

Osnove programskog inženjerstva. Životni ciklus razvoja programske podrške.

01/13

Programska potpora

❑ Programska oprema/podrška/potpota, softver (software)

- dio računalnog sustava koji nema fizikalnih dimenzija
- opći pojam za sve vrste programa, programskih jezika itd
- Skup elemenata ili objekata u jedinstvenoj “konfiguraciji” koju čine računalni programi + podaci + dokumentacija
- svojstva:
 - složenost, podložnost pogreškama,
 - ne troši se, teško mjerljiv,
 - stari, dugo se koristi,
 - lako se kopira (zajedno s pogreškama)

❑ Primijenjena programska potpora = Računalna aplikacija (application)

- namjenski program, primjenska programska oprema
- računalom podržano rješenje jednog ili više poslovnih problema ili potreba

❑ Informacijski sustav = sustav aplikacija za upravljanje ljudskim aktivnostima

Jednokorisničke, samostalne aplikacije

- ❑ “standalone”, početkom 90-ih, dBase, Clipper, ZIM, ...

The screenshot displays a DOS-based standalone application interface. At the top, a status bar shows 'MR' and the date '25.05.93'. Below this is a 'MAIN MENU' with options: Grad, Os, and Ze. The 'Grad' option is selected, leading to a screen for entering 'Grad' data. This screen includes fields for 'Sifra zemlje' (set to 'HR') and 'Naziv zem' (set to 'Zemlja'). A sub-menu is visible for 'Zemlja', showing '1/1' and 'Postanski Sifra zemlje : HR'. The main data entry screen shows 'Naziv grada' (set to 'Zagreb') and 'Adresa' (set to 'Srednjaci bb'). At the bottom, it shows 'Zemlja rodjenja : AAA' and a list of function keys: 'F5-Zoom F6-Window F10-Save ESC'. A legend at the bottom of the 'Zemlja' sub-menu indicates: 'F5 ENTER F2-Add F3-Delete F4-Change F8-Browse F9'.

```
MR 25.05.93
```

```
MAIN MENU
Grad
Os
Ze
```

```
Grad Add
Sifra zemlje : HR
Naziv zem Zemlja 1/1
Postanski Sifra zemlje : HR
Naziv grada Naziv zemlje : Republika Hrvatska.....
F5 ENTER F2-Add F3-Delete F4-Change F8-Browse F9
```

```
Naziv grada : Zagreb.....
Adresa : Srednjaci bb.....
Zemlja rodjenja : AAA
F5-Zoom F6-Window F10-Save ESC
```

Poslužiteljske aplikacije

- ❑ serverske, 90-ih, Informix, Oracle, ...

OSOBA: Sljed Preth Unos Izmj Ostalo Lista Rasp Zap ...
 Postavljanje uvjeta za dohvat zapisa

===== (LALIĆ MARIJAN) ===== (2/3) ===== 120/261 =====

Sprema (71) (Visoka VII/1)
 Br.svjed. () Dat. () Izdao ()
 Zanimanje (54131) (ORGANIZATORI SISTEMA)
 Stamb.stanje (3) (Privremeni)
 Krvna grupa ()
 Zdrav.broj ()

Ured za obranu (0) (**)
 Osobni UPD (33207) (SIN)
 Početak voj.roka ()
 Vojska voj.roka (0) (Nepozn)
 Razl.prest.voj.roka ()
 Sudjel. u dom.ratu (D)

		Broj naloga 04 VS 0129	
		Datum knj. 09/11/2004	
Broj dokumenta _____		Orig. broj _____	
Vrsta dokumenta BKA		BANKOVNI IZVOD	
Mj. troška/pri. 30		Pregled : radnik	
Konto 99	radnik	naziv	
Iznos kuna			
Dug/Pot D			
Datum dokumenta 05	90125	AGATIĆ	(-) NENAD
Analitika R	90737	AGIĆ	(-) DARKO
Program	00953	AGLIĆ ALJINOVIĆ	(ANDRIJA) ANDREA
Aktivnost	00914	ALEKSIĆ-MASLAČ	(KREŠIMIR) KARMELA
Pror. glava	90431	ALIĆ	(-) MLADEN
Izvor financir.	90617	AMBRUŠ	(-) DAVORIN
	90731	ANDRASSY	(-) MLADEN
	90037	ANDREIĆ	(-) VALENTIN
Pozicija	00899	ANDRES	(LADISLAV) DALIBOR
Opis	01300	ANĐELINIĆ	(MARKO) MATKO
Valuta / devize			
arhk-ff	traz_fm	Akt()	

Klijentske aplikacije

- ❑ “debeli” klijenti, kraj 90-ih, Microsoft Access, Visual Basic, Java, ...

Dokument

Vrsta: Predračun Broj: 102 Datum: 09.09.96 Prethodni: Broj:

Partner: Dječji vrtić "Vrapčići" Br. ulaznog dokumenta:

Datum valute: 20.09.96 Dana valute: 12 10 % Rabat Račun PP: ☐

Otprema: HPT Plaćanje: Virman Ukupno: 432.00 Datum storna:

Ostali troškovi: manipulacija + poštarina Gotovina: 18.00 Datum ažuriranja:

Slovima: četristopedeset kuna

Zaglavlje: Napomena:

Šifra	Naziv	Količina
1	Pokaži što znaš	8.00
2	Pokušaj nešto novo	8.00
3	Učimo opažati	8.00

Stavka: 1 od 3

Dokument: 1 od 6

F3 - Brisanje F5 - Tablica F7 -

Vrsta: Abies alba Mill.

Osnovno Autori Osobine OsobineNN Invazivne Sinonimi **Multimedija** Ugroženost Komentar EBDC IDWG Opis

Autor: Slika Del

Datum: Thumb ☐ Sakri

Tip: Slika Skanirano ☒ Javno ☐

Objekt: cvat/inflorescence

Tehnika: skanirano iz publikacije/scan from publication

Pohrana:


Referenca: Šilić, Č.

Godina: 1973

Naslov: Atlas drveća i gramlja.

Lokalitet:

Opaska:



Mobilne i distribuirane aplikacije

❑ Internet, džepne, “tanki” klijenti, *remoting*, 2000-, .NET, J2EE

Subscribe Pretplatnik

Server name: voz

Virtual directory: rep

Publication: gg

Database: gg

Username: sa

Password: ***

Database Path: \My

Poslovi

Osoba: ☐ ozura

Datum: ☐ 7.8.06

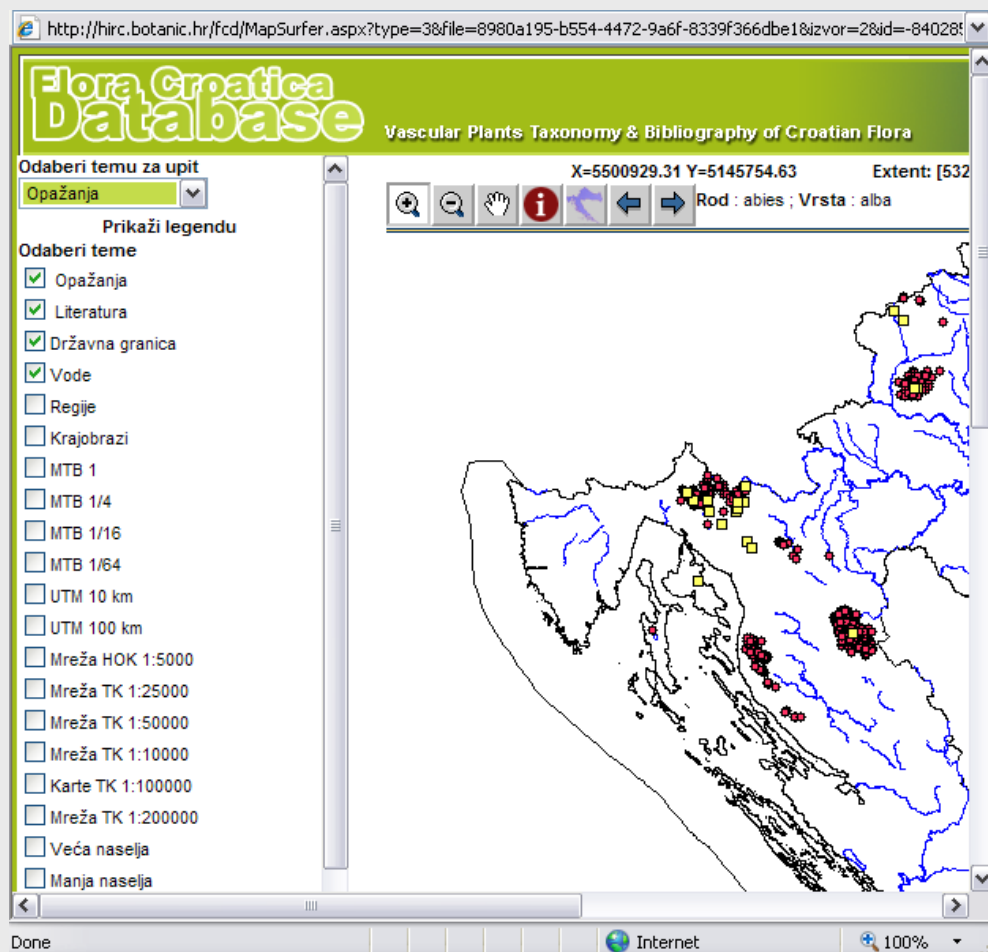
Projekt: ☐ !AktivanJavan

Posao: ☐ backup

RbrPosa	Osoba	DatPosao	KratP
254662	anaj	21.2.04	[VER]
254663	duje	14.2.04	111
254664	tošo	19.2.04	111
254665	anaj	19.2.04	111
254666	anaj	21.2.04	ACI
254667	duje	21.2.04	111
254668	tošo	12.2.04	111

Sati Sum 31975,65

View



Programsko inženjerstvo

❑ Programsko inženjerstvo (software engineering)

- “Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is **the application of engineering to software**. (2) The study of approaches as in (1).” [IEEE Std 610.12-1990, 1994]
 - IEEE Std 610.12-1990 "Standard Glossary of Software Engineering Terminology", IEEE, New York, 1994, p. 67.
 - IEEE = Institute of Electrical and Electronic Engineers, <http://www.ieee.org>
- **sistematičan, discipliniran i mjerljiv pristup razvoju, primjeni i održavanju softvera**
- **primjena inženjerskog pristupa na programsku opremu**
- Programsko inženjerstvo je inženjerska disciplina koja obuhvaća sve aspekte izrade programske opreme. [Sommerville, 2004]

❑ Područje programskog inženjerstva

- poslovi kojima se oblikuje i razvija programska oprema
- sustavna primjena prikladnih alata i tehnika na čitav proces razvoja programske potpore

Modeli procesa razvoja

❑ Model procesa

- Općenito: Plan razvoja, koji navodi opće postupke razvoja programskog proizvoda.
- Preciznije: Definicija koja kaže koje aktivnosti treba obaviti, tko ih treba obaviti i u kojoj ulozi; kojim redoslijedom, koji će proizvodi biti razvijeni i kako ih vrednovati.

❑ Korišteno nazivlje

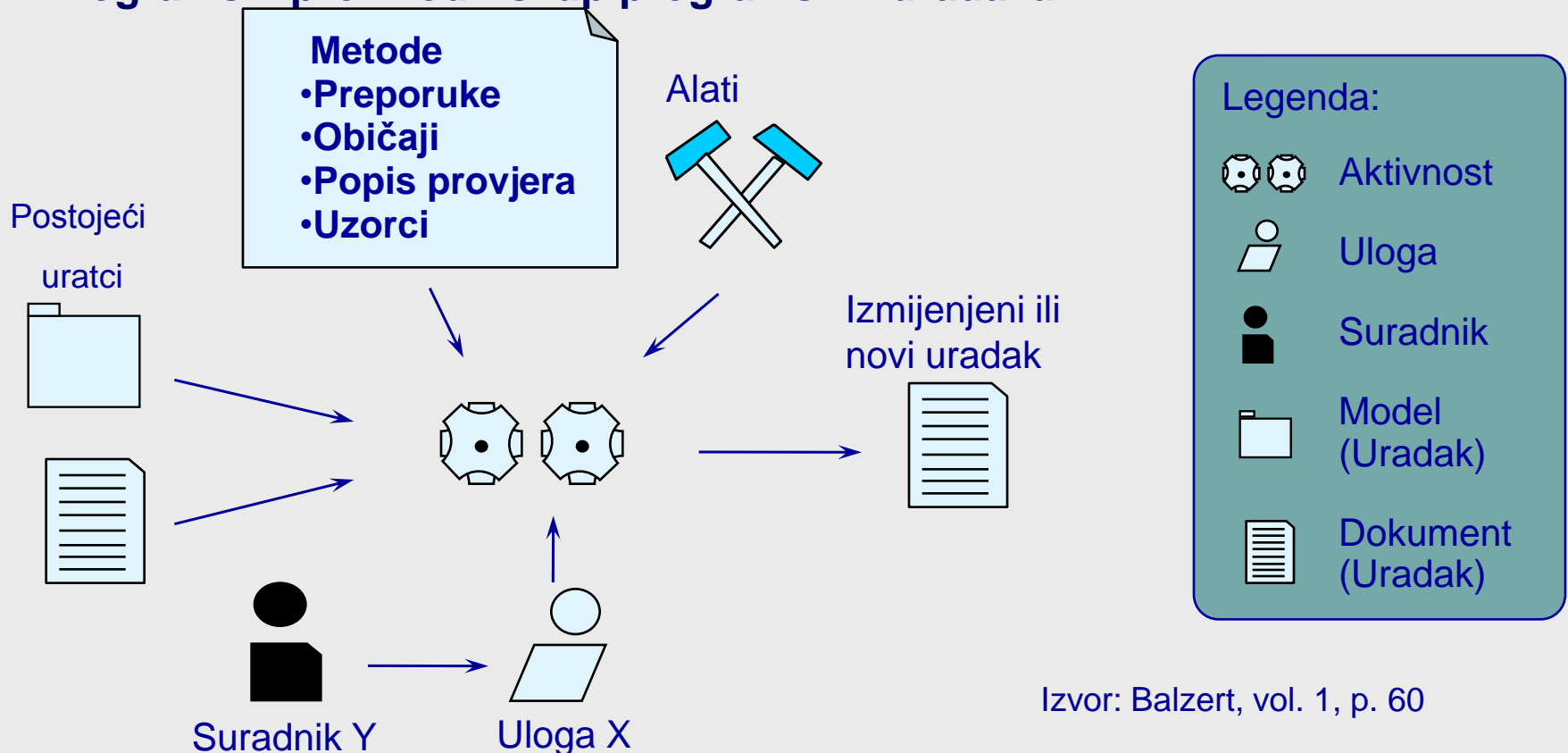
- Modeli programskih procesa, Modeli faza, Modeli životnih ciklusa, Modeli razvoja softvera, Modeli projekta
- nazivi su slični, ali ima razlika, npr.
 - model životnog ciklusa: uključuje razvoj i održavanje
 - model razvoja softvera: bez održavanja

❑ SDLC = software/systems development life-cycle

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

Aktivnosti procesa

- ❑ Aktivnost - podproces u modelu procesa
- ❑ Uloga - suradnik koji obavlja određeni posao, npr. voditelj projekta, arhitekt / specijalist za projektiranje, programer, administrator BP
- ❑ Programski uradak - dokument, model ili program
- ❑ Programski proizvod - skup programskih uradaka



Izvor: Balzert, vol. 1, p. 60

Životni ciklus programske potpore



Faze životnog ciklusa

☐ Planiranje

- Utvrđivanje ciljeva (poslovne koristi)
- Analiza izvedivosti
- Izrada plana rada
- Ekipiranje projekta
- Upravljanje projektom

☐ Analiza

- Prikupljanje informacija
- Modeliranje procesa
- Modeliranje podataka
- Specifikacija zahtjeva

☐ Projektiranje, oblikovanje

- Dizajn arhitekture
- Dizajn sučelja
- Dizajn baze podataka i datoteka
- Dizajn programa

☐ Izrada, ugradnja (implementacija)

- Konstrukcija
- Instalacija

☐ Primjena

- Rad
- Održavanje

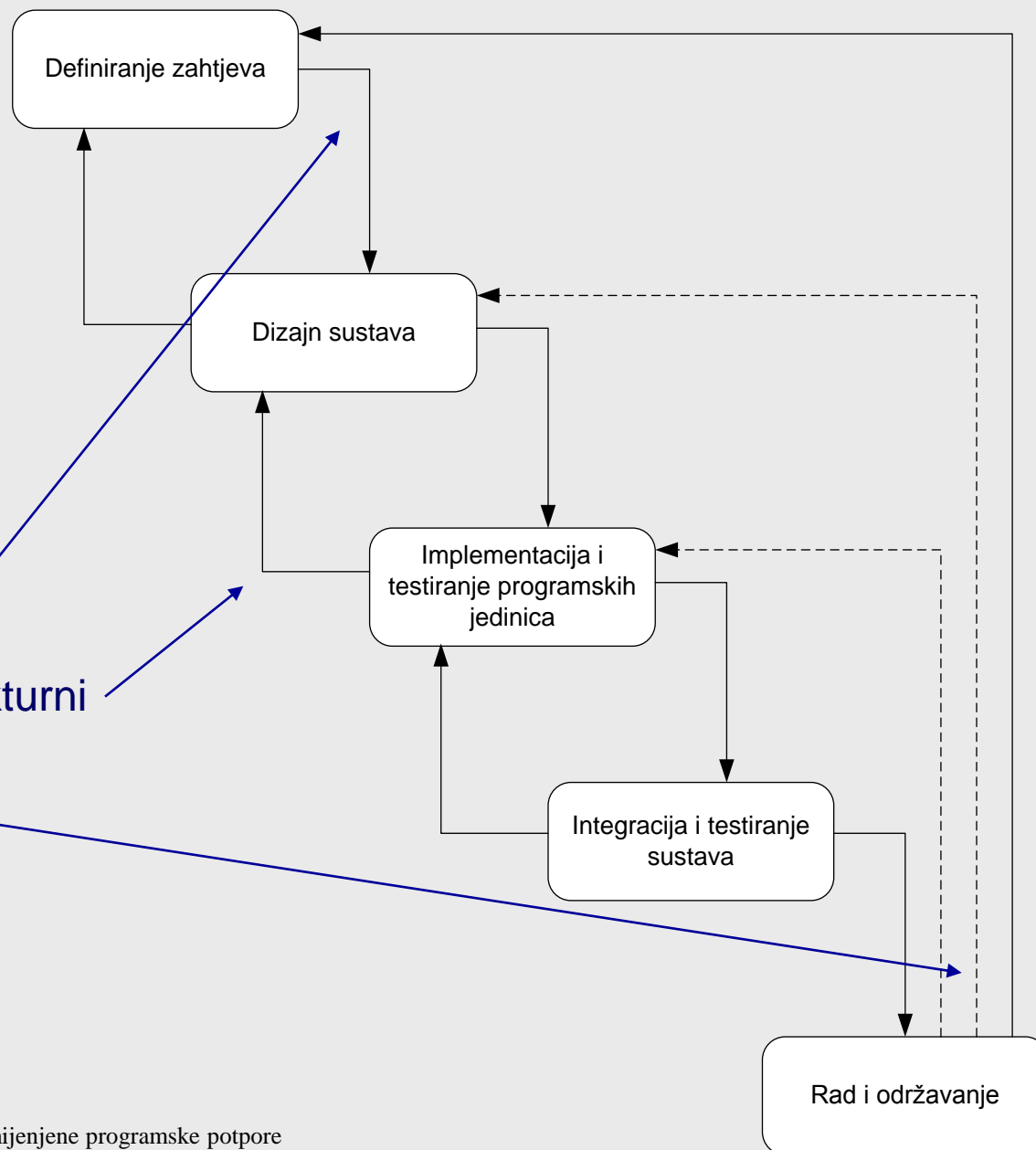
Vodopadni model razvoja

- ❑ **Definiranje zahtjeva (*requirements analysis and definition*)**
 - definira funkcionalnost programske potpore prema zahtjevima korisnika.
- ❑ **Dizajn sustava (*system and software design*)**
 - definira cjelokupnu arhitekturu programske podrške
 - grubi model sustava razrađuje se u detaljni opis izvedbe
- ❑ **Ugradnja i testiranje jedinica (*implementation and unit testing*)**
 - faza kodiranja tijekom koje se zahtjevi prevode u programski kod
 - programske jedinice zasebno se testiraju provjerom naspram specifikacije
- ❑ **Integracija i testiranje sustava (*integration and system testing*)**
 - programske jedinice povezuju se u cjelinu
 - provjerava se odgovara li programska potpora zahtjevima korisnika
- ❑ **Primjena i održavanje (*deployment/operation and maintenance*)**
 - započinje predajom sustava korisnicima na uporabu
 - tijekom održavanja uklanjaju se naknadno uočene neispravnosti te se sustav proširuje i poboljšava prema potrebama

Vodopadni model (waterfall)

❑ Varijante

- klasični
- pseudostrukturni
- radikalni



Varijante vodopadnog modela

❑ Klasični vodopadni model

- slijedno napredovanje iz faze u fazu
- razvoj započinje tek kad su svi zahtjevi dobro dokumentirani
- nisu dozvoljene naknadne promjene rezultata prethodnih faza
- problem u slučaju pogrešaka ili novih/promijenjenih zahtjeva
- uvođenje prema gore (bottom up): moduli, podsustavi, sustav
- sustav nije upotrebljiv dok nije gotov u potpunosti
- problem predodžbe o produktu na temelju pisane specifikacije

❑ Pseudostrukturni vodopadni model

- povratna veza i mogućnost promjene rezultata prethodnih faza
- uvođenje prema dolje: moduli na višim, pa na nižim razinama
- primjena tehnika strukturiranog programiranja

❑ Strukturni (radikalni)

- aktivnosti različitih faza mogu se obavljati istovremeno
- korištenje rječnika podataka, 4GL i generatora aplikacija
- prikladan kada se unaprijed ne zna konačni izgled sustava

Razvoj prototipa

❑ Prototipiranje (prototyping)

- Koristi se kada korisnik ne može točno definirati svoje informacijske potrebe prije nego što se izgradi informacijski sustav.
- Razlog tomu može biti nedostatak postojećeg sustava na kojem bi korisnik zasnivao svoje potrebe ili pak teška vizualizacija budućeg sustava.

❑ Izrada prototipa

- Izgrađuje se sustav koji zadovoljava neke osnovne, inicijalne potrebe.
- U radu s takvim sustavom korisnik otkriva svoje informacijske potrebe te se sustav modificira kako bi se zadovoljile te potrebe.
- Postupak korištenje sustava i modificiranja istog iterativno se ponavlja, a informacijske potrebe korisnika otkrivaju korištenjem sustava.

❑ Izrada prototipa pogodna je u onim okruženjima gdje je teško definirati konkretni model sustava te u okruženjima gdje se informacijske potrebe korisnika mijenjaju ili razvijaju.

Brzo prototipiranje (rapid prototyping)

❑ funkcionalni prototip

- nedovršeni proizvod koji simulira rad pravog sustava
- zamjenom pravim rješenjima nastaje radni sustav

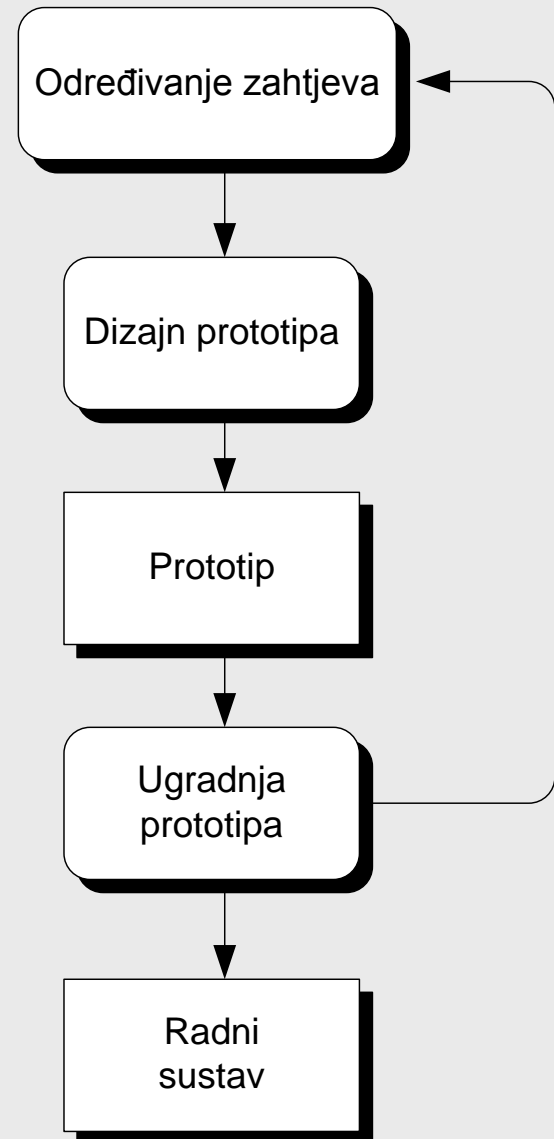
❑ prikladno za male projekte (tzv. one-man)

❑ Prednosti:

- iteracije promjena – korisnici se smiju predomisliti
- povećanje kreativnosti i brzine razvoja

❑ Nedostaci:

- “zaboravljanje” da prototip nije pravi sustav
- mogući neuspjeh zamjene prototipa radnim sustavom
- dokumentacija proizlazi iz izrade, uz opasnost da pisane specifikacije nikad neće biti napravljene
- nemogućnost ispravne procjene i planiranja resursa



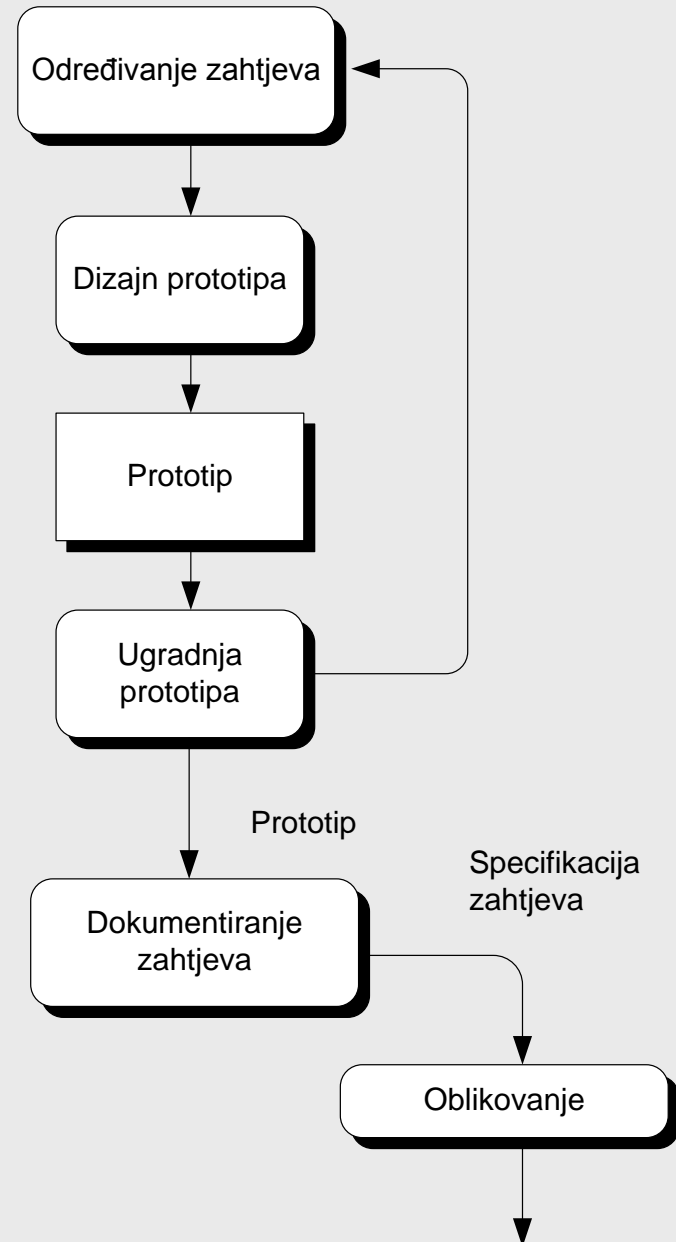
Ograničeno/strukturirano (constrained prototyping)

❑ nefunkcionalni prototip

- prikaz izgleda

❑ izrada prototipa kao sredstvo određivanja zahtjeva

- u određenom trenutku se prekida i slijedi faza oblikovanja sustava



Iterativni postupak razvoja

❑ Unified software development process (UDP)

- izvorno Objectory
- danas IBM Rational Unified Process (RUP)

❑ Iterativni i inkrementalni razvoj

- softver se razvija i objavljuje po dijelovima
- glavne faze obavljaju se kroz niz iteracija
 - svaka iteracija obavlja se standardnim životnim ciklusom koji uključuje analizu, oblikovanje, ugradnju i provjeru
 - rezultat iteracije je proizvod završne kakvoće (production-quality), provjeren i integriran, koji zadovoljava podskup ukupnih zahtjeva
 - isporuke mogu biti interne ili prema korisnicima

❑ RUP sadrži niz "predložaka" razvojnih procesa (roadmaps) za različite modele razvoja i tipove projekata

Faze i koraci razvoja

Core Workflows (Disciplines)

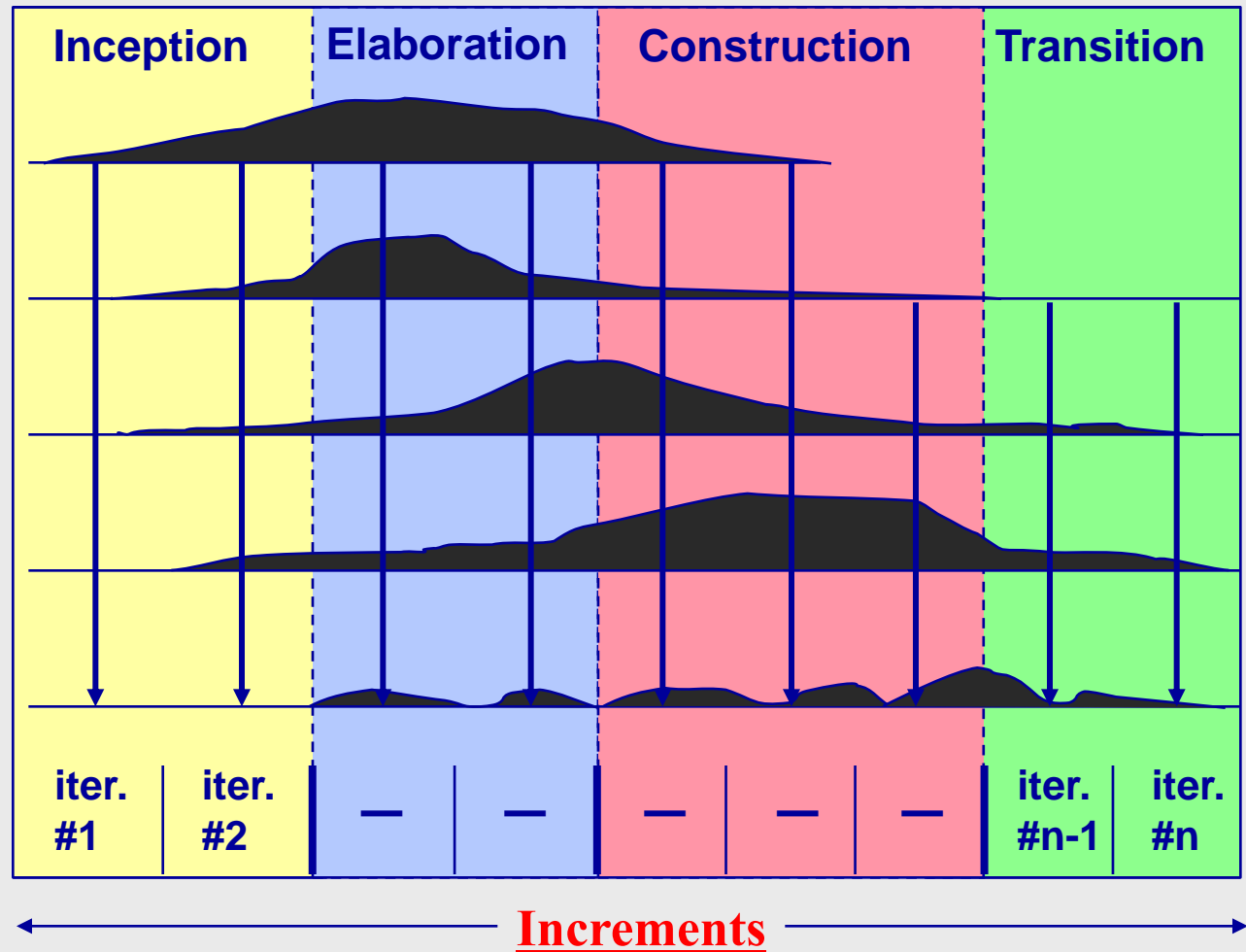
Requirements

Analysis

Design

Implementation

Testing



© Rational Software

Glavne faze razvoja

❑ Počinjanje (Inception)

- opravdanje razloga za pokretanje projekta
- prikupljanje najvažnijih zahtjeva (10% detaljno)
- određivanje dosega projekta

❑ Elaboracija (Elaboration)

- prikupljanje detaljnih zahtjeva (80%)
- globalna (high-level) analiza i dizajn
- ustanovljavanje osnovne arhitekture
- planiranje konstrukcije

❑ Konstrukcija, gradnja (Construction)

- prikupljanje ostalih zahtjeva + promjene zahtjeva
- razrada arhitekture i izrada sustava
- kontinuirana integracija

❑ Prijelaz (Transition)

- beta testiranje, podešavanje performansi, poduka korisnika
- provjera prihvatljivosti i zadovoljstva korisnika

❑ Post-implementacija (Post-deployment)

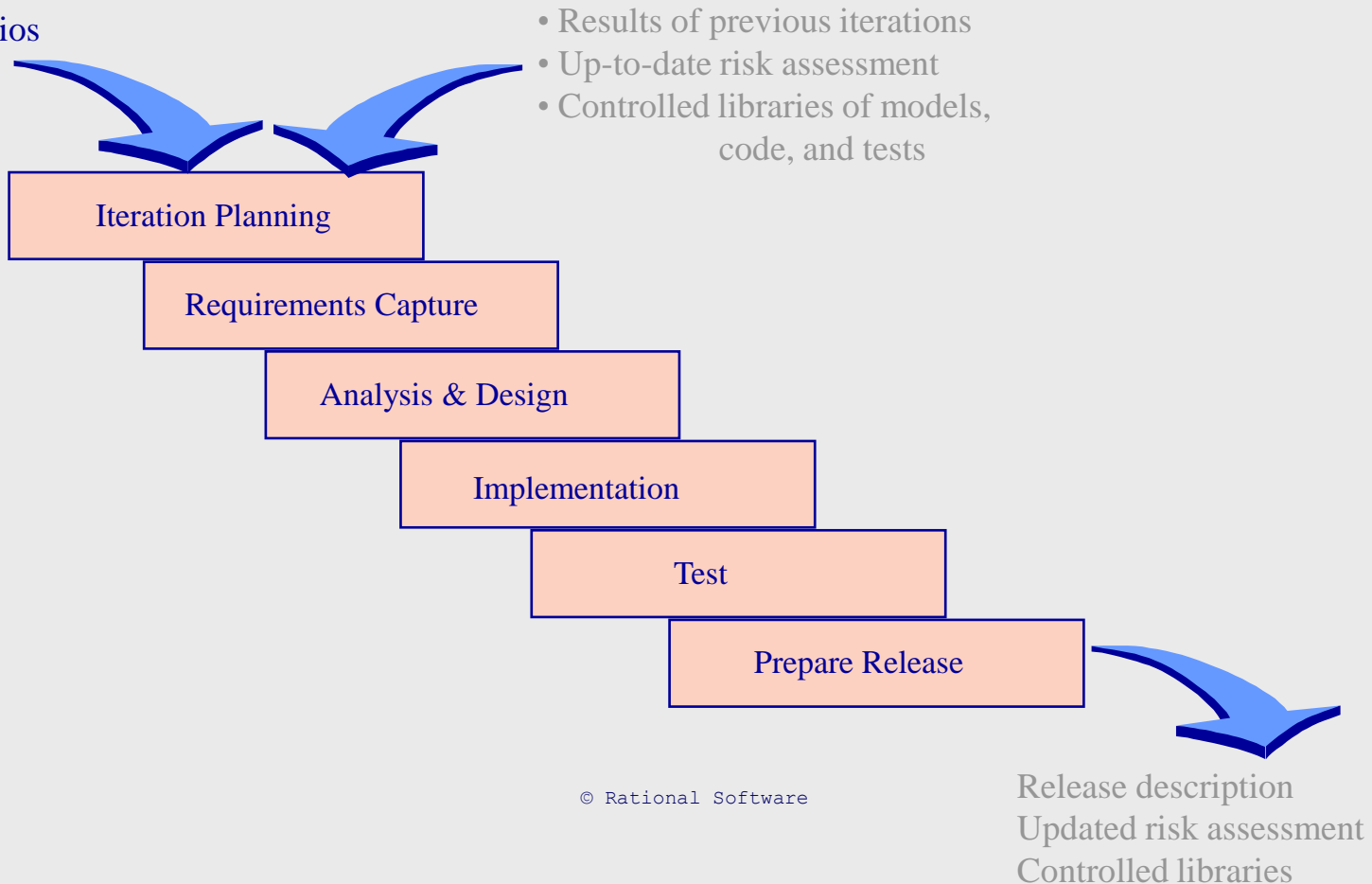
- nastavak evolucijskog razvoja
- uz očuvanje integriteta aplikacije

Životni ciklus iteracije

❑ Mini-vodopadni model razvoja (Mini-Waterfall)

- usitnjeni standardni životni ciklus razvitka
- zasnovan i vođen na slučajevima korištenja

Selected scenarios



Aktivnosti životnog ciklusa iteracije

❑ Planiranje iteracije (Iteration planning)

- određivanje svrhe (ciljeva)
- određivanje kriterija za provjeru
- izrada detaljnog plana koji se uključuje u opći plan razvoja
 - kontrolne točke za nadzor napretka (milestones)
 - postupno rješavanje i pregled

❑ Određivanje zahtjeva (Requirements Capture)

- određivanje slučajeva korištenja koji će se ugraditi u iteraciji
- ažuriranje objektnog modela "otkrivenim razredima" i vezama
- izrada plana provjere za iteraciju

Aktivnosti životnog ciklusa iteracije

❑ Analiza i oblikovanje (Analysis & Design)

- određivanje razreda koji će biti ugrađeni u iteraciji
- ažuriranje objektnog modela
- ažuriranje dokumentacije arhitekture
- početak razvoja procedura provjere

❑ Izrada (Implementation)

- automatsko generiranje koda na temelju modela dizajna
- ručno generiranje koda za operacije
- kompletiranje test procedura
- provjera elemenata i integracijska provjera

Aktivnosti životnog ciklusa iteracije

❑ Provjera (Test)

- integracija i provjera s ostalim dijelovima (prijašnjim izdanjima)
- obuhvat i pregled rezultata testiranja
- provjera rezultata provjera prema kriterijima provjere
- prosudba (ocjena) iteracije

❑ Priprema izdanja

- sinkronizacija koda i modela dizajna
- pohrana proizvoda iteracije u knjižnice

Određivanje iteracija

❑ Broj i trajanje iteracija

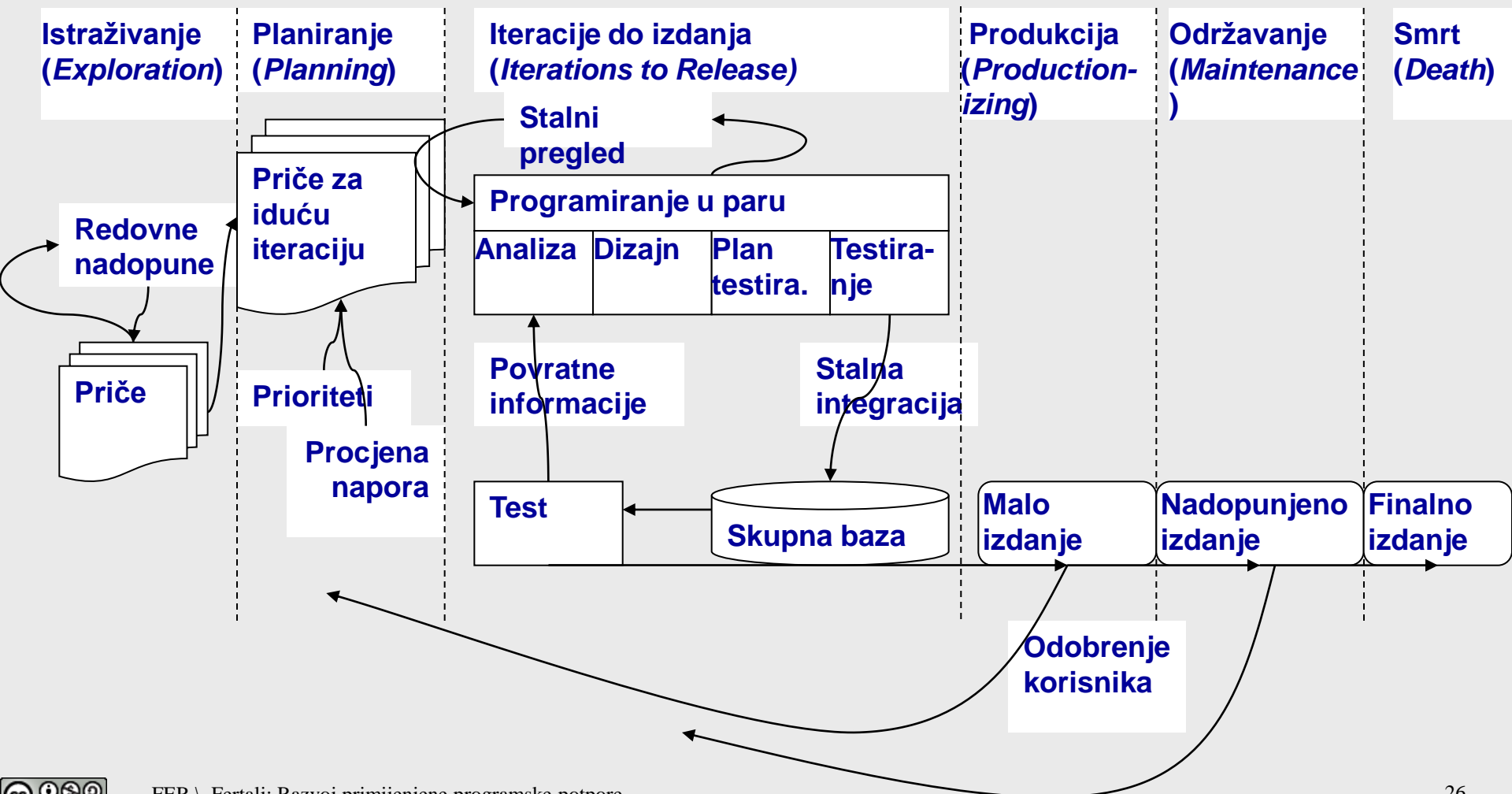
- za projekte do 18 mjeseci, otprilike 3 do 6 iteracija
- uobičajeno podjednakog trajanja
- mogu varirati ovisno o fazi (u konstrukciji duže)

❑ Prva iteracija

- najčešće najteža
- zahtijeva pripremu okruženja, ekipe i posla
- opasna zbog moguće pretjeranog optimizma
- ako se krivo procijeni može izazvati pomake i neželjene učinke (smanjenje broja iteracija, ukupno slabije rezultate)

Ekstremno programiranje

- ❑ Agilni razvojni proces
- ❑ Vrijednosti (values): komunikacija, jednostavnost, povratne informacije, hrabrost, uvažavanje (respect)



Faze ekstremnog programiranja

❑ Istraživanje

- Korisnici bilježe svoje priče na kartice
- Svaka kartica sadrži jednu mogućnost programa.
- Projektni tim se pobliže upoznaje s alatima, tehnologijom i postupcima projekta.
- Radi se prototip sustava za testiranje tehnologije i varijanti arhitekture sustava.
- Faza istraživanja traje nekoliko tjedana do nekoliko mjeseci

❑ Planiranje

- Postavlja prioritete na korisničke priče (tj. svojstva programskog rješenja)
- Planira se doseg prvog malog izdanja i vrijeme za pojedinu karticu
- Zatim se određuje cjelokupni vremenski raspored.
- Rok za izdavanje prvog malog izdanja obično je unutar dva mjeseca.
- Faza planiranja traje nekoliko dana.

Faze ekstremnog programiranja

❑ Iteracije do izdanja

- Uključuje nekoliko iteracija sustava prije prvog izdanja.
- Vremenski raspored iz faze planiranja se razlaže u više iteracija
- Pojedina iteracija traje jedan do četiri tjedna.
- Prva iteracija stvara takav sustav koji 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.

❑ Produkcija

- Dodatno testiranje i provjera performansi sustava prije isporuke klijentu.
- Razrješenje primjedbi na sustav te odlučivanje da li će se riješiti u ovom izdanju.
- Iteracije trajanja tri do najviše tjedan dana.
- Zakašnjele nove ideje i prijedlozi se dokumentiraju i njihova implementacija odgađa.

Faze ekstremnog programiranja

- ❑ **Nakon što je prvo izdanje pušteno u produkciju,**
 - XP projekt mora istovremeno održavati softver u primjeni i proizvoditi nove iteracije
 - Zbog toga se brzina implementacije smanjuje
 - Održavanje može zahtijevati nove članove projektnog tima i promjenu strukture tima.

- ❑ **Faza smrti je blizu kada klijent nema više novih kartica s pričama**
 - Podrazumijeva se da sustav zadovoljava sve zahtjeve (npr. pouzdanost i stabilnost).
 - Vrijeme u XP projektu da se konačno napiše sva korisnička dokumentacija budući da više nema promjena na arhitekturi, dizajnu i kodu sustava.
 - Smrt može nastupiti i kada sustav ne ispunjava sva korisnička očekivanja, ili ako postane preskup za daljnji razvoj.

Reference

☐ Software Engineering, Not Computer Science

- *“A scientist builds in order to learn; an engineer learns in order to build.”
(Fred Brooks)*
- <http://stevemcconnell.com/psd/04-senotcs.htm>

☐ <http://www.sei.org/>

☐ <http://www.rspa.com/>

☐ <http://www.comp.lancs.ac.uk/computing/resources/lanS/>

☐ <http://www-306.ibm.com/software/rational/>

☐ <http://www.extremeprogramming.org/>

☐ <http://www.agilemodeling.com/>



Priprema za projekt

☐ Formiranje projektnih ekipa

- Formira predavač nakon što dobije potpunu evidenciju s upisa

☐ Proglasiti voditelja

- Ekipa proglasi voditelja
- Voditelj se dojavljuje e-poštom koordinatoru predmeta
- Voditelj po želji preuzima instalacijski CD/DVD

☐ Instalirati VS.NET i dodatne komponente

- svatko za sebe prema RPPP-UputeTFS s www.fer.hr/predmet/rppp

☐ Intervju i korisnička priča

- uvodno iznošenje zahtjeva korisnika na 2. predavanjima
- pojašnjenje zahtjeva korisnika na 3. predavanjima
- dodatne informacije na labosu i konzultacijama

Osnove platforme .NET i jezika C#

Jezik C# i *.NET Framework*

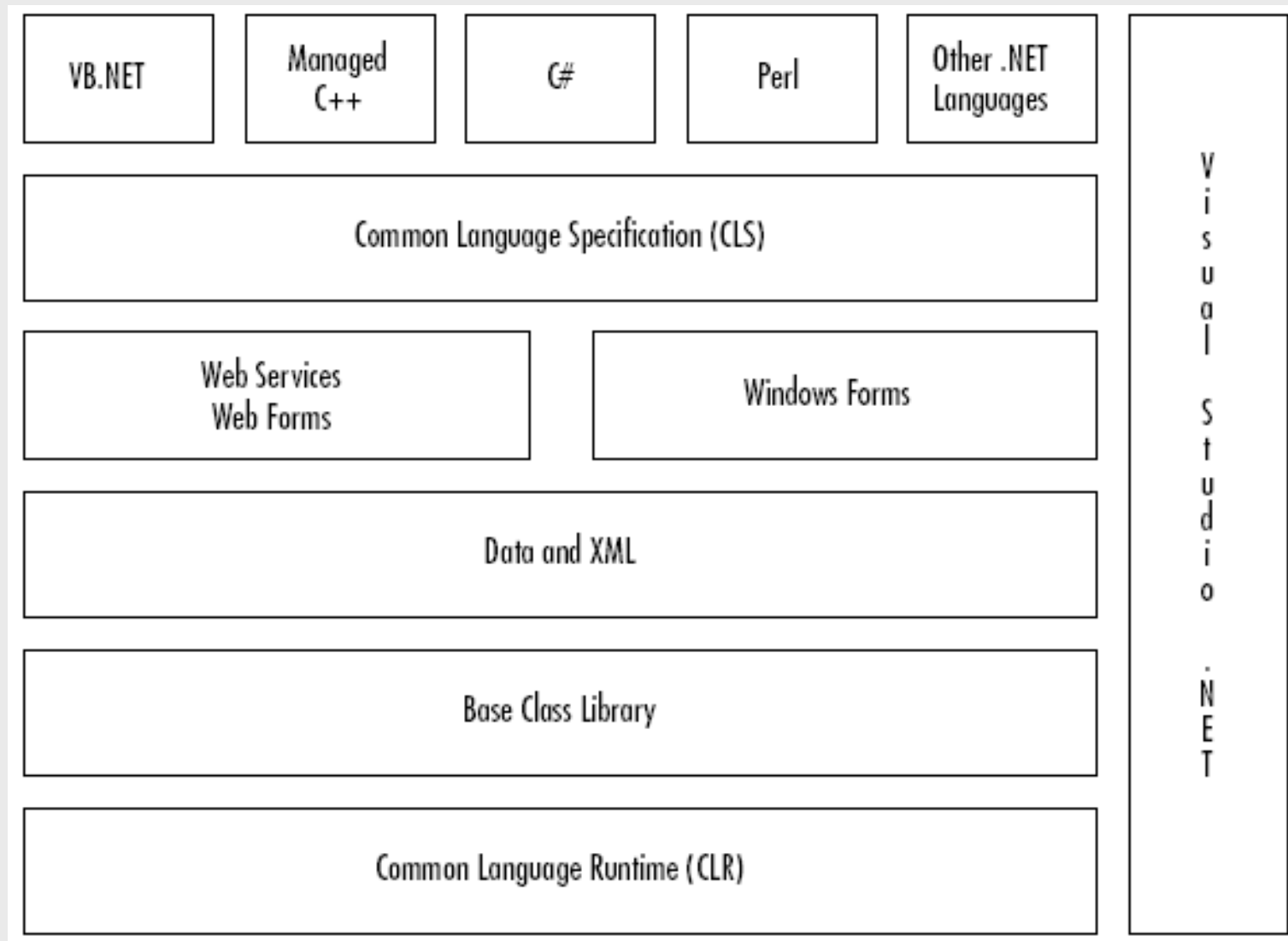
❑ Jezik C# - Anders Hejlsberg et al.

- vođen događajima (event-driven),
- objektno usmjeren (object-oriented),
- otvoren mreži (network-aware),
- vizualan (visual) – interaktivno sučelje razvojne okoline i aplikacija

❑ Microsoft .NET Framework

- nastao s idejom iste osnovice za izradu lokalnih i Internet aplikacija
- neovisnost o jeziku
 - jezici Visual Basic .NET, Visual C++ .NET, C# i drugi
 - zajednička knjižnica osnovnih razreda Base Class Library (BCL) ili Framework Class Library (FCL)
 - zajednički, opći tipovi podataka Common Type System (CTS)
 - zajednički pogon programa - Common Language Runtime (CLR)
- neovisnost o platformi ... na kojoj postoji CLR
 - iako zamišljena, .NET je namijenjen uglavnom Windows platformi

Arhitektura .NET



Microsoft .NET Framework

❑ C# prevoditelj

- prevodi C# izvorni kod (.cs datoteke) u poseban međujezik MSIL

❑ MS Intermediate Language (MSIL)

- MSIL se izvodi u virtualnom stroju a ne izravno na procesoru računala

❑ Common Language Runtime (CLR)

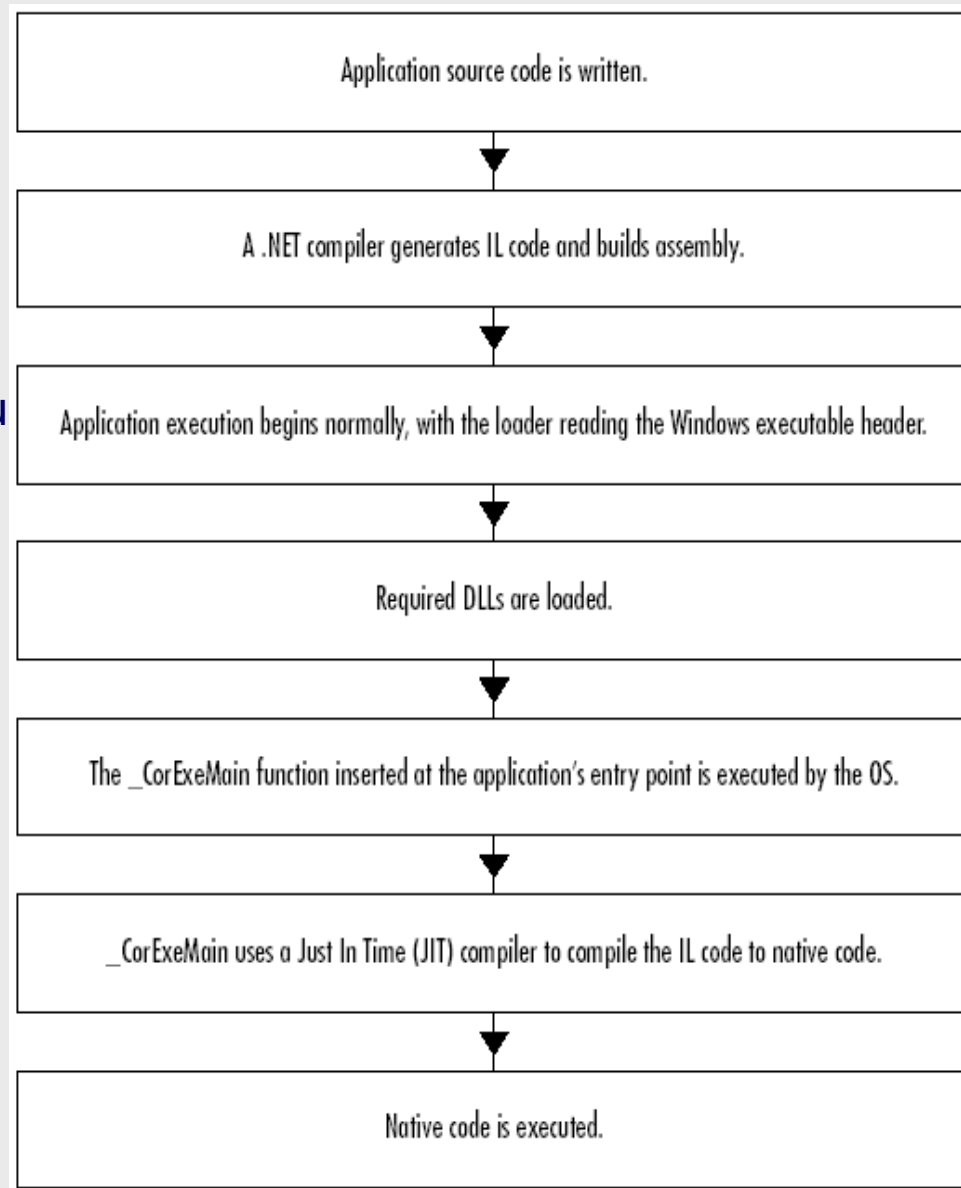
- zajednički pogon programa
- interpretira MSIL naredbe
- koristi JIT prevodilac

❑ Just-In-Time compiler

- pri prvom pokretanju programa prevodi MSIL u strojni kod

❑ Asemblij (Assembly)

- skup prevedenih razreda



Elementi jezika

❑ Identifikatori

- nazivi razreda, varijabli, ... koje određuje programer
- sastoje se od slova, znamenki i podcrte, prvi znak mora biti slovo
- npr. System, Pozdrav

❑ Ključne riječi (keywords)

- rezervirane riječi jezika, posebni identifikatori koji u jeziku imaju predefinirano značenje
- definicije objekata, naredbe i direktive za prevođenje
- npr. using, class, static

❑ Programske strukture

- odvajanje naredbi s ;
- blokovi programa - { }

❑ C# razlikuje velika i mala slova!

❑ Komentari

- Retkovni komentar – tekst od znakova "//" do kraja retka
- Blok komentar – tekst oblika /* ... */, može se rasprostirati u više redaka

Struktura C# programa

☐ Program

- sadrži jedan ili više razreda

☐ Razred (klasa, *class*)

- sadrži članove – svojstva (varijable) i metode tj. postupke (potprograme)

☐ Ne postoje slobodni potprogrami ni globalne varijable

- sve se zbiva unutar tijela razreda

☐ Ulazna točka (entry point)

- postupak `static void Main`

☐ Ne koristi se `#include`

- svaki prevedeni razred ima manifest - definiciju sučelja koju mogu koristiti drugi programi, uključujući one napisane u nekom drugom jeziku

☐ Primjer Osnove\PozdravKonzola

```
// korištenje knjižnice
using System;

class Pozdrav// zaglavlje razreda
// tijelo razreda
{
    // metoda Main
    static void Main(string[] args)

    // blok naredbi
    {
        // naredba WriteLine

        Console.WriteLine("Pozdrav!");
    }
}
```

Konzolna i grafička aplikacija

❑ Konzolna aplikacija:

📁 Osnove\PozdravKonzola

- **Prostor imena System**
- **Postupak Main**

```
using System;

class Pozdrav
{
    static void Main(string[] args)
    {
        Console.WriteLine("Pozdrav!");
    }
}
```

Ispis na konzolu

❑ Grafička aplikacija:

📁 Osnove\PozdravGraficka

Argumenti
komandne linije

```
using System;

class Pozdrav
{
    static void Main(string[] args)
    {
        System.Windows.Forms.
            MessageBox.Show("Pozdrav!");
    }
}
```

Ispis na prozor

Ponovno korištenje koda (reuse) drugog programa

❑ Poziv metode istog razreda

- izvorna datoteka:
 - vozdra.cs
- prevođenje:
 - csc vozdra.cs
- program:
 - vozdra.exe
- ispis:
 - Pozdrav!
 - C#

```
public class Vozdra {  
    static void Main() {  
        System.Console.WriteLine("Pozdrav!");  
        VisokiC ();  
    }  
    public static void VisokiC() {  
        System.Console.WriteLine("C#");  
    }  
}
```

❑ Korištenje drugog, prethodno prevedenog razreda

- izvorna datoteka:
 - proba.cs
- prevođenje:
 - csc /reference:vozdra.exe proba.cs
- program:
 - proba.exe
- ispis:
 - C#

```
class Proba {  
    static void Main () {  
        Vozdra.VisokiC();  
    }  
}
```



Ponovno korištenje koda unutar istog programa

❑ Primjer Osnove\Reuse, Program.cs i Pozdrav.cs

```
using System;

public class Pozdrav
{
    public Pozdrav()
    {
        Poziv("iz lokalnog");
    }

    public static void
        Poziv(string poruka)
    {
        System.Console.WriteLine(
            "Pozdrav " + poruka);
    }
}
```

```
using System;

class Proba
{
    static void Main()
    {
        Console.WriteLine("Glavni!");
        Pozdrav.Poziv("iz glavnog");
        Pozdrav p = new Pozdrav();
    }
}
```

Ključne riječi

- ❑ Sve ključne riječi definirane su malim slovima !!

abstract	as	base	bool	break
byte	case	catch	char	checked
class	const	continue	decimal	default
delegate	do	double	else	enum
event	explicit	extern	false	finally
fixed	float	for	foreach	get
goto	if	implicit	in	int
interface	internal	is	lock	long
namespace	new	null	object	operator
out	override	params	private	protected
public	readonly	ref	return	sbyte
sealed	set	short	sizeof	stackalloc
static	string	struct	switch	this
throw	true	try	typeof	uint
ulong	unchecked	unsafe	ushort	using
value	virtual	void	volatile	while

Standardni ulaz/izlaz

☐ Čitanje standardnog ulaza

- `public static string ReadLine();`
 - `Console.ReadLine()` – čita niz znakova
- `public static int Read();`
 - `Console.Read()` – čita znak

☐ Pisanje na standardni izlaz

- `public static void Write(...);`
 - piše argument(e) i ostaje u istom retku
 - `Console.Write("Pozdrav! ");`
- `public static void WriteLine(...);`
 - piše argument(e) i znak za skok u novi redak
 - `Console.WriteLine("Pozdrav! ");`

☐ Primjeri primjene slijede u narednim programima

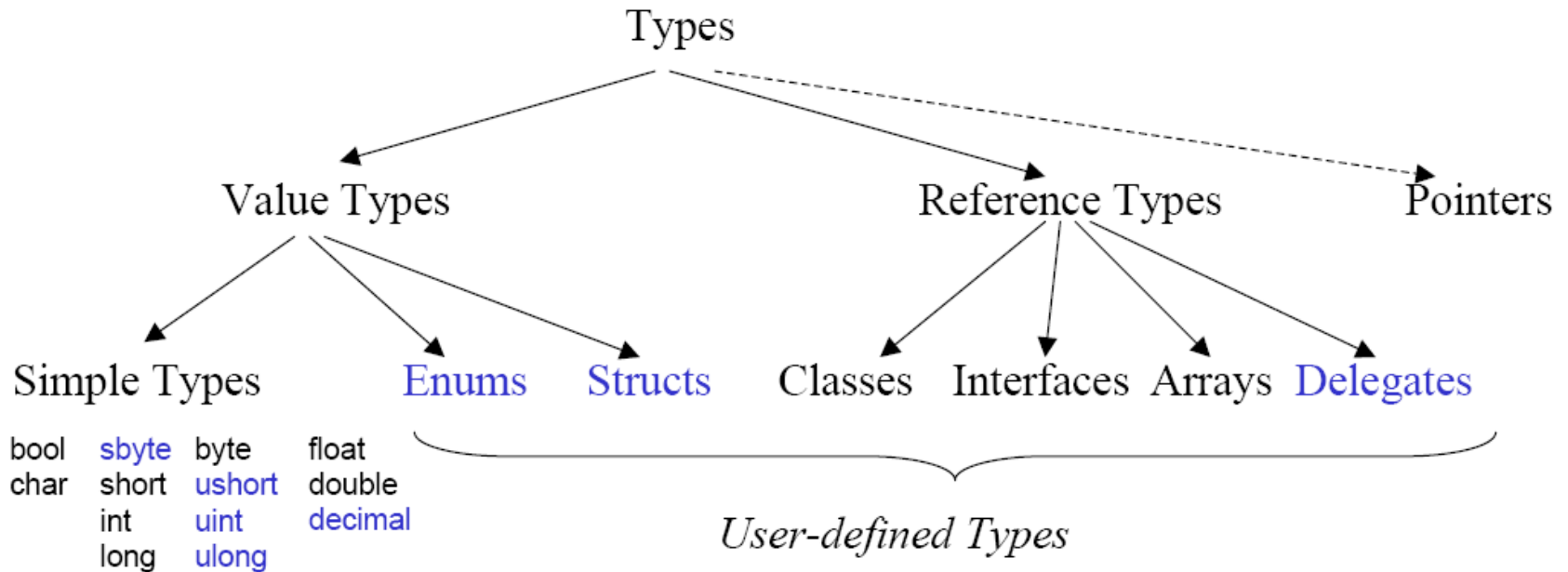
☐ Za više informacija pogledati Help

Formatirani ispis, numerički formati

Character	Description	Examples	Output
C or c	Currency	<code>Console.Write("{0:C}", 2.5);</code> <code>Console.Write("{0:C}", -2.5);</code>	\$2.50 (\$2.50)
D or d	Decimal	<code>Console.Write("{0:D5}", 25);</code>	00025
E or e	Scientific	<code>Console.Write("{0:E}", 250000);</code>	2.500000E+005
F or f	Fixed-point	<code>Console.Write("{0:F2}", 25);</code> <code>Console.Write("{0:F0}", 25);</code>	25.00 25
G or g	General	<code>Console.Write("{0:G}", 2.5);</code>	2.5
N or n	Number	<code>Console.Write("{0:N}", 2500000);</code>	2,500,000.00
X or x	Hexadecimal	<code>Console.Write("{0:X}", 250);</code> <code>Console.Write("{0:X}", 0xffff);</code>	FA FFFF

Tipovi podataka

- ❑ C# ima zajednički sustav tipova – Common Type System (CTS)
- ❑ Svi tipovi izvode se iz osnovnog tipa System.Object
 - "sve je objekt"



Tipovi podataka

❑ Osnovne kategorije tipova podataka:

- Vrijednosti (value types) – sadrže podatke
 - osnovni tipovi (int, float, char...), enum, struct
- Reference (reference types) – pokazivači (pointers)
 - razredi (class types),
 - sučelja (interface types),
 - delegati (delegate types),
 - polja (array types) ,
 - znakovni nizovi (strings)

❑ Varijable i pridruživanje vrijednosti

- varijabla - imenovana memorijska lokacija određenog tipa i poznate veličine te vrijednosti koja se u njoj nalazi a koja se može promijeniti
- prije uporabe (zadavanja ili korištenja vrijednosti) varijablu treba deklarirati

```
int broj1;      // <type> <variable-name>;  
broj1 = 45;     // <variable-name> = <exp>
```

Ugrađeni tipovi podataka

C# Type	.Net Framework (System) type	Broj bajtova	Interval brojeva
sbyte	System.Sbyte	1	-128 do 127
short	System.Int16	2	-32768 do 32767
int	System.Int32	4	-2147483648 do 2147483647
long	System.Int64	8	-9223372036854775808 do 9223372036854775807
byte	System.Byte	1	0 do 255
ushort	System.UInt16	2	0 do 65535
uint	System.UInt32	4	0 do 4294967295
ulong	System.UInt64	8	0 do 18446744073709551615
float	System.Single	4	$\pm 1.5 \times 10^{-45}$ do $\pm 3.4 \times 10^{38}$ 7 točnih znamenki
double	System.Double	8	$\pm 5.0 \times 10^{-324}$ do $\pm 1.7 \times 10^{308}$ 15 točnih znamenki
decimal	System.Decimal	12	$\pm 1.0 \times 10^{-28}$ do $\pm 7.9 \times 10^{28}$ 28 točnih znamenki
char	System.Char	2	Unicode znak (16 bit)
bool	System.Boolean	1 2	true ili false

Aritmetički operatori

Operator	Značenje
+	zbiranje
-	oduzimanje
*	množenje
/	dijeljenje
%	ostatak (modulo) pri dijeljenju (radi i s realnim operandima)
++	operator uvećanja za 1
--	operator smanjenja za 1

- ☐ **Znak plus (+) također predstavlja operator ulančavanja (konkatenacije) nizova znakova**

Operatori usporedbe

❑ Usporedbeni (relacijski) operatori

- Operatori `==` i `!=` mogu se koristiti sa svim tipovima podataka
- ostali usporedbeni operatori mogu koristiti samo s tipovima podataka koji podržavaju usporedbene operatore.
- Npr. `bool` vrijednosti ne mogu se uspoređivati operatorima `<`, `<=`, `>`, `>=`
- Pri usporedbi referenci uspoređuju se adrese (pokazivači) na objekte
 - Izuzetak su nizovi znakova.

Operator	Značenje
<code>==</code>	jednako
<code>!=</code>	različito
<code>></code>	veće
<code>>=</code>	veće ili jednako
<code><</code>	manje
<code><=</code>	manje ili jednako

Logički operatori

- ❑ **Logički operatori (bool i bitovni)**
 - Operandi moraju biti tipa bool i rezultat logičke operacije je tipa bool.
 - U ovu grupu mogu se pribrojiti i unarni *true* i *false*
- ❑ **Operatori & i | uvijek ispituju vrijednost svih operanada u izrazu.**
- ❑ **Reducirani operatori && i || uvijek ispituju logičku vrijednost samo prvog operanda a ostalih po potrebi.**

Operator	Značenje
&	AND
	OR
^	XOR (isključivi OR)
&&	reducirani AND
	reducirani OR
!	NOT
~	komplement

Ostali operatori

- ❑ Operator pridruživanja: =
- ❑ Skraćeno pridruživanje:
 - +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=
- ❑ Posmak: <<, >>
- ❑ Uvjetni operator: (?:) // *cond-expr ? expr1 : expr2*
- ❑ Veličina tipa: sizeof(*type*)
- ❑ Tip objekta:
 - npr. `System.Type type = typeof(int);`
- ❑ Operator is - expression is type
 - provjerava da li je izraz određenog tipa
- ❑ Pristup članu: . (točka) // *name1 . name2*
- ❑ Indeksiranje: []
- ❑ Indirekcija i adresa: * -> [] &

```
int n = 3;
if (n is int)
...
```

Prioritet operatora

Category	Operators
Primary	<code>(x)</code> , <code>x.y</code> , <code>f(x)</code> , <code>a[x]</code> , <code>x++</code> , <code>x--</code> , <code>new</code> , <code>typeof</code> , <code>sizeof</code> , <code>checked</code> , <code>unchecked</code>
Unary	<code>+</code> , <code>-</code> , <code>!</code> , <code>~</code> , <code>++x</code> , <code>--x</code> , <code>(T)x</code>
Multiplicative	<code>*</code> , <code>/</code> , <code>%</code>
Additive	<code>+</code> , <code>-</code>
Shift	<code><<</code> , <code>>></code>
Relational	<code><</code> , <code>></code> , <code><=</code> , <code>>=</code> , <code>is</code>
Equality	<code>==</code>
Logical AND	<code>&</code>
Logical XOR	<code>^</code>
Logical OR	<code> </code>
Conditional AND	<code>&&</code>
Conditional OR	<code> </code>
Conditional	<code>?:</code>
Assignment	<code>=</code> , <code>*=</code> , <code>/=</code> , <code>%=</code> , <code>+=</code> , <code>-=</code> , <code><<=</code> , <code>>>=</code> , <code>&=</code> , <code>^=</code> , <code> =</code>

Razredi

```
class Tocka
```

```
{
```

```
    public int cx;
```

```
    public int cy;
```

```
    public Tocka(int x, int y){
```

```
        cx = x; cy = y;
```

```
    }
```

```
    public int Kvadrant(){
```

```
        if (cx > 0){
```

```
            if (cy > 0) return 1;
```

```
            else return 4;
```

```
        }else
```

```
        {
```

```
            if (cy > 0) return 2;
```

```
            else return 3;
```

```
        }
```

```
    }
```

```
}
```

❏ Primjer 📁 Osnove\RazredTocka

→ Javne varijable

→ Konstruktor

→ Javni postupak

↑ Instanciranje objekta

```
Tocka t = new Tocka(1,-2);  
Console.WriteLine("Točka ({0},{1})  
je u {2}.kvadrantu.", t.cx,  
t.cy, t.Kvadrant());
```

↓ Poziv postupka

Vrijednosti i reference

- ❑ **Koju vrijednost imaju varijable `val1` i `val2` nakon sljedećeg programskog odsječka?**

```
int val1 = 0;
int val2 = val1;
val2 = 5;
```

- ❑ **Koju vrijednost imaju varijable `ref1` i `ref2` nakon sljedećeg programskog odsječka?**

```
class Razred
{
    public int Value = 0;
}
Razred ref1 = new Razred();
Razred ref2 = ref1;
ref2.Value = 123;
```

→ `new` operator stvara novu instancu razreda `Razred` i vraća pokazivač na nju

→ `ref2` pokazuje gdje i `ref1`

- ❑ **Primjer**  **Osnove\VrijednostiReference**

Nabrajanje (Enum)

❑ Pobrojani tip (enumerator)

- Korisnički tip vrijednosti koji nasljeđuje `System.Enum`
- Sastoji se od imenovanih konstanti
- Temeljni tip podataka pobrojanog tipa je `int`

❑ Primjer Osnove\Enum

```
enum Dani
{
    Ponedjeljak, Utorak, Srijeda,
    Cetvrtak, Petak, Subota, Nedjelja
}
```

```
int x = (int) Dani.Ponedjeljak;
int y = (int) Dani.Utorak;
Console.WriteLine("Pon={0}", x);
Console.WriteLine("Uto={0}", y);
```



Dani počinju od 0.

❑ Ako želimo da dani idu od 1. nadalje:

```
enum DaniPoRedu
{
    Ponedjeljak=1, Utorak, Srijeda, Cetvrtak,
    Petak, Subota, Nedjelja
}
```

Naredbe za upravljanje programskim tokom

❑ Selekcija

- if (expression) statement1 [else statement2]
- switch (*expression*) { case *constant-expression*: *statement jump-statement* [default: *statement jump-statement*] }

❑ Petlje

- while (expression) statement
- for ([initializers]; [expression]; [iterators]) statement
- **foreach (*type identifier in expression*) statement**
- do statement while (expression);

❑ Skokovi

- break , continue, goto
 - goto *identifier*;
 - goto case *constant-expression*;
 - goto default;
- return [*expression*];

```
int[] numbers =
    {4, 5, 6, 1, 2, 3, -2, -1, 0};
for(int i=0; i<numbers.length; i++){
    Console.WriteLine(numbers[i]);
}
foreach (int i in numbers){
    Console.WriteLine(i);
}
```

Polja

❑ Polja određene duljine:

```
int[] a = new int[10]; // inicijalno vrijednosti 0
int[] b = new int[] { 1, 2, 3};
int[] c = { 1, 2, 3};
```

❑ Za duljinu polja moguće je navesti i varijablu

```
int n = 3;
int[] d = new int[n];
```

❑ Dohvat elementa na mjestu i: `b[i]`

❑ Broj elemenata u polju: `b.Length`

❑ Primjer Osnove\Polja

```
for (int i = 0; i < b.Length; i++)
{
    Console.Write(b[i]);
}
```

Rad s poljima

❑ Neki od statičkih postupaka razreda `System.Array` su:

- `Sort` – sortira polje ili dio polja, opcionalno uz vlastiti usporednik
- `BinarySearch` – binarno pretražuje polje i vraća indeks traženog elementa
 - bitovni komplement indeksa prvog elementa većeg od traženog, kad traženi nije u polju a vrijednost traženog je manja od preostalih elemenata polja
 - bitovni komplement broja elemenata polja, kad traženi nije u polju a vrijednost traženog je veća od vrijednosti najvećeg u polju
- `Copy` – kopira čitavo polje ili dio polja polja u novo polje
- `Reverse` – obrće redoslijed članova polja

```
System.Array z = new int[] { 3, 1, 2 };
System.Array.Sort(z);
Console.WriteLine("Tražim 3 na ind.{0}", BinarySearch(z, 3));
...
Array o = new int[3];
Array.Copy(z, o, 2);
Array.Reverse(o);
```

Struktura Char

❑ Tip vrijednosti koji služi za pohranu jednog znaka

- `char znak = '٣'; // arapski broj 3`
- `char znak = '\n'; // novi red`
- `char znak = '\u0663'; // kod u Unicode tablici za arapski br. 3`

❑ Neki atributi i postupci

- `ToUpper` pretvara znak u odgovarajuće *Unicode* veliko slovo
 - npr. `System.Char.ToUpper(znak);`
- `ToLower` pretvara znak u odgovarajuće *Unicode* malo slovo
- `IsDigit` pripada li znak skupini *Unicode* decimalnih znamenaka
- `IsLetter` pripada li znak skupini *Unicode* slova
- `MaxValue` najveća moguća vrijednost za `Char`
- `MinValue` najmanja moguća vrijednost za `Char`

Razred String

❑ **System.String (isto što i string) – niz znakova**

- jednom postavljena vrijednost ne može se promijeniti.
- postupci na *stringu* stvaraju novi (promijenjeni) *string*.
- Općenito, objekti koji nisu string imaju postupak `ToString()`, npr.
`123.ToString()`

❑ **Primjer Osnove\String**

- `string a = "Primjer za String";`
- Duljina: `a.Length // 17`
- Uklanjanje znakova: `a.Remove(7,4) // PrimjerString`
- Velika slova: `a.ToUpper() // PRIMJER ZA STRING`
- Mala slova: `a.ToLower() // primjer za string`
- Podniz: `a.Substring(11) // String`
- Uklanjanje vodećih i pratećih praznina `a.Trim()`
- Rastavljanje stringa na dijelove:
`string[] stringovi = a.Split(new char[] { ' ' });`
`// niz s početka daje {"Primjer", "za", "String"}`

Razred StringBuilder

❑ `System.Text.StringBuilder` – promjenjiv niz znakova

- Sve promjene primjenjuju se na izvornom nizu.

❑ Primjer Osnove\String

- `StringBuilder a= new StringBuilder("Primjer za String");`
- Umetanje: `a.Insert(7,'i') // Primjeri za String`
- Dodavanje: `a.Append("Builder") //Primjeri za StringBuilder`

Konverzija tipova

❑ Implicitna konverzija – bez navođenja operatora konverzije

- Standardno int->double, int->long, ...

❑ Eksplicitna konverzija – Primjer: Osnove\Konverzija

- operator konverzije tipa (cast)

```
int c = (int) 4.5;
```

- ToString()

```
int a = 154;  
string s = a.ToString();
```

- Parse()

```
int c = Int32.Parse(s);
```

- razred Convert

```
decimal d = Convert.ToDecimal(c);
```

Boxing/Unboxing

❑ **Boxing**

- pretvorba nekog vrijednosnog tipa podataka u referencijski tip `object`
- alocira instancu i kopira vrijednost u novostvoreni objekt

❑ **Unboxing**

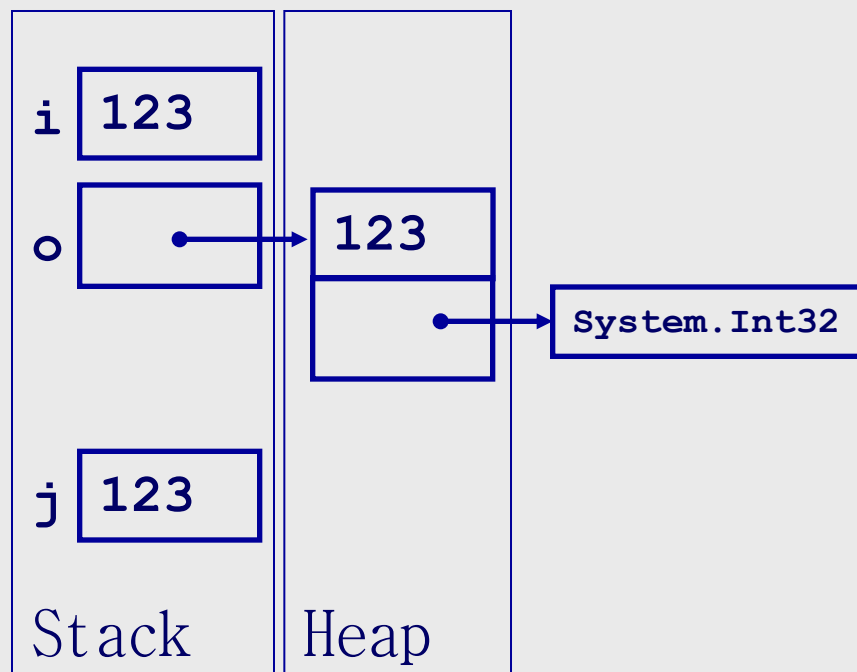
- izdvajanje vrijednosnog tipa podataka iz tipa `object`
- kopiranje vrijednosti (objekta) s reference

❑ **Primjer** 📁 **Osnove\Boxing**

```
int i = 123; // A value type
```

```
object o = i; // Boxing
```

```
int j = (int) o; // Unboxing
```



Nulabilni tipovi

❑ **Varijable mogu imati nedefiniranu vrijednost**

❑ **Deklaracija standardnog tipa sadrži upitnik (?)**

```
int? num = null;  
if (num.HasValue == true) // isto što i if (!(num == null))  
    Console.WriteLine("num = " + num.Value);  
else  
    Console.WriteLine("num = Null");
```

❑ **String se ne deklarira kao nulabilan**

- String može biti prazan (tzv. null-string) ali to ne znači da je null
- `string s = string.Empty` // isto što i `s = ""`
- postavljanje na null znači da nije ni prazan
- `string s = null;` // vrijedi `s != ""`, `s == null`

Prostor imena (namespace)

❑ Izvorni .NET kod je organiziran u prostore imena (namespaces)

- C# program se može sastojati od više datoteka
- Svaka datoteka sadrži jedan ili više prostora imena
- Isti prostor imena može se protezati (deklarirati) u više datoteka.

❑ Prostor imena

- Prostor imena sadrži definicije razreda, struktura, sučelja, pobrojanih tipova, delegata i deklaraciju drugih prostora imena
- Prostor imena u kojem su deklarirani svi ostali prostori imena i tipovi podataka u .NET Frameworku je System
- Ako se u programu eksplicitno ne definira prostor imena, C# kod je sadržan u globalnom imeniku (*global namespace*)

❑ Koncept prostora imena omogućuje postojanje istih imena u različitim prostorima imena

- Jedinstvenost imena tipova u prostoru imena i samog prostora imena osigurana je preko tzv. potpunih imena (*fully qualified names*)

Neki .NET Framework prostori imena

Namespace	Opis
System	Osnovni razredi i tipovi podataka (npr. int , char , float)
System.Data	Razredi koji čine ADO.NET, koji se koristi za pristup bazama podataka i rukovanje podacima.
System.Drawing	Razredi za crtanje i grafiku.
System.IO	Čitanje i pisanje podataka, npr. u datotekama.
System.Threading	Simultano izvođenje programa, višenitnost (multithreading).
System.Windows.Forms	Razredi za kreiranje grafičkog sučelja
System.Xml	Razredi za obradu XML datoteka.
System.String	Razredi za rad sa nizovima znakova (string).
System.Char	Razredi za rad sa znakovnim tipom podataka.
System.Collection	Razredi za rad s kolekcijama (listama).
System.Collections.Generic	Razredi za rad s generičkim (tipiziranim) kolekcijama.

Definiranje prostora imena

- ❑ **Prostor imena N1 je član globalnog prostora imena**
 - Njegovo puno ime je N1
- ❑ **Prostor imena N2 je član prostora N1**
 - Njegovo puno ime je N1.N2
- ❑ **Razred C1 je član od N1**
 - Njegovo puno ime je N1.C1
- ❑ **Ime razreda C2 se pojavljuje dva puta, ali je njegovo puno ime jedinstveno**
 - N1.C1.C2 ili
 - N1.N2.C2.

```
namespace N1{           // N1
    class C1{           // N1.C1
        class C2{       // N1.C1.C2
            }
        }
    }
    namespace N2{       // N1.N2
        class C2{       // N1.N2.C2
            }
        }
    }
}
```

Korištenje prostora imena

❑ Referenciranje koda u prostoru imena obavlja se

- kvalificiranim imenom, od naziva prostora imena i identifikatora razreda
 - npr. `System.Console`
- direktivom `using`
 - eksplicitno se uvode sva imena sadržana u prostoru imena
 - npr. `using System;` referencira imenik u kojem je definiran razred `Console`, pa se `Console` može direktno koristiti
- zamjenskim imenom – koristi se za definiranje skraćenih naziva i uklanjanje neodređenosti kad u različitim imenicima postoje razredi istog imena.

```
using System; // navod imenika
using stdout = System.Console; // alias razreda
class Pozdrav {
    static void Main(string[] args) {
        System.Console.WriteLine("Kvalificirano!");
        Console.WriteLine("Izravno!"); // radi using System
        stdout.WriteLine("Zamjenski!"); // radi using stdout =
    }
}
```

Zadaci za vježbu


- ❑ **Napisati program koji iz polja stringova ispisuje one koji sadržavaju zadani podniz.**

-  Osnove\Podniz


- ❑ **Napisati program koji**

- učitava string sa komandnog retka
- mijenja ga u velika slova
- ispisuje njegove znakove u obrnutom poretку.

- ❑ **Napisati razred Kalkulator**

- Napiši postupke za oduzimanje, zbrajanje, množenje i dijeljenje.
- Napiši glavni program za rad s kalkulatorom.
-  Osnove\Kalkulator

- ❑ **Demonstrirati generator pseudoslučajnih brojeva**

- Upotrijebi razred `System.Random`.
-  Osnove\RandomBrojevi

Reference

Resursi

- IntroductionToCS
- Core CSharp and .NET Quick Reference

Korisni portali

- <http://www.codeproject.com>
- <http://sourceforge.net>
- <http://www.codeguru.com>
- <http://www.csharp-help.com>
- <http://www.csharp-station.com>
- <http://www.c-sharpcorner.com>