

Projekt izrade aplikacije

2/13

Projekt

❑ Projekt

- Projekt je vremenski određeno nastojanje da se proizvede jedinstven proizvod, usluga ili rezultat. [PMBOK – Project Management Book of Knowledge, PMI]
- Projekt je niz jedinstvenih, složenih i povezanih aktivnosti koje imaju određeni cilj i koji se mora postići u zadanom vremenskom roku, u okviru zadanog proračuna i u skladu sa specifikacijama. [Wisocky, Beck and Crane]

❑ Vremenska određenost

- Svaki projekt mora imati jasno određen početak i kraj. Projekti mogu biti kratki ili trajati godinama, ali će svakako završiti.
- Projekt završava u trenutku kada postane jasno da su ciljevi projekta dostignuti ili kada se zaključi da ciljevi projekta ne mogu ili neće biti dostignuti.

❑ Jedinstvenost

- Projekt se odnosi na rad na nečemu što prije nije postojalo i što se razlikuje od rezultata nastalih sličnim projektima.

Upravljanje projektom

❑ Upravljanje, rukovođenje projektom (Project management)

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

❑ Upravljanje projektom uključuje

- Planiranje
 - Utvrđivanje zahtjeva
 - Postavljanje jasnih i ostvarivih ciljeva
 - Uravnoteženje zahtjeva na kvalitetu, doseg, vrijeme i trošak,
 - Prilagodbu interesima i očekivanjima zainteresiranih strana – dionika (eng. Stakeholders)
- Organiziranje
 - Formiranje projektnog tima
- Raspoređivanje obaveza
 - Tko što i kada treba napraviti
- Usmjeravanje
 - Nadgledanje, omogućavanje izvršenja
- Kontroliranje
 - Provjera učinka i rezultata

Uloge na projektu

☐ **Korisnik, Korisnik usluga, Klijent (User, Customer, Client)**

- osoba ili grupa, naručilatelj ili krajnji korisnik

☐ **Sponzor projekta (project sponsor)**

- Osoba ili grupa koja osigurava (financijske) resurse za projekt

☐ **Voditelj projekta (project manager)**

- Osoba imenovana kako bi ostvarila ciljeve projekta

☐ **Resursi projekta**

- Osobe, oprema, usluge, materijal, budžet ili druga sredstva.

☐ **Projektna ekipa**

– Svi članovi ekipe, uključujući upravljačke, a u nekim slučajevima i sponzora

- voditelj – upravljanje projektom
- sistem analitičar – određivanje potreba, specifikacija zahtjeva i dizajna
- projektant/arhitekt – uspostava osnovne arhitekture
- razvojniki (developer, builder) – kodiranje, testiranje
- administrator baza podataka – administriranje DBMS
- sistem inženjer / sistem administrator – administriranje OS i mreže

Dokumentiranje projekta

❑ Povelja projekta (Project Charter)

- Dokument kojim pokretač projekta ili sponzor odobrava projekt i ovlašćuje voditelja za primjenu organizacijskih resursa u provedbi projekta.

❑ Plan projekta = Plan upravljanja softverskim projektom

- IEEE Standard for Software Project Management Plans 1058-1998
- dokument koji opisuje sveukupnu organizaciju projekta

❑ Primjeri

- PlanProjekta.dot
- Firma-PlanProjekta.doc

❑ Plan može sadržavati raspored

- SoftwareDevelopment.mpp
- PlanRazvoja.mpp
- Firma-PlanProjekta.mpp

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

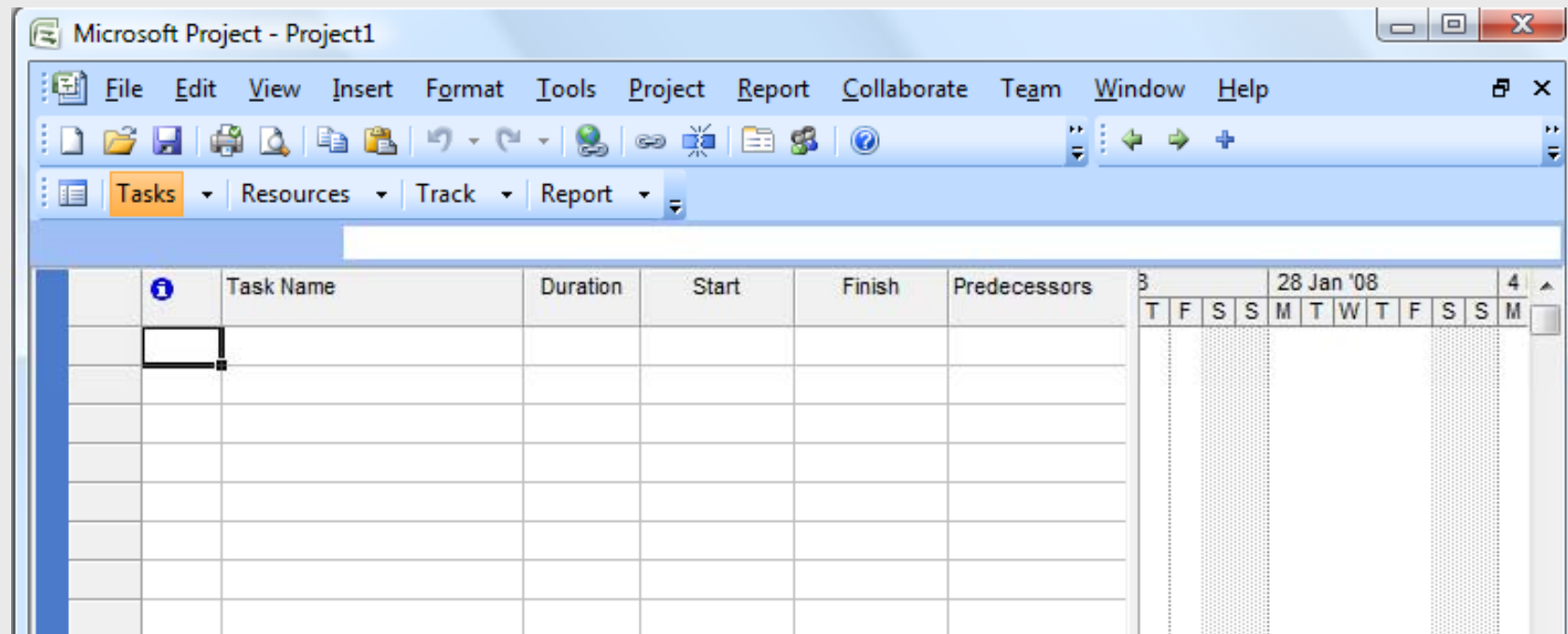
Izrada plana projekta

❑ MS Project - sustav za upravljanje projektima

- klijentski program za mrežno planiranje
- poslužitelj za upravljanje projektom – evidentiranje planova, rizika te dokumentacije

❑ Koraci izrade plana projekta

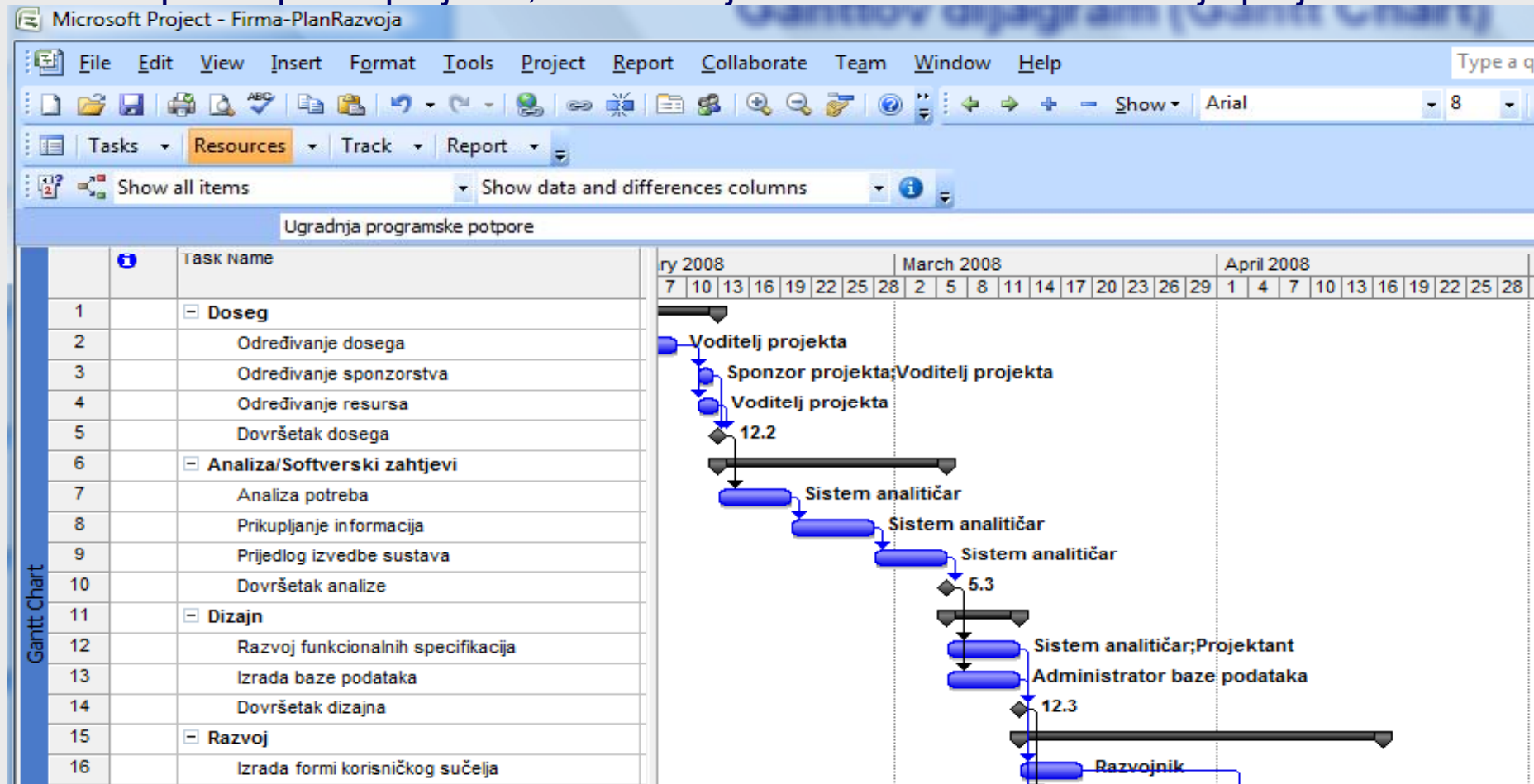
- Izrada liste zadataka
- Izrada hijerarhije zadataka (work breakdown structure)
- Procjena trajanja zadataka
- Izrada ovisnosti među zadacima
- Dodjela resursa



Osnovni pogled na plan projekta

❑ Ganttov dijagram (Gantt Chart)

- tablični prikaz zadatka na lijevoj strani i grafički prikaz na desnoj strani
- prikaz plana projekta, unos i izmjene zadatka te analiziranje projekta

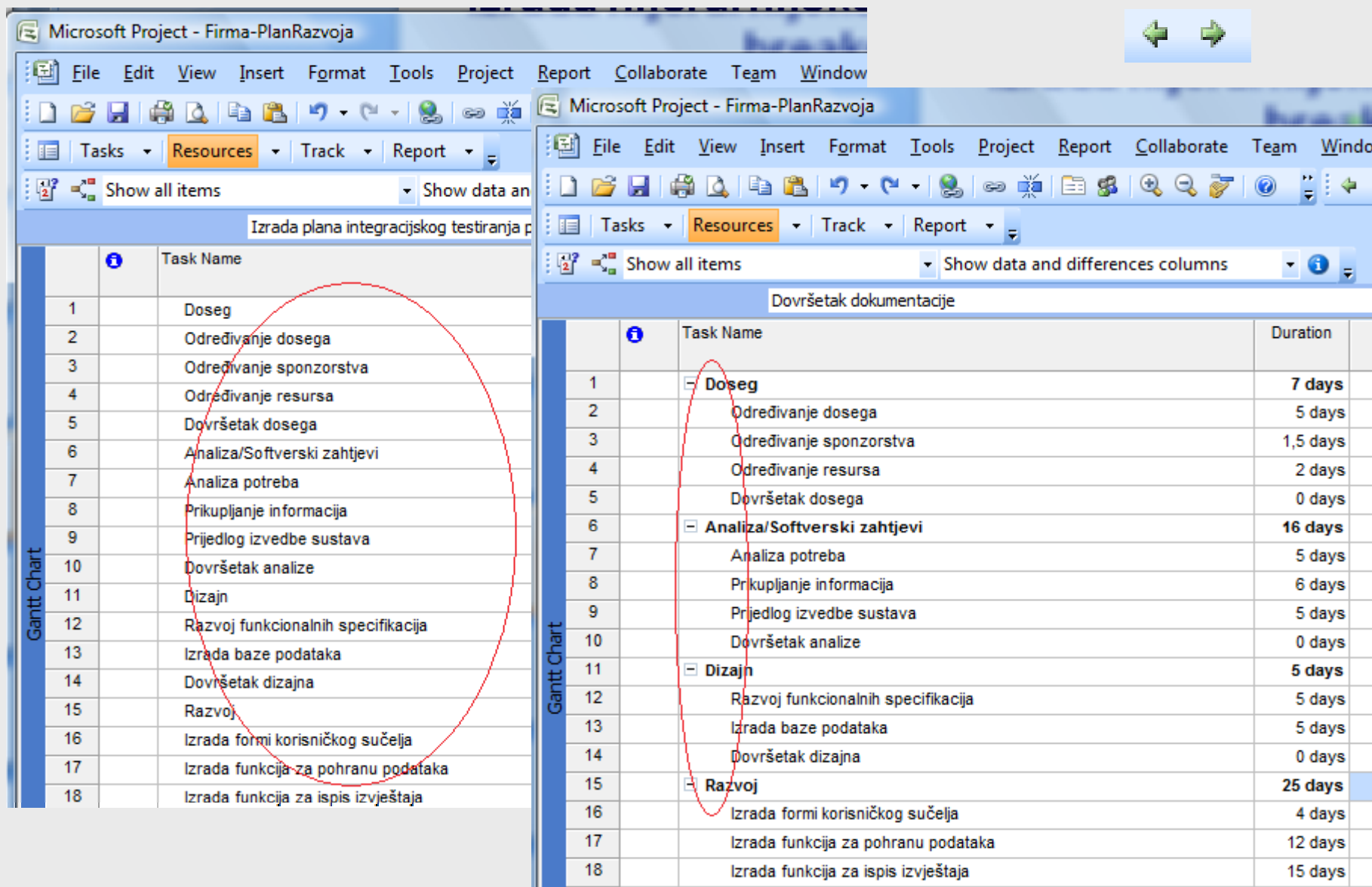


Izrada liste zadataka

❑ Zadaci – osnovni gradbeni elementi svakog projekta

- predstavljaju posao koji se mora obaviti da bi se postigao cilj projekta
- opisuju tijek događaja, trajanja i zahtjeva za resursima na projektu
- primitivni zadaci
 - zadaci koji se dekompozicijom ne mogu podijeliti na jednostavnije zadatke
- skupni zadaci (summary tasks)
 - zbrajaju trajanje i troškove primitivnih zadataka
 - trajanje, datum te izračunate vrijednosti se automatski izvode iz skupa primitivnih zadataka
- prekretnice ili miljokazi (milestones)
 - ključni događaj ili krajnji rok odnosno cilj koji treba postići
 - trajanja 0
 - služe za provjeru stupnja dovršenosti drugih zadataka
 - pomak ključnog događaja ima za posljedicu vremenski preraspored

Lista i hijerarhija zadataka



Izrada hijerarhije zadataka

☐ Faza - grupa povezanih zadataka koji se odnose na fazu projekta

- Zbirni zadaci se odnose na faze

☐ WBS (work breakdown structure)

- hijerarhijska lista faza, zadataka i milijokaza
- osnova za pregledni raspored projekta

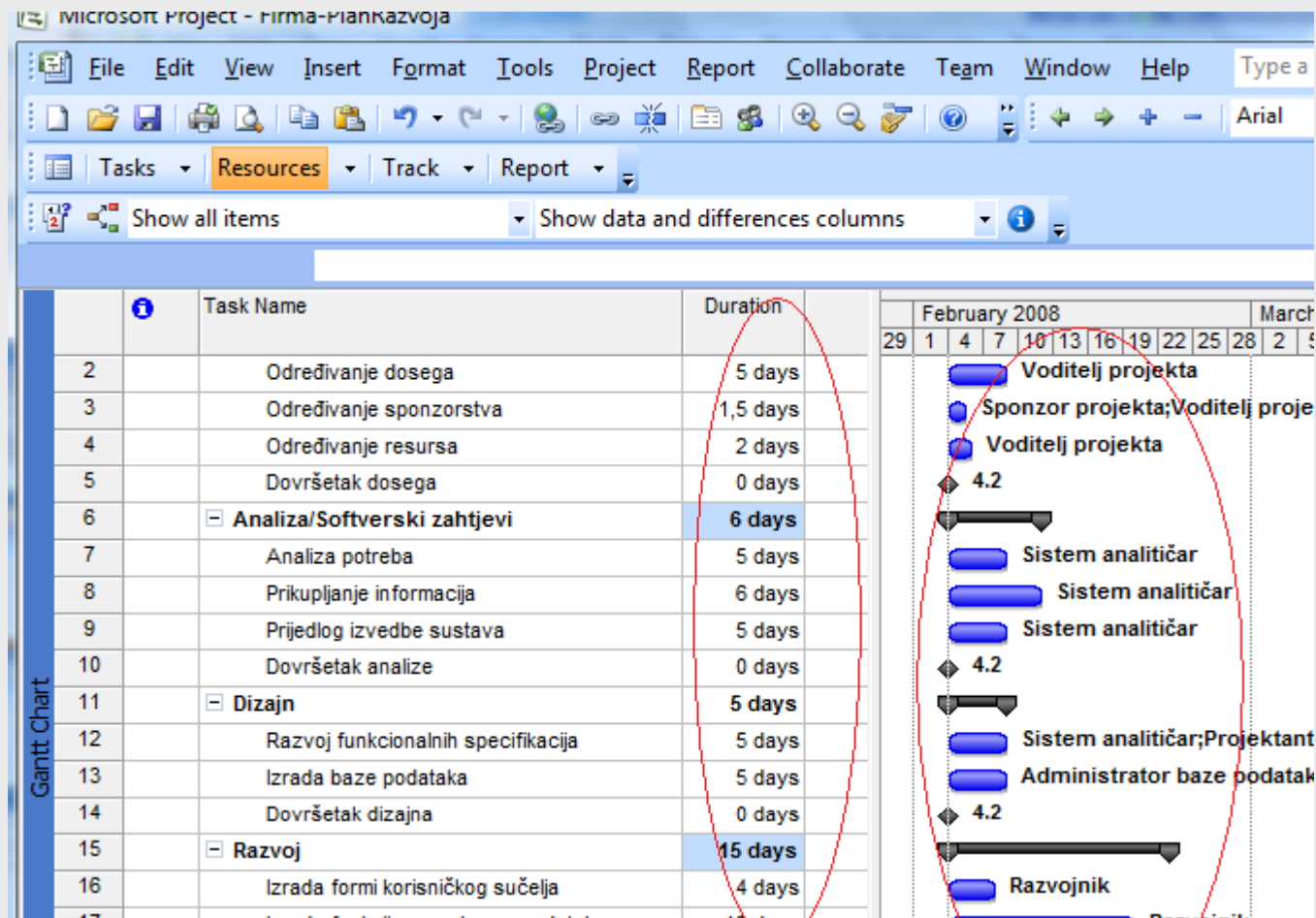
☐ Dva su pristupa razvoju zadataka i faza:

- Planiranje s vrha prema dolje (Top-down)
 - pristup od općeg prema specifičnom
 - identificira glavne faze i rezultate projekta prije dodavanja zadataka potrebnih za završetak tih faza
 - složeni projekti mogu imati nekoliko slojeva faza;
- Planiranje s dna prema dolje (Bottom-up)
 - pristup od specifičnog prema općem
 - identificira što više zadataka najnižeg sloja prije grupiranja u faze

Procjena trajanja zadataka

❑ Trajanje zadatka

- očekivana količina vremena za završetak zadataka
- Tools/Options/Schedule – odabir željene vremenske jedinice



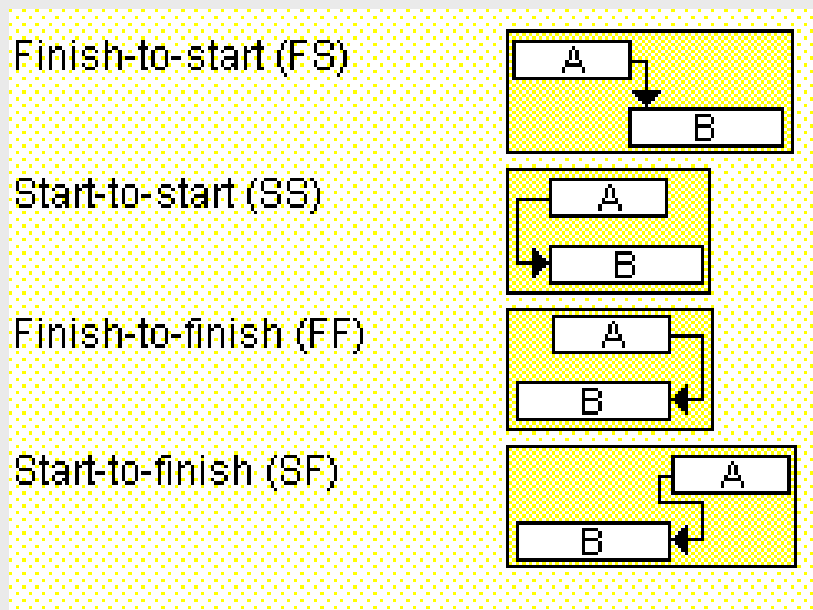
Međuzavisnost zadataka

❑ Projekt može zahtijevati da zadaci budu napravljeni u određenom redoslijedu

- Niz – iza jednog slijedi drugi zadatak
- Zavisnost – sljedbenik (successor) može biti izvršen ako je dovršen prethodnik (predecessor)
- Bilo koji zadatak može biti prethodnik jednom ili više sljedbenika

❑ Odnosi između zadataka:

- Finish-to-start (FS) – završni datum prethodnika jest početni sljedbenika
- Start-to-start (SS) - početni datum prethodnika utvrđuje početni sljedbenika
- Finish-to-finish (FF) - završni datum prethodnika utvrđuje završni sljedbenika
- Start-to-finish (SF) - početni datum prethodnika utvrđuje završni sljedbenika



Definiranje zavisnosti među zadacima

Microsoft Project - Firma-PlanRazvoja

File Edit View Insert Format Tools Project Report Collaborate Team Window Help Type a question for help

Tasks Resources Track Report

Show all items Show data and differences columns

Razvoj funkcionalnih specifikacija

	Task Name	Work	Actual Work	Remaining Work	Predecessors	Resource Names
7	Analiza potreba	40 hrs	0 hrs	40 hrs	5	Sistem analitičar
8	Prikupljanje informacija	48 hrs	0 hrs	48 hrs	7	Sistem analitičar
9	Prijedlog izvedbe sustava	40 hrs	0 hrs	40 hrs	8	Sistem analitičar
10	Dovršetak analize	0 hrs	0 hrs	0 hrs	9	
11	Dizajn	120 hrs	0 hrs	120 hrs		
12	Razvoj funkcionalnih specifikacija	80 hrs	0 hrs	80 hrs	10	Sistem analitičar;Proj
13	Izrada baze podataka	40 hrs	0 hrs	40 hrs	10	Administrator baze p
14	Dovršetak dizajna	0 hrs	0 hrs	0 hrs	13;12	
15	Razvoj	504 hrs	0 hrs	504 hrs		

Name: Razvoj funkcionalnih specifikacija Duration: 5d ☒ Effort driven Previous Next

Start: Thu 6.3.08 Finish: Wed 12.3.08 Task type: Fixed Units % Complete: 0%

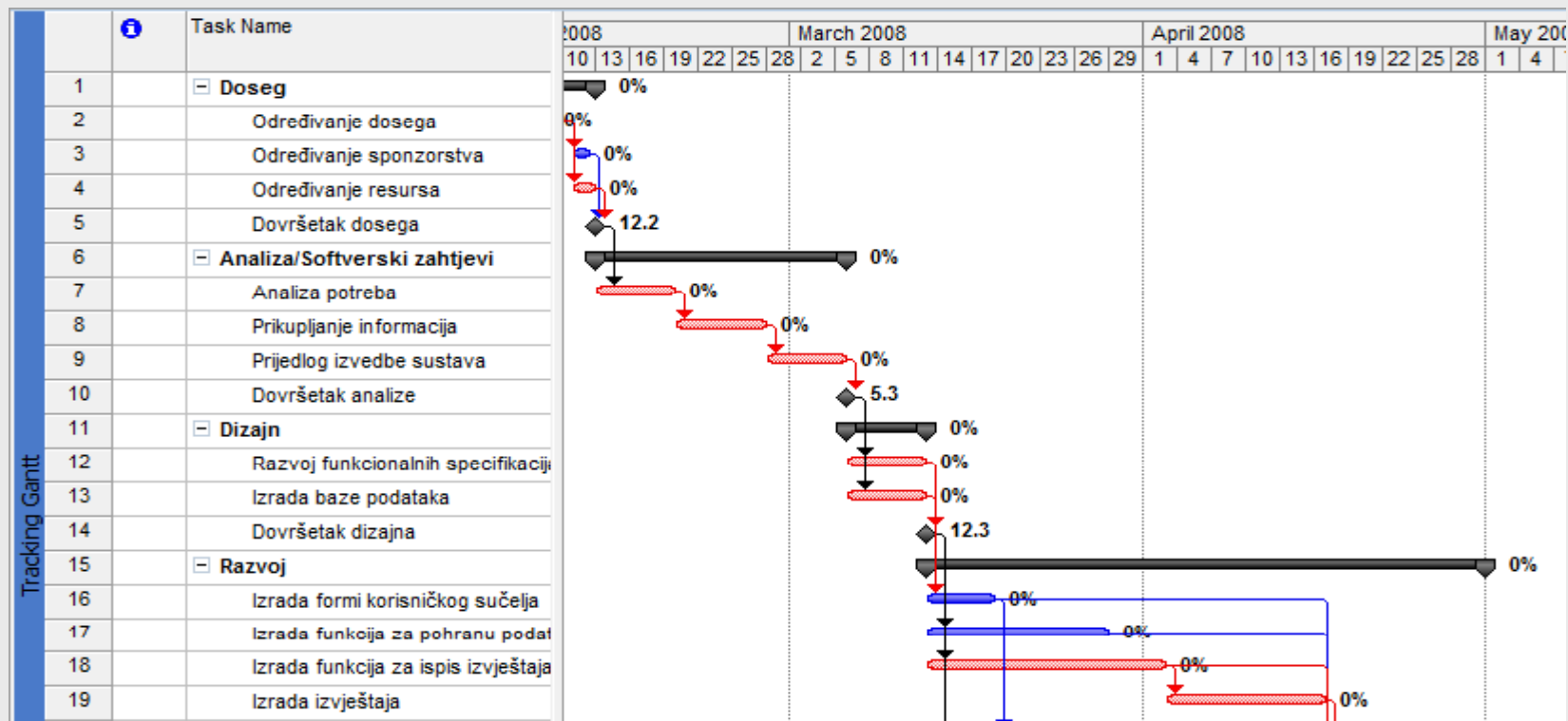
ID	Resource Name	Units	Work	ID	Predecessor Name	Type	Lag
4	Sistem analitičar	100%	40h	10	Dovršetak analize	FS	0d
5	Projektant	100%	40h				

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)



Resursi

❑ Resursi - sredstva

- ljudi, oprema i materijal potrebni za obavljanje zadataka

❑ Vrste resursa:

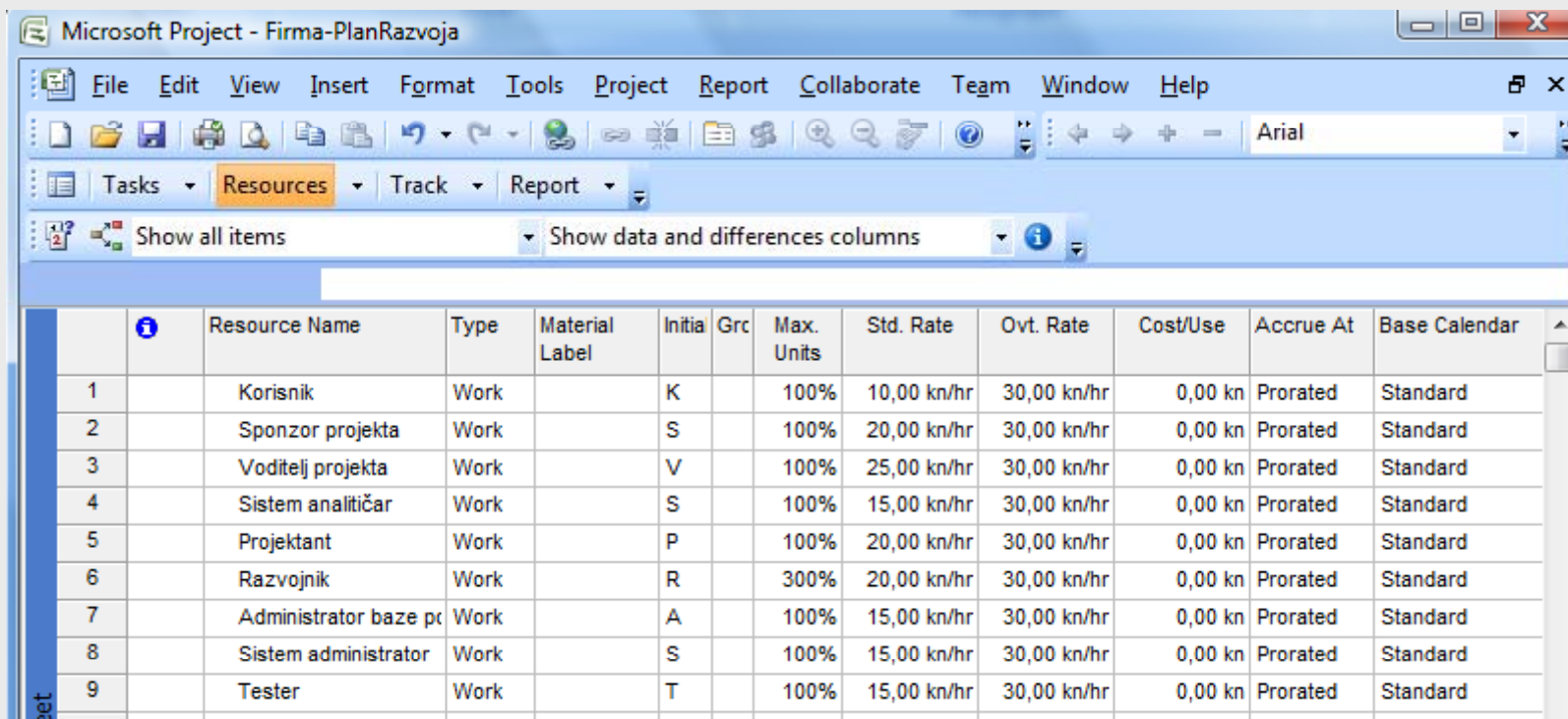
- Resursi rada (work resources)
 - ljudi (ograničeno vrijeme rada)
 - oprema (neograničeno vrijeme rada)
- Resursi materijala (material resources)
 - potrošni materijal koji predstavlja projektni utržak
 - daje informaciju o brzini konzumiranja resursa

❑ Dva važna pogleda na resurse:

- Raspoloživost - u koje vrijeme određeni resurs može raditi na zadatku i koliko posla može obaviti
- Trošak – koliko novca će biti potrošeno na resurse

Raspodjela resursa

- ❑ Unos resursa i pratećih podataka (dostupnost, trošak)
- ❑ Maksimalne jedinice (max. units)
 - prikazuju vrijednosti raspoloživosti resursa u postocima
 - 100% predstavlja jednog čovjeka punog radnog vremena
 - 300% predstavlja tri čovjeka punog radnog vremena



		Resource Name	Type	Material Label	Initial	Grc	Max. Units	Std. Rate	Ovt. Rate	Cost/Use	Accrue At	Base Calendar
1		Korisnik	Work		K		100%	10,00 kn/hr	30,00 kn/hr	0,00 kn	Prorated	Standard
2		Sponzor projekta	Work		S		100%	20,00 kn/hr	30,00 kn/hr	0,00 kn	Prorated	Standard
3		Voditelj projekta	Work		V		100%	25,00 kn/hr	30,00 kn/hr	0,00 kn	Prorated	Standard
4		Sistem analitičar	Work		S		100%	15,00 kn/hr	30,00 kn/hr	0,00 kn	Prorated	Standard
5		Projektant	Work		P		100%	20,00 kn/hr	30,00 kn/hr	0,00 kn	Prorated	Standard
6		Razvojniki	Work		R		300%	20,00 kn/hr	30,00 kn/hr	0,00 kn	Prorated	Standard
7		Administrator baze pc	Work		A		100%	15,00 kn/hr	30,00 kn/hr	0,00 kn	Prorated	Standard
8		Sistem administrator	Work		S		100%	15,00 kn/hr	30,00 kn/hr	0,00 kn	Prorated	Standard
9		Tester	Work		T		100%	15,00 kn/hr	30,00 kn/hr	0,00 kn	Prorated	Standard

Kalendar i dodjela resursa

❑ Kalendar resursa

- osnova po kojoj je resursu dodijeljeno obavljanje posla
- MS Project automatski za svaki resurs kreira (standardni) kalendar resursa
- za pojedinačnu prilagodbu uvažavaju se radni i neradni dani resursa
- Primjer:
 - ako kalendar evidentira radno vrijeme četvrtka i petka od 13-17 sati, 100% raspoloživosti nekog resursa ne znači 40 satno tjedno radno vrijeme, nego 8 satno tjedno radno vrijeme

❑ Dodjelom resursa zadatku, dodjeljuje mu se određeni posao

- Posao – količina utroška resursa za završetak zadatka
- Posao = Trajanje * Jedinice (Work = Duration * Units)

❑ Raspoređivanje temeljem napora (effort-driven scheduling)

- metoda planiranja koja se koristi kod dodavanja ili brisanja nekog resursa sa zadatka
- početni rad zadatka ostaje konstantan bez obzira koliko se dodatnih resursa dodijeli

Raspored resursa

Microsoft Project - Firma-PlanRazvoja

File Edit View Insert Format Tools Proj

Tasks Resources Track Report

Show all items Show data

Voditelj projekta

Task Name

1 Doseg

2 Određivanje dosega

3 Određivanje sponzorstva

4 Određivanje resursa

5 Dovršetak dosega

6 Analiza/Softverski zahtjevi

7 Analiza potreba

8 Prikupljanje informacija

1,5 days Mon 11.2.08 Tue 12.2.08 2

2 days Mon 11.2.08 Tue 12.2.08 2

0 days Tue 12.2.08 Tue 12.2.08 3;4

26 days Wed 13.2.08 Wed 19.3.08

5 days Wed 13.2.08 Tue 19.2.08 5

6 days Wed 20.2.08 Wed 27.2.08 7

Sponzor projekta: Vo

Voditelj projekta

0%

Name: Određivanje resursa Duration: 2d Effort driven Previous Next

Start: Mon 11.2.08 Finish: Tue 12.2.08 Task type: Fixed Units % Complete: 0%

ID	Resource Name	Units	Work	ID	Predecessor Name	Type	Lag
3	Voditelj projekta	100%	16h	2	Određivanje dosega	FS	0d

Change Working Time

Resource calendar for 'Sistem administrator':

Base calendar: Standard

Legend:

- Working
- Nonworking
- Edited working hours
- Exception day
- Nondefault work week

Working times for 5 February 2008:

- 8:00 to 12:00
- 13:00 to 17:00

Based on: Default work week on calendar 'Standard'.

Exceptions

Name	Start	Finish

Work Weeks

January 2008

M	T	W	Th	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29		

Details... Delete



Zadaci

☐ Projektirati i izraditi sustav za upravljanje projektima

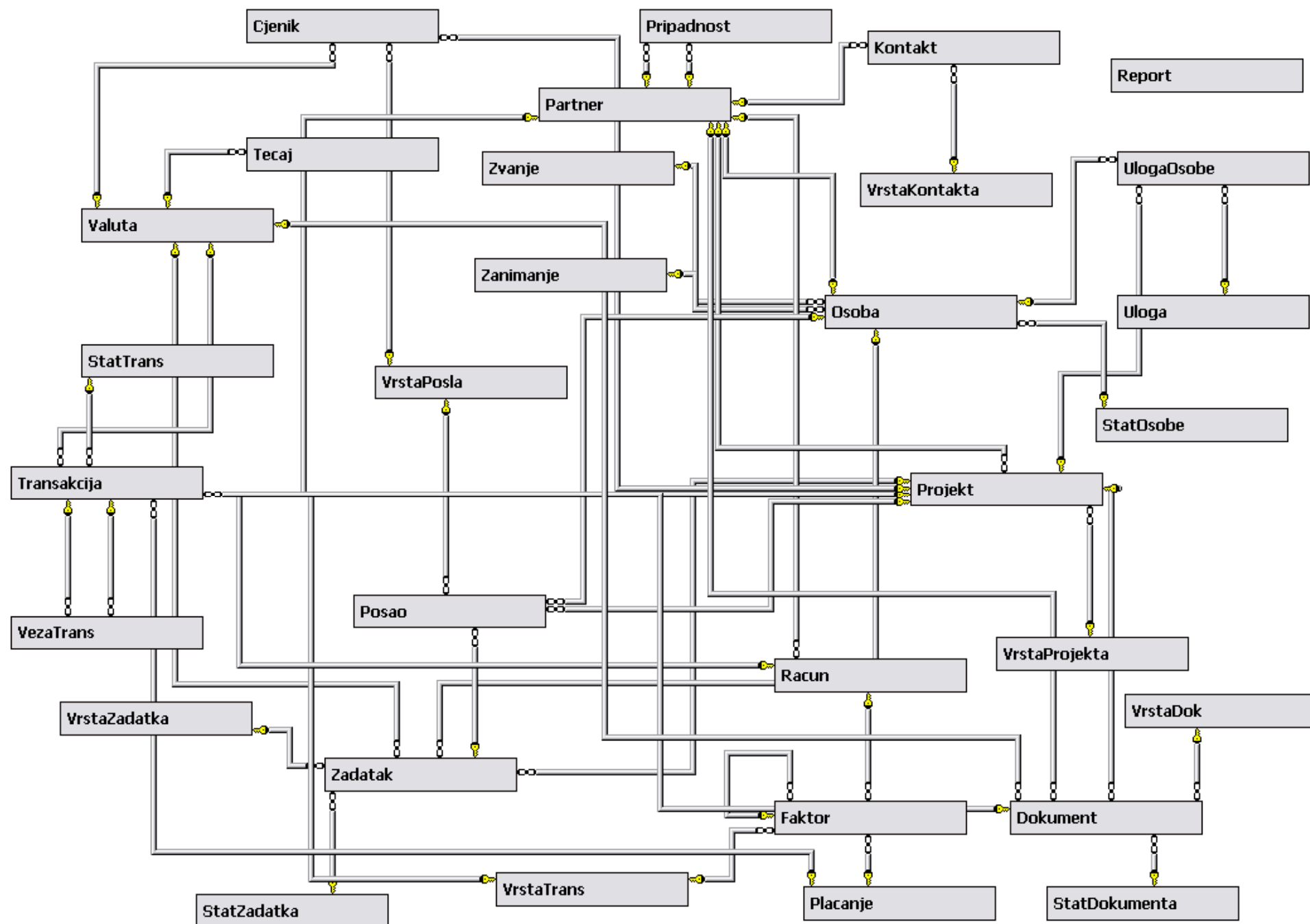
- Kao rezultat rada na projektu svaki član mora napraviti po jednu smislenu zaslonsku masku na Windows, WWW i PPC platformi.
- Integraciju specifikacije posla, zajedničkih programskih knjižnica te ostale projektne dokumentacije radi ekipa zajednički.

☐ Napraviti početni plan i raspored projekta

- Kakav sustav ? Kako dobar sustav ? Kojim redom ? Kojim resursima ?

☐ Domaća zadaća: Dokumentirati početni plan projekta

- Uspostaviti životni ciklus razvoja na razini ekipe
- Planirati faze razvoja
- Popisati nositelje zadataka
- Odrediti prekretnice s obzirom na realno stanje
- PlanProjekta.doc i PlanProjekta.mpp - zajednički za ekipu



Integrirana razvojna okruženja

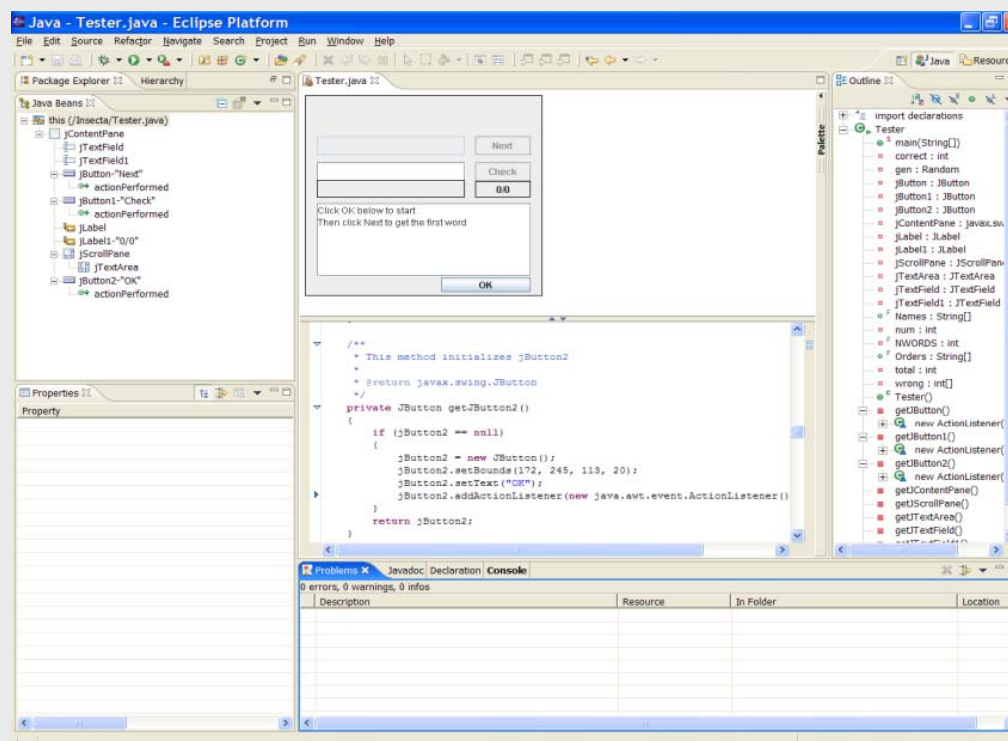
Integrirana razvojna okruženja

❑ Integrated Development Environment (IDE) – okruženje, okolina za programiranje koje integrira

- uređivač izvornog koda (source code editor)
- kompilator (compiler) i/ili interpreter (interpreter)
- alat za izradu grafičkog korisničkog sučelja (GUI)
- sustav za ispravljanje pogrešaka (debugger)
- pomagalo za kontrolu verzija (version control system)
- alate za automatsku izgradnju (build-automation tools)
- alate za objektno orijentirano programiranje (npr. Class Browser)
- itd

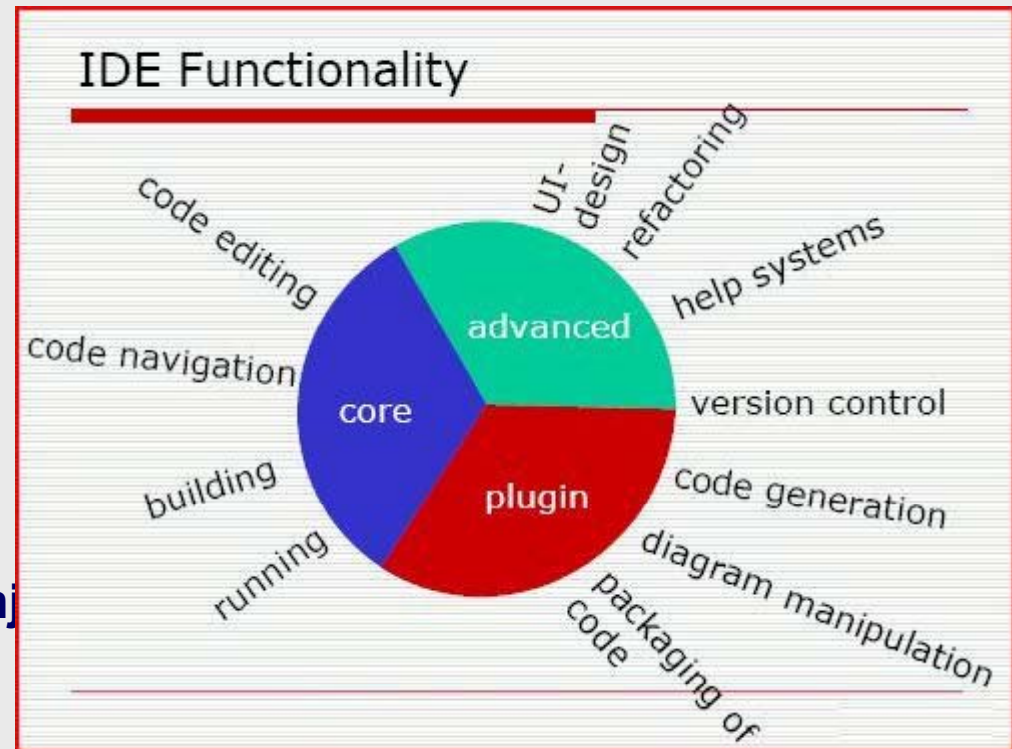
Primjeri IDE

- ❑ **Prvotno namijenjenj jednom programskom jeziku:**
 - Visual Basic, Delphi, Jbuilder....
- ❑ **Moderna okruženja podržavaju više jezika**
 - Visual Studio .NET : C++, C#, Visual Basic.NET, ASP.NET
 - Eclipse - Eclipse primarno okruženje za Javu, ali sadrži umetke (plugins)
 - C/C++, Python, Perl, Ruby, Fortran, Cobol i dr.



Funkcionalnost i namjena IDE

- ❑ Označavanje sintakse (eng. syntax highlighting)
- ❑ Označavanje pogrešaka (eng. error highlighting)
- ❑ Pristup dokumentaciji (eng. documentation access)
- ❑ Kretanje kodom (eng. code navigation)
- ❑ Generiranje koda pomoću predložaka koda (eng. code generation through code templates)
- ❑ Podrška za preradu, refaktoriranje (eng. *refactoring*)
- ❑ Različite razine analize koda



Visual Studio .NET

☐ Visual Studio.NET je IDE za razvoj

- konzolnih aplikacija, desktop aplikacija, web aplikacija, web servisa, ...
- na platformi .NET Framework platformi (Windows PC, PocketPC, ...)

☐ Podržani jezici

- C++, C#, VB.NET, J#, ...

☐ Mogućnost dogradnje dodatnih razvojnih alata ...

☐ Verzije

- trenutna: Visual Studio .NET (2005) + .NET Framework 2.0
- uskoro: Visual Studio .NET (2008) + .NET Framework 3.5

☐ Izdanja (edition)

- Visual Studio Standard Edition
- Visual Studio Professional Edition
- Visual Studio Team System

Primjena VS.NET na izradu aplikacija

☐ Konzolne aplikacije

- pokreću se s komandne linije i ne uključuju grafičko sučelje

☐ Windows desktop aplikacije

- aplikacije s prozorima (forms)
- Upotrebljavaju .NET Framework, koji trebaju za izvođenje.

☐ Windows servisi

- su aplikacije koje rade u pozadini.
- izvode programirane zadatke (scheduled tasks) ili obrađuju zahtjeve (npr. s mreže)

☐ ASP.NET aplikacije

- dinamičke web aplikacije, također podržane .NET Frameworkom.

☐ ASP.NET web servisi

- potprogrami na webu

☐ Windows mobilne aplikacije

- izvode se na uređajima koji imaju Compact framework (Pocket PC, Smartphone).

☐ Ostale aplikacije

- MFC/ATL/Win32 aplikacije – u C++-u , ne trebaju .NET Framework.
- Visual Studio dodaci (add-ins)
- projekti za instalaciju aplikacija, rad sa bazama, kreiranje izvještaja, ...

Mogućnosti okoline Visual Studio (features)

❑ IntelliSense

- pomoć uređivanju izvornog koda, predlaganje elemenata naredbi

❑ Dizajnerski pogled (WSYIWYG)

- oblikovanje korisničkog sučelja za Windows aplikacije, ASP.NET aplikacije i Windows mobilne aplikacije

❑ Otklanjanje pogrešaka (Debugging)

- prolaz naredbu po naredbu tijekom izvođenja
- pregled i promjena sadržaja varijabli (auto, local) i praćenje izraza (watch)
- moguće ulaženje u pozvane procedure

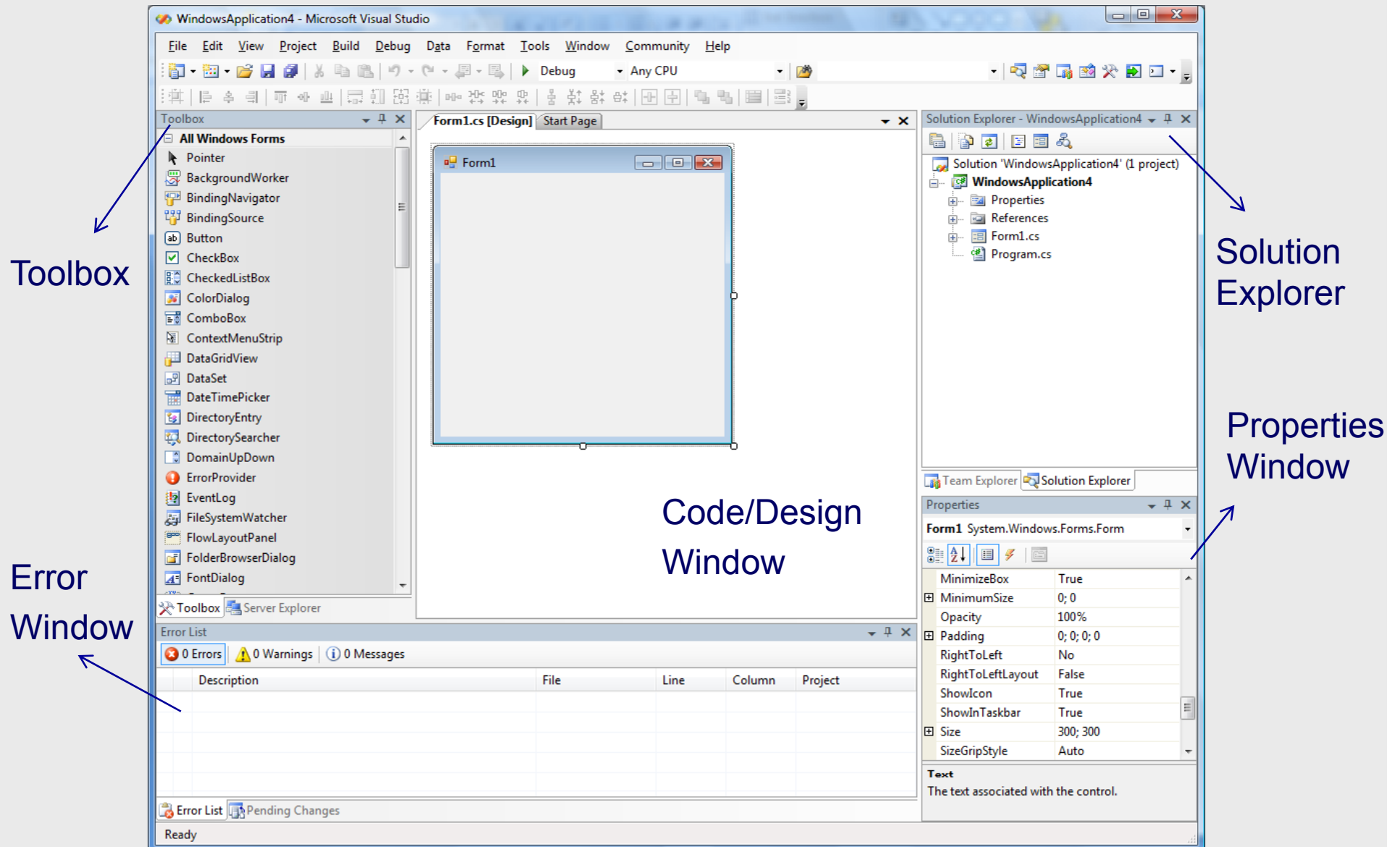
❑ Interaktivna pomoć (Help)

- integrirana s elektroničkom knjižnicom hipertekstova MS Developer Network
- pregled svih elemenata jezika i platforme te elementi priručnika

❑ Organizacija programskog koda

- Project – skup datoteka koje predstavljaju istu aplikaciju ili objektnu datoteku dobivenu prevođenjem (assembly)
- Solution – skup projekata koji čine kompletno poslovno rješenje

Sučelje razvojne okoline Visual Studio



Sučelje razvojne okoline Visual Studio

☐ **Code/Design Window**

- Prozor u kojem se piše kod ili dodaju kontrole u grafičkom (design) pogledu.

☐ **Solution Explorer**

- Prozor s hijerarhijskom strukturom izvornih i pratećih datoteka aplikacije (projekti, datoteke, reference, ...)

☐ **Team Explorer**

- Prozor za rad s rješenjima za timski rad, ukoliko je instaliran klijent za TFS.

☐ **Toolbox**

- Prozor s kontrolama koje se metodom drag&drop stavljaju na grafički pogled.

☐ **Error Window**

- Prozor u kojem se ispisuju poruke o pogreškama i upozorenja prilikom prevođenja programa.

☐ **Locals**

- Prozor koji je vidljiv prilikom pokretanja programa. Služi za praćenje stanja varijabli tijekom izvođenja programa.

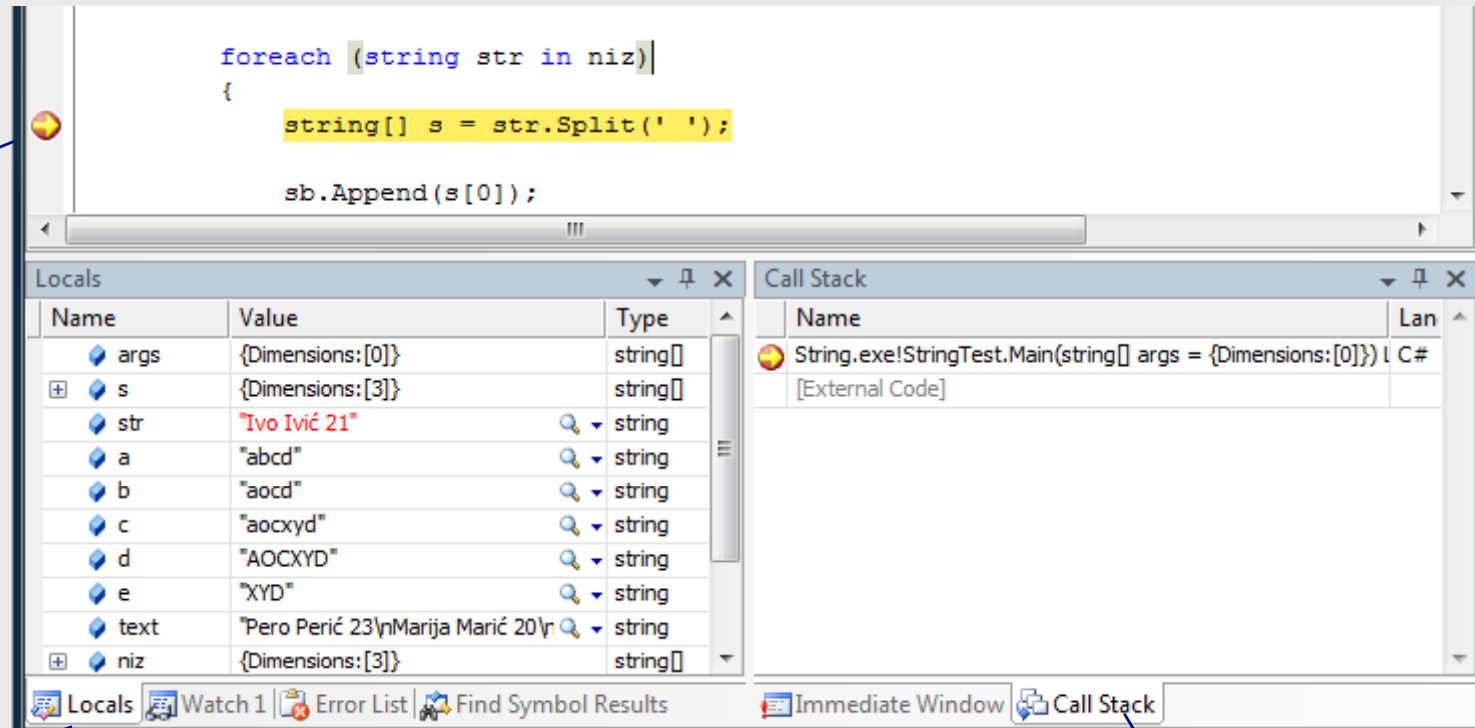
☐ **Properties Window**

- Prozor za podešavanje svojstava (properties) pojedinog objekta.

☐ **Iz izbornika View mogu se dodati razni drugi prozori.**

Praćenje rada aplikacije

Breakpoint
– Točka
prekida



Locals – praćenje
vrijednosti varijabli

CallStack - Praćenje
poziva postupaka

- ☐ Watch – nadzor vrijednosti odabranih varijabli
- ☐ ImmediateWindow – izvršavanje naredbi u pogonu
- ☐ StepInto, StepOver (Debug)

Zadatak za vježbu

- ❑ **Dijelovi razvojne okoline mogu biti različito razmješteni, mogu plutati ili biti usidreni.**
 - Razmjestiti pojedine elemente
 - Pokazati/sakriti neke dijelove
 - Provjeriti dijelove izbornika View
 - Provjeriti postavke u Tools / Options
 - Resetirati razvojnu okolinu na tvorničke postavke.

- ❑ **Visual Studio 2005 Guided Tour**
 - http://msdn.microsoft.com/vstudio/tour/vs2005_guided_tour/default.htm

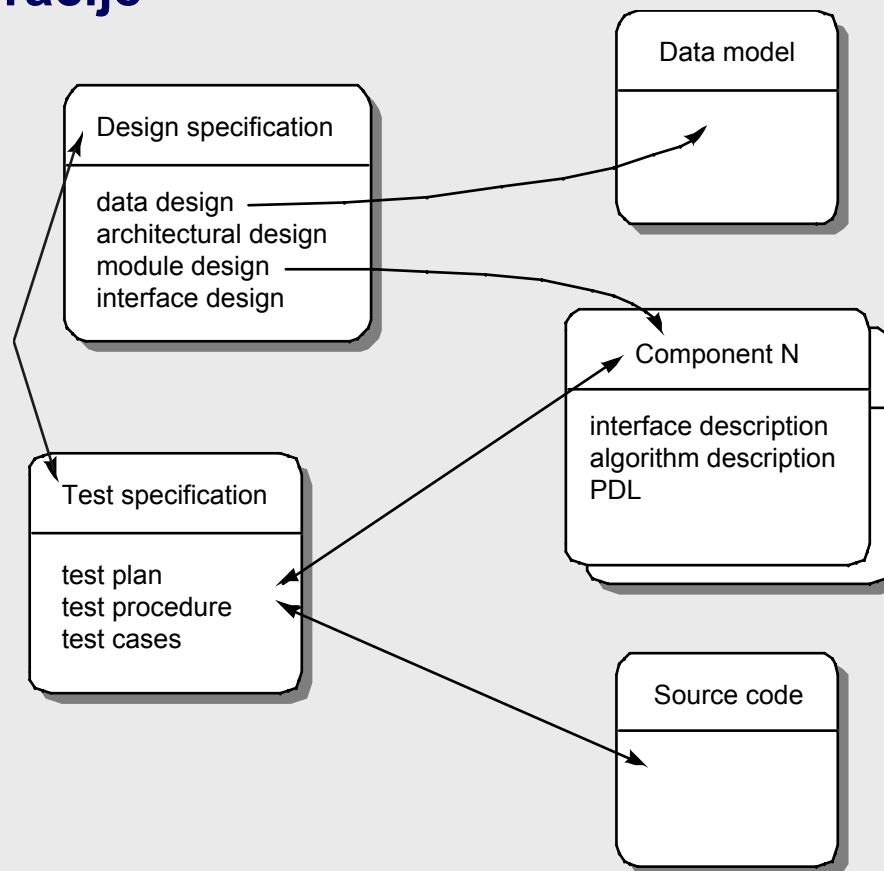
Kontrola programskog koda

Upravljanje konfiguracijom

❑ Element konfiguracije (IEEE)

- agregacija hardvera i/ili softvera koja se tretira kao jedinka u procesu upravljanja konfiguracijom

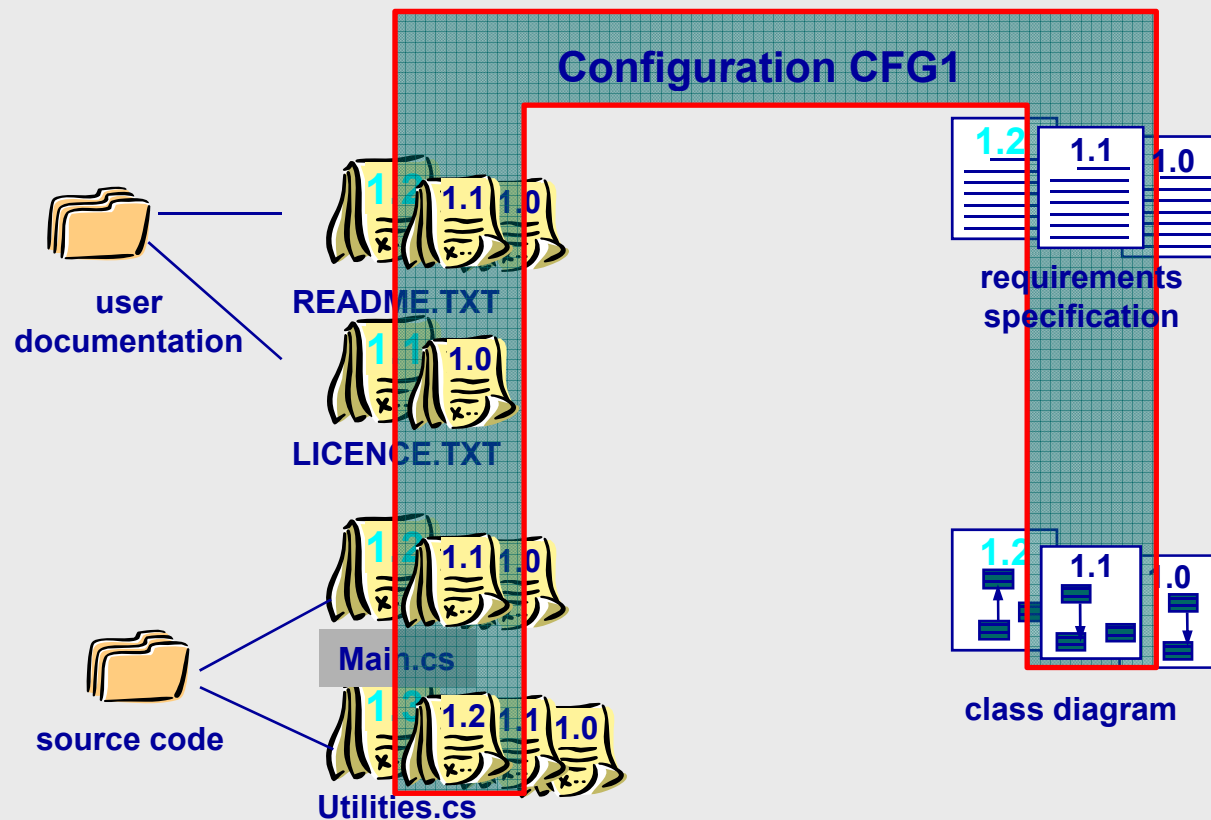
❑ Objekti konfiguracije



Upravljanje konfiguracijom

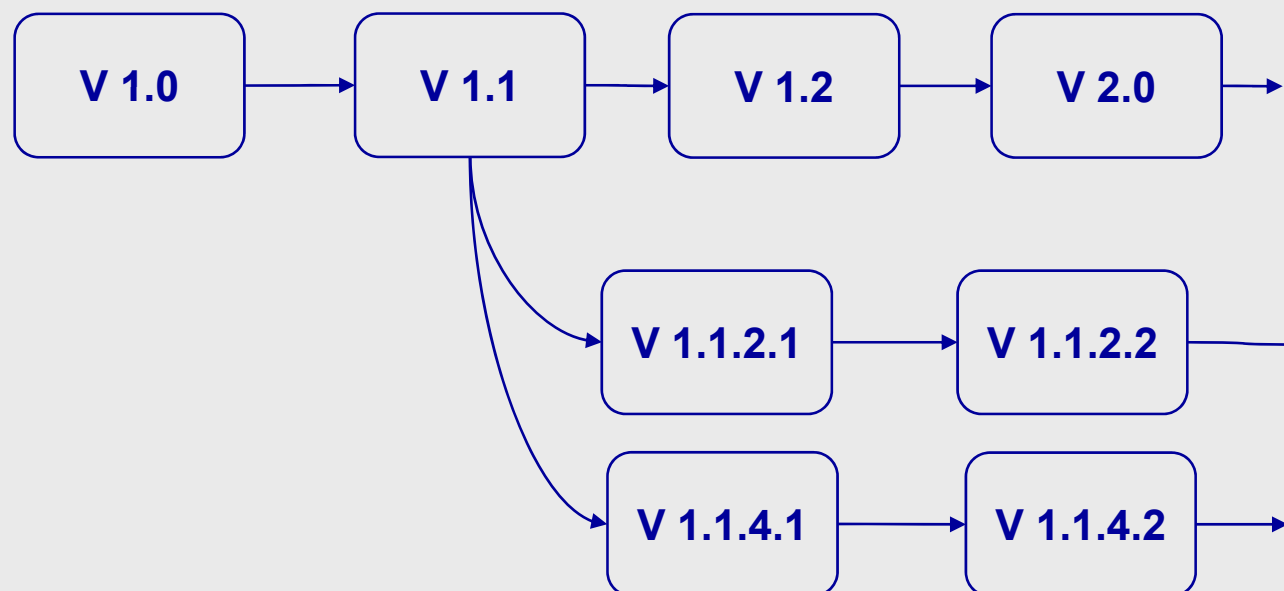
□ Konfiguracija

- imenovani skup konfiguracijskih elemenata u određenoj točki životnog ciklusa



Verzije konfiguracije

- verzija, inačica (version) – određeno izdanje (issue, release) proizvoda
- objava, isporuka (release) – originalna verzija u primjeni, npr. zadnja v2.0
- revizija (revision) – ona koja se koristi umjesto originalne, podrazumijeva izmjene u određenim vremenskim intervalima, npr. V1.2
- varijanta (variant) – alternativa originalu (hardverska platforma, različiti jezik), živi paralelno s njim, npr. v1.1.2.1
- osnovica (Baseline) – specifikacija proizvoda formalno provjerena i usvojena, koja služi kao temelj razvoja i koja se mijenja samo kroz formalnu proceduru kontrole promjena, IEEE (IEEE Std. No. 610.12-1990)



Kontrola verzija

❑ Kontrola verzija (Version control) = verzioniranje

- kombinira procedure i alate radi upravljanja različitim verzijama objekata konfiguracije, koji nastaju softverskim procesima

❑ Mogućnosti sustava kontrole verzija

- baza projekata (project database) ili riznica (repository)
 - pohranjuje sve relevantne objekte konfiguracije
- verzioniranje
 - razlikovanje pohranjenih inačica objekata konfiguracije
- pomagalo za izradu (make facility)
 - prikuplja relevantne objekte i proizvodi određenu verziju softvera
- praćenje problema (issues tracking), praćenje bugova (bug tracking)
 - bilježenje i praćenje statusa tema koje se odnose na pojedine objekte konfiguracije

Označavanje konfiguracije

❑ Verzija objektne datoteke (assembly) određena je s četiri broja:

`<major version>.<minor version>.<build number>.<revision>`

- major version - mijenja se prilikom znatne promjene u (npr. kod redizajna koji prekida vertikalnu kompatibilnost sa starijim verzijama)
- minor version - mijenja se prilikom znatne promjene, ali uz zadržavanje kompatibilnosti s prethodnim verzijama
- build number - predstavlja ponovno prevođenje istog koda (npr. prilikom promjene platforme, procesora i slično)
- revision - primjenjuje se npr. prilikom izdavanja sigurnosnih zakrpa i sličnih manjih promjena

Automatsko i ručno verzioniranje

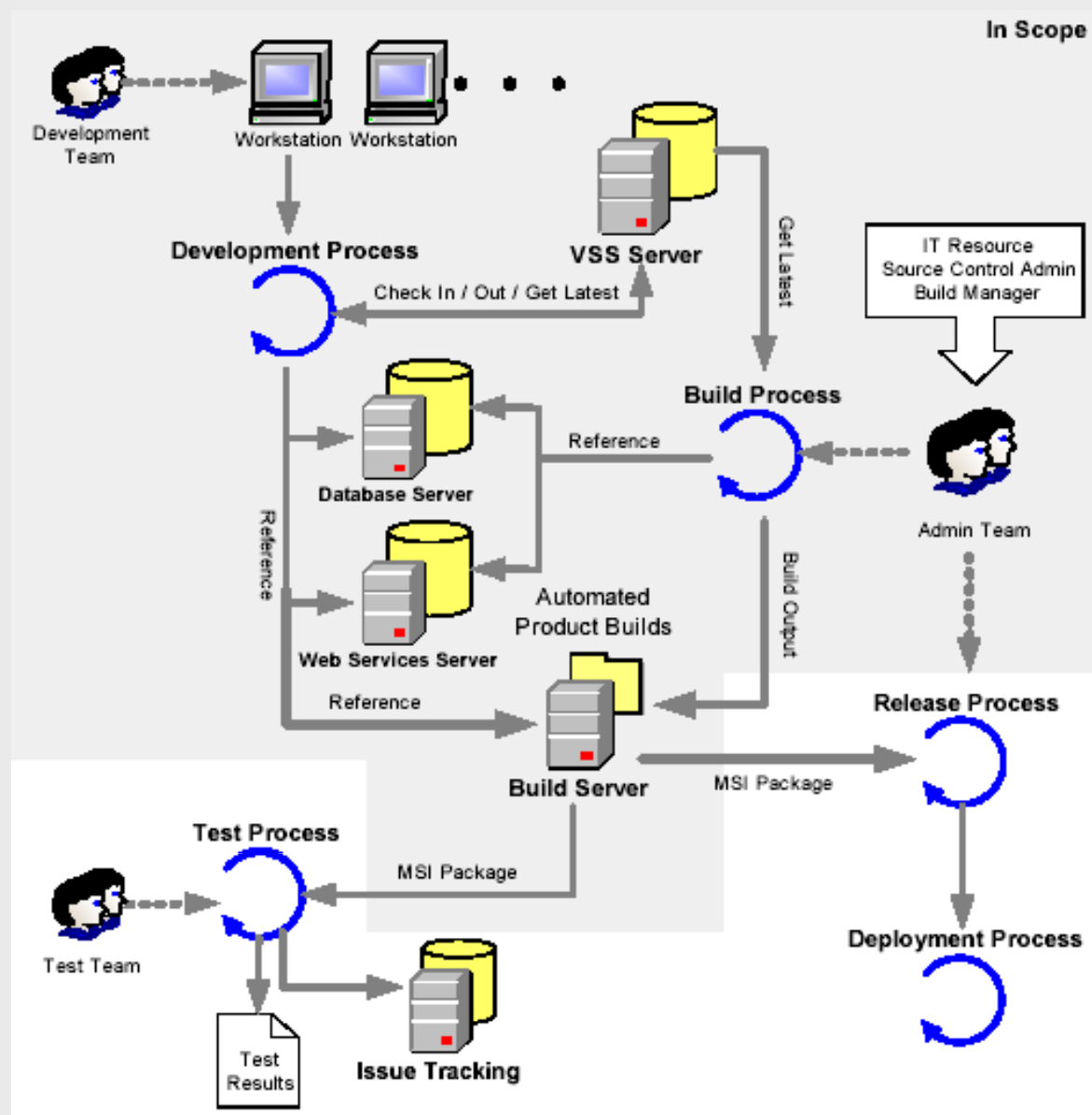
❑ Automatsko označavanje

- prednosti:
 - eliminacija ručnog rada (npr. pisanja i izvedbe skripti)
 - ne postoje dvije inačice s istom oznakom
- nedostaci:
 - oznaka elementa ne podudara se s oznakom cijelog sustava
 - novi brojevi ovise o danu i vremenu prevođenja
 - verzija se mijenja pri svakom prevođenju, neovisno o tome jesu li se dogodile promjene ili ne

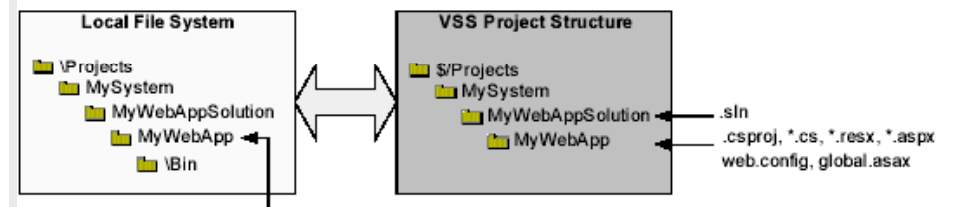
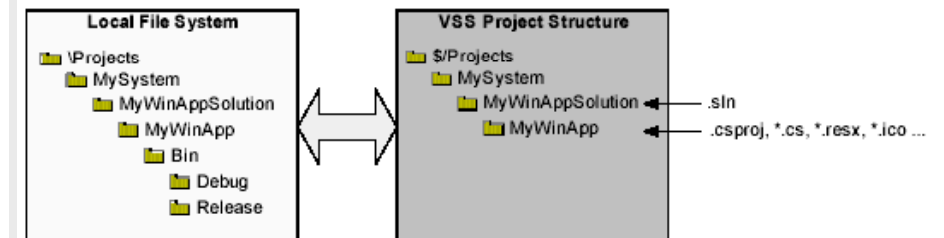
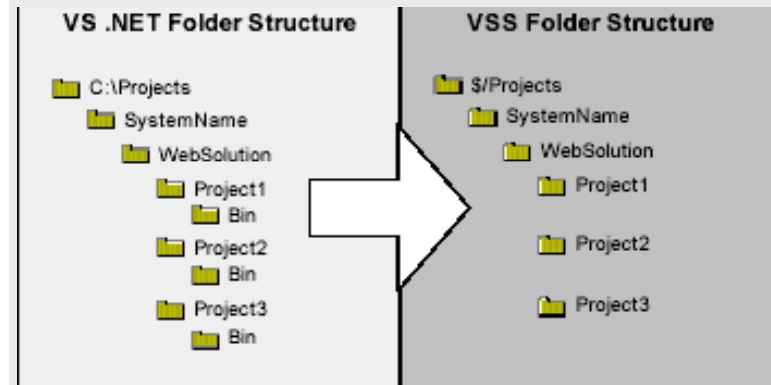
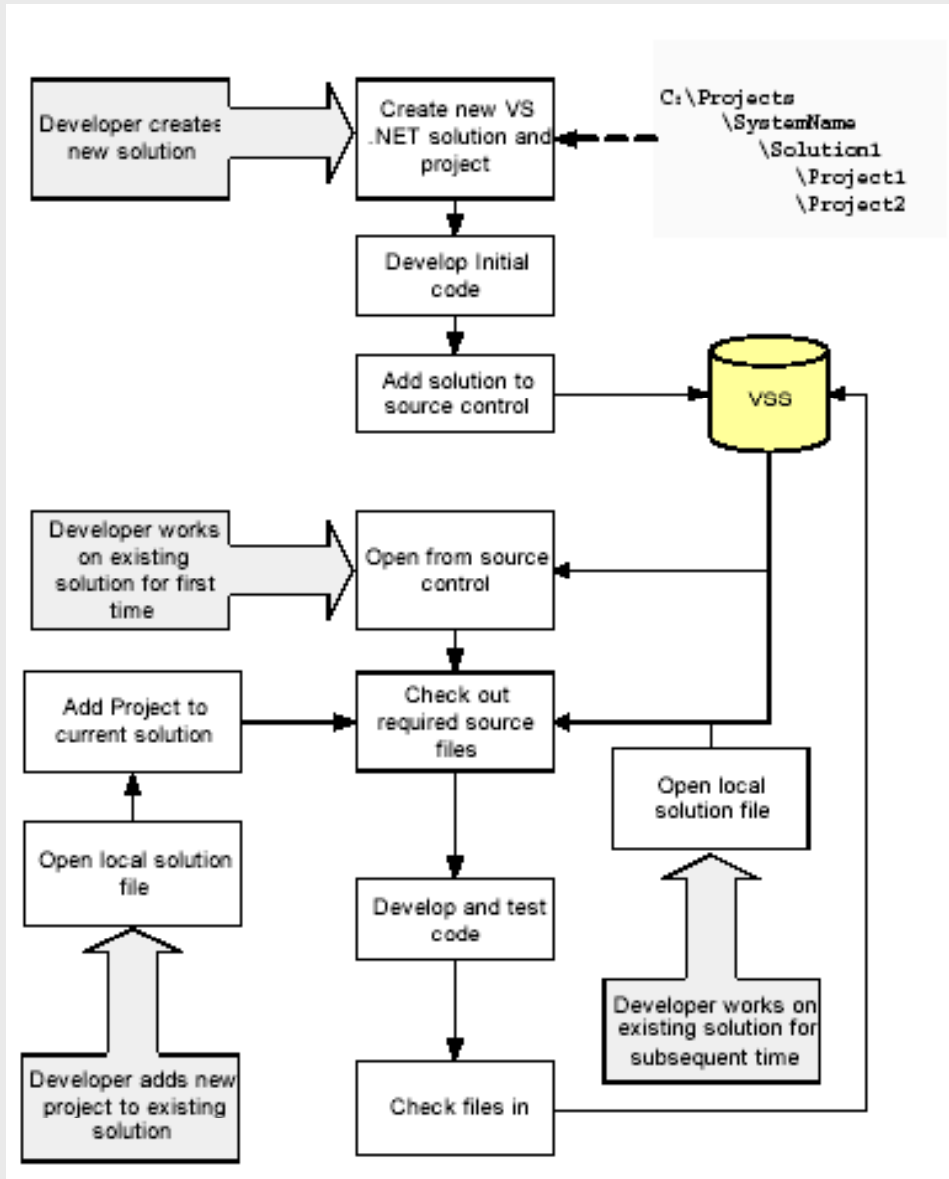
❑ Ručno verzioniranje

- prednosti:
 - potpuna kontrola nad brojevima verzije
 - moguća je sinkronizacija između verzije pojedinih komponenti i verzije cijelog sustava
- nedostaci:
 - verzioniranje se mora raditi ručno
 - moguće je napraviti više različitih objektnih datoteka s istim oznakama

Kontrola verzija (version control)

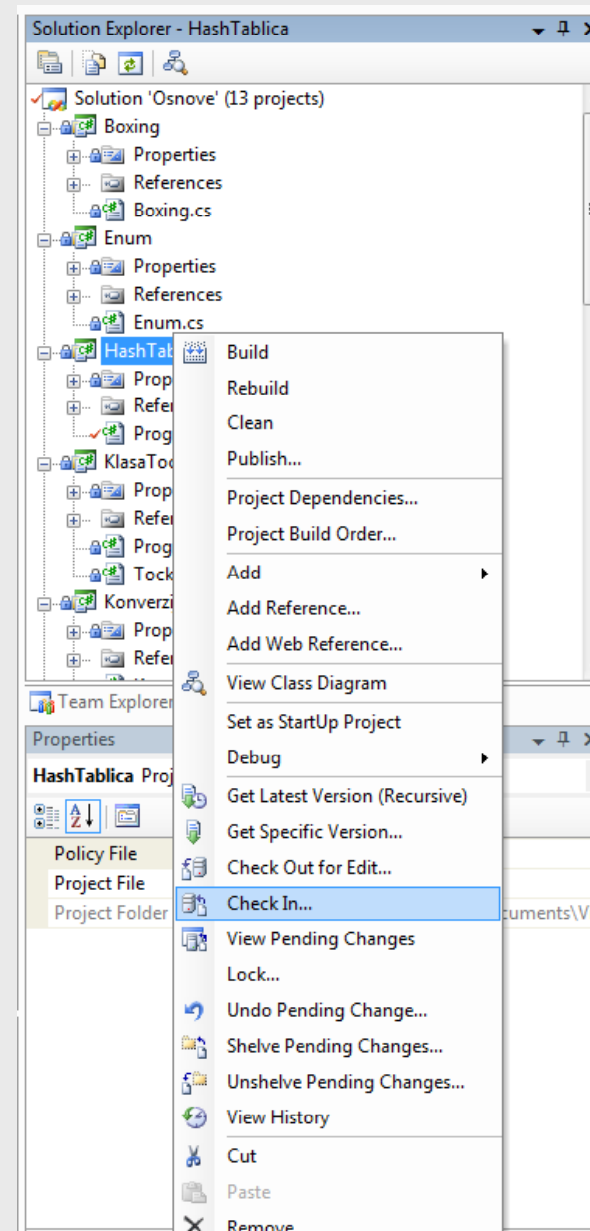


Kontrola verzija

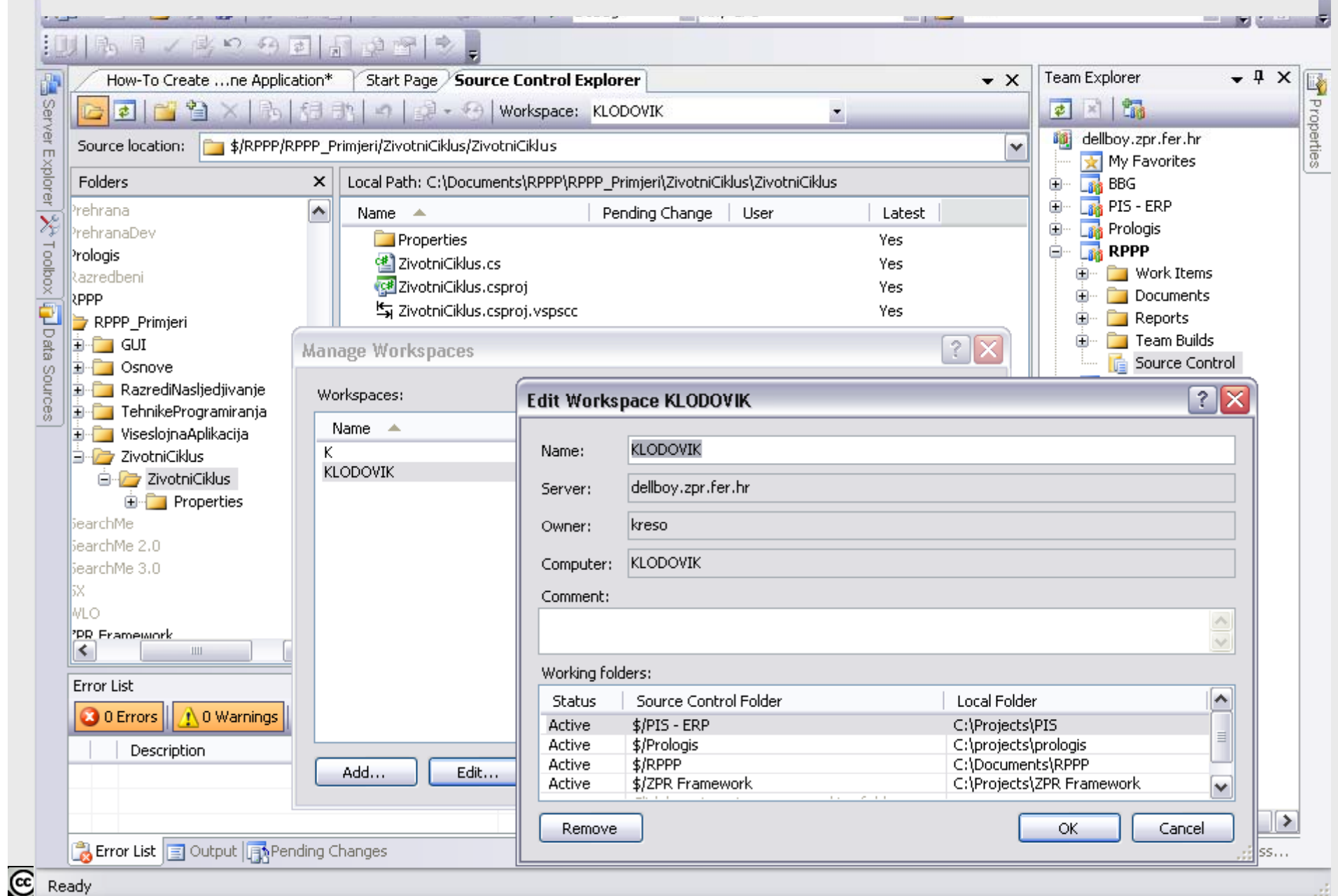


Timski razvoj programa

- ☐ Dodavanje datoteke u SourceControl
 - ☐ SolutionExplorer - Desni klik na projekt - Add to SourceControl (pogledati detaljnije u uputama)
- ☐ CheckIn
 - ☐ *upload* datoteke na poslužitelj
 - ☐ otključavanje drugim korisnicima
- ☐ CheckOut for Edit ...
 - ☐ *download* s poslužitelja
 - ☐ zaključavanje drugim korisnicima
- ☐ Undo Pending Changes
 - ☐ otkazivanje izmjena uz otključavanje
 - ☐ povratak na prethodnu verziju
- ☐ View History
 - ☐ pregled povijesti verzija



Timski razvoj programa



Kontrola izvornog koda

The screenshot displays the Visual Studio Source Control Explorer interface. The main window shows the 'Source Control Explorer' tab with the workspace 'KLODOVIK'. The source location is set to '\$/RPPP/RPPP_Primjeri/ZivotniCiklus/ZivotniCiklus'. The left pane shows a tree view of folders, including 'RPPP_Primjeri' and its subfolders. The right pane shows a table of files with columns for Name, Pending Change, User, and Latest. A context menu is open over the file 'ZivotniCiklus.cs', listing various actions such as 'View', 'Get Latest Version', 'Check Out for Edit...', 'Delete', 'Rename', 'Undo Pending Changes...', 'Check In Pending Changes...', 'Shelve Pending Changes...', 'View History', 'Compare...', 'Branch...', 'Merge...', 'Move...', 'Apply Label...', 'Properties...', and 'Refresh'.

Source location: `$/RPPP/RPPP_Primjeri/ZivotniCiklus/ZivotniCiklus`

Local Path: `C:\Documents\RPPP\RPPP_Primjeri\ZivotniCiklus\ZivotniCiklus`

Name	Pending Change	User	Latest
Properties			Yes
ZivotniCiklus.cs			Yes
ZivotniCiklus			Yes
ZivotniCiklus			Yes

History - `C:\Documents\RPPP\RPPP_Primjeri\ZivotniCiklus\ZivotniCiklus`

Chang...	Change	User	Date
10559	add	ivana	20.1

Ready

43



Domaća zadaća

- ☐ **Proučiti upute o uspostavi i korištenju TFS (TFS-Upute)**
- ☐ **Instalirati TFS dodatak razvojnoj okolini Visual Studio**
- ☐ **Uspostaviti kontrolu programskog koda projekta ekipe**
 - Isprobati osnovne funkcije, dodavanjem i zaključavanjem izvornih datoteka
 - Isporučiti u mapu projekta početni plan projekta

Računalom podržano programsko inženjerstvo

CASE pomagala

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

❑ Gornji CASE (Upper CASE, Front-End CASE)

- Automatizacija ili podrška ranih faza - planiranje, analiza i oblikovanje
 - modeliranje zahtjeva
 - oblikovanje baza podataka, prevođenje modela u 3NF
 - generiranje sheme BP (SQL naredbe)

❑ Donji CASE (Lower CASE, Back-End CASE)

- potpora fazama razvoja: detaljni dizajn, konstrukcija, ugradnja, održavanje
- generatori programske opreme (application generator, code generator)
- alati za održavanje programske potpore (version control) i prepravke (reengineering)
- alati za izradu programske dokumentacije (documentation tool)
- alati za povratno inženjerstvo (reverse engineering), npr. analizatori izvornog koda, alati za restrukturiranje koda

❑ Integrirana CASE pomagala (ICASE - Integrated CASE)

- cjeloviti CASE sustavi koji pokrivaju sve faze razvoja, a sadrže i dodatne module za povratno inženjerstvo, nadzor i upravljanje izvedbom te osiguranje kvalitete

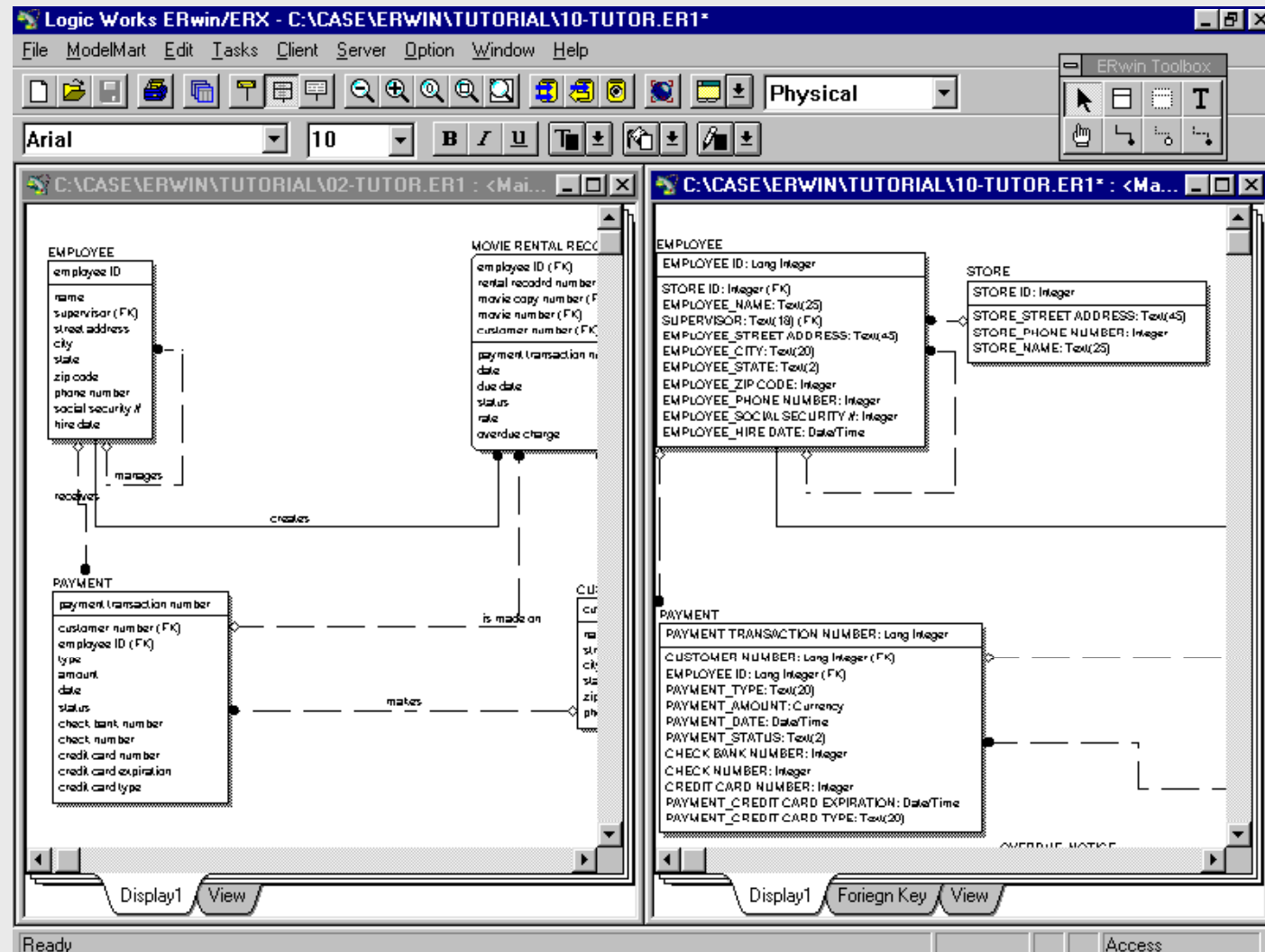
Funkcije pomagala i razvojni procesi

□ Slijede primjeri (dostupnih?) pomagala

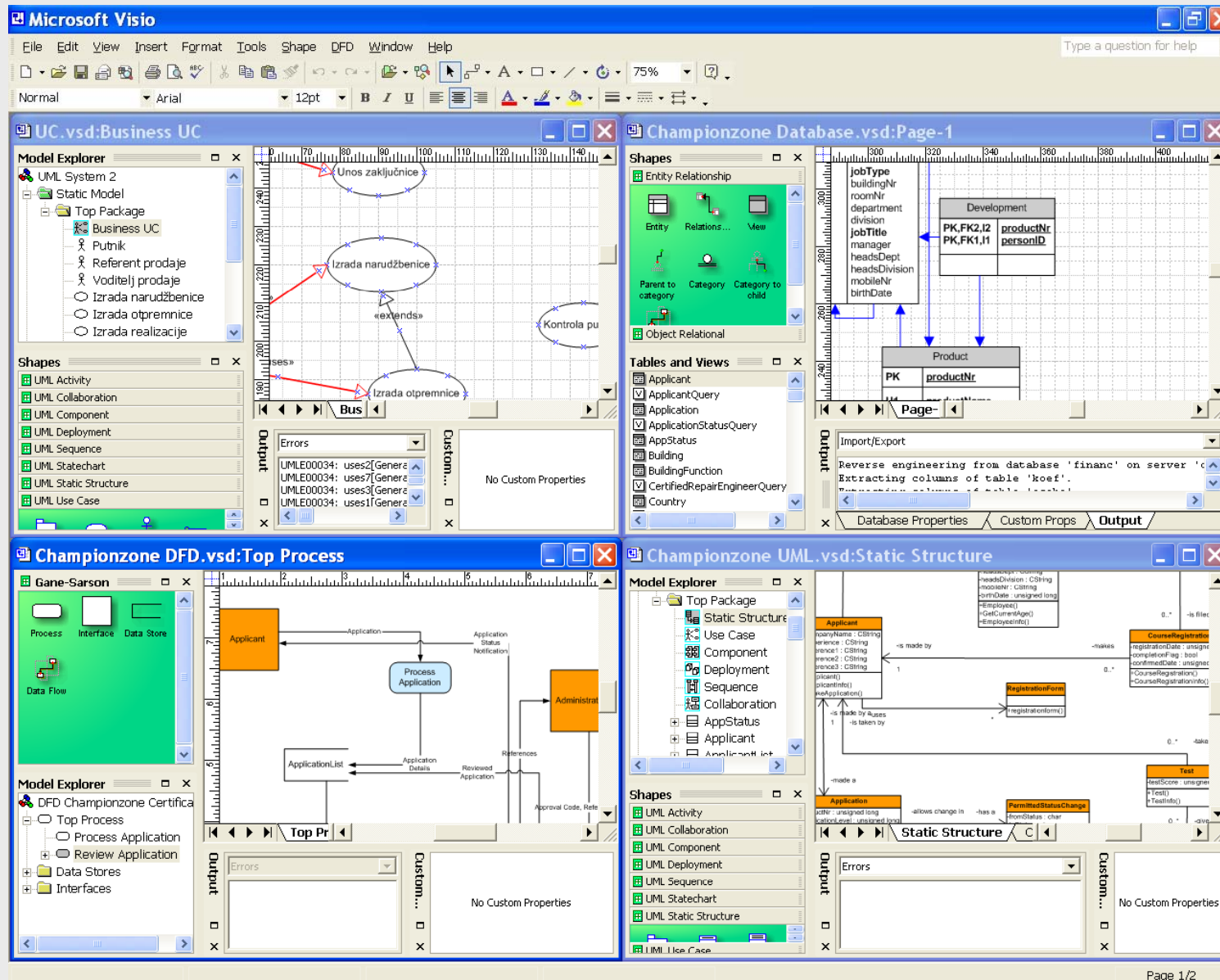
Alati za reinženjerstvo			•	
Alati za testiranje			•	•
Alati za debugiranje			•	•
Alati za analizu programa			•	•
Alati za procesiranje koda		•	•	
Potporna metodologije	•	•		
Alati za prototipiranje	•			•
Upravljanje konfiguracijom		•	•	
Upravljanje promjenama	•	•	•	•
Alati za dokumentiranje	•	•	•	•
Alati za modeliranje	•	•	•	•
Alati za planiranje	•	•	•	•
	Specifikacija	Dizajn	Implementacija	Testiranje

Primjer alata za oblikovanje BP

- ❑ CA AllFusion ERwin Data Modeler – dizajn i reinženjersvo BP, različite notacije (IDEF1X, IE, DM)



MS Visio - univerzalno pomagalo za oblikovanje



Integracija CASE alata u razvojnu okolinu

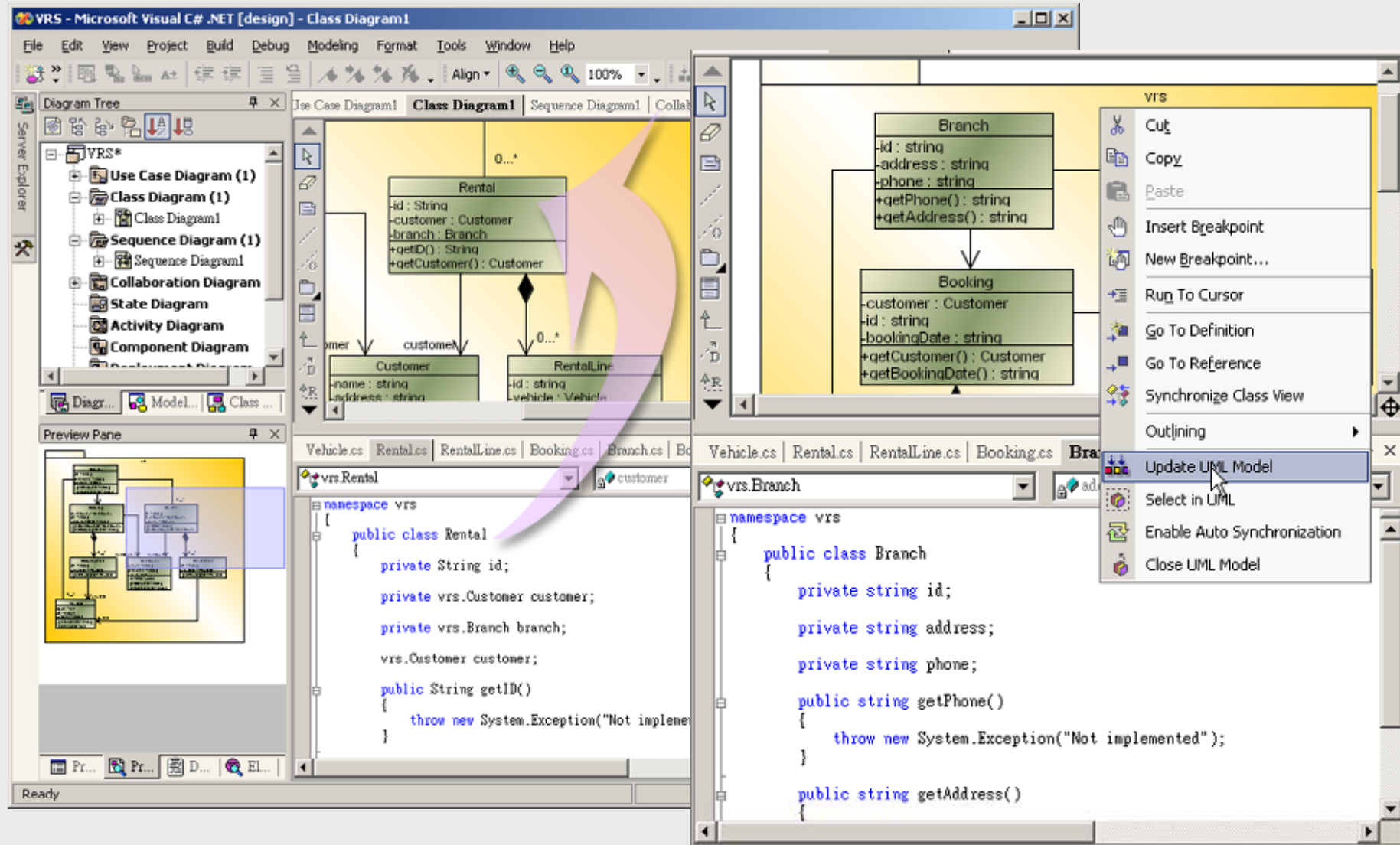
☐ Visual Paradigm for UML

- Svi UML dijagrami
- Analiza teksta
- CRC kartice

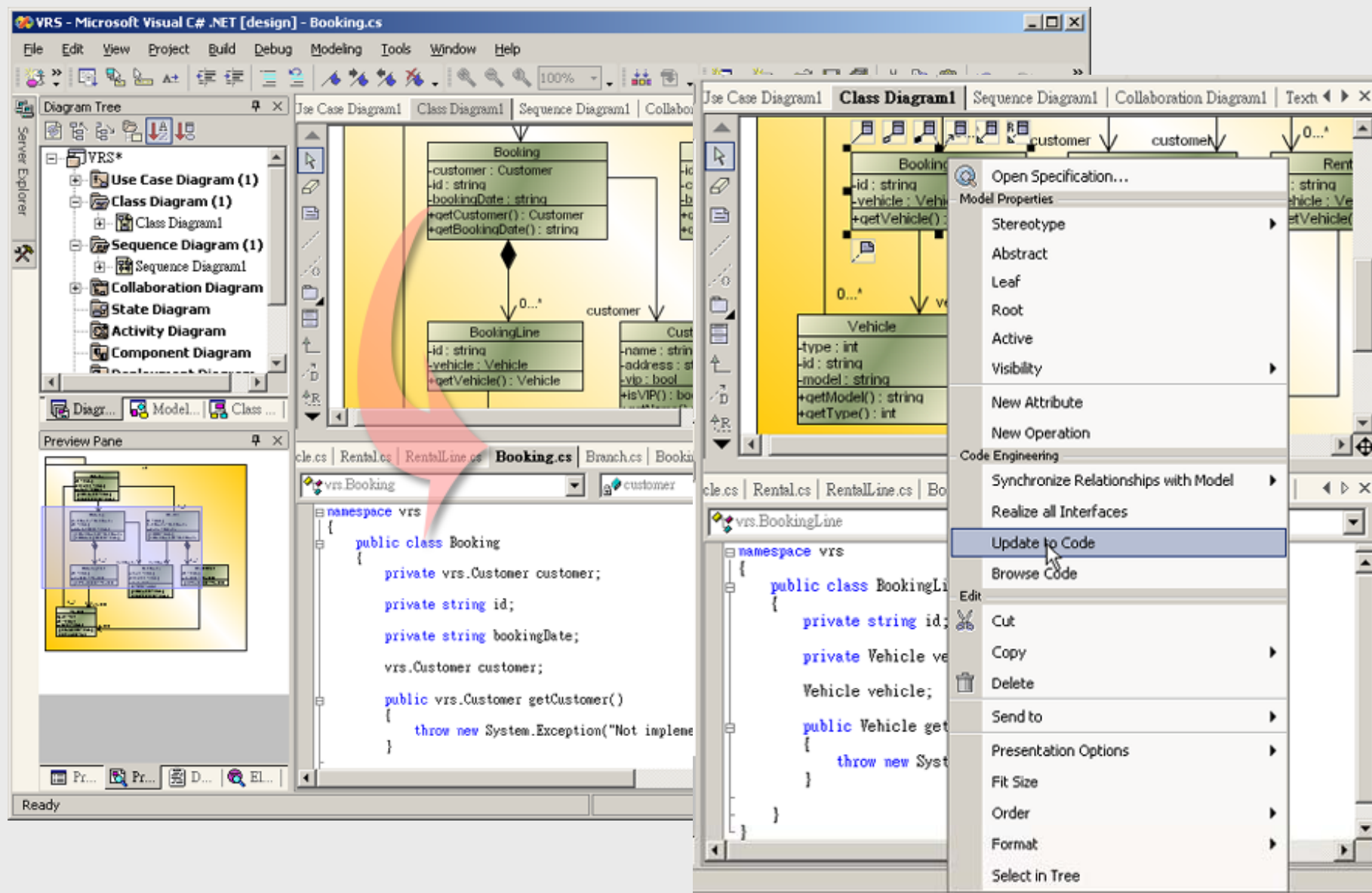
☐ Visual Paradigm for SDE (Smart Development Environment)

- integracija s Visual Studio, Eclipse, JBuilder, ...
- reverzno inženjerstvo
- generiranje izvornog koda
- Community Edition
- <http://www.visual-paradigm.com>

Reverzno inženjerstvo



Generiranje izvornog koda



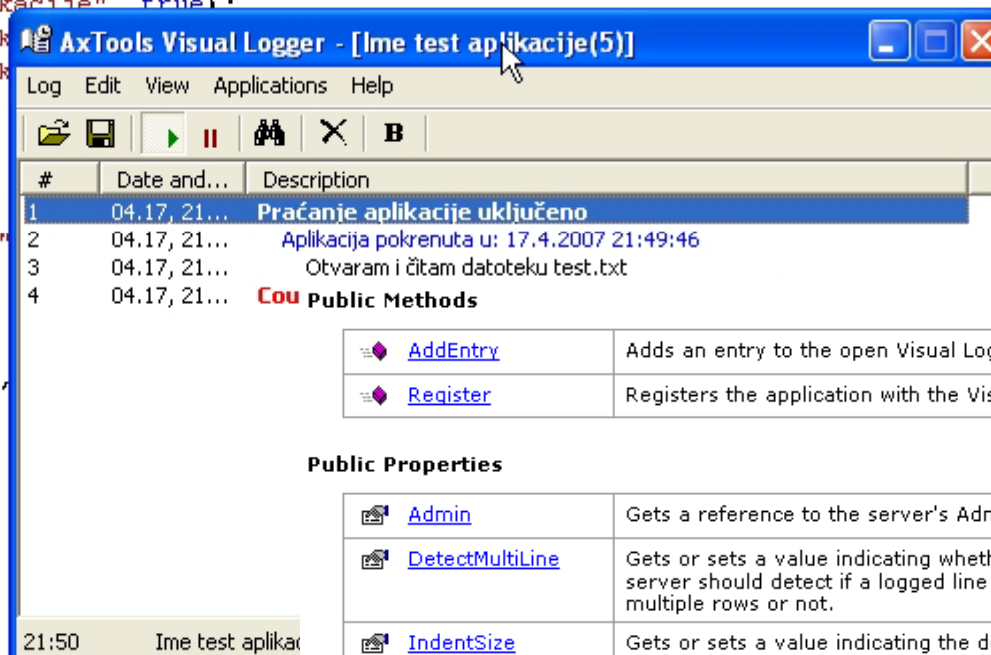
Pomagalo za praćenje rada aplikacije

❑ Visual Logger (freeware)

- <http://www.axtools.com>
- može se koristiti prilikom razvoja (debugging i trace) ili nakon isporuke aplikacije za praćenje njenog rada i kasniju analizu zabilježenih informacija (logovi se pošalju programerima na analizu)

```
static void Main(string[] args)
{
    AxVisualLogger.VisualLoggerClass logger = new AxVisualLogger.VisualLoggerClass();
    logger.Register("Ime test aplikacije", true);
    logger.AddEntry("Praćanje aplikacije uključeno");
    logger.AddEntry("Aplikacija pokrenuta u: 17.4.2007 21:49:46");
    logger.AddEntry("Otvaram i čitam datoteku test.txt");

    try
    {
        logger.AddEntry("Otvaram i čitam datoteku test.txt");
        File.ReadAllText("test.txt");
    }
    catch (Exception ex)
    {
        logger.AddEntry(ex.Message);
    }
}
```



AddEntry	Adds an entry to the open Visual Logger list.
Register	Registers the application with the Visual Logger server.

Public Properties

Admin	Gets a reference to the server's Admin object.
DetectMultiLine	Gets or sets a value indicating whether the Visual Logger server should detect if a logged line is spanned across multiple rows or not.
IndentSize	Gets or sets a value indicating the default text nesting level in the Visual Logger list.
UseIndentBar	Gets or sets a value indicating whether the logger should use pipe (" ") characters to outline the text nesting.

Generatori koda [Herrington]

☐ **Prevođenje koda (Code munging)**

- prijevod programskog koda napisanog u jednom programskom jeziku u drugi, regularnim izrazima ili jednostavnom obradom (parsiranjem) i ugrađenim predlošcima (templates)

☐ **Ekspanzija koda (Inline code expanding)**

- generiranje supstitucijom posebnih oznaka (markups)

☐ **Generiranje miješanim kodom (Mixed-code generation)**

- čita izvorni kod i zamjenjuje ga npr. analizom posebne vrste komentara

☐ **Djelomično generiranje razreda (Partial-class generation)**

- generiranje skupova razreda temeljem apstraktnih definicija

☐ **Generiranje slojeva (Tier or layer generation)**

- generiranje cjelokupnog sloja aplikacije, npr. temeljem UML modela (model-driven generation)

CodeSmith

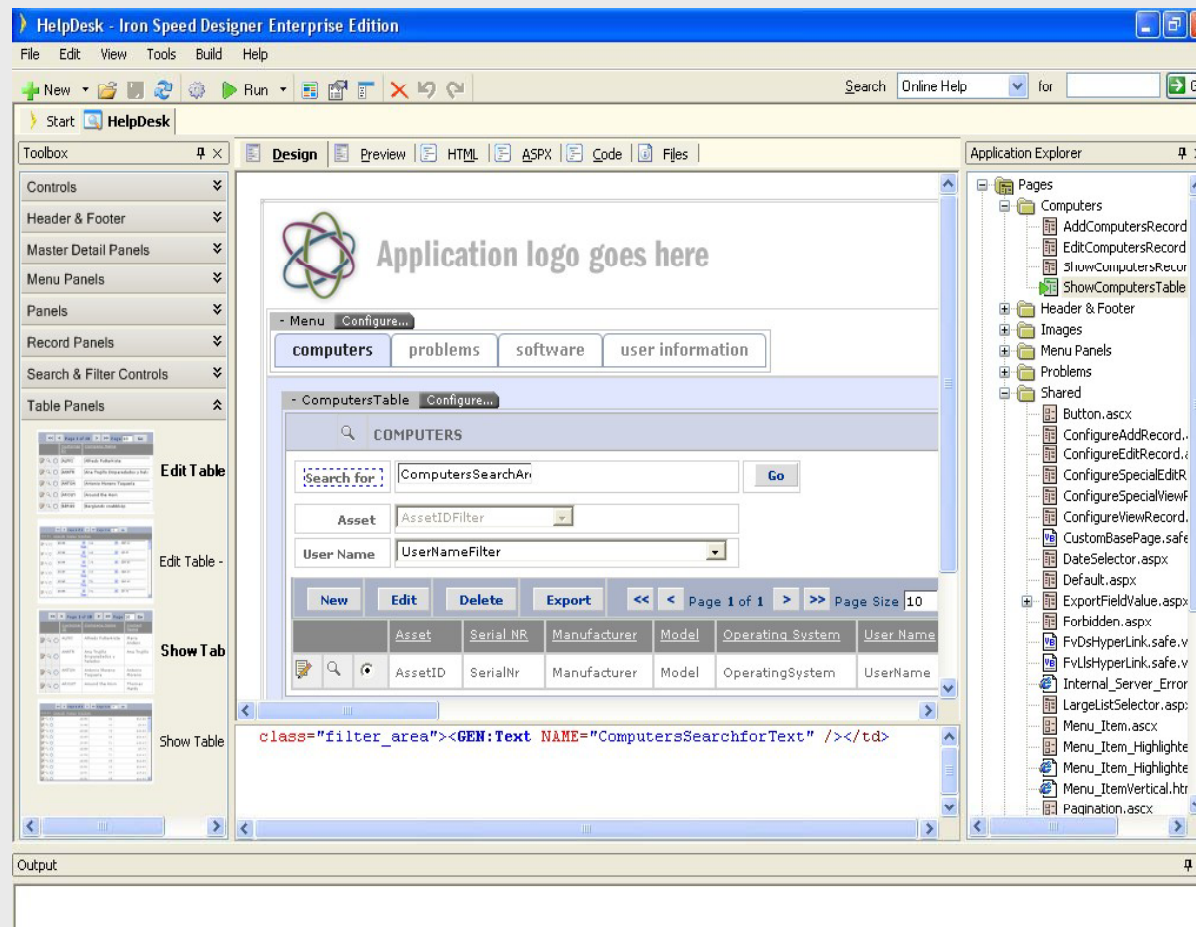
- ❑ Predlošci za C#, VB, J# ...
- ❑ Prilagodba sintaksom sličnom ASP.NET

```
4 Description: Create a list of properties from a database table
5 --%>
6 <%@ CodeTemplate Language="C#" TargetLanguage="C#" Debug="False" Description="Create a list of
7 <%@ Property Name="SourceTable" Type="SchemaExplorer.TableSchema" Category="Context" Descript
8 <%@ Map Name="CSharpAlias" Src="System-CSharpAlias" Description="System to C# Type Map" %>
9 <%@ Assembly Name="SchemaExplorer" %>
10 <%@ Import Namespace="SchemaExplorer" %>
11
12 <% foreach (ColumnSchema column in this.SourceTable.Columns) { %>
13 private <%= CSharpAlias[column.SystemType.FullName] %> _<%= StringUtil.ToCamelCase(column.Name)
14
15 public <%= CSharpAlias[column.SystemType.FullName] %> <%= StringUtil.ToPascalCase(column.Name)
16 {
17     get { return _<%= StringUtil.ToCamelCase(column.Name) %>; }
18     set { _<%= StringUtil.ToCamelCase(column.Name) %> = value; }
19 }
20
21 <% } %>
```


Iron Speed

❑ Generator .NET Web Aplikacija

- generira GUI, web page logic, DAL
- do 80% of aplikacije
- ima čarobnjaka, predloške, prilagodbu izbornika, ...



MyGeneration (besplatan!)

❑ Zasnovan na predloščima

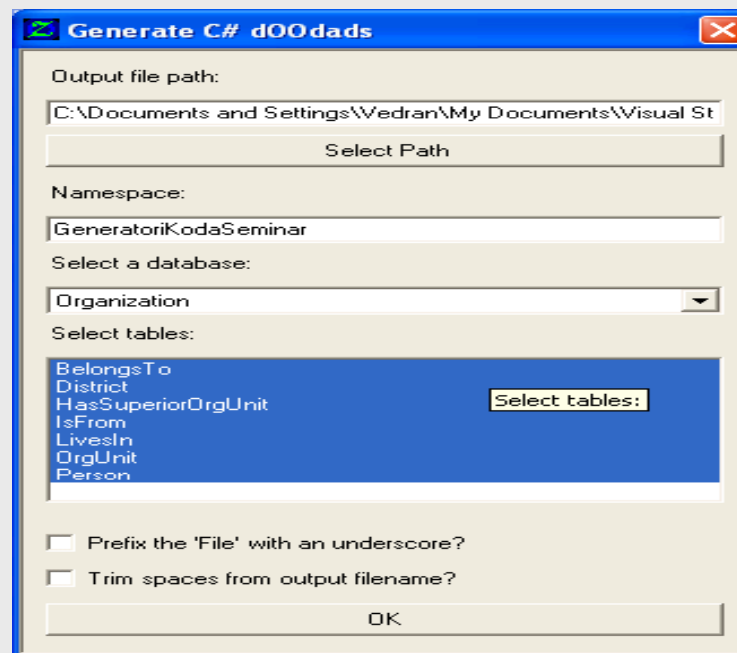
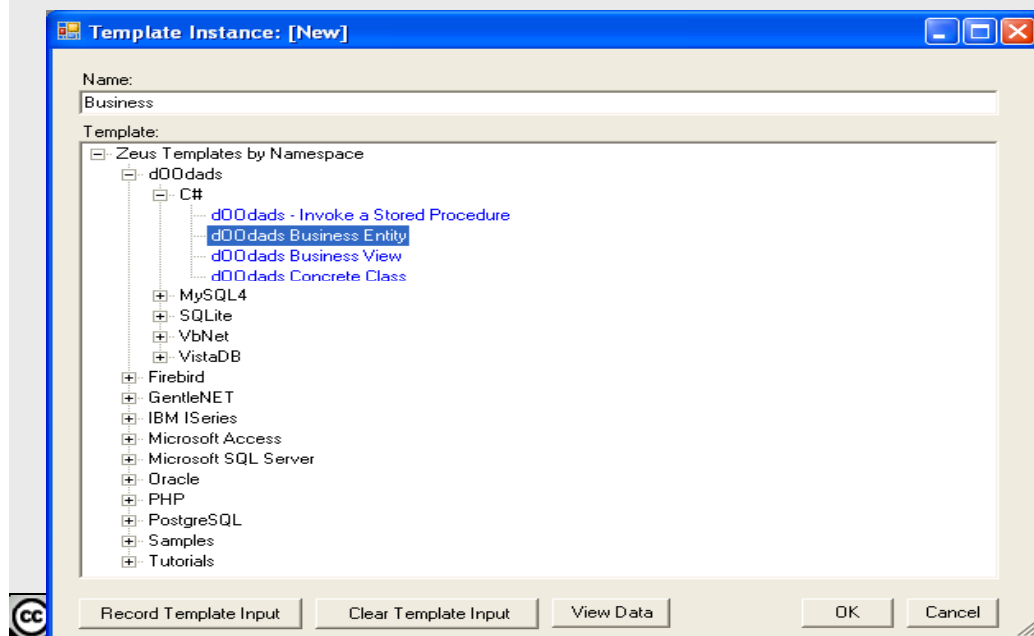
- prilagodba korištenjem JScript, VBScript, C#, VB.NET.

❑ Podrška arhitekturama

- dOODads, EntitySpaces, EasyObjects.NET/EntLib, Gentle.NET, Opf3, NHibernate, Microsoft's DAAB, DotNetNuke, iBatis.

❑ Podrška bazama podataka

- Microsoft SQL, Oracle, IBM DB2, PostgreSQL, Microsoft Access, FireBird, Interbase, VistaDB, SQLite, MySQL, Advantage, Pervasive



Pouke glede generatora

☐ Pravilno korišteni

- Skraćuju vrijeme kodiranja
- Smanjuju broj pogrešaka
- Povećavaju učinkovitost prilagodbe programa
- Doprinose pisanju konzistentnog i kvalitetnog koda

☐ Mogući rizici

- Ovisnost o dobavljaču alata
- Loše aplikacije kao rezultat loših predložaka
- Zlorababa alata ili uporaba neadekvatnih alata

Reference

❑ Upravljanje projektima

- www.pmi.org
- <http://www.pmi-croatia.hr>
- <http://www.construx.com/>
- <http://office.microsoft.com/en-us/help/HA011361531033.aspx>

❑ IDE , CASE

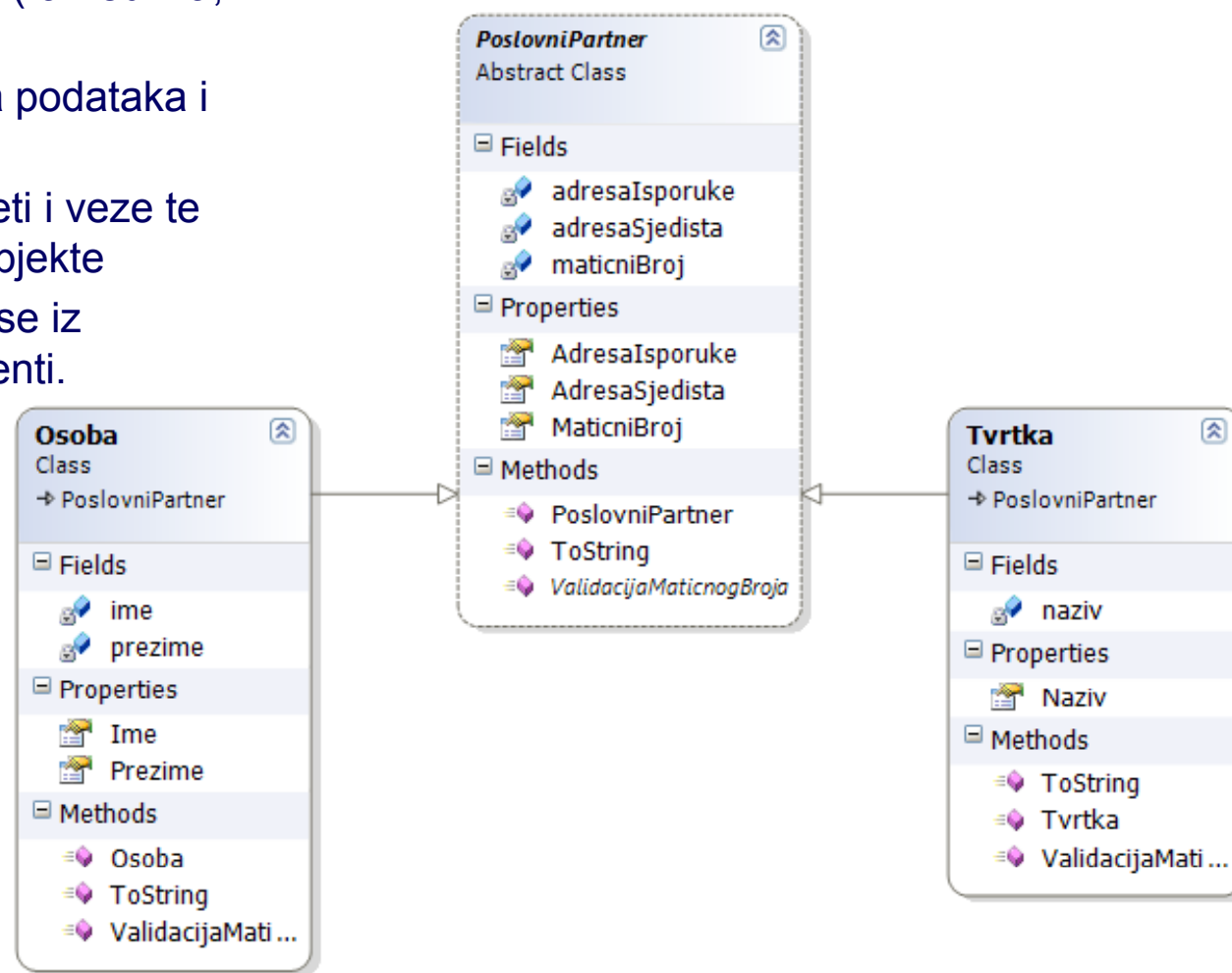
- <http://www.cs.queensu.ca/Software-Engineering/tools.html>
- <http://www.codegeneration.net/>
- <http://www-306.ibm.com/software/awdtools/developer/rosexde/>
- <http://www.visual-paradigm.com>
- <http://www.axtools.com>
- <http://www.eclipse.org/>
- <http://www.visible.com/>
- <http://www.popkin.com/>

Objektno orijentirano programiranje

Osnove objektno orijentiranog programiranja

❑ Objektno usmjerena paradigma

- entiteti iz stvarnog svijeta opisuju se apstraktnim objektima (razredima, klasama objekata)
- integracija oblikovanja podataka i procesa
- tokovi podataka, entiteti i veze te procesi integrirani u objekte
- složeni sustavi grade se iz pojedinačnih komponenti.



Koncepti objektnog pristupa

❑ Objekt (Object)

- objekt - svaka pojava, pojam ili predmet (stvarni ili zamišljeni) o kojem se prate podaci i načini rukovanja podacima (procesi)
- pojava objekta (object instance) – pojedinačna pojava nekog tipa objekta, s vlastitim stanjem i ponašanjem

❑ Apstrakcija (Abstraction)

- Obuhvat glavnih aspekata uz ispuštanje nevažnih detalja

❑ Učahurivanje (Encapsulation)

- skrivanje ugradnje od korisnika objekta
- objekti su “crne kutije” čijim se podacima rukuje putem ugrađenih metoda

❑ Modularnost (Modularity)

- Razlaganje složenih sustava u (lakše) upravljive dijelove.

❑ Hijerarhija (Hierarchy)

- Hijerarhijsko uređivanje apstrakcija

Koncepti objektnog pristupa

❑ Razred (Class)

- tip objekta (abstract data type) – generalizacija grupe objekata, zajednički naziv za sve objekte sa zajedničkim svojstvima (property) i načinima rukovanja (process, method), npr. Proizvod
- opis grupe objekata koji imaju jednake attribute, operacije, veze i semantiku
- objekt je instanca razreda, s vlastitim vrijednostima svojstava
- razred je apstraktna definicija objekta

❑ Svojstvo (Property), atribut (Attribute)

- definira skup vrijednosti koje instanca može poprimiti za neku karakteristiku
- može biti nekog tipa, npr. integer, Boolean, Adresa

❑ Postupak (Method), Operacija (Operation)

- implementacija usluge koja može biti zatražena od objekta razreda
- ima prototip, signaturu (naziv, parametre, povratnu vrijednost)

❑ Višeobličje (Polymorphism)

- mogućnost "skrivanja" različitih ugradnji iza zajedničkog sučelja
- postupci koje objekti različitih razreda tumače zavisno o specijalizaciji

Koncepti objektnog pristupa

❑ Nasljeđivanje (Inheritance)

- podrazred (subclass) nasljeđuje svojstva i metode nadrazreda (superclass), npr. Igla i Avion nasljeđuju Proizvod

❑ Podsustav (Subsystem)

- kombinacija paketa razreda
- realizacija jednog ili više sučelja koje definiraju ponašanje razreda

❑ Komponenta (Component)

- nezavisan, zamjenjiv, ne-trivijalan dio sustava
- može biti izvorni kod, pogonska inačica ili izvršni program

❑ Veza, asocijacija (Association)

- smisljena poveznica između dva ili više dijelova

Razredi

Razredi i članovi razreda

- ❑ **U jeziku C# razredi se sastoje od članova (members):**
 - Konstante (constants)
 - Atributi (fields)
 - Konstruktori (constructors)
 - Destruktori (destructors)
 - Postupci (methods)
 - Svojstva (properties)
 - Indekseri (indexers)
 - Operatori (operators)
 - Događaji (events)
 - Statički konstruktori (static constructors)
 - Ugniježđeni tipovi (nested types)
- ❑ **Razred je obrazac prema kojem se stvaraju instance razreda - objekti**
 - `new` – operator koji stvara novu instancu
 - `this` – referenca na trenutnu instancu razreda
 - razred s početnim postupkom (`Main`) ne mora se eksplicitno instancirati

Modifikatori tipova i članova

❑ Modifikatori pristupa razredima i članovima

- `public` – pristup nije ograničen
- `protected` – pristup ograničen na razred i naslijeđene razrede
- `internal` – pristup ograničen na program u kojem je razred definiran
- `protected internal` - pristup ograničen na razred i naslijeđene razrede u programu u kojem je razred definiran
- `private` - pristup ograničen na razred u kojem je član definiran

❑ Ostali modifikatori

- `abstract` – razred može biti samo osnovni razred koji će drugi naslijeđivati
- `const` – atribut (polja) ili lokalna varijabla je konstanta
- `extern` – postupak se implementira negdje drugdje
- `new` – modifikator koji skriva naslijeđenog člana od člana osnovnog razreda
- `readonly` – polje poprima vrijednost samo u deklaraciji ili pri instanciranju
- `sealed` – razred ne može biti naslijeđen
- `static` – zajednički član svih instanci razreda (ne kopija nastala s `instancom`)
- `virtual` – postupak ili dostupni član koji može biti nadjačan u naslijeđenom razredu

Konstante (constants) i atributi (fields)

❑ Konstante

- Članovi kojima se pristupa operatorom za pristup članu
`className.memberName`

❑ Modifikator `const`

- Vrijednost konstante određena je pri prevođenju i ne može se mijenjati
- Konstante su po prirodi statičke, pa ih se ne može eksplicitno proglasiti statičkim ni pristupati im preko reference na instancu objekta

❑ Atributi

- Varijable kojima se može promijeniti vrijednost pristupom članu
- Nepromjenjivi atributi – modifikator `readonly`
 - Koriste se kada nije dozvoljena njihova promjena, a vrijednosti im nisu poznate u vrijeme prevođenja

❑ Primjer Razredi\Razred

```
public int var = 0;  
public readonly int neDiraj;  
//konstanta  
public const int konst = 1;
```

Konstruktori

❑ Konstruktor je postupak koji se obavlja pri stvaranju instance

- Standardni (default) konstruktor, preopterećenje konstruktora
- Ne vraća vrijednost
- Ime mu je jednako imenu razreda

❑ Primjer Razredi\Razred

```
class Razred
{
    //atributi
    public readonly int neDiraj;
    //konstruktori
    public Razred() {
        neDiraj = 0;
    }
    public Razred(int neDiraj0) {
        neDiraj = neDiraj0;
    }
}
```

```
Razred r1 = new Razred();
Razred r2 = new Razred(13);
```

Destruktori

- ❑ **Destruktor je postupak koji se automatski poziva neposredno prije uništenja objekta od strane sakupljača smeća (Garbage Collector).**

- Naziv destruktora jednak je nazivu razreda s predznakom '~'
- Ne vraća vrijednost

- ❑ **Primjer**  **Razredi\Razred**

```
//destruktor
~Razred() {
    Console.WriteLine("Destruktor");
}
}
```

- ❑ **Sakupljač smeća (garbage Collector)**

- Oslobađa memoriju koja je bila zauzeta referencama koje više ne postoje.
- Sakupljanje smeća obavlja tijekom izvođenja programa.
- Ne može se unaprijed točno odrediti kada će se memorija osloboditi, to jest kada će se destruktor pozvati.

Postupci (metode)

❑ Postupak je programska funkcija pridružena razredu.

- Skrivanje člana (učahurivanje - enkapsulacija)
- Pristup skrivenom članu javnim postupcima

❑ Primjer: 📁 Razredi\Postupci

```
class Postupak
{
    private int localVar = -1;
    public void SetVar(int var) { localVar = var; }
    public int GetVar() { return localVar; }
}
```

```
Postupak p = new Postupak();
Console.WriteLine("Var prije " + p.GetVar());
p.SetVar(3);
Console.WriteLine("Var poslije " + p.GetVar());
```


Postupci (metode)

❑ Argumenti:

- standardno, bez modifikatora – "call by value"

```
public static void SwapByVal (int a, int b);  
SwapByVal (a, b);
```

- ref modifikator – argumenti su reference ("call by reference")
 - Prije poziva postupka argumenti MORAJU biti inicijalizirani
 - ref parametru argument mora eksplicitno biti predan navođenjem ref.

```
public static void SwapByRef (ref int a, ref int b);  
SwapByRef (ref a, ref b);
```

- out modifikator – izlazni argumenti
 - U trenutku poziva out postupka argumenti ne moraju biti inicijalizirani.
 - Pri izlasku iz postupka out argumenti MORAJU biti postavljeni.

```
public static void TestOut(out int val, int defval);  
TestOut(out i, 13);
```

❑ Primjer Razredi\Postupci

Postupci (metode)

❑ Prijenos varijabilnog broja argumenata:

- Ključna riječ `params`
 - koristi se kad broj argumenata nije unaprijed poznat
 - polje argumenata može biti bilo kojeg tipa

❑ Primjer: Razredi\Argumenti

```
public static void DumpParameters(params object[] args){  
    Console.WriteLine("Params: ");  
    for(int iArg = 0; iArg < args.Length; iArg++)  
        Console.WriteLine("{0}:{1}", iArg, args[iArg])  
}
```

```
DumpParameters("jen", "dva", 3);  
DumpParameters();
```

❑ Main je postupak koji preuzima argumente pri pozivu programa

- `static void Main(string[] args)`
- navođenje `params` je opcionalno
- Primjer `DumpParameters(args);`

Svojstva

- ❑ Svojstvo je postupak pristupa zaštićenim varijablama instance:

```
class Temperatura{  
    private float T;  
    public float Celsius{  
        get { return T - 273.16f; }  
        set { T = value + 273.16f; }  
    }  
    public float Fahrenheit{  
        get { return 9f / 5 * Celsius + 32; }  
        set { Celsius = (5f / 9) * (value - 32); }  
    }  
}
```

```
Temperatura X = new Temperatura();  
X.Fahrenheit = 70;  
Console.WriteLine("{0} = {1}", X.Fahrenheit, X.Celsius);
```

- ❑ Primjer  Razredi\SvojstvaIndekseri
- ❑ Ako ne želimo dozvoliti promjenu varijable, ne implementiramo set.

Indekseri

❑ Indekseri

- omogućavaju korištenje objekta kao niza (pristup elementima razreda pomoću operatora [])
- Sintaksa uobičajena za svojstva (get i set)

❑ Primjer: Razredi\SvojstvaIndekseri

```
public Temperatura this[int index]    // Indeksir
{
    set{
        if (index >= 0 && index < nizTemperatura.Length)
            nizTemperatura[index] = value;
        else throw new Exception("Pogreška!");
    }
    get{
        if (index >= 0 && index < nizTemperatura.Length)
            return nizTemperatura[index];
        else throw new Exception("Pogreška!");
    }
}
```

Preopterećenje postupaka (method overloading)

❑ Postupci jednakog imena definirani u istom razredu

- moraju imati različitu signaturu (prototip), tj. različiti tip ili redoslijed argumenata
- uobičajeno obavljaju isti posao – nad različitim tipovima podataka

❑ Primjer: Razredi\PreopterećenjePostupaka

```
static double Maximum(double x, double y, double z){  
    Console.WriteLine("double Maximum( double x, double y,  
        double z )");  
    return Math.Max(x, Math.Max(y, z));  
}  
  
static int Maximum(int x, int y, int z){  
    Console.WriteLine("int Maximum( int x, int y, int z )");  
    return Math.Max(x, Math.Max(y, z));  
}
```

```
// koji Maximum se poziva?  
Console.WriteLine("maximum1: " + Maximum(1, 3, 2));  
Console.WriteLine("maximum2: " + Maximum(1.0, 3, 2));
```

Preopterećenje operatora (operator overloading)

❑ Operatori koji se mogu preopteretiti

- unarni: + - ! ~ ++ -- true false
- binarni: + - * / % & | ^ << >> == != > < >= <=

❑ Primjer: Razredi\PreopterecenjeOperatora

```
public static ComplexNumber operator +(ComplexNumber x,
    ComplexNumber y){
    return new ComplexNumber(x.Re + y.Re, x.Im + y.Im);
}

public static ComplexNumber operator *(ComplexNumber x,
    ComplexNumber y){
    return new ComplexNumber(x.Re * y.Re - x.Im * y.Im, x.Re *
        y.Im + y.Re * x.Im);
}
```

```
Console.WriteLine(x + " + " + y + " = " + (x + y));
Console.WriteLine(x + " - " + y + " = " + (x - y));
Console.WriteLine(x + " * " + y + " = " + (x * y));
```

Statički članovi

- ❑ **static** se može primijeniti na atribute, postupke, svojstva, operatore i konstruktore
- ❑ **Primjer**  **Razredi\StatickiClanovi**
- ❑ **static varijable**
 - sve instance razreda dijele istu kopiju varijable razreda

```
private static int brojac = 0;  
public StaticRazred() {  
    brojac++;  
}
```

- ❑ **static postupci**
 - Statički član pripada tipu, a ne instanci.
 - **Poziv:** ImeRazreda.ImeStatickogClana (...);

```
StaticRazred.Brojac;  
StaticRazred.Zbroji(5, 3);
```

```
static int Zbroji(int x, int y) {  
    return x + y;  
}
```

Zadaci za vježbu

❑ Implementirati razred `Planet`

- Konstruktor neka prima radijus, gravitaciju i ime planeta.
- Razred sadrži i statičku varijablu za brojanje instanciranih planeta, svojstva `Ime`, `Radijus`, `Gravitacija` i postupak `GetCount()`.
- Demonstrirati rad implementiranog razreda.

 `Razredi\Planeti`

❑ Implementirati razred `PlanetCollection`

- Razred predstavlja listu planeta.
- Iskoristiti razred iz prethodnog zadatka s planetima.
- Napisati postupak `Add` za dodavanje planeta.
- Planetima se može pristupati preko indeksa.

❑ Implementirati razred `Nebo` koji

- sadrži `Planet` njegovu udaljenost od njegovog sunca.
- U razred `PlanetCollection` dodati ime sunca
- Dodati postupak sortiranja planeta po udaljenosti od sunca,
- te postupak za računanje “udaljenosti” dvaju planeta.



Zadaci

☐ Domaća zadaća

- Kreirati jedno programsko rješenje (solution) na TS
 - Svaki član ekipe unutar toga kreira vlastiti projekt izvornog koda
 - Svaki projekt mora sadržavati barem jedan razred koji demonstrira ugradnju različitih tipova članova (organizacija, zaposlenik, korisnik, projekt, zadatak, posao, dokument, honorar, ...)

☐ Zadatak projekta

- Odabrati CASE pomagalo za projektiranje i dokumentiranje

Reference

❑ C# School Book – Free ebook

- <http://www.programmersheaven.com/2/CSharpBook>

❑ Objects, Classes, and Structs (C# Programming Guide)

- Objects, Classes, Structs, Inheritance, ...
- <http://msdn2.microsoft.com/en-us/library/ms173109.aspx>