

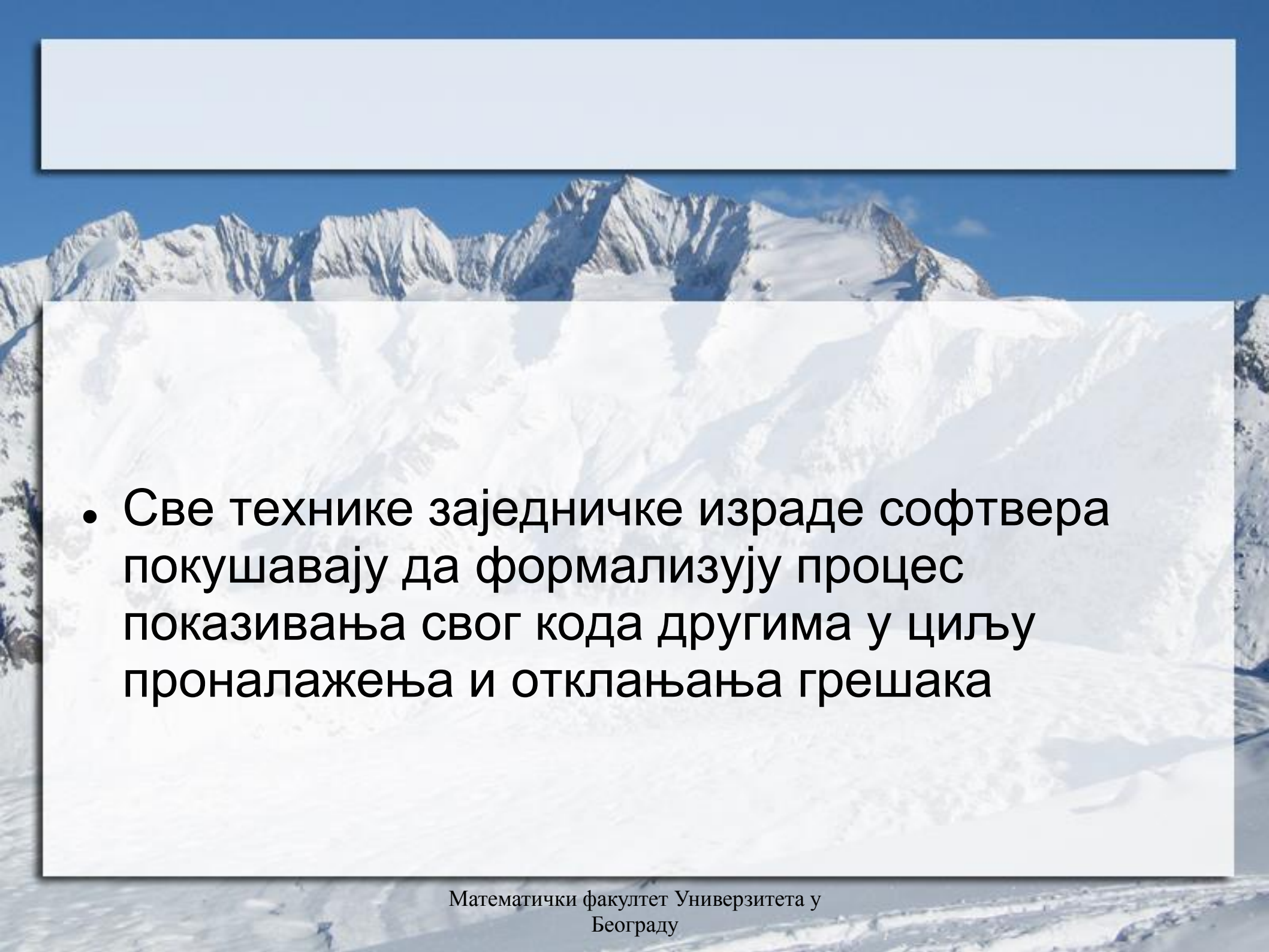
Семинарски из развоја софтвера 2

Тема: Заједничка израда софтвера
(eng. Collaborative Construction)

Студент: Сара Попадић
mr10017@alas.matf.bg.ac.rs

Увод

- Термин “collaborative construction” настао је у различитим фирмама независно готово у исто време, око 2000. године
- Мотивација: понекад када некоме објашњавамо проблем, који решавама, тражећи помоћ од саговорника, сетимо се решења без његове помоћи у тренуцима излагања проблема

- 
- Све технике заједничке израде софтвера покушавају да формализују процес показивања свог кода другима у циљу проналажења и отклањања грешака

Преглед техника за заједнички развој

- Програмирање у пару (*eng.pair programming*)
- Формална инспекција(*eng.formal inspection*)
- Неформални технички прегледи (*eng.unformal technical reviews*)
- Читање документа (*eng.document reading*)
- И многе друге технике у којима програмери деле одговорност за креирање кода

Основна идеја

- Све технике заједничког развоја (развој у сарадњи са другима), упркос њиховим разликама, су базиране на идеји да онај ко развија софтвер не може да сагледа слабе тачке у свом раду, а неко “са стране” (ко не ради на том делу кода) другачије види код и/или проблем

Заједничка израда допуњује друге технике за контролу квалитета

- Примарни циљ заједничког развоја је побољшање квалитета софтвера
- Ефикасност уочавања грешака при оваквом начину рада је знатно већи него при тестирању програма(пројекта)

Додатна предност

- Смањење времена потребног за развој, а то смањује трошкове израде

Програмирање у пару

- Један куца код, а други члан пара гледа код (током куцања), тражи грешке и размишља о решењу
- замена улога/задатака после неког временског рока
- Данас се ова техника често примењује

Кључ успеха за програмирање у пару

- Поштовати стандарде кодирања
- Не допустити да се програмирање у пару претвори у гледање
 - Онај ко не програмира требало би да буде активан учесник у раду:
 - Анализира код
 - Размишља о томе шта следеће да се програмира
 - Како тестирати код и побољшати га

Кључ успеха за програмирање у пару

- Не инсистирати на програмирању у пару за решавање једноставних задатак
- Редовна промена улога/задатака у пару (ротирање)
- Усагласити темпо куцања и праћења
- Обоје морају добро видети монитор
- Избегавати упаривање новајлија
 - бар један је раније програмирао у пару

Кључ успеха за програмирање у пару

- Не приморавати оне који нису у добрим односима да раде у пару
- Одредити тим лидера
 - Одговоран за резултат рада
 - Задужен за комуникацију са другима, који раде на пројекту (ван пара)

Добре стране програмирања у пару

- Чланови пара постају искренији један према другом и међусобно се охрабрују да би одржали висок ниво квалитета иако су под протиском због времена за завршетак
- Побољшава се квалитет кода, а читљивост и разумљивост кода расту

Тенички прегледи(*technical reviews*)

- Проучавани су дуже него програмирање у пару, јер су резултати били импресивни:
 - IBM је открио да сваки сат инспекције замењује чак 100 сати кодирања (тестирања и исправљања грешака)
 - Hewlet Packard је уштедео \$21.500.000 за годину дана
 - Студија о великим програмима је утврдила да сваки сат инспекције замењује 33 сата одржавања кода и инспекција повећана 20 пута је ефикаснија од тестирања
 - И тако даље

Ефекати при техничком прегледу (грешке)

- Не само да се уочавање грешака ефикасније ради него тестирањем, већ заједничка израда проналази више врста (типова) грешака него што то чини тестирање
- Рецезент може да уочи:
 - Нејасне поруке о грешкама
 - Неадекватне коментаре
 - Понављање делова кода који би требало да буду консолидовани...

Ефекати при техничком прегледу

- Програмер када зна да ће његов код него прегледати, ради много пажљивије
- Ово је одличан начин провере квалитета програма, чак иако би тестирање било довољно добро за откривање грешака
- Рецезенти дају повратну информацију програмерима

Формалне инспекције(formal inspections)

- Michel Fagan, IBM
- Инспекција је посебан начин прегледа кода, који се показао као веома ефикасан у отривању грешака и исплативији од тестирања
- Пажња рецезента је усмерана на делове кода са којима је раније било проблема
- Нагласак је на проналажењу грешака, а не на њиховом отклањању

Формалне инспекције(formal inspections)

- Рецезенти на састанак инспекције доносе листу са проблемима које су открили
- Различите улоге су додељене различитим учесницима
- Модератор инспекције није аутор у пројекту
- Модератор је прошао специјалну обуку
- Подаци о свим инспекцијама се чувају, да би побољшали следеће инспекције
- Главни менаџер не присуствује састанку инспекције, док технички лидери могу присуствовати

Резултати инсекција

- Око 60% грешака се открије. Боље технике су само: тестирање прототипа и high-volume beta testing
- Смањен је број грешака за 20-30%, у односу на мање формалне инспекције
- За процену напретка:
 - Да ли се посао обавља?
 - Да ли се посао добро обавља?

Формалне инспекције

- Програмери који су у тиму који користи формалне инспекције брзо напредују

Концепт колективног власништва

- У неким моделима развоја софтвера програмеру “припада” само онај део кода на ком ради и постоје знанична или незванична ограничења у вези са изменом туђег кода
- Код колективног власништва сав код је “власништво” групе и сваки члан групе може мењати сваки део кода

Предности колективног власништва

- Код је бољег квалитета (јер више људи ради на истом)
- Ризик да неко напусти пројекат се смањује
- Исправљање грешака је брже

Програмирање у пару/формалне инспекције

- Програмирање у пару даје квалитет кода сличан као формалне инспекције
- Цена целокупне израде пројекта програмирањем у пару повећава трошкове у односу на самосталан рад(једне особе) за 10-25%, али смањује време развоја за 45%, што може бити веома значајно. То није предност у односу на инспекције које дају сличне резултате.

Закључак

- Заједнички развој -> већи проценат успешности у проналажењу грешака и ефикасније од тестирања
- Различите врсте грешака



Хвала на пажњи!

Сара Попадић