

# Oblikovni obrasci u programiranju

## Uvodno predavanje

Siniša Šegvić

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Zavod za elektroniku, mikroelektroniku  
računalne i intelligentne sustave

# SADRŽAJ UVODNOG PREDAVANJA

## Motivacija za načela i obrasce programskog oblikovanja:

- problemi vezani uz **složenost interakcije** među komponentama
- obuzdavanje složenosti **apstrakcijom** i **organizacijom**

## Oblikovne osnove za ublažavanje interakcijskih problema:

- ne-funkcionalni zahtjevi: ne odnose se na funkcionalnost sustava
- utjecaj tih zahtjeva na dinamiku razvoja

## Uklapanje oblikovanja u **razvojne procese**:

- kako i kada planirati organizaciju programa?
- usporedba "lakih" i "teških" razvojnih modela (metodologija)

## O predmetu:

- glavne teme, prepostavljeno znanje
- način održavanja nastave, razdioba bodova, literatura

# MOTIVACIJA: PROBLEMI

Zašto je programsko inženjerstvo **teško**?

- Problemi vezani uz funkcionalnost programa:
  - sofisticirana funkcionalnost
  - postizanje adekvatne brzine izvođenja
  - ostvarivanje lakog, intuitivnog i ugodnog korištenja
- Problemi vezani uz poslovno okruženje:
  - ograničeni vremenski i materijalni resursi
  - neprecizni i promjenljivi funkcionalni zahtjevi
- Problemi vezani uz **organizaciju**:
  - **složenost interakcije** među komponentama programa
  - otežavaju zadovoljavanje svih ostalih zahtjeva

Zašto je **teško** organizirati interakciju komponenata?

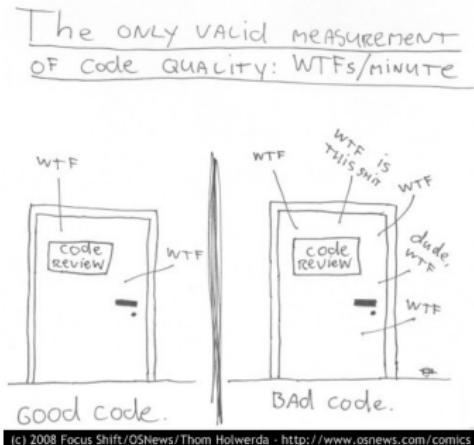
- strahovito puno načina za organiziranje sustava (**eksp. složenost!**)
- prikladnost organizacije vidljiva tek pri održavanju (**kasno!**)

Organiziranje **interakcije**: ključni zadatak **programskog oblikovanja!**

# MOTIVACIJA: SLOŽENOST [BROOKS75,BROOKS95]

Složenost interakcije je nezaobilazna poteškoća razvoja programa:

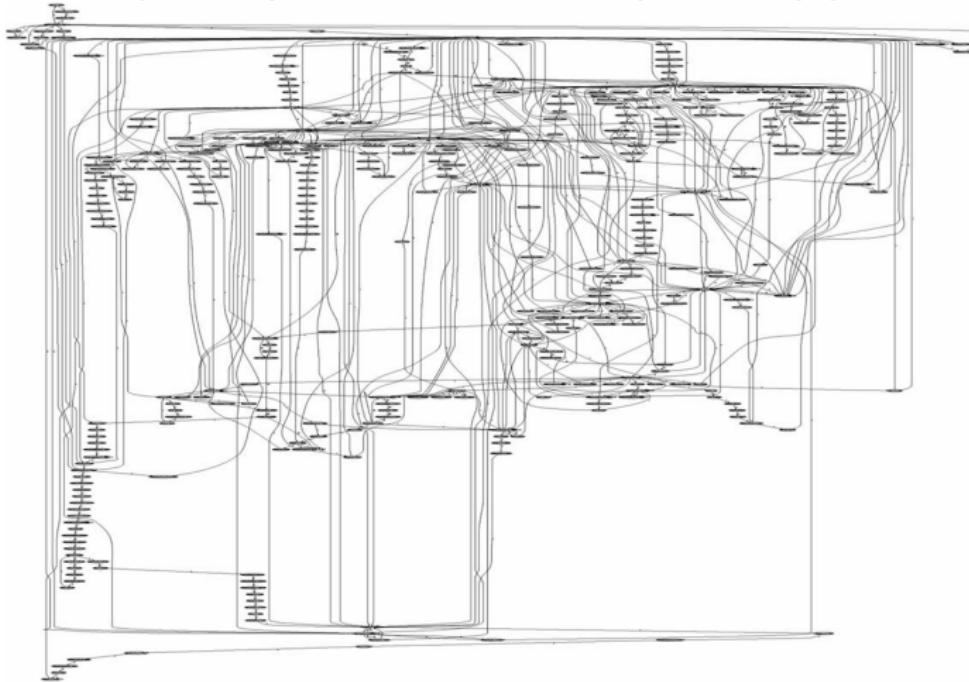
- implementirati jedan program s 30 funkcionalnosti puno teže nego 30 zasebnih programa
  - funkcionalnosti međudjeluju, najčešće negativno
- ne postoji tehnologija koja bi ubrzala razvoj za red veličine
  - no silver bullet
- ako projektu koji kasni dodijelimo nove inženjere --- kašnjenje će se povećati (mythical man-month)
- jedan pristup bio bi izdvojiti funkcionalnosti u zasebne programe i povezati ih skriptnim jezikom (ali to ne možemo uvijek!)
- **tema kolegija:** pristupi za smanjivanje međuvisnosti komponenata te ublažavanje njenih štetnih efekata



(c) 2008 Focus Shift/OSNews/Thom Holwerda - <http://www.osnews.com/comics>

## MOTIVACIJA: PRIMJER 1

Graf sistemskih poziva pri dohvatu stranice preko http protokola:



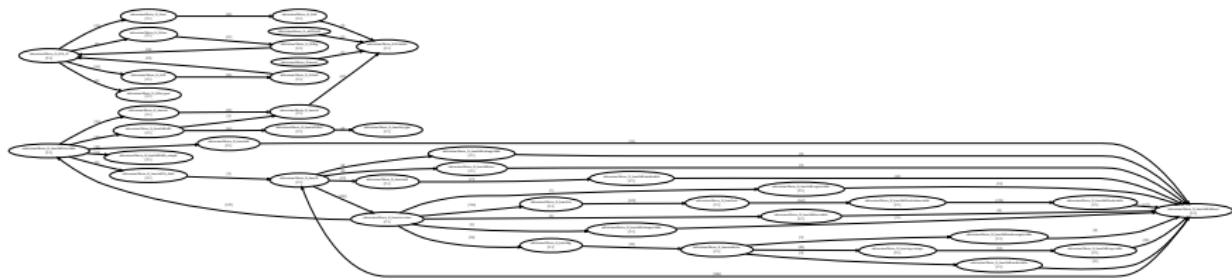
<https://ma.ttias.be/system-calls-in-apache-linux-vs-iis-windows/>

Razmatramo dodatak: uvođenje nove metode autentikacije

- što je teže, natipkati algoritam ili skužiti kako ga optimalno uklopiti?

## MOTIVACIJA: PRIMJER 2A

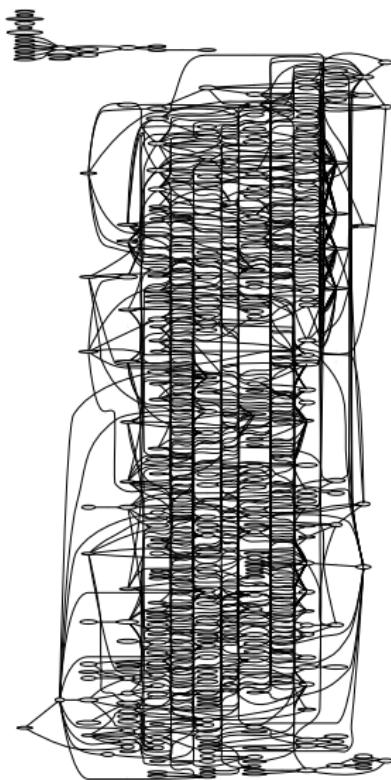
Zašto je složenost programskog sustava problematična?



(graf ovisnosti komponenata jednog paketa biblioteke libsvn)

## MOTIVACIJA: PRIMJER 2B

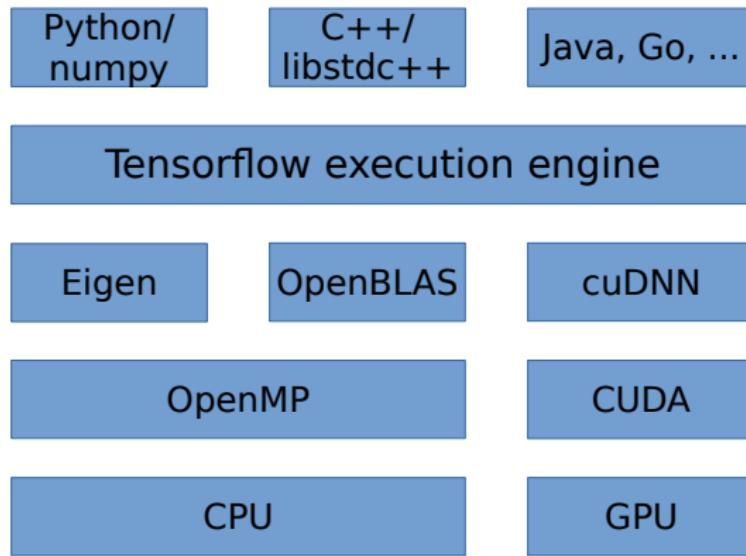
Zato što iza jedne stanojite granice složenosti struktura programa postaje neprozirna...



## MOTIVACIJA: PRIMJER 3

Do dobre koncepcije dolazimo **apstrakcijom** (skrivanjem nebitnog)

Moderni programski okviri **apstrahiraju** programske biblioteke i sklopovske arhitekture (klijentski kod **ne mora** znati za to)



**Tema kolegija:** načela i pristupi za prepoznavanje korisnih apstrakcija.

# MOTIVACIJA: OOUP

Zašto mislimo da bi vam ovaj kolegij mogao biti **interesantan**?

- **Ciljana publika:** budući inženjeri računarsva
  - AKA profesionalni programeri
- **Činjenica:** jednostavni programi **napisani** (sazrijevanje!)  
(početni položaji zauzeti, ad-hoc pristupi ne pale)
- **Zaposlenje 1:** održavanje **mastodonta**  
(stabilno okruženje, razumijevanje postojeće **organizacije**)
- **Zaposlenje 2:** rad na novom **ambicioznom proizvodu**  
(dinamično okruženje, prilagođavanje **organizacije** domeni)
- U oba slučaja težimo **održivoj evoluciji** proizvoda
- Vidjet ćemo da ključ uspješne evolucije predstavlja adekvatna programska **organizacija**!

# MOTIVACIJA: Fokus OOUP-a

Fokus predmeta: **organiziranje** komponenata u veći sustav

- koristit ćemo idiomatski pristup *s vrha prema dnu*;  
razmatrat ćemo **rješenja**: načela, idiome, obrasce
- treba nam duboko poznavanje tehnika koje omogućavaju oblikovanje fleksibilnih programskih komponenata
  - mogu se 'saviti' kako bi 'upile' promjene
  - mogu se koristiti na različite načine iz različitih projekata
- posebnu pažnju ćemo pokloniti **dinamičkom polimorfizmu**
  - mehanizme izvedbe u C-u, C++-u i Pythonu proučava prva laboratorijska vježba
- čitanje i pisanje računalnih programa: **osnovna kompetencija** inženjera računarstva
  - izvorni kod možemo promatrati kao formalnu specifikaciju funkcionalnosti programskega sustava

# MOTIVACIJA: PROGRAMIRANJE



<https://www.commitstrip.com/en/2016/08/25/a-very-comprehensive-and-precise-spec/>

# MOTIVACIJA: JEZIK VS OBLIKOVANJE

Zašto poznavanje mehanike jezika **nije dovoljno**?

- uspješni jezici su nesavršeni i opterećeni prošlošću
- dani problem se može izvesti na mnogo korektnih načina:  
jezični pokazatelji za njihovo vrednovanje **prelokalni** i nedovoljni!
- primjer standardne biblioteke C-a: zašto `gets` **ne valja**?

```
NAME
    gets - get a string from standard input (DEPRECATED)

SYNOPSIS
    char *gets(char *s);

DESCRIPTION
    Never use this function.
```

- nema garancije da **jezično** korektna i jasna komponenta ima zadovoljavajuća svojstva i u **širem** oblikovnom kontekstu!

## MOTIVACIJA: JEZIK VS OBLIKOVANJE (2)

Primjer: komponenta `Image` izravno koristi vanjsku biblioteku `acmePdf v1.0`

```
class Image{
    // ...
    void load(char const* path);
};

void Image::load(char const* path){
    switch (util::extension(path)){
        // ...
        case "pdf":{
            ImageFmt fmt;
            acmePdfGetFmt(path, fmt);
            this->reformat(fmt);
            acmePdfLoad(path, this->getBuffer());
        }
    };
}
```

Ovakve ovisnosti o vanjskim komponentama mogu stvoriti **probleme**.

- procjena razvojnog napora za libpoppler (pdf reader, GPL): 50+ osoba-godina (COCOMO).

## MOTIVACIJA: JEZIK VS OBLIKOVANJE (3)

Razmotrimo sljedeći (sasvim mogući) scenarij:

- U acmePdf v1.0 otkriveni neugodni bugovi
- acmePdf v2.0 izdan pod nezgodnom licencom
- korisnici naše komponente su u nevolji (vendor lock-in anti pattern)

Neki korisnici mogu podnijeti novu licencu, drugi ne: odjednom imamo dvije verzije komponente (muka održavati...)

Rješenje:

- izdvojiti metodu load u zasebnu komponentu
- ta komponenta mora ispitati je li acmePdf dostupna (**introspekcija**)
- u slučaju nedostupnosti bacamo **iznimku**

Nema garancije da **jezično** korektna i jasna komponenta ima zadovoljavajuća svojstva i u **širem** oblikovnom kontekstu!

# SADRŽAJ UVODNOG PREDAVANJA

- Motivacija za načela i obrasce programskog oblikovanja:
  - problemi vezani uz složenost interakcije među komponentama
  - obuzdavanje složenosti apstrakcijom i organizacijom
- **Oblikovne osnove** za ublažavanje interakcijskih problema:
  - **ne-funkcionalni zahtjevi:** **ne odnose** se na funkcionalnost sustava
  - utjecaj tih zahtjeva na **dinamiku razvoja**
- Uklapanje oblikovanja u **razvojne procese**:
  - kako i kada planirati organizaciju programa?
  - usporedba "lakih" i "teških" razvojnih modela (metodologija)
- **O predmetu:**
  - glavne teme, pretpostavljeno znanje
  - način održavanja nastave, razdioba bodova, literatura

## OBLIKOVNE OSNOVE: CILJEVI RAZVOJA

Koja svojstva programskega sistema želimo ostvariti?

1. **korektnost:** program obavlja svoj posao  
(algoritmi, strukture podataka)
2. **zadovoljavajuća performansa:** program radi dovoljno brzo  
(napredni algoritmi i strukture podataka, arhitektura računala)
3. **ugodan izgled:** lijepo korisničko sučelje  
(ergonomija, grafičko oblikovanje)
4. **lako održavanje:** razumijevanje, ispitivanje, popravljanje  
(programska oblikovanje, dokumentacija)
5. **fleksibilnost** (podatnost): lako nadogradnjava, toleriranje  
promjena  
(programska oblikovanje)

**Statička** (1-3) vs. **dinamička** (4-5) svojstva programa

**Organizacija** - ključ dinamike programskega sistema!

## OBLIKOVNE OSNOVE: VAŽNOST DINAMIKE RAZVOJA

Dinamička svojstva su važna jer se pokazuje da se do dobrih programa  
**ne dolazi** ``iz prve''

Primjer **realnog** scenarija nakon pola godine razvoja:

- nepotpuna** korektnost:  
(polovična funkcionalnost, novi zahtjevi, bube švabe)
- brzina **možda** i prihvatljiva
- korisnici **nezadovoljni** lakoćom korištenja

Uz malo dobre sreće, v1.0 može biti prihvatljiva

- investitori pristaju financirati novu verziju
- čvrsti uvjet nastavka: dinamička svojstva programa!

## OBLIKOVNE OSNOVE: VAŽNOST DINAMIKE RAZVOJA (2)

Zanimljiva priča iz života:

- 1998: Google se ponudio Yahooou za \$1e6
  - Page i Brin (tada znanstveni novaci) su htjeli nastaviti studij...
  - Yahoo odbija, Page i Brin u garaži prijateljice otvaraju d.o.o
- 2002: Yahoo nudi Googleu \$3e9
  - Google traži \$5e9
  - Yahoo odbija
- 2008: Microsoft nudi Yahooou \$40e9
  - Yahoo odbija, Microsoft kupuje Fast za \$1.2e9
- 2018: Google vrijedi preko \$500e9
  - Yahoo prodan za \$5e9 (2016)

Google je nadjačao Yahoo (i ostale) jer se brže i bolje **razvijao**:

- bolji program za indeksiranje interneta (engl. web crawler)
- bolji algoritam za rangiranje stranica (pagerank)
- bolje i brže korisničko sučelje (kratki kontekst, samo jedna slika)

## OBLIKOVNE OSNOVE: KAKO POSTIĆI DOBRU DINAMIKU

Dobru dinamiku programskog projekta omogućit će kôd:

- kojeg je lako **ispitati** (engl. testable)
  - koliko brzo možemo provjeriti sumnju da komponenta X ne radi?
- kojeg je lako **razumjeti** (engl. readable)
  - hoće li se snaći programer kojeg smo zaposlili prošli tjedan?
- kojeg je lako **opraviti** odnosno **izmijeniti** (engl. maintainable)
  - koliko brzo možemo onemogućiti zbrajanje popusta?
- kojeg je lako **nadograditi** (engl. extensible)
  - ako promijenimo A, hoćemo li morati mijenjati B, C, D?

Navedena svojstva (tzv. -ilities, non-functional requirements) postižu se prikladnom **organizacijom!**

- to je upravo predmet proučavanja našeg kolegija

Nabrojali smo dobra **svojstva** gotovih programa;  
**kako** do takvih programa doći?

## SADRŽAJ UVODNOG PREDAVANJA

Motivacija za načela i obrasce programskog oblikovanja:

- problemi vezani uz složenost interakcije među komponentama
- obuzdavanje složenosti apstrakcijom i organizacijom

**Oblikovne osnove** za ublažavanje interakcijskih problema:

- ne-funkcionalni zahtjevi: **ne odnose** se na funkcionalnost sustava
- utjecaj tih zahtjeva na dinamiku razvoja

Uklapanje oblikovanja u **razvojne procese**:

- kako i kada planirati organizaciju programa?
- usporedba "lakih" i "teških" razvojnih modela (metodologija)

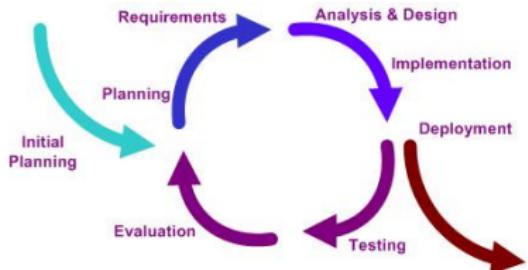
O predmetu:

- glavne teme, prepostavljeno znanje
- način održavanja nastave, razdioba bodova, literatura

# PROCES: KLASIČNO VS AGILNO OBLIKOVANJE

Model razvojnog procesa propisuje smjernice razvoja programa

- Royce 1970 - vodopadni model:  
zahtjevi → oblikovanje → izvedba → ispitivanje → održavanje
- 198x,199x: iterativne modifikacije vodopadnog modela
- 21. stoljeće: pojava agilnih razvojnih procesa
  - iterativno dodavanje prioritetne funkcionalnosti: piecemeal growth vs. masterplan
  - kratke iteracije, dnevna testiranja, česta komunikacija, prilagodljivost, tehnička izvrsnost
  - prihvaćanje realnosti: promjenljivi zahtjevi, nepredvidljivi problemi, heterogenost razvojnog tima, ...



# PROCES: AGILNI MODELI

**Nedostatci** (i "nedostatci") agilnih modela razvojnog procesa:

- nema garancije da program stvarno radi!  
(što ako se bug manifestira u avionu na 3000m visine?)
- fokus na kôd, umjesto na oblikovanje: divlji zapad!



**Prednosti** agilnih modela razvojnog procesa:

- agilni modeli zahtijevaju **kontinuirano** oblikovanje
  - u početku malo znamo o domeni i ne možemo oblikovati
- **ideja:** svjesno alocirati resurse u ovisnosti o odnosu između cijene razvoja i prihvatljivog rizika
- nema garancije da agilno razvijeni program uistinu radi, ali to ne nude ni alternativni pristupi

## PROCES: EVOLUCIJA, INOVACIJA

Činjenica: uspješni programi današnjice nastali **evolucijom**

- Windows Vista (50 MLoC) ← 86-DOS (1980)
- Linux 2.6 (5 MLoC) ← Linux 0.01 (1991, 10 KLoC)

Zaključak: programiranje nije slično **građevinarstvu!**

- bar ne toliko koliko to implicira vodopadni model
- najkorisniji programi su oni koje još nitko nije napisao  
(naše implementacije lako se kopiraju i prepravljaju)

Naš prirodni habitat je **fronta** tehnološke ekspanzije

- profit se odselio s osobnog računala: tablični kalkulator je davno napisan (VisiCalc, 1979, 7e5 primjeraka, Apple II)
- generativni modeli, kriptovalute, mobilne i umrežene primjene, oglašavanje; razumijevanje slike, govora, prirodnog jezika; ugrađene i robotičke primjene...

# PROCES: SLOC COUNT LINUX

sloc count linux-2.6.28.7 (kazala)

SLOC	Directory	SLOC - by - Language (Sorted)
3340479	drivers	ansic=3336030, yacc=1688, asm=1136, perl=829, lex=779, sh=17
1338785	arch	ansic=1127349, asm=209975, sh=615, yacc=307, lex=300, awk=96, python=45, pascal=41, sed=29, perl=28
545820	fs	ansic=545820
377581	net	ansic=377581
356592	sound	ansic=356409, asm=183
250442	include	ansic=248816, cpp=1515, pascal=75, asm=36
74639	kernel	ansic=74334, perl=305
36459	mm	ansic=36459
32743	crypto	ansic=32743
25316	security	ansic=25316
24193	scripts	ansic=14432, perl=4707, cpp=1791, sh=1175, yacc=967, lex=742, python=379
17146	lib	ansic=17146
10742	block	ansic=10742
7712	Documentation	ansic=5615, sh=1022, perl=857, lisp=218
5243	ipc	ansic=5243
2658	virt	ansic=2658
2283	init	ansic=2283
1803	firmware	asm=1598, ansic=205
833	samples	ansic=833
493	usr	ansic=491, asm=2

## SADRŽAJ UVODNOG PREDAVANJA

Motivacija za načela i obrasce programskog oblikovanja:

- problemi vezani uz složenost interakcije među komponentama
- obuzdavanje složenosti apstrakcijom i organizacijom

**Oblikovne osnove** za ublažavanje interakcijskih problema:

- ne-funkcionalni zahtjevi: **ne odnose** se na funkcionalnost sustava
- utjecaj tih zahtjeva na dinamiku razvoja

Uklapanje oblikovanja u **razvojne procese**:

- kako i kada planirati organizaciju programa?
- usporedba "lakih" i "teških" razvojnih modela (metodologija)

**O predmetu:**

- glavne teme, pretpostavljeno znanje
- način održavanja nastave, razdioba bodova, literatura

## O PREDMETU

Proučavat ćemo elemente **programske organizacije** na razinama **komponente** (.5 kLoC), te **paketa** ili podsustava (5 kLoC)

- organizacijska **načela**, te **oblikovni obrasci** kojima se ona postižu za učestale razrede problema
- izabrane programske tehnike za ostvarivanje oblikovnih ciljeva (polimorfizam, dinamičke biblioteke, ugovorno oblikovanje)

Prepostavljamo izvorni kôd **opće namjene**

- komponente mogu biti dio mrežne usluge, mobilne aplikacije, ugradbenog uređaja, autonomnog robota, istraživačkog kôda, ...

Gradivo je većim dijelom **agnostično** s obzirom na operacijski sustav, programski jezik i model razvojnog procesa.

Prepostavljamo osnovna znanja iz domene objektno orijentiranog programiranja (dinamički polimorfizam ćemo temeljito ponoviti).

## O PREDMETU: UVJETI, BODOVI

**Aktivnosti:** predavanja, vježbe (C, C++, Python, Java, ?), međuispit, završni ispit, klasični ispit

### Kalendar nastave:

kraj ožujka:	L1
sredina travnja:	L2
kraj travnja:	MI
sredina svibnja:	L3
početak lipnja:	L4
sredina lipnja:	ZI
početak srpnja:	KI

### Kontinuirana provjera:

laboratorij:	10 (A) + 10 (B)
ispiti:	40, 40
preduvjet:	50% laboratorijskih vježbi

### Klasični ispit:

preduvjet:	50% laboratorijskih vježbi
------------	----------------------------

**Burza grupa:** tjedan-dva prije odgovarajuće vježbe (pratite oglase!).

**Ocenjivanje:** 2: 50%, 3: 63%, 4: 76%, 5: 89%.

Mogućnost dobivanja **bonus bodova** za: korisne sugestije, prijedloge novih tema ili vježbi, ekstra zadatke, seminare.

- javiti se e-mailom bilo kojem nositelju ili asistentu

## O PREDMETU: PLAN

Što ćemo raditi tijekom ovog semestra?

1. načela programske organizacije:
  - motivacijski primjer, tehnike programiranja
  - načela logičkog i fizičkog oblikovanja
2. osnovni oblikovni obrasci
3. ostali oblikovni obrasci

**Vježbe** će se izvoditi u C-u, C++-u, te Pythonu ili Javi

- neke vježbe treba izvesti u C-u (1.1, 3.1.1) i C++-u (1.3-1.5)  
(pogledajte cppcpp, javite kako da ga poboljšamo)
- možete riješiti samo jednu od vježbi 3.1.2, 3.1.3, i 3.1.4
- ostale vježbe možete pisati u proizvolnjem jeziku
- prevoditelj i operacijski sustav su proizvoljni
- termin za nadoknadu **jedne** vježbe: krajem semestra

## O PREDMETU: EKIPA

Predavanja:

- Siniša Šegvić
- (Marko Čupić)
- Marin Oršić

Laboratorijske vježbe:

- Ivan Sabolić
- Anja Delić
- Ivan Martinović
- 8 demonstratora

## O PREDMETU: LITERATURA

Obrasci i načela programskog oblikovanja:

- Design Patterns; Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides; Addison-Wesley; 1995
- Agile Software Development: Principles, Patterns, and Practices; Robert C. Martin; Prentice Hall; 2002
- Large-Scale C++ Software Design; John Lakos; Addison-Wesley; 1996
- Head First Design Patterns; Elisabeth Freeman, Eric Freeman, Bert Bates, Kathy Sierra; O'Reilly Media, Inc.; 2004
- The Elements of Programming Style; Brian W. Kernighan, P. J. Plauger; Computing Mcgraw-Hill; 1978

## O PREDMETU: LITERATURA (2)

### Modeli razvojnog procesa

- The Mythical Man-Month; F. Brooks; Addison Wesley; 1995
- The Pragmatic Programmer; A. Hunt, D. Thomas; Addison Wesley; 2000
- Generative Programming: Methods, Tools, and Applications; Krysztof Czarnecki, Ulrich Eisenecker; Addison-Wesley Professional; 2000

## O PREDMETU: LITERATURA (3)

**Literatura:** C++, generičko programiranje

- [More] Effective {C++|STL}; S. Meyers; Addison Wesley; 1996
- Inside the C++ object model; Stanley Lippman; Addison-Wesley Professional; 1996
- Effective Modern C++; S. Meyers; O'Reilly Media; 2014
- C++ FAQs; Marshall P. Cline, Greg Lomow, Mike Girou; Addison-Wesley Professional; 1998
- Demistificirani C++; Julijan Šribar i Boris Motik; Element; 2006
- Generic Programming and the STL: Using and Extending the C++ Standard Template Library; Addison-Wesley Professional; Matthew H. Austern; 1998