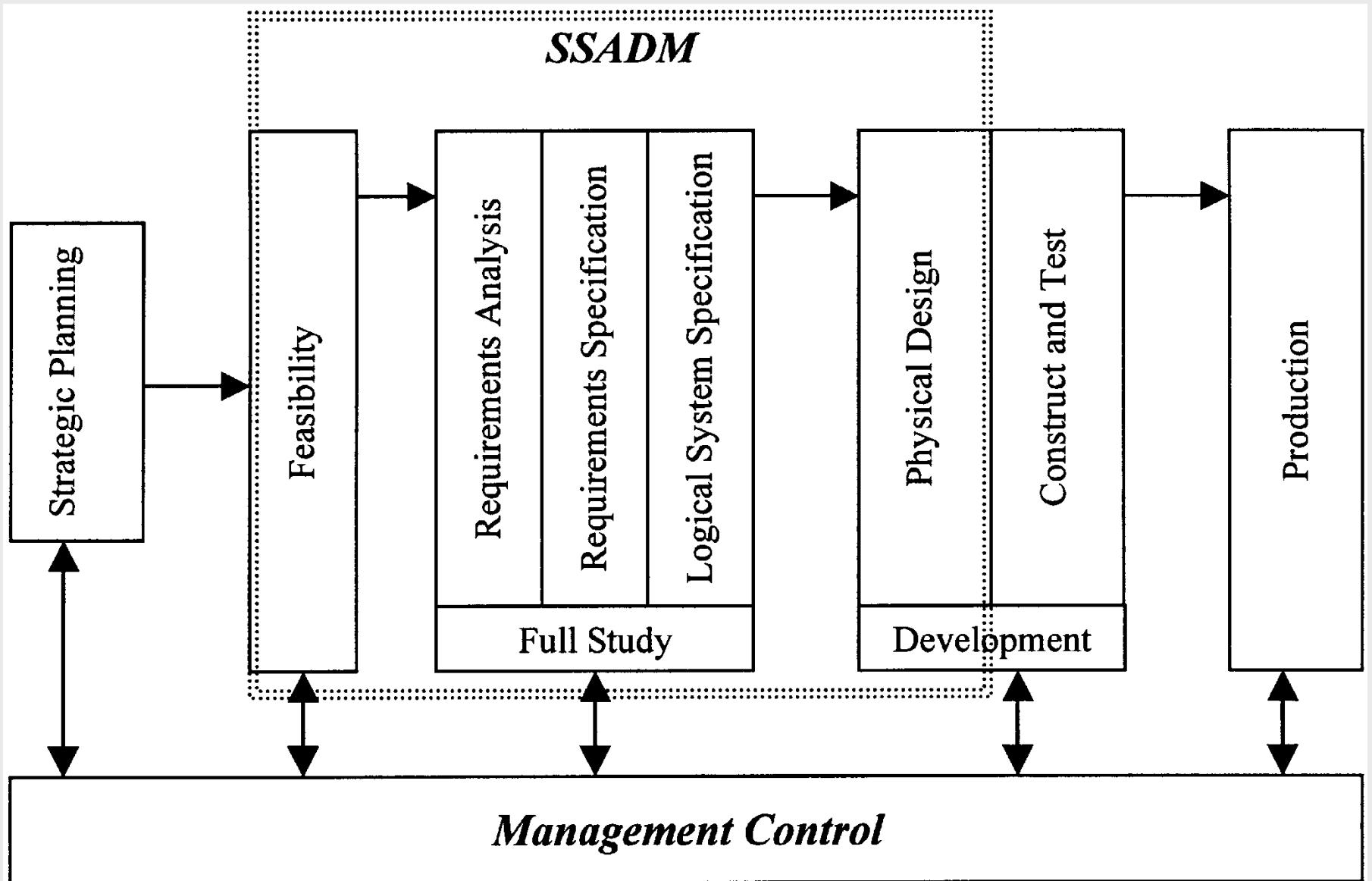
□ Faze – moduli, dalje razloženi u stadije (stages)

- 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)

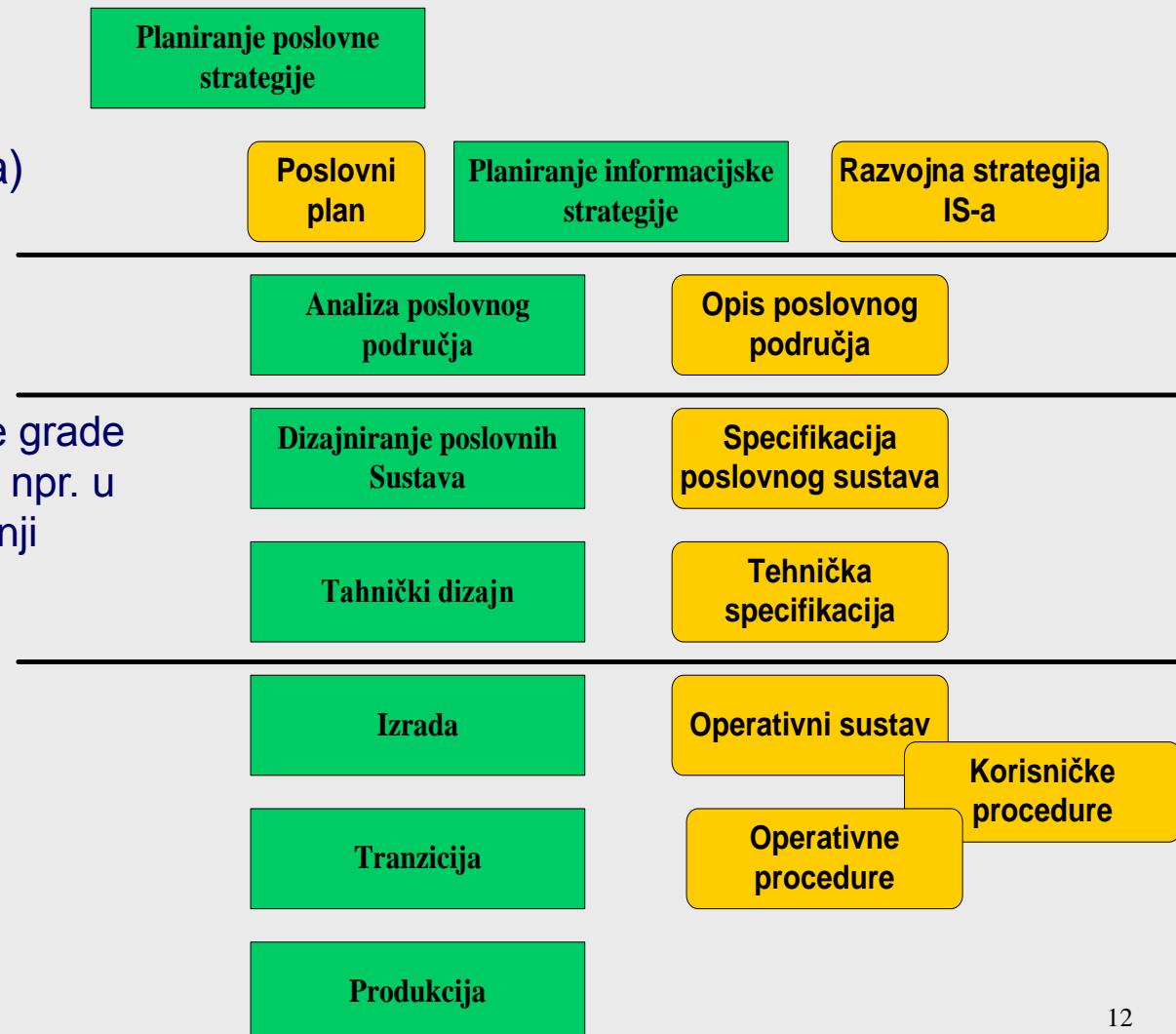
Životni ciklus prema SSADM



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

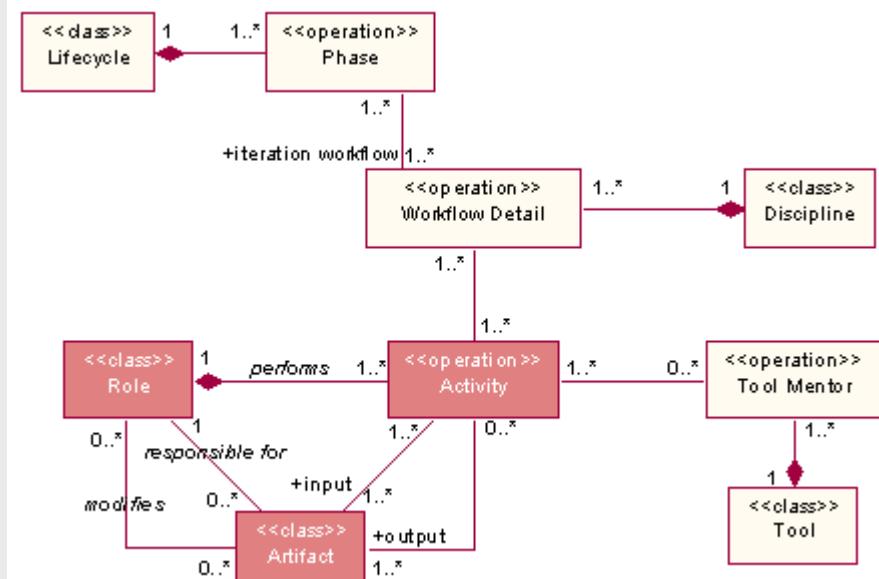
Ujedinjeni razvojni proces softvera

□ Unified software development process (UDP)

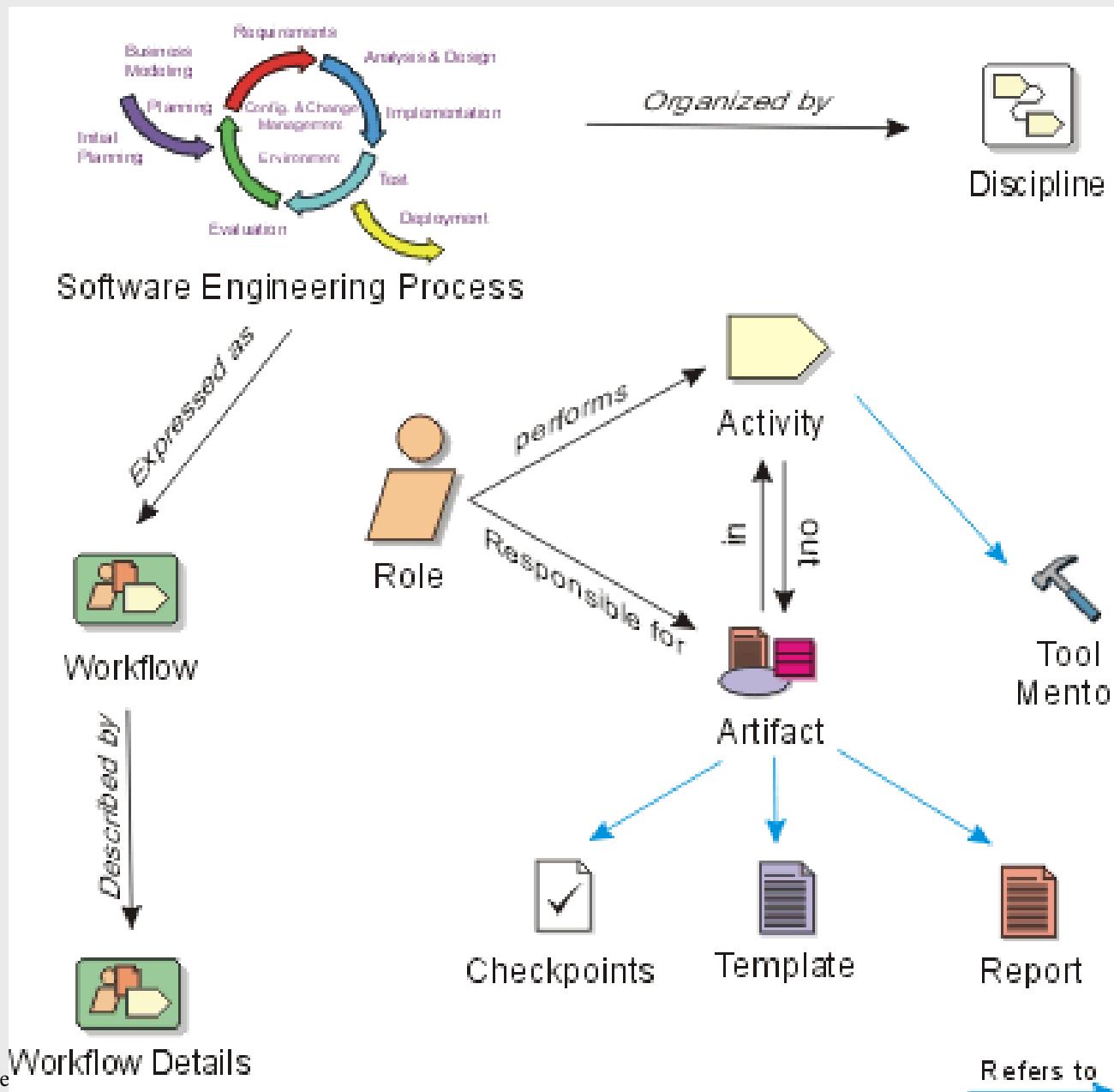
- okruženje za razvoj softvera temeljeno na iterativnom i inkrementalnom procesu
- najpoznatija komercijalna inačica – IBM Rational Unified Process (RUP)

□ 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

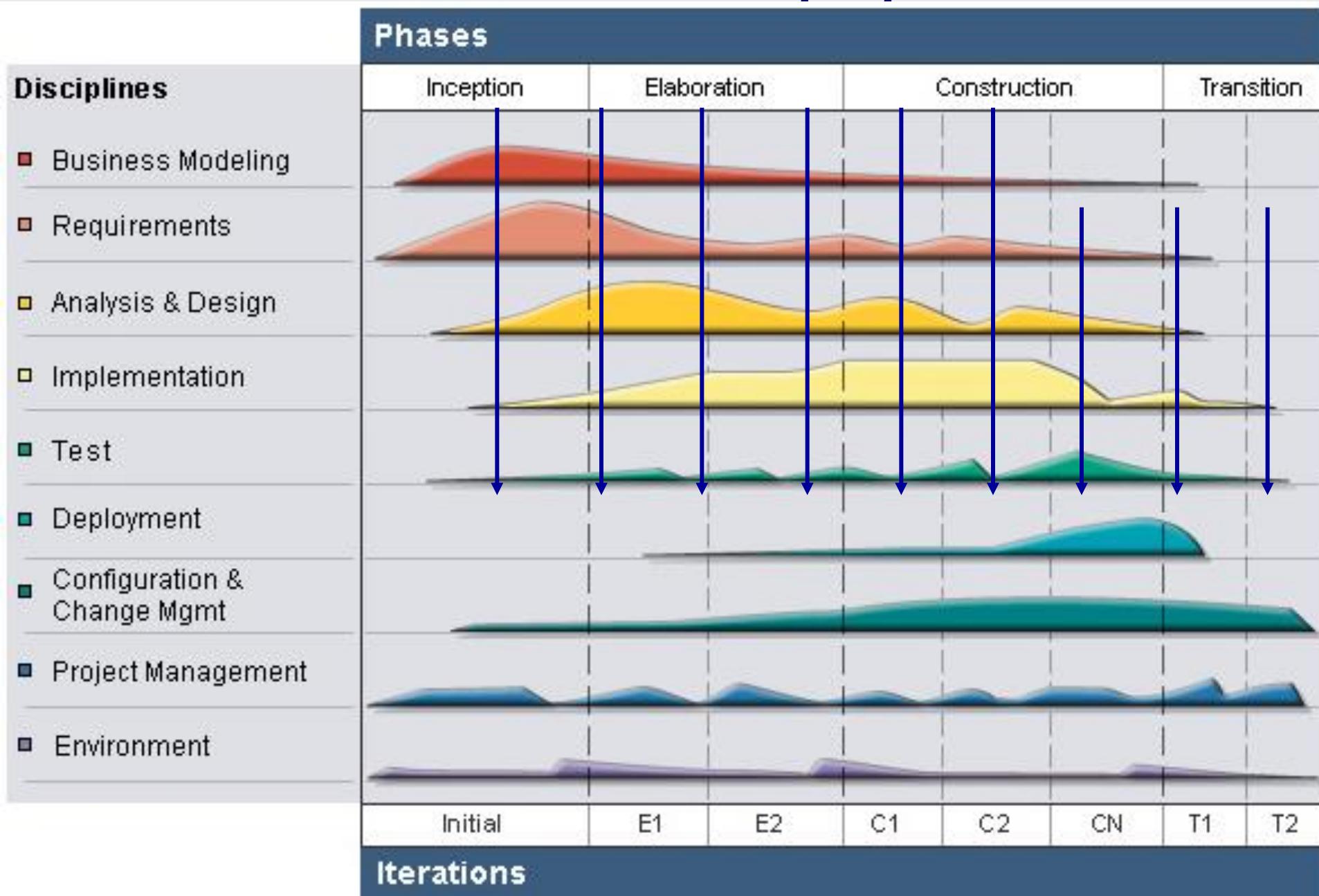
Načela (Process essentials)

- **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

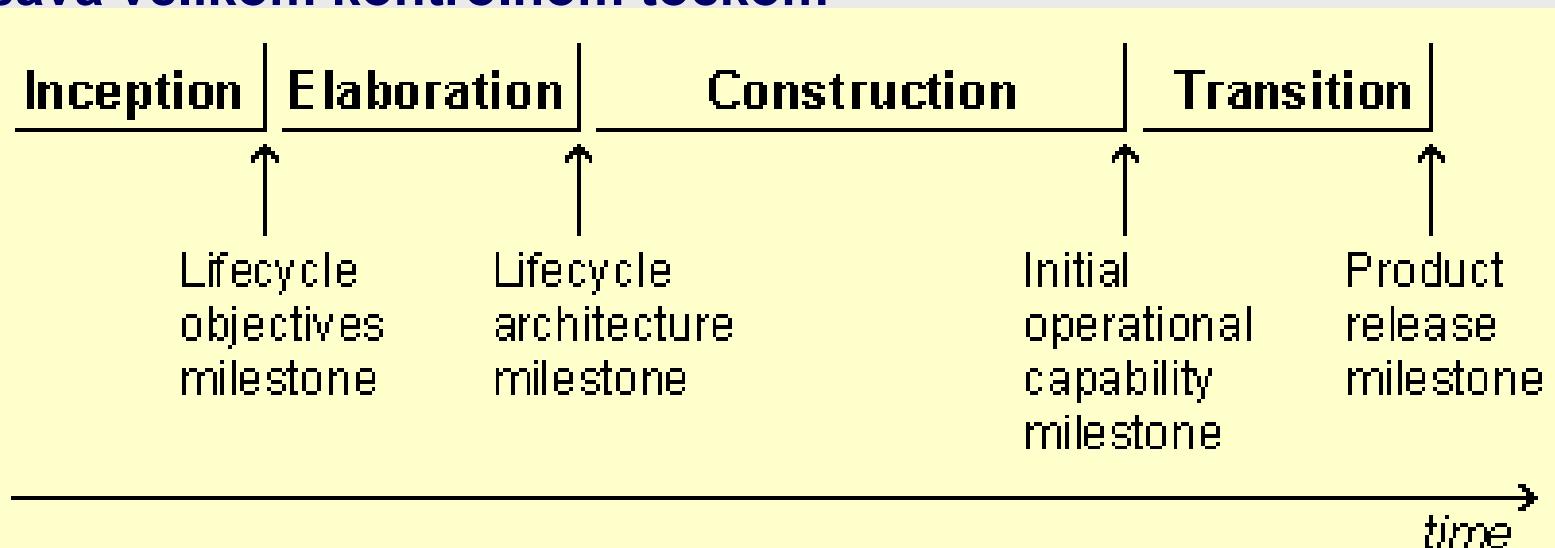
- **Prototip:** Inkrementalno graditi i testirati proizvod
 - iterativan i inkrementalan pristup - što ranije otkrivanje i razrješenje problema
- **Evaluacija:** Redovito procjenjivati rezultate
 - izvješća o statusu – identifikacija, komuniciranje i razrješenje problema
 - procjena na kraju svake iteracije – 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 usmjerenja na logički pogled i funkcionalne zahtjeve
 - Dizajn usmjeren 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

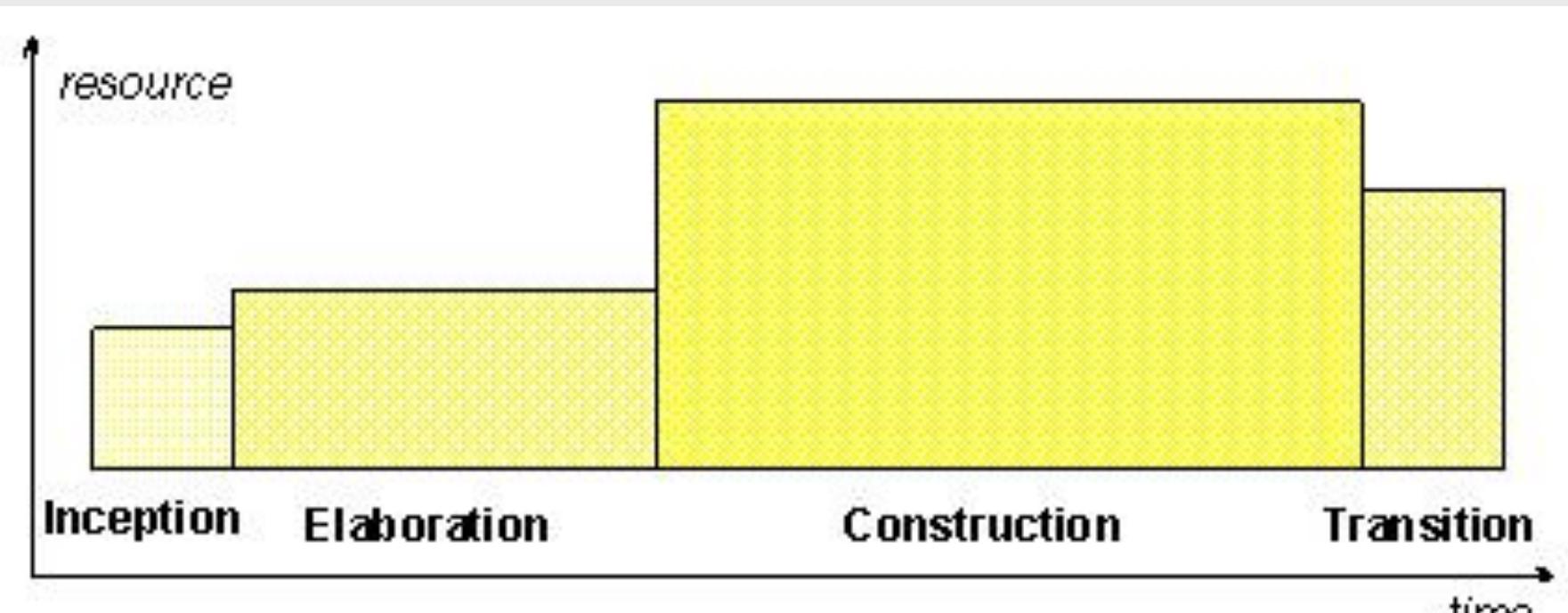
- Izvršavanje planova uvođenja u primjenu
- Dovršavanje korisničke dokumentacije i uputa
- Poduka krajnjih korisnika i održavatelja
- Testiranje programskog rješenja na lokaciji isporuke
- Izrada isporuke (release) konačnog programskog rješenja
- Prikupljanje povratne informacije od krajnjih korisnika
- 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

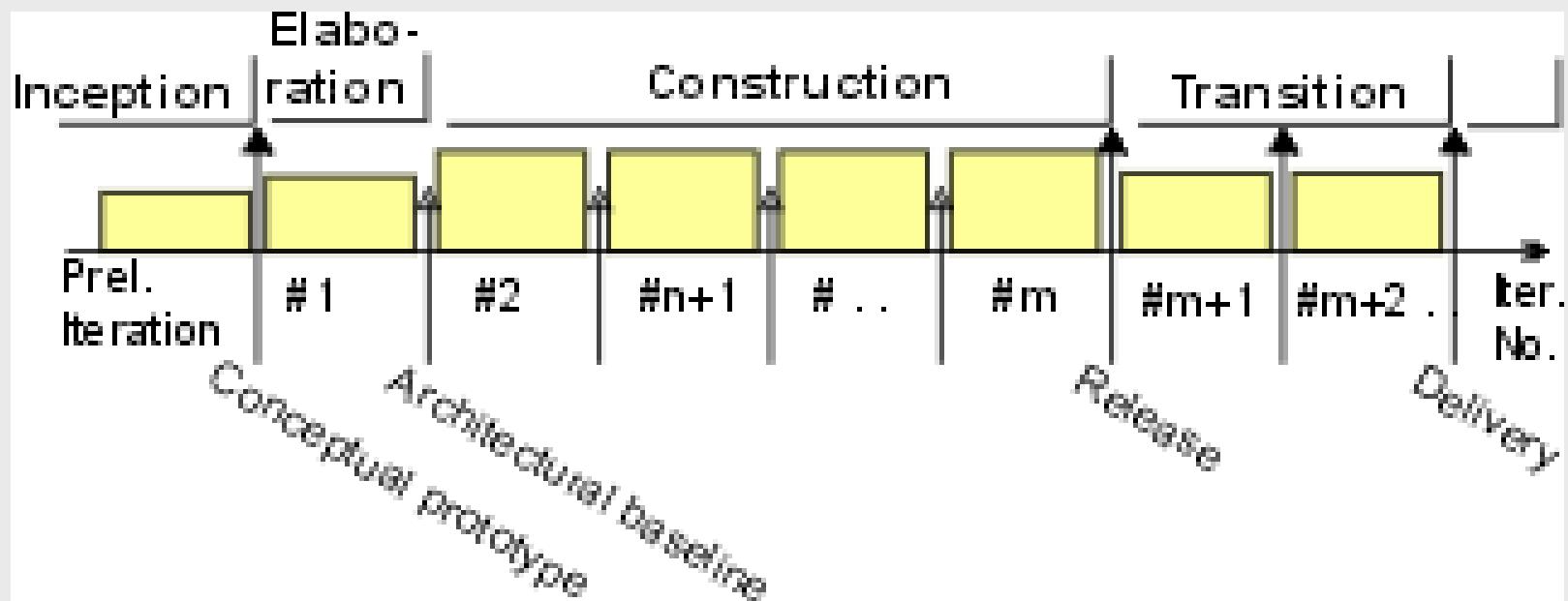
	Počinjanje (Inception)	Elaboracija (Elaboration)	Konstrukcija (Construction)	Prijelaz (Transition)
uložen napor (u odnosu na ukupan)	5%	20%	65%	10%
trajanje (u odnosu na ukupno)	10%	30%	50%	10%



Strategije iterativne provedbe RUP projekta (1)

□ Inkrementalna strategija

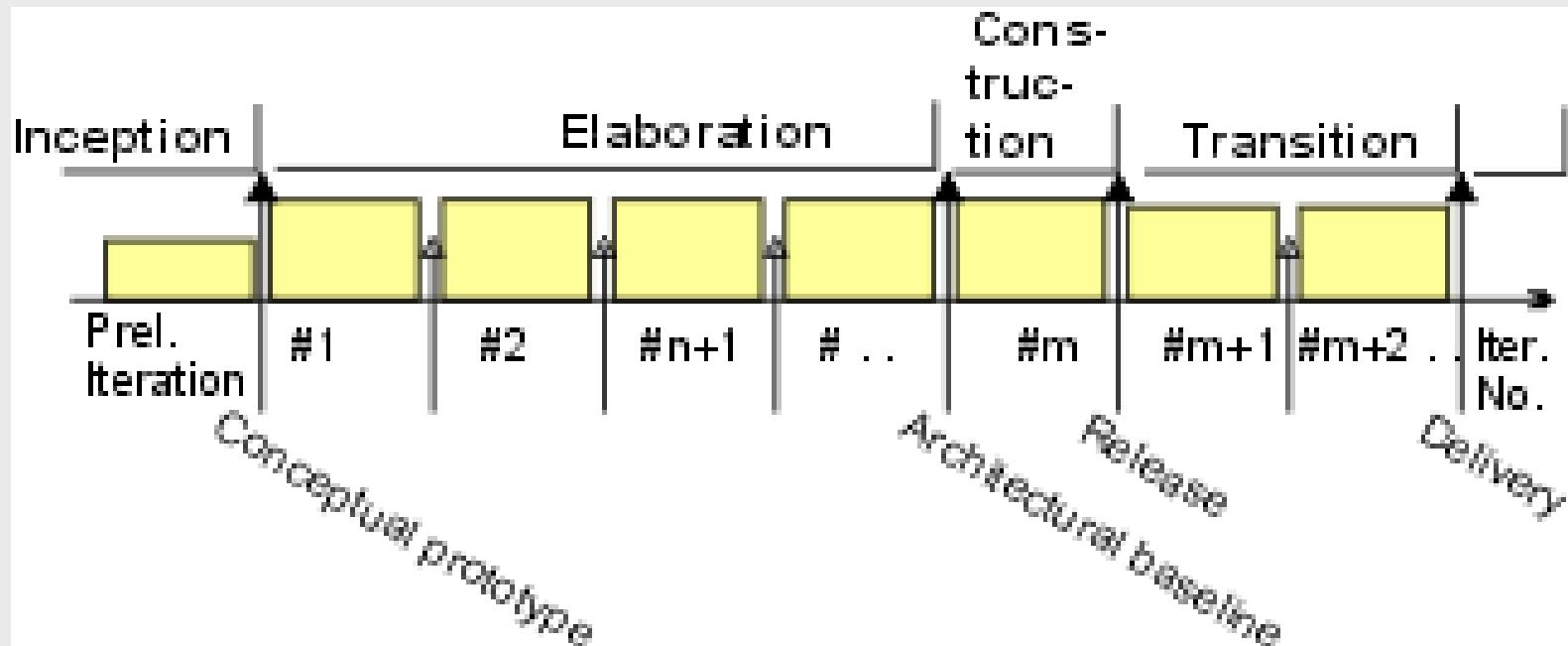
- poznata domena problema
- vrlo dobro poznati rizici
- iskusan projektni tim



Strategije iterativne provedbe RUP projekta (2)

□ Evolucijska strategija

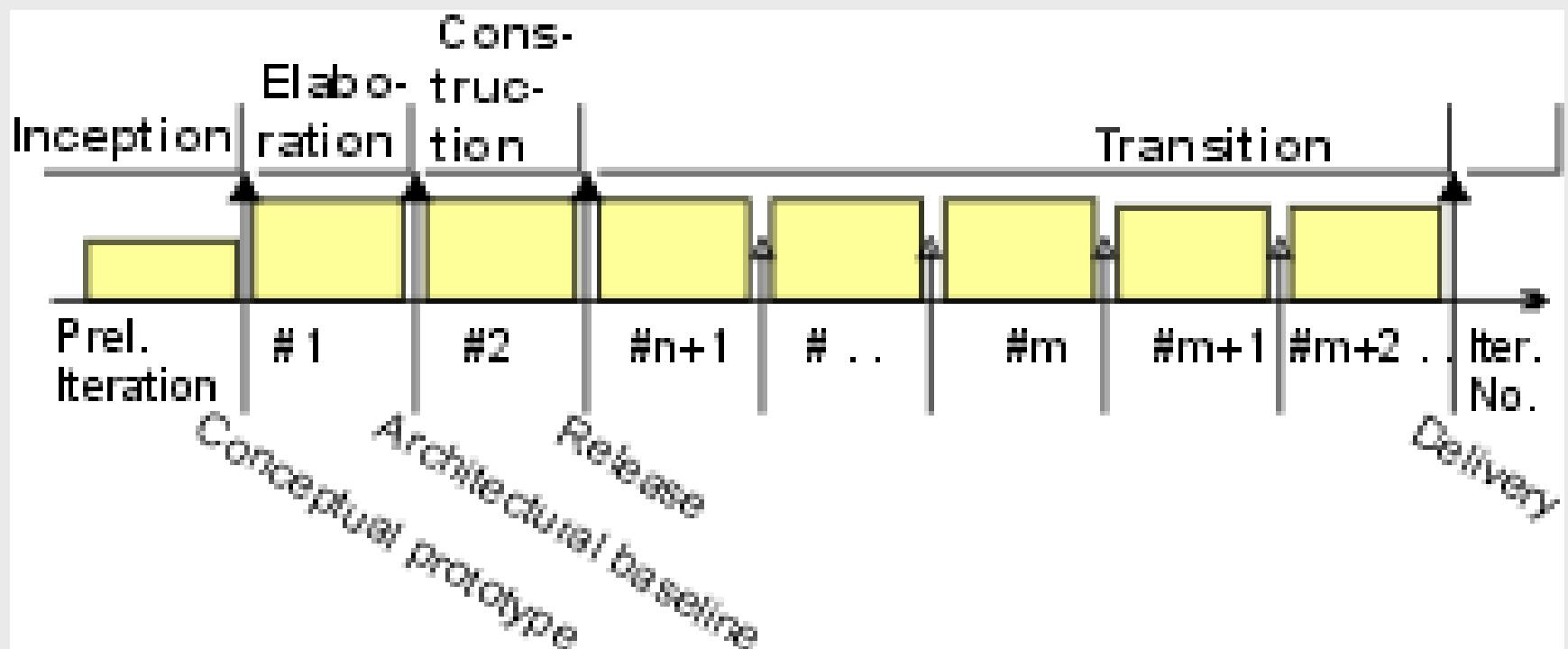
- nova ili nepoznata domena problema
- neiskusan projektni tim



Strategije iterativne provedbe RUP projekta (3)

□ Strategija inkrementalne isporuke

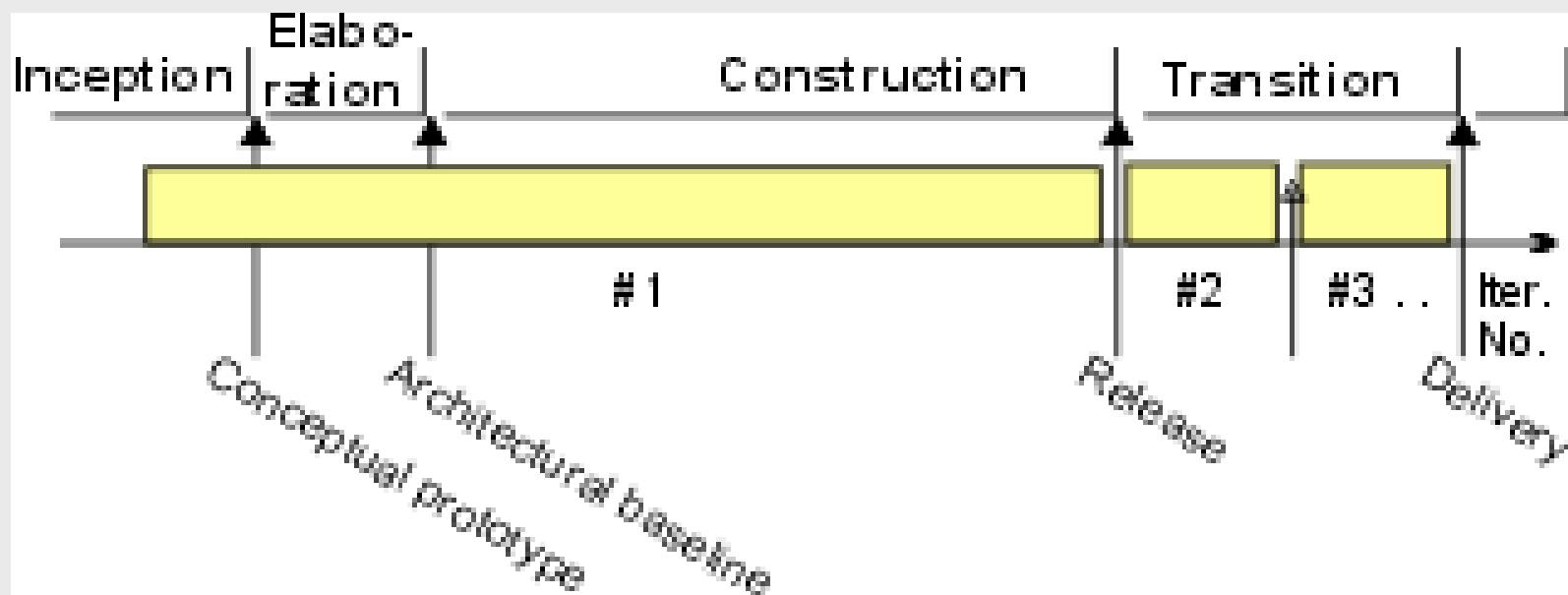
- poznata domena problema
- arhitektura i zahtjevi mogu biti stabilizirani rano u razvojnom ciklusu
- malo novog i nepoznatog u projektu
- iskusan projektni tim
- inkrementalne isporuke su vrlo važne i imaju visoku vrijednost za korisnika



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* – pojednostavljena 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

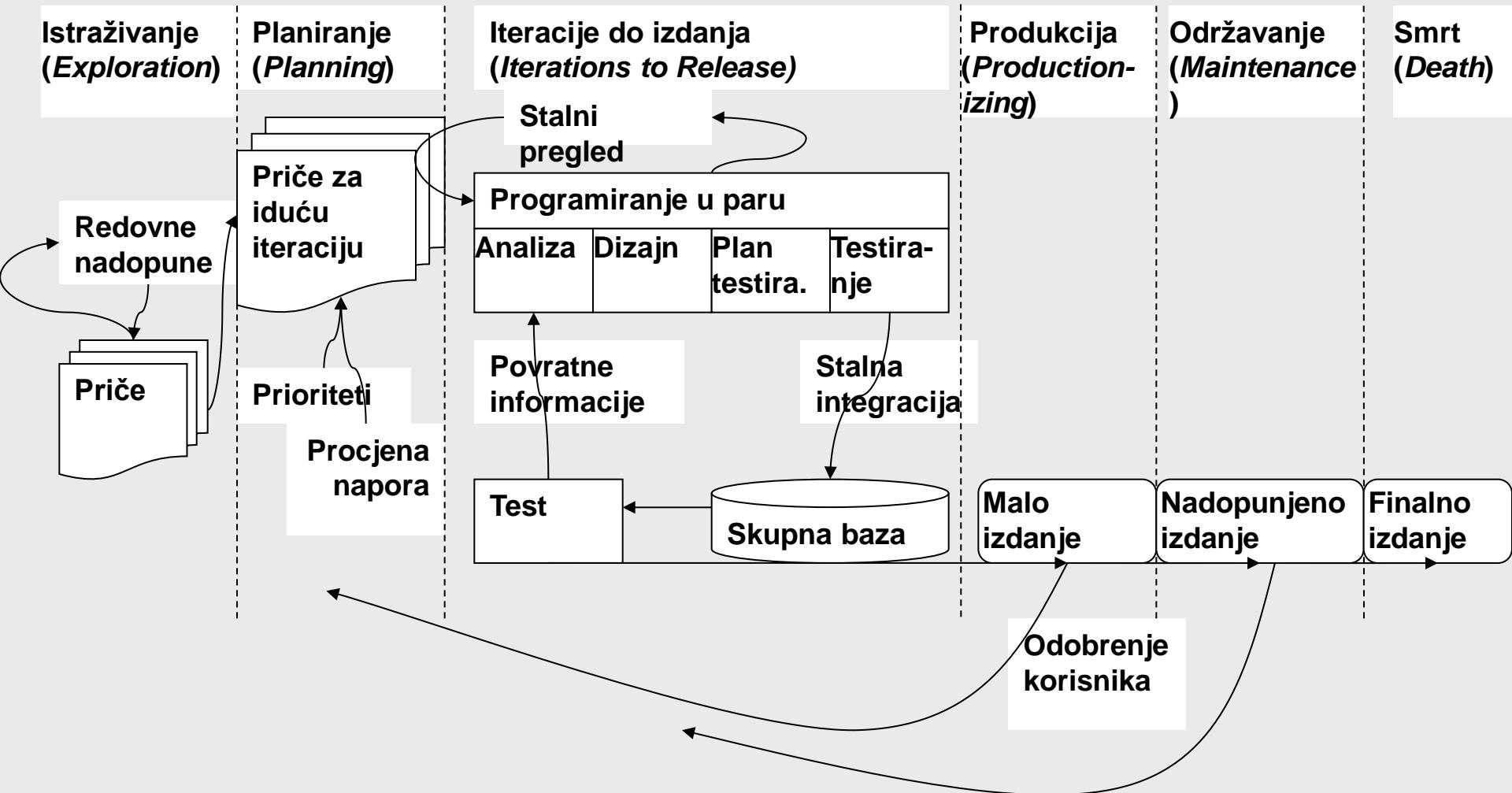
- Zadovoljstvo korisnika ranim i kontinuiranim isporukama softvera
- Promjene zahtjeva se željno prihvaćaju, čak i u kasnoj fazi razvoja
- Česta i što ranija isporuka softvera koji radi – 2x/mj do 1x/nekoliko mj
- Česta (dnevna) suradnja „poslovnjaka“ i razvojnika
- Motiviranje pojedinaca za rad u projektu – okruženje, povjerenje
- Usmena komunikacija - najbolja metoda za razmjenu informacija
- Glavna mjeru 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 (analiza zahtjeva)

- Korisnici bilježe svoje priče na kartice (!?)
- 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 (predviđanje i procjena)

- 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 (razvoj)

- 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)

- Verbalna i elektronička, učestala / stalna, svih dionika

□ Jednostavnost (simplicity)

- Najjednostavniji mogući dizajn (KISS)
- kontinuiranim refaktoriranjem i minimizacijom dokumentacije

□ Povratne informacije (feedback)

- Od korisnika i unutar tima – što ranije te češće

□ Hrabrost (courage)

- Akcije i (teške, nepopularne) odluke
- npr. odbacivanje dijelova (YAGNI), ili veće promjene u kasnoj fazi projekta

□ Uvažavanje (respect)

- Izbjegavanje promjena 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, tjedan započinje sastankom izbora priča
 - tjedan ne mora započeti u ponedjeljak
- **Kvartalni ciklus (Quarterly Cycle)**
 - grublje, na dulje staze, razvoj se planira kvartalno („rolling wave“)
- **Rezerva (Slack)**
 - Zadaci niskog prioriteta koji mogu biti odbačeni ako projekt kasni

Osnovne XP prakse (2)

□ Smještaj ekipe (Sit Together)

- Kolocirana ekipa, otvoreni prostor

□ Cjelovitost i zajedništvo ekipe (Whole Team)

- Cjelovitost kompetencija, osjećaj pripadnosti

□ Informativno radno okruženje (Informative Workspace)

- Ploče, „visible wall graphs”, CMS/GSS/PMS

□ Energičan rad (Energised Work)

- Odmorni programeri – produktivnost – ograničeni prekovremeni rad

□ Programiranje u paru (Pair Programming)

- Vodič (driver) i promatrač (observer, navigator)
- Zamjena uloga ali i partnera

Osnovne XP prakse (3)

❑ Inkrementalni dizajn (Incremental Design)

- Nema velikog oblikovanja unaprijed (BDUF – big design up-front)
- dizajn kao kontinuirani proces malih koraka – refaktoriranje

❑ Test prije programiranja (Test-First Programming)

- Testovi trebaju biti napisani prije kodiranja te automatizirani

❑ Desetominutna 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)

- Svakih nekoliko sati ili nakon promjena → dnevno

XP ekipa

- Upravitelj (manager, big boss)
 - Upravljanje ekipom, rješavanje problema # team lead
- Trener (coach)
 - Savjet, nadzor, kontrola (issue control) # tech lead
- Programer, razvojnik (developer)
 - kodiranje, pisanje testova, refaktoriranje
- Tester
 - Izrada i izvođenje testova, održavanje alata za testiranje
- Klijent, korisnik (customer)
 - piše priče i testove prihvatljivosti (!?), određuje prioritete
- Druge uloge
 - Tracker, Trainer, Doomsayer, ...
 - Product Manager, Domain Expert, Interaction Designer, Business Analyst ...

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)

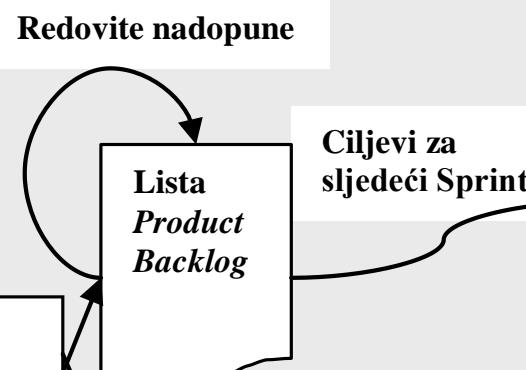
- razvoj iterativnim ciklusima, takozvanim sprintovima
- sprintovi - okvirno jednakog trajanja, 30 dana (prema knjizi)
 - tjedan do tri u praksi
- sprint ima sve faze klasičnog ciklusa (RUP ima mini-vodopad !)
 - zahtjeve, analizu, dizajn, evoluciju, test i isporuku
- tri do osam sprintova dok sustav ne bude spreman za distribuciju

□ Poslije igre (post-game)

- priprema sustav za izdanje kroz integraciju, testiranje i druge aktivnosti

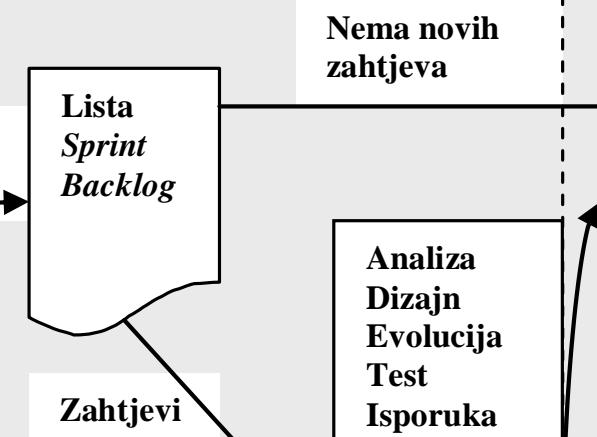
Scrum proces

Prije igre



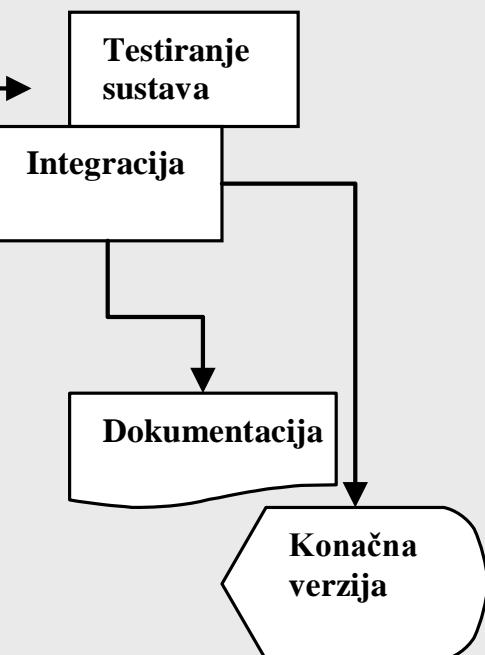
Ciljevi za sljedeći Sprint

Razvoj (Igra)



Nema novih zahtjeva

Poslije igre



Testiranje sustava

Integracija

Dokumentacija

Konačna verzija

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

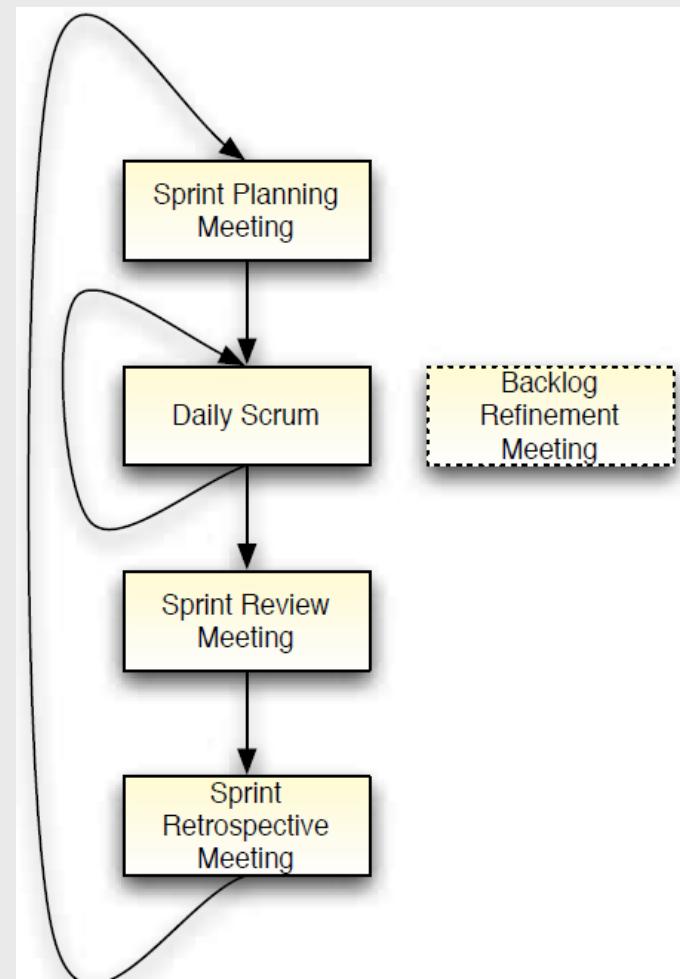
- demonstracija inkrementa na kraju sprinta
- Vlasnik deklarira "dovršeno"
- ostalo ide u naredni sprint

□ Retrospektiva sprinta

- samoanaliza procesa

□ Pročišćavanje preostalog posla

- podjela, procjena, prioriteti ...



Materijali i internet adrese

- 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.featuredrivendevlopment.com>
- <http://www.scrumalliance.org/>
- <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

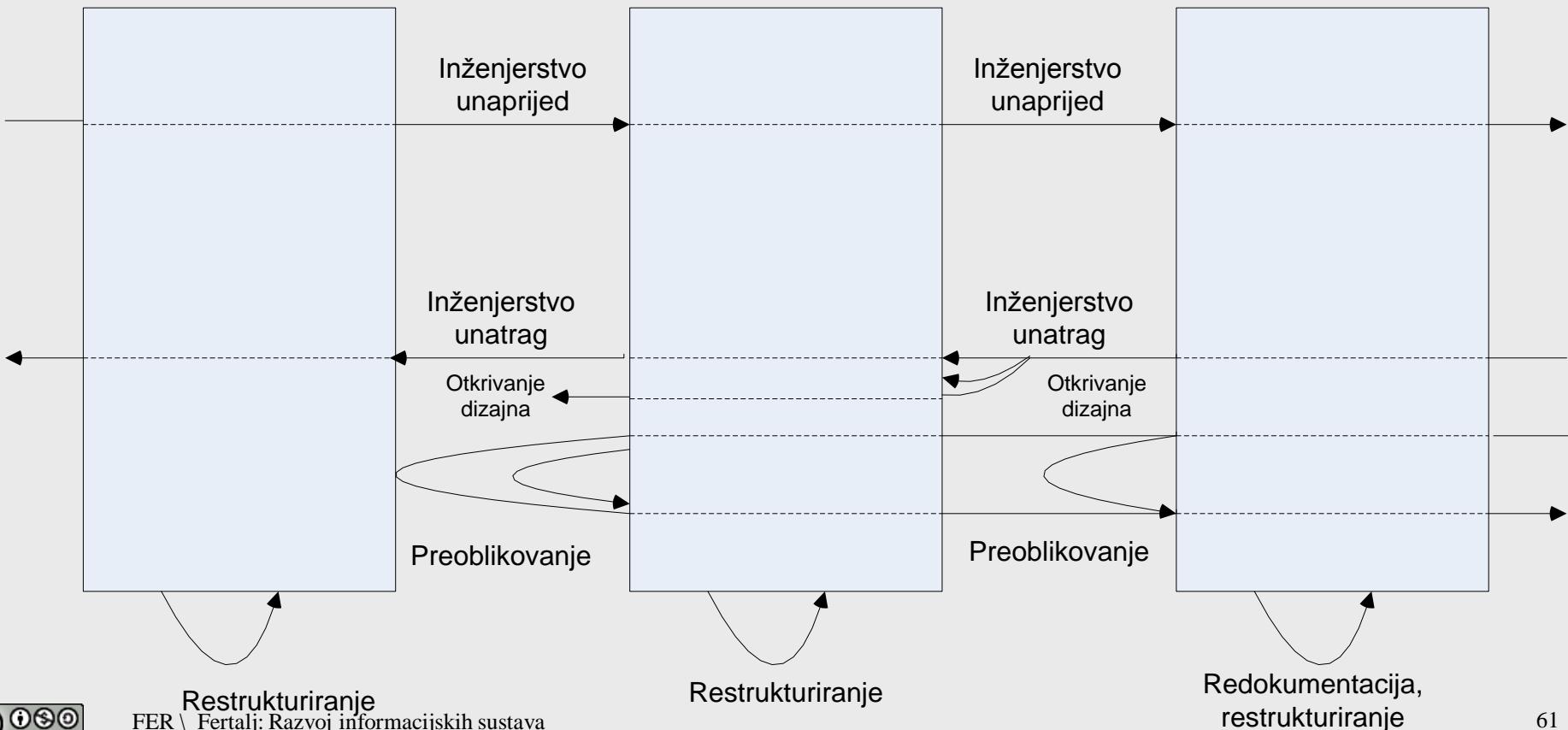
□ Preoblikovanje programske podrške (software reengineering)

- 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

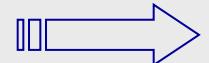
- **Inženjerstvo prema naprijed (forward engineering)**
 - tradicionalni postupak - s viših razina apstrakcije prelazimo na projektiranje i izvođenje
 - oznaka „naprijed“ - samo da se napravi razlika prema inženjerstvu unatrag
- **Inženjerstvo unatrag (reverse engineering) - obrnuto, reverzno, povratno**
 - 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)



- mjerjenje 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

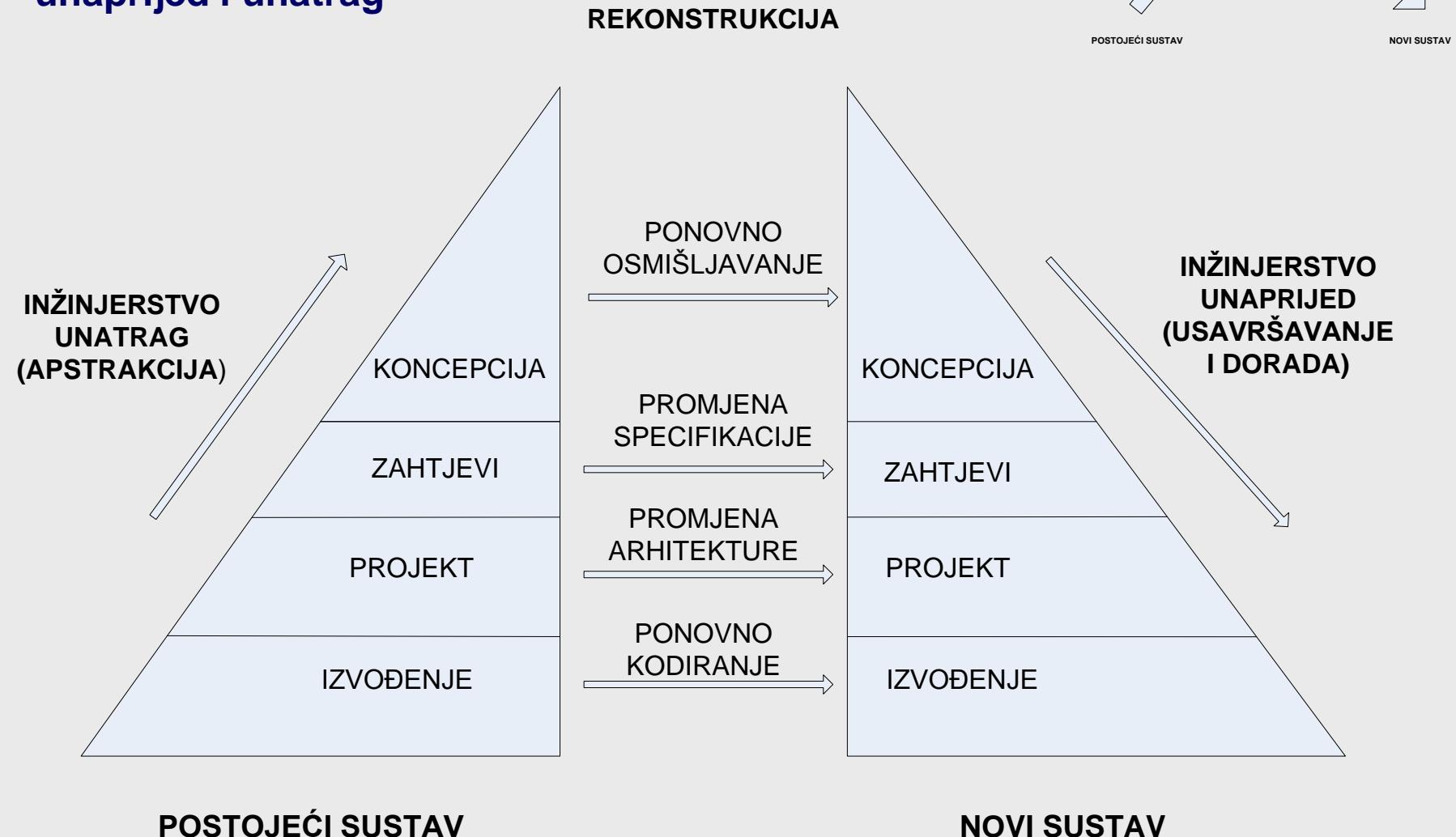
□ Mjerenja programske potpore (software measurements)

- *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* – funkcijске 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

□ Općeniti alati, alati općenito

- <http://www-01.ibm.com/software/rational/?lnk=mprSO-rati-user>
- 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>

□ Ostalo

- <http://modeling-languages.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

2013/14.11

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 IS koji zamjenjuje odvojene podsustave 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.

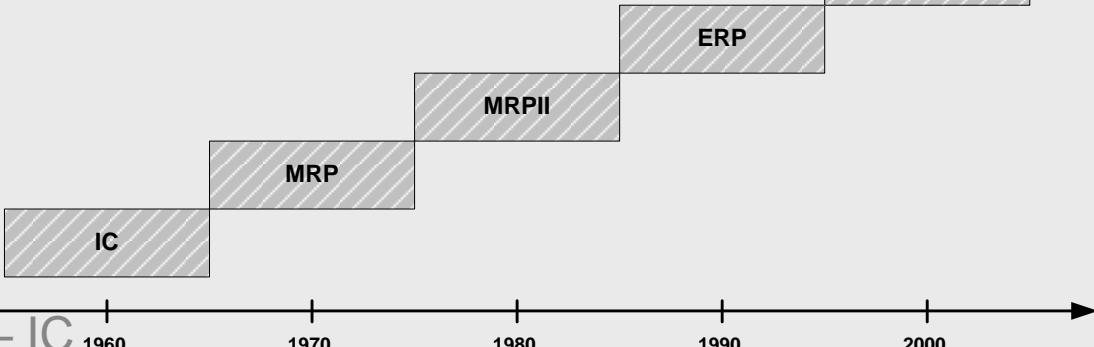
□ Upravljanje kvalitetom

- 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



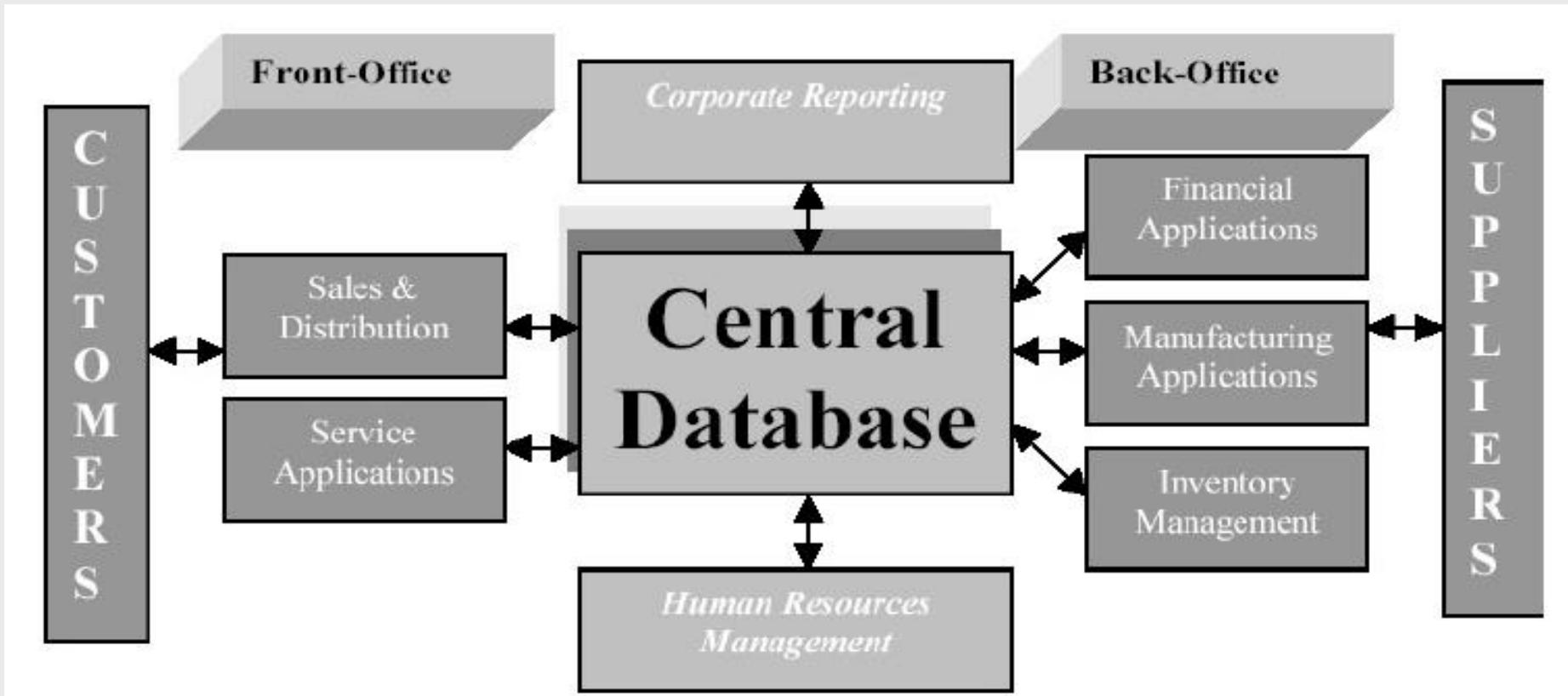
□ Razvoj

- 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

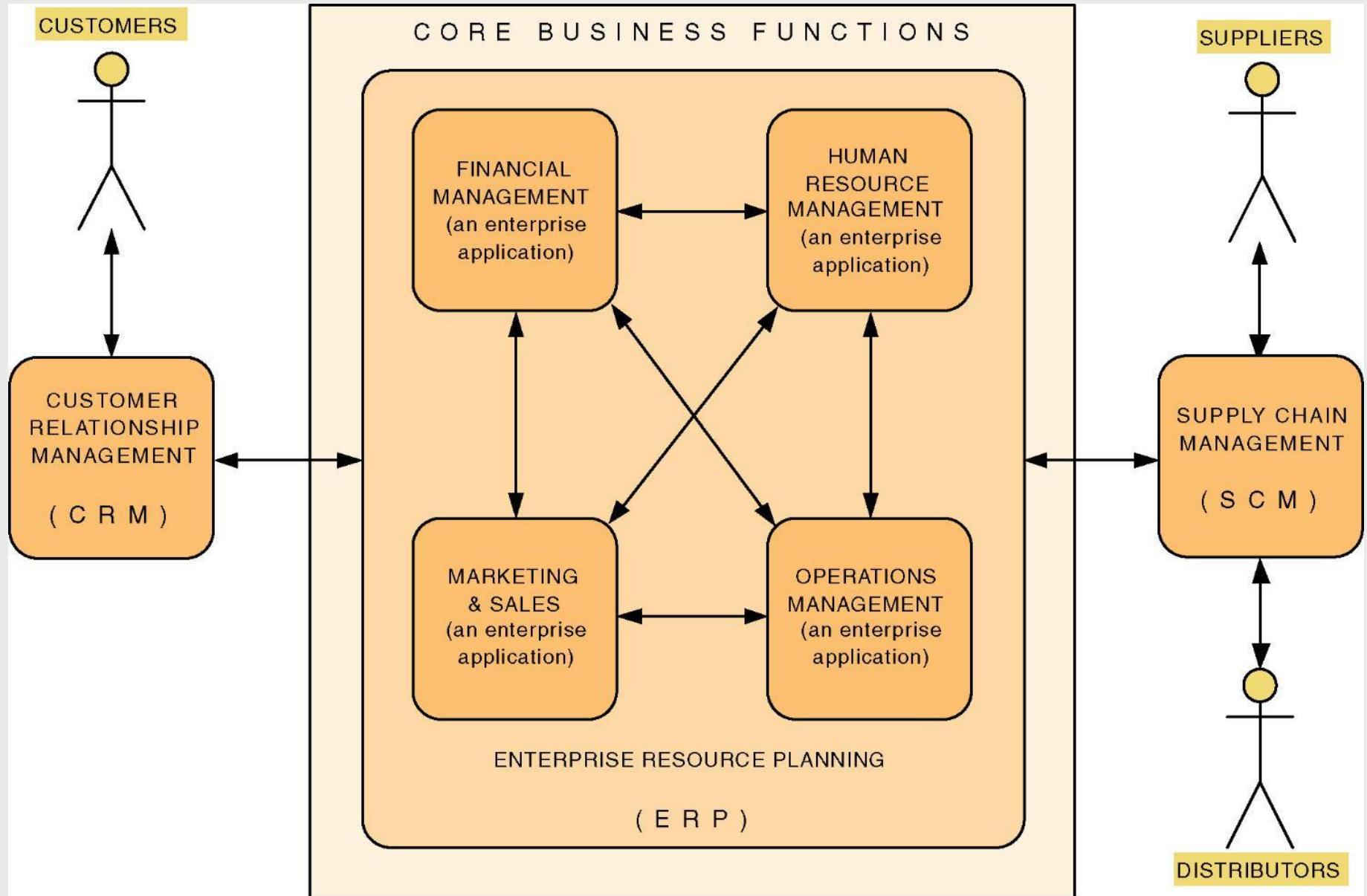
- orientacija na Internet, vanjski moduli, e-bussines rješenja
- napredno poslovno planiranje (*APS – Adwanced Planning and Scheduling*),
- automatizirana prodaja (*SFA – Sales Force Automation*),
- poslovna inteligencija (*BI – Business Inteligence*),
- upravljanje odnosom s kupcima (*CRM – Customer 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

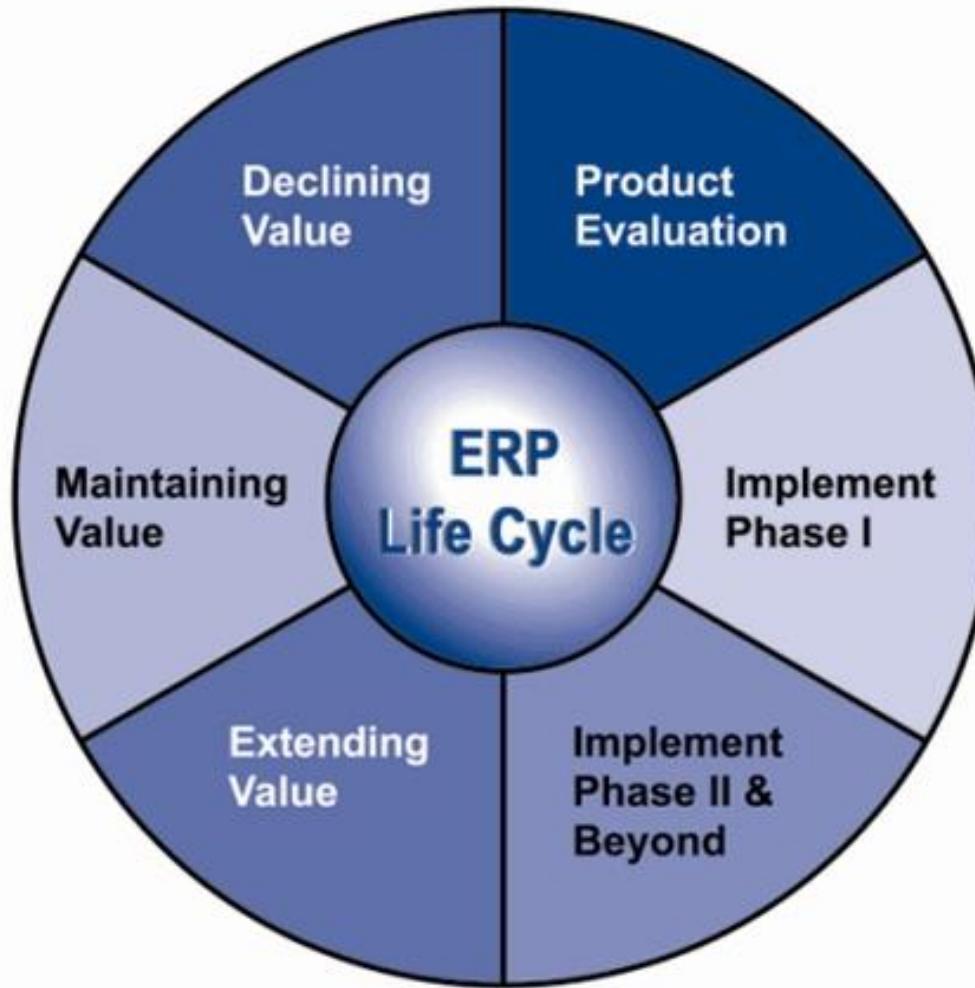
□ Supply Chain Management (SCM)

- 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**

Reference

□ Reference

- M.Bradford, Modern ERP: select, implement & use today's advanced business systems, 2010, www.modernerp.com

□ Projekti vrednovanja ERP sustava

- https://www.zpr.fer.hr/projekt.php?sif_proj=22
- https://www.zpr.fer.hr/projekt.php?sif_proj=29
- <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)

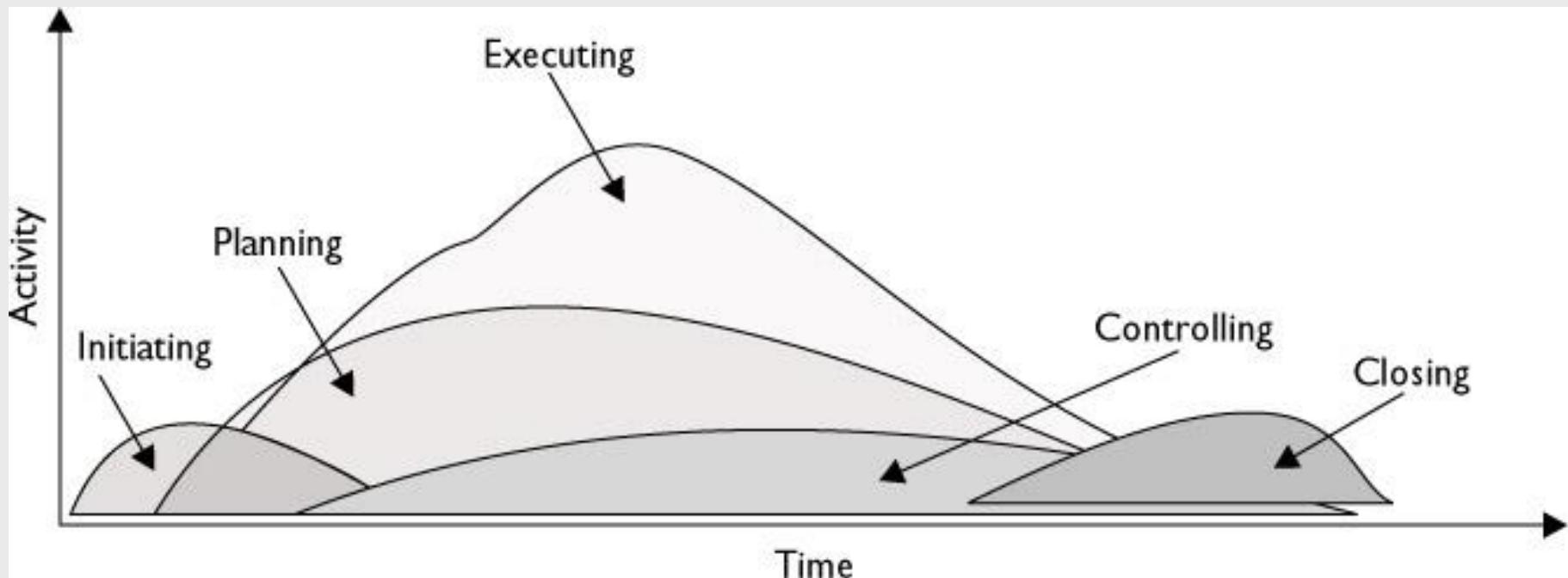
- primjena znanja, vještina, alata i tehnika u projektnim aktivnostima da bi se ispunili projektni zahtjevi [PMI]

□ “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)

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

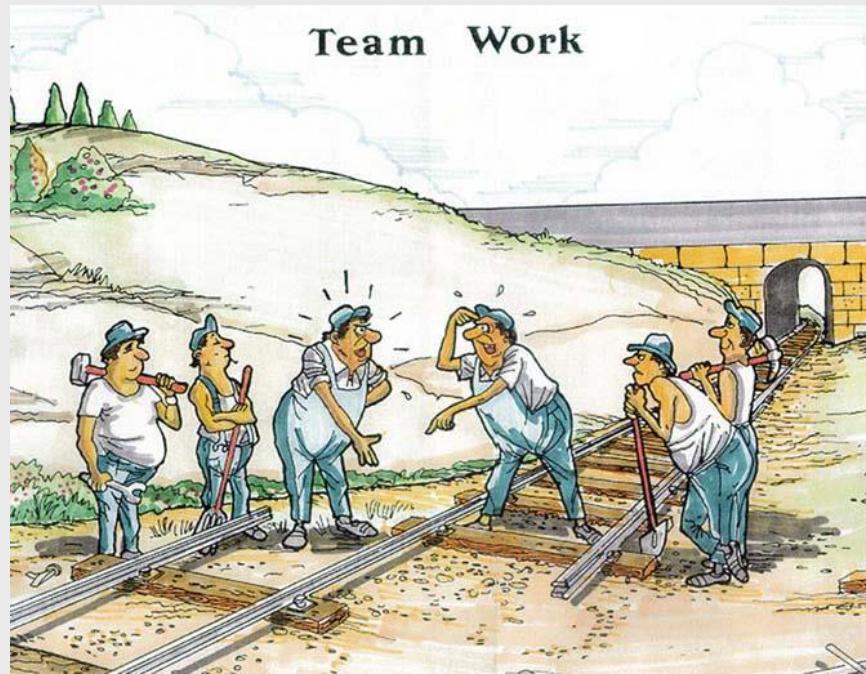
- za posao treba vremena i resursa onoliko koliko je raspoloživo

Organizacija projekta

Organizacija i ekipni rad

□ Ekipni rad (teamwork)

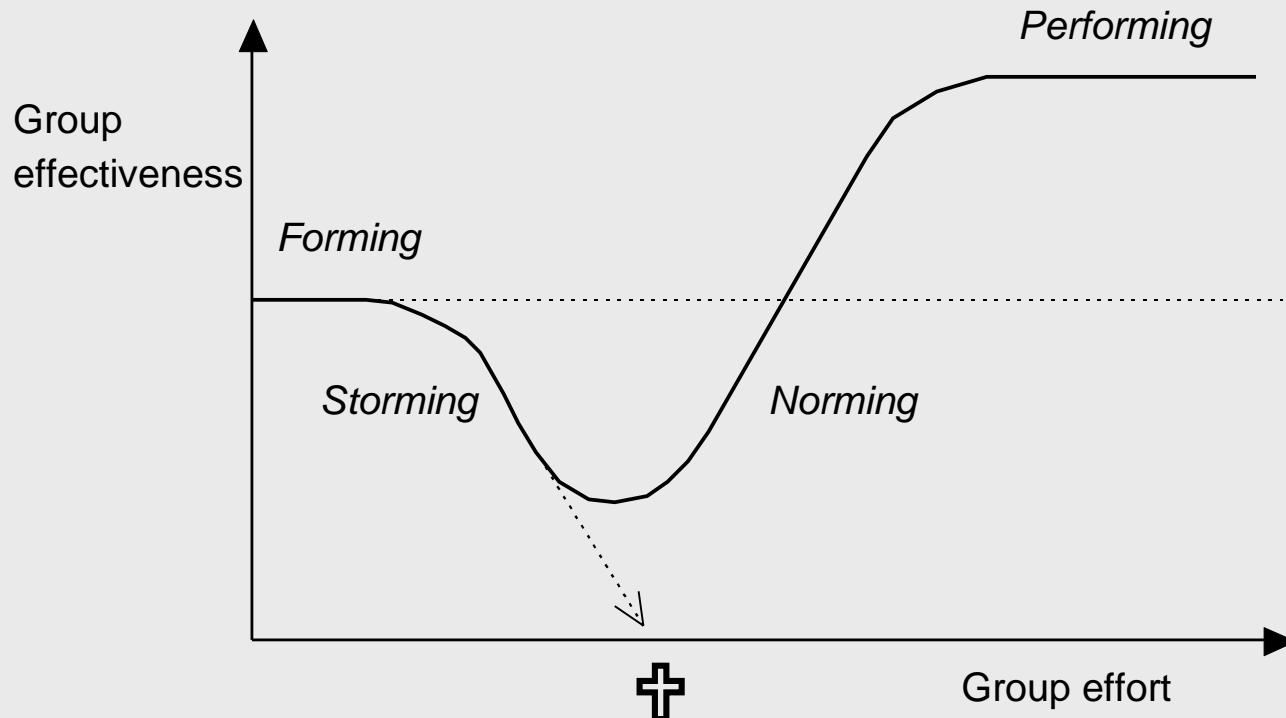
- Ekipa, tim - grupa međuzavisnih pojedinaca koji zajedno rade na ostvarivanju zajedničkog cilja te dijele odgovornost za završetak posla
- svaki član ima specifičnu ulogu (ili uloge) i odgovornost(i)
- Prednosti:
 - kvalitetnije donošenje odluka – "ne treba nam soliti pamet"
 - motivacija članova – "zajedno smo jači"
 - inovativnost – "dvoje zna više nego jedan, ..."
- Sinergija
 - $2+2=5$?



Razvoj ekipe

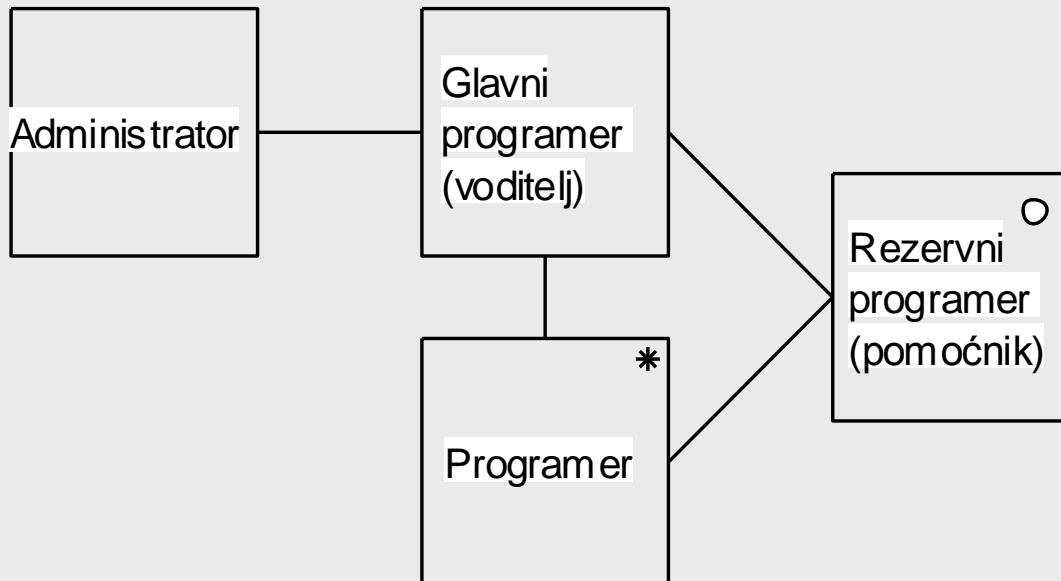
□ Razvoj ekipe (Forming, Storming, Norming, Performing)

- 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

- Upravitelj (manager, big boss)
 - Upravljanje ekipom, rješavanje problema # team lead
- Trener (coach)
 - Savjet, nadzor, kontrola (issue control) # tech lead
- Programer, razvojnik (developer)
 - kodiranje, pisanje testova, refaktoriranje
- Tester
 - Izrada i izvođenje testova, održavanje alata za testiranje
- Klijent, korisnik (customer)
 - piše priče i testove prihvatljivosti (!?), određuje prioritete
- Druge uloge
 - Tracker, Trainer, Doomsayer, ...
 - Product Manager, Domain Expert, Interaction Designer, Business Analyst ...

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

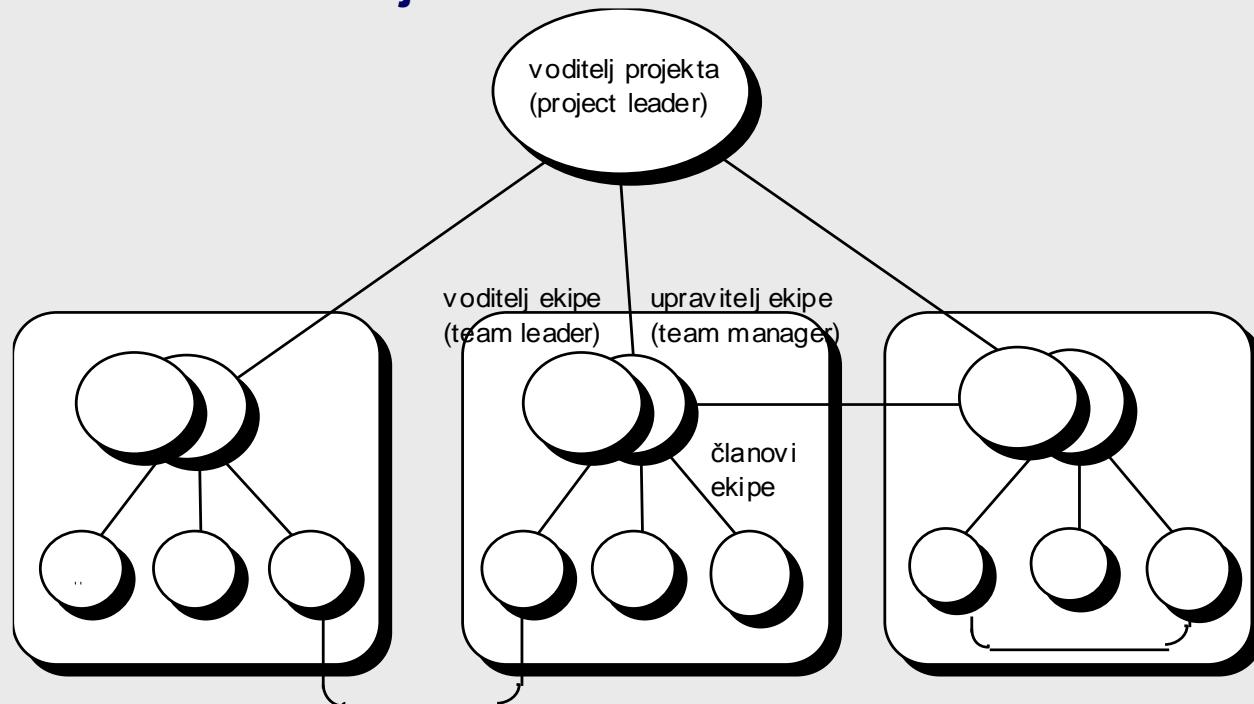
Ekipa može imati i dva voditelja

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
- **Osnajivanje (*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

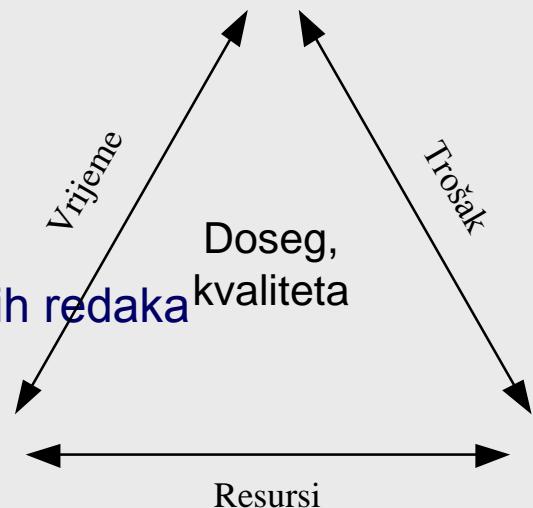
Planiranje projekta

□ 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

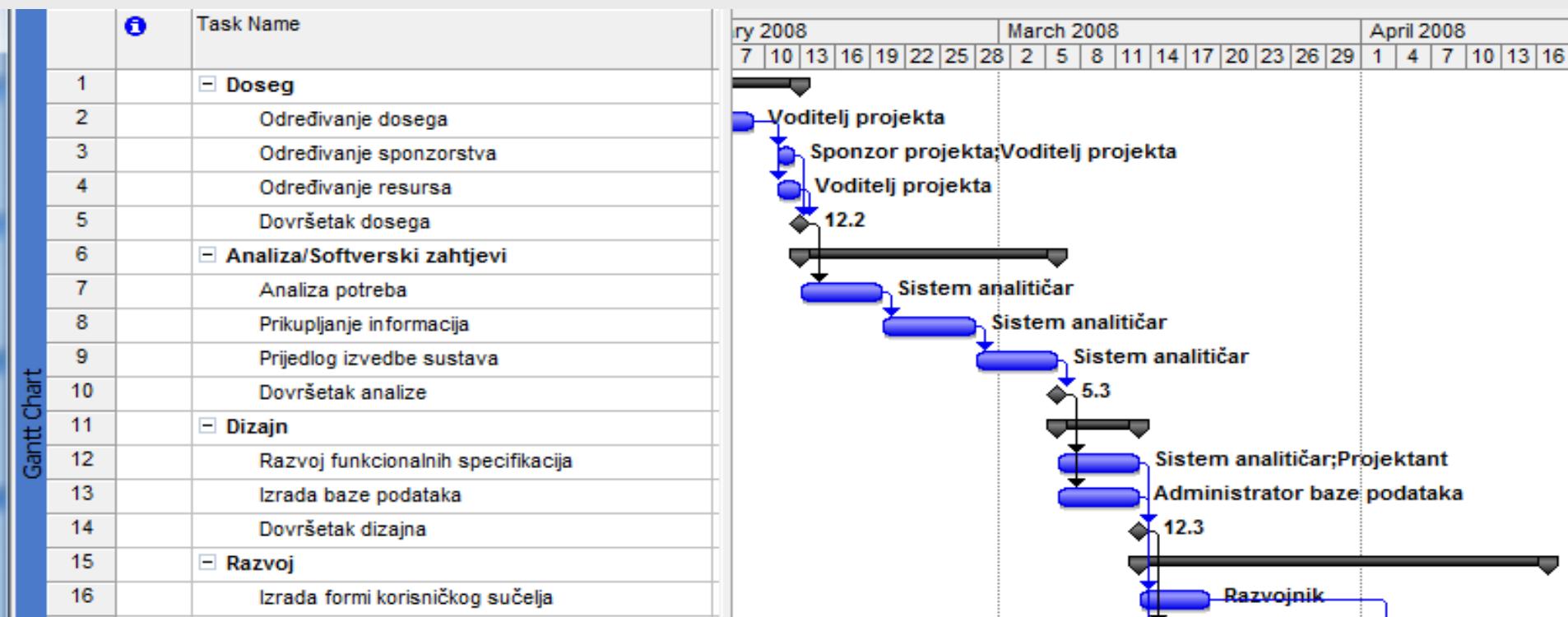
Vremenski raspored projekta

- **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

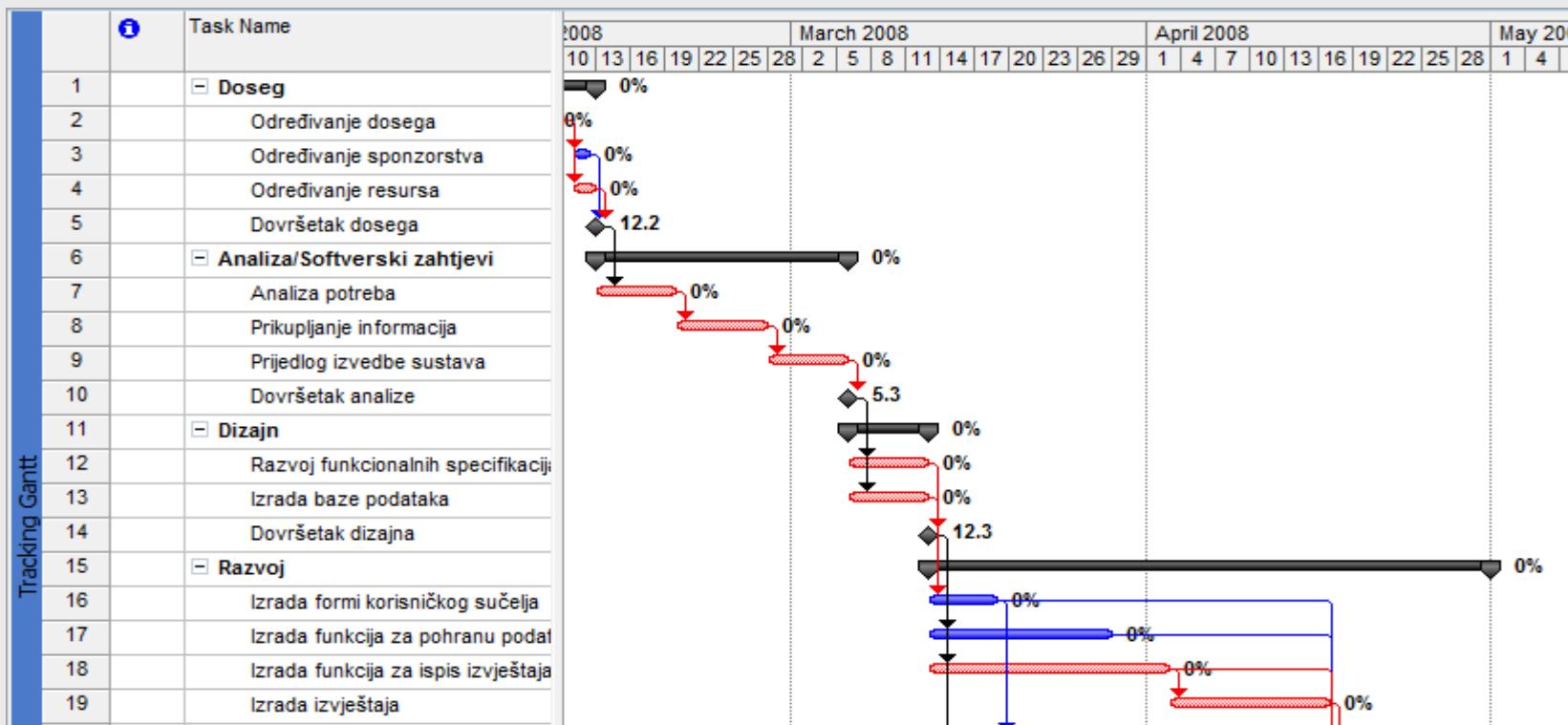


Kritični put

□ 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

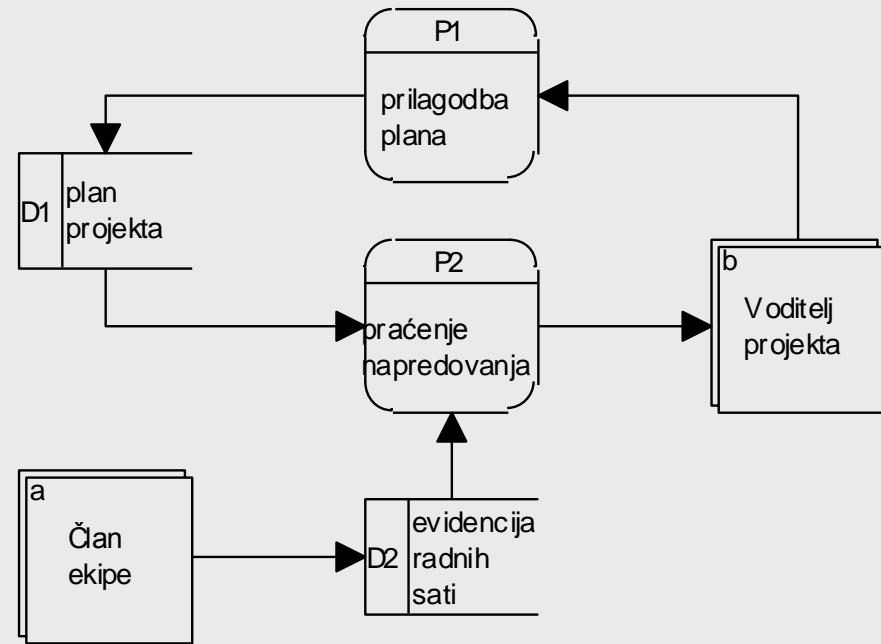
Postupci nadzora i kontrole

□ 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

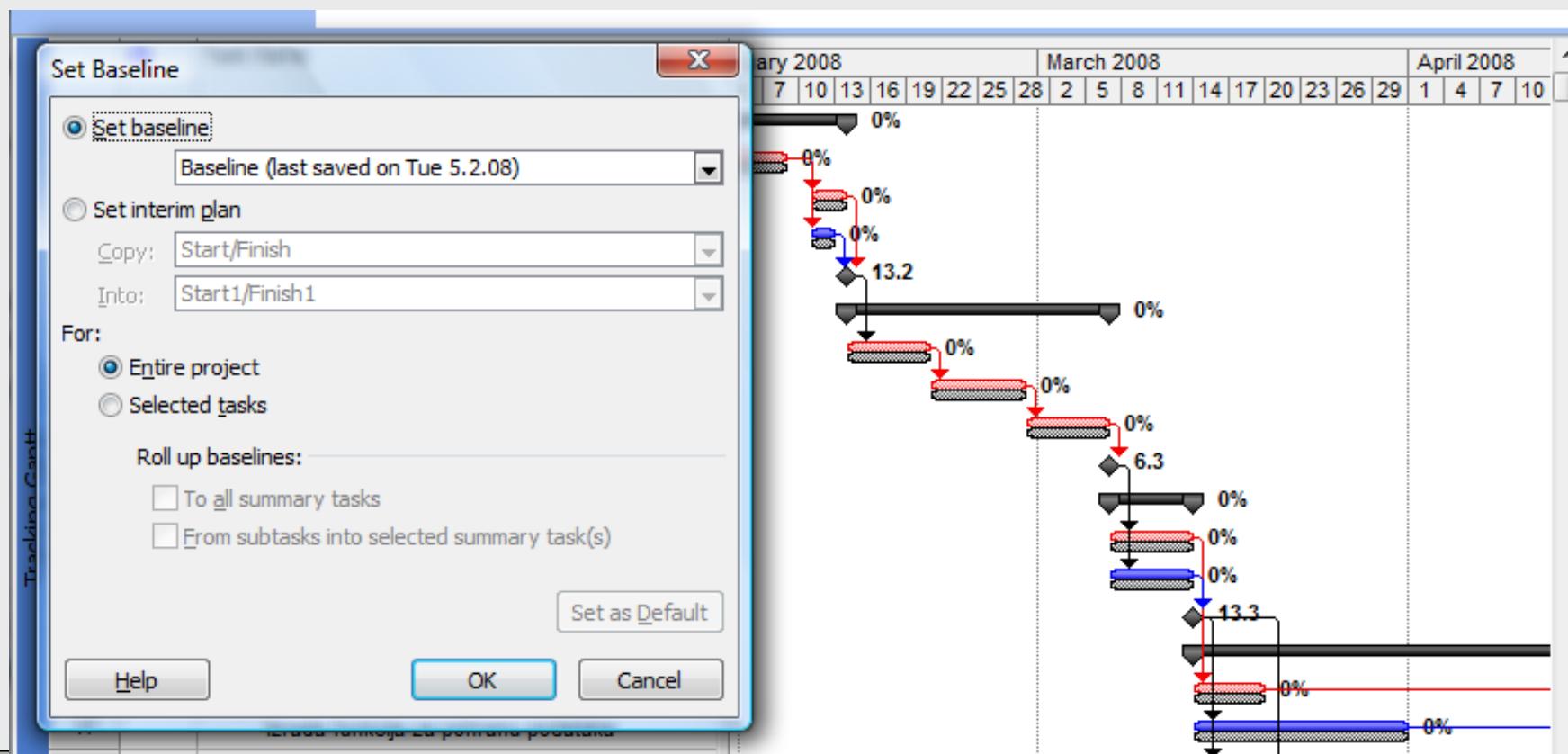


Zadatak	Izvršitelji	Planirani početak	Planirani završetak	Prioritet	Planirano trajanje	Stvarno trajanje	Status	% ispunjenja	Stvarni završetak	Kašnjenje	Komentar

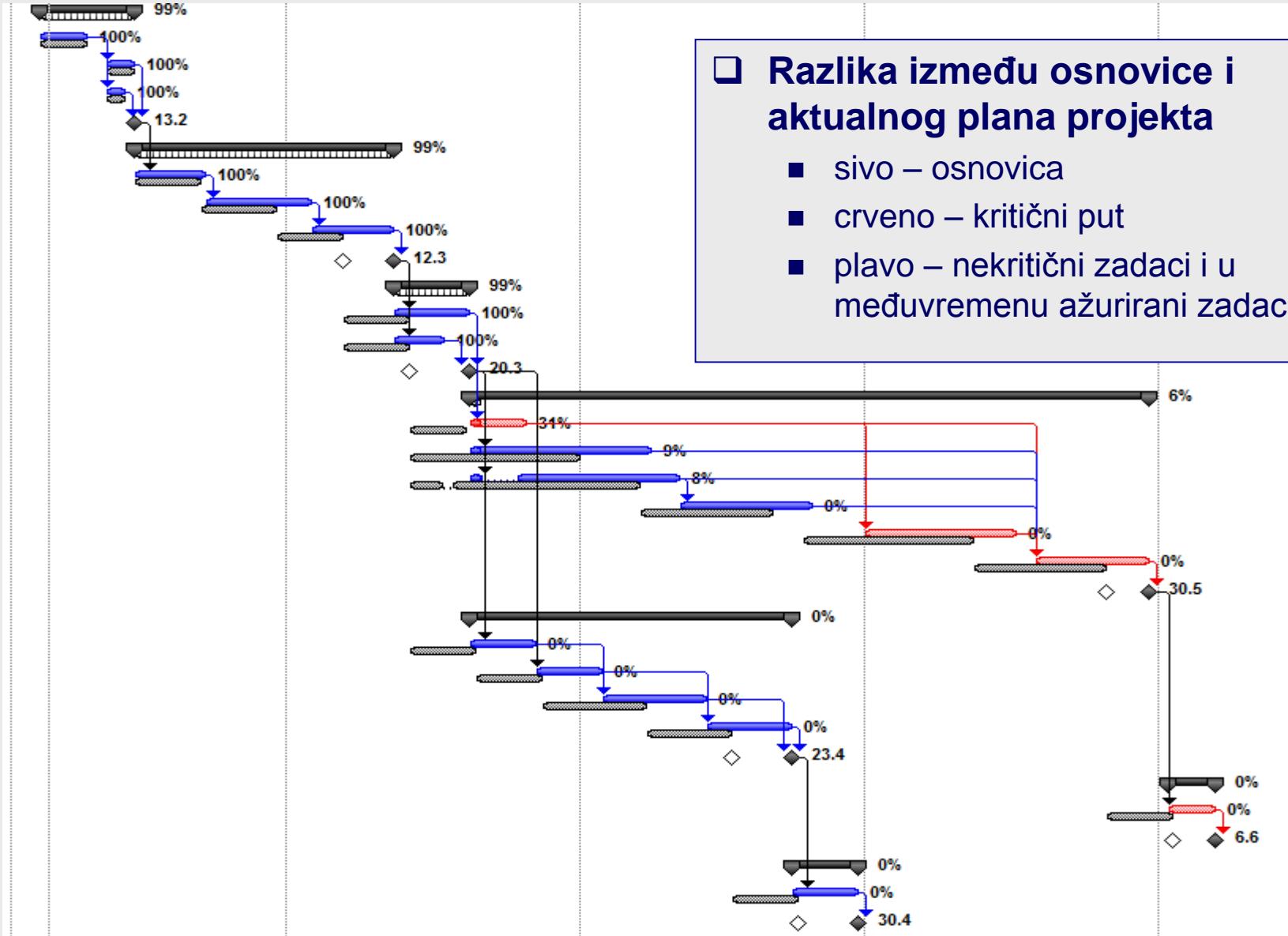
Praćenje napretka – spremanje osnovice

□ Osnovica (baseline) – plan u odnosu na koji se prati napredak

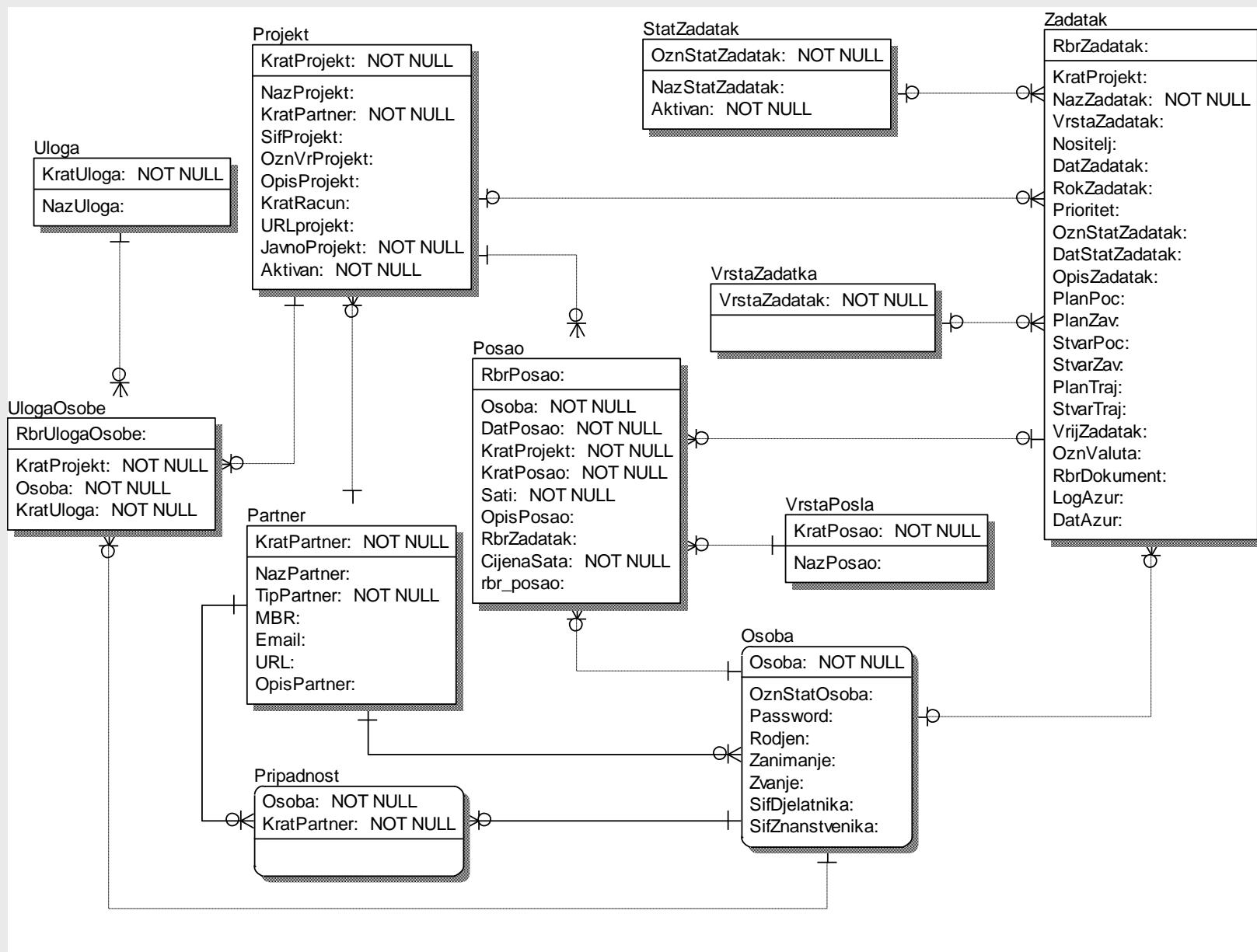
- skup važnih vrijednosti u projektnom planu, kao što je planirani datum početka, završni datum i trošak zadatka, resursa
- primjer (MSP) - plan snimljen u osnovicu plana projekta
- Tools/Tracking/Set Baseline...



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		

Događaji

Trenutno nema nadolazećih događaja. Da biste dodali novi događaj, kliknite "Dodaj novi događaj".

Dodaj novi događaj

Obavijesti

Broj aktivnih taskova i potrebnog vremena

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 ne razvija developer, moguće je obaviti udaljeno debugovanje. Dokumenti/Razvoj (<http://bob.zpr.fer.hr/projekti/PROLOGIS/Debugging.docx>).

[\(Dodatne najave...\)](#)

Dodaj novu najavu

Osoba	DatPosla	KratProjekta	KratPosla	Sati	Opis
-------	----------	--------------	-----------	------	------



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 odgovljačenje 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

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/>

□ Tehnike

- http://www.tutorialspoint.com/management_concepts/

□ Alati

- <http://www.smashingmagazine.com/2008/11/13/15-useful-project-management-tools/>