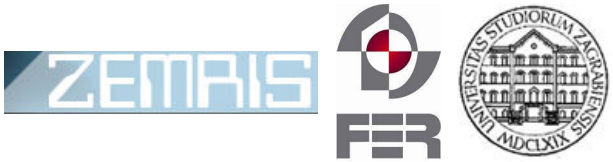

Oblikovni obrasci u programiranju

Uvodno predavanje

Siniša Šegvić

Zavod za elektroniku, mikroelektroniku,
računalne i inteligentne sustave
Fakultet elektrotehnike i računarstva
Sveučilište u Zagrebu

SADRŽAJ

Uvod u predmet Oblikovni obrasci u programiranju

- **Motivacija**: sazrijevanje programskog inženjerstva
- **Ciljevi**: koja su željena svojstva programskog sustava?
- **Metodologije**: kako izgraditi programski sustav bez neugodnosti?
- **O predmetu**: kratki opis, literatura, način održavanja nastave, teme

MOTIVACIJA

Zašto mislim da bi vam OOUP mogao biti **interesantan**?

- **Ciljana publika**: budući profesionalci programskog inženjerstva
- **Činjenica**: sazrijevanje područja
  (početni položaji zauzeti, ad-hoc pristupi ne pale)
- **Zaposlenje 1**: održavanje mastodonta
  (stabilno okruženje, razumijevanje postojeće arhitekture)
- **Zaposlenje 2**: rad na novom ambicioznom proizvodu
  (dinamično okruženje, prilagođavanje arhitekture domeni)
- U oba slučaja težimo održivoj evoluciji proizvoda
- Vidjet ćemo da je arhitektura ključ evolucije!

MOTIVACIJA (2)

Fokus predmeta: projektiranje **arhitekture** programskog sustava

- arhitektonska pitanja se slabo susreću u nastavi
  umjesto: mehanika jezika, *odozdo prema gore*
- ovdje koristimo idiomatski pristup *s vrha prema dnu*;
  razmatramo rješenja: načela, idiome, obrasce

Zašto poznavanje pravila programskog 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!
- **banalni** primjer: što "rade" sljedeća tri reda C-a?

```
_(__,___,____){___/__<=1?_(__,___+1,____):!(___%__)?_(__,___+1,0):___%__==___/
__&&!____?(printf("%d\t",___/__),_(__,___+1,0)):___%__>1&&___%__<___/__?_(__,1+
___,____+!(___/__%(___%__))):___<__*__?_(__,___+1,____):0;}main(){_(100,0,0);}
```

- Nekritičko inzistiranje na nespornoj lokalnoj vrlini šteti!

MOTIVACIJA (3)

Program od prije nakon **pažljivog** formatiranja:

```
_(__,___,____){
  ___/__<=1?
    _(__,___+1,____):
    !(___%__)?
      _(__,___+1,0):
      ___%__==___/__ && !____?
        (printf("%d\t",___/__),_(__,___+1,0)):
        ___%__>1 && ___%__<___/__?
          _(__,1+___,____+!(___/__%(___%__))):
          ___<__*__? _(__,___+1,____) : 0;
}
main(){
  _(100,0,0);
}
```

- Nema garancije da korektan i **lokalno** jasan program ima zadovoljavajuća svojstva i u širem kontekstu!
- Dobra svojstva postižemo **eksplicitnim** (i lucidnim) naporom

CILJEVI

Koja svojstva programskog sustava želimo ostvariti?

1. **korektnost**: program obavlja svoj posao
   (algoritmi, strukture)
2. **zadovoljavajuća performansa**: program radi dovoljno brzo
   (algoritmi, strukture)
3. **lako korištenje**: korisnici brzo uče vješto rukovanje
   (ergonomija, dizajn)
4. **lako održavanje**: razumijevanje, ispitivanje, nadogradnja
   (arhitektura, dokumentacija)
5. **podatnost** (fleksibilnost): otpornost na promjene
   (arhitektura)

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

Arhitektura - ključ dinamike programskog sustava!

CILJEVI (2)

Realni scenarij nakon pola godine razvoja:

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

U optimističnom slučaju, v1.0 donekle prihvatljiva

- investitor pristaje financirati novu verziju
- dinamička svojstva proizvoda: uvjet uspješne evolucije!

CILJEVI (3)

Dobra dinamika programskog projekta znači:

- lako **ispitivanje**
  (neke programe je lakše ispitati od drugih!)
- lako **razumijevanje**
  (hoćemo li se snaći i za godinu dana?)
- laka **nadogradnja**
  (koliko brzo možemo ugraditi novu funkcionalnost?)
- lako **mijenjanje** implementacije
  (što se događa kad se zahtjevi promijene, ili naše razumijevanje postane bolje?)

Navedena svojstva postižu se prikladnom arhitekturom!

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

METODOLOGIJE

Razvojna metodologija propisuje smjernice razvoja programa

- Royce 1970 - vodopadni model (the waterfall model):
  zahtjevi → oblikovanje → izvedba → ispitivanje → održavanje
- Brooks 1986 - “no silver bullet”:
  produktivnost neće porasti za red veličine sljedećih 10 g!
- Brooks 1995 - “no silver bullet refired”:
  još 10 godina ništa (ipak, OOP, COTS korisni)!
- 21. stoljeće: pojava agilnih metodologija
  - piecemeal growth vs. masterplan
  - prihvaćanje realnosti: promjenljivi zahtjevi, nepredvidljivi problemi, heterogenost razvojnog tima, ...
  - koncentriranje na kôd koji radi i sposobne ljude
  - kratke iteracije, dnevna testiranja, česta komunikacija, gajenje tehničke izvrsnosti, prilagodljivost

METODOLOGIJE (2)

Kritike agilnih metodologija:

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

Neki **odgovori**:

- agilna metodologija propisuje **kontinuirano** oblikovanje
  (u početku malo znamo o domeni i ne možemo oblikovati)
- **svojevoljno** alocirati resurse u ovisnosti o:
  - odnosu između cijene razvoja i očekivanog kolača
  - prihvatljivom riziku
- Ne možemo znati da program stvarno radi, ali:
  - **kritični** programi se i dandanas kvare
  - formalna verifikacija košta

METODOLOGIJE (3)

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 **građevini**!

- bar ne toliko koliko to implicira vodopadni model
- naša implementacija lako se prepravlja i besplatno kopira

Naš prirodni habitat je **fronta** tehnološke ekspanzije

- profit se odselio s osobnog računala: tablični kalkulator je davno napisan (VisiCalc, 1979, 7e5 primjeraka, Apple II)
- ugrađeni sustavi; obrada slike, govora, prirodnog jezika; web aplikacije, oglašavanje, umreženi servisi; ...

O PREDMETU

O čemu se ovdje radi?

- elementi **programske arhitekture** na razinama komponente (.5 kLoC), te paketa ili podsustava (5 kLoC)
- razmatraju se arhitektonska **načela**, te **oblikovni obrasci** kojima se ona postižu za učestale razrede problema
- pretpostavljaju se osnovna znanja iz domene objektno orijentiranog programiranja
- gradivo je većim dijelom **agnostično** s obzirom na:
  - operacijski sustav
  - programski jezik
  - metodologiju razvoja
- Ipak, naglasak na jeziku C++

O PREDMETU (2)

Aktivnosti: predavanja, domaće zadaće, vježbe (C, C++, ?), međuispiti, završni ispit

Kalendar nastave:

prvi ciklus:	4P+L+MI
drugi ciklus:	4P+L+MI
treći ciklus:	5P+L+ZI

Bodovanje:

domaće zadaće	0
laboratorij	25
ispiti	20, 25, 30

Ocjenjivanje:

- prag prolaza je 50% bodova
- ocjene prema razdiobi (15%,35%,35%,15%)

O PREDMETU (3)

Što ćemo raditi tijekom tri ciklusa ovog semestra?

1. načela programske arhitekture: logičko i fizičko oblikovanje

   vježba: neprozirni tipovi, načelo inverzije ovisnosti
2. obrasci (objektno orijentiranog) oblikovanja

   vježba: obrazac proxy za višedimenzionalna polja
3. obrasci generičkog programiranja (C++)

   vježba: pametni pokazivači

Vježbe će se izvoditi u standardnom jeziku C++
(prevodioc i operacijski sustav proizvoljni)

O PREDMETU (4)

Literatura: obrasci i načela programske arhitekture

- Agile Software Development: Principles, Patterns, and Practices; Robert C. Martin; Prentice Hall; 2002
- Large-Scale C++ Software Design; John Lakos; Addison-Wesley; 1996
- Design Patterns; Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides; Addison-Wesley; 1995
- Modern C++ Design: Generic Programming and Design Patterns Applied; A. Alexandrescu; Addison-Wesley; 2001
- The Elements of Programming Style; Brian W. Kernighan, P. J. Plauger; Computing Mcgraw-Hill; 1978
- Head First Design Patterns; Elisabeth Freeman, Eric Freeman, Bert Bates, Kathy Sierra; O'Reilly Media, Inc.; 2004

O PREDMETU (5)

Literatura: razvojne metodologije

- 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
- Extreme Programming Adventures in C#; Ron Jeffries; Microsoft Press; 2004

O PREDMETU (6)

Literatura: C++, generic programming

- [More] Effective {C++|STL}; S. Meyers; Addison Wesley; 1996
- 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 Computing Series; Matthew H. Austern; 1998
- C++ Templates: The Complete Guide; by David Vandevoorde, Nicolai M. Josuttis Addison-Wesley Professional; 2002