Модели процеса развоја софтвера

Развој софтвера (2)

проф. др Милан Гњатовић

Факултет за компјутерске науке
Универзитет Џон Незбит

15.03.2016.

Садржај

Са претходног предавања

- Са претходног предавања:
  - Развој и коришћење софтвера су подложни грешкама.
  - Неисправни или неадекватни софтвери могу да доведу до критичних последица.
  - Зато је увођење инжењерске дисциплине у процес развоја софтвера неопходно.
- Моделовање процеса развоја софтвера је корак у том смеру.

Модел процеса развоја софтвера

- Основна намена модела је да установи редослед фаза развоја софтвера и критеријуме за прелазак из једне фазе у другу, укључујући:
  - критеријуме за утврђивање комплетности извршења тренутне фазе, и
  - критеријуме за избор и започињање наредне фазе.
- Другим речима, модел даје одговоре на следећа питања везана за развој софтвера:
  - До када треба извршавати тренутну активност?
  - Шта треба урадити следеће?
- Појам модела процеса није исто што и појам софтверске методе.

„Имплементирај и поправи“ (енг. *code-and-fix*)

- Модел из најранијих дана развоја софтвера.
- Укључује два корака:
  - напиши кôд,
  - реши проблеме у кôду.
- . . . , тј. прво се врши имплементирање, а тек потом се разматрају питања спецификације, дизајна, тестирања и одржавања.
- Нови термин: „имплементација“:
  - Имплементација се односи на део развоја софтвера у коме се функционалност софтвера реализује кроз програмски кôд.

Недостаци овог приступа (и поуке)

- Недостатак:
  - Након бројних преправки, изворни кôд постаје лоше структуриран, што наредне преправке чини још тежим.
- Поука:
  - Дизајн софтвера треба да претходи имплементацији.
- Нови термин: „дизајн“:
  - Дизајн[1] се односи на део развоја софтвера у коме се одлучује КАКО ће се софтвер реализовати.

[1] Више о дизајну на следећим предавањима.

Недостаци овог приступа (и поуке)

- Недостатак:
  - Чак и ако је добро дизајниран, нема гаранције да ће софтвер одговарати захтевима и потребама крајњег корисника.
- Поука:
  - Спецификација софтвера треба да претходи дизајну.
- Нови термин: „спецификација“:
  - Спецификација[2] се односи на део развоја софтвера у коме се одлучује ШТА софтвер треба да ради.

[2] Више о спецификацији на неком од следећих предавања.

Недостаци овог приступа (и поуке)

- Недостатак:
  - Преправке су тешке, јер кôд није припремљен за тестирање и измене.
- Поуке:
  - Тестирање и измене треба препознати као засебне фазе у развоју софтвера.
  - Иако се тестирање и измене врше након (макар парцијалне) имплементације, планирање ових фаза треба извршити у ранијим фазама развоја софтвера.
  - Другим речима, морамо знати функционалне и друге захтеве у односу на које се софтвер тестира.

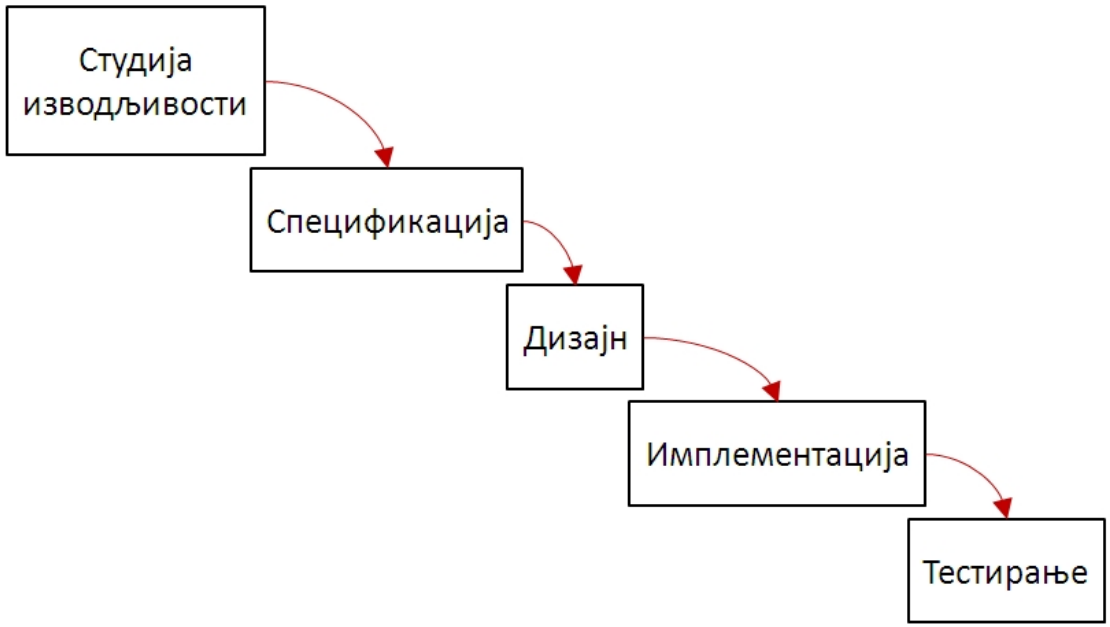
Секвенцијални модел (енг. *waterfall model*)

- Секвенцијални модел је развијен да би решио ове недостатке.
- Овај модел подразумева развој софтвера у секвенцијалним фазама[3]:
  - студија изводљивости,
  - спецификација,
  - дизајн,
  - имплементација,
  - тестирање.

[3] Код различитих аутора се јављају детаљнији спискови фаза секвенцијалног модела. Овде су приказане основне фазе које су карактеристичне за све варијације.

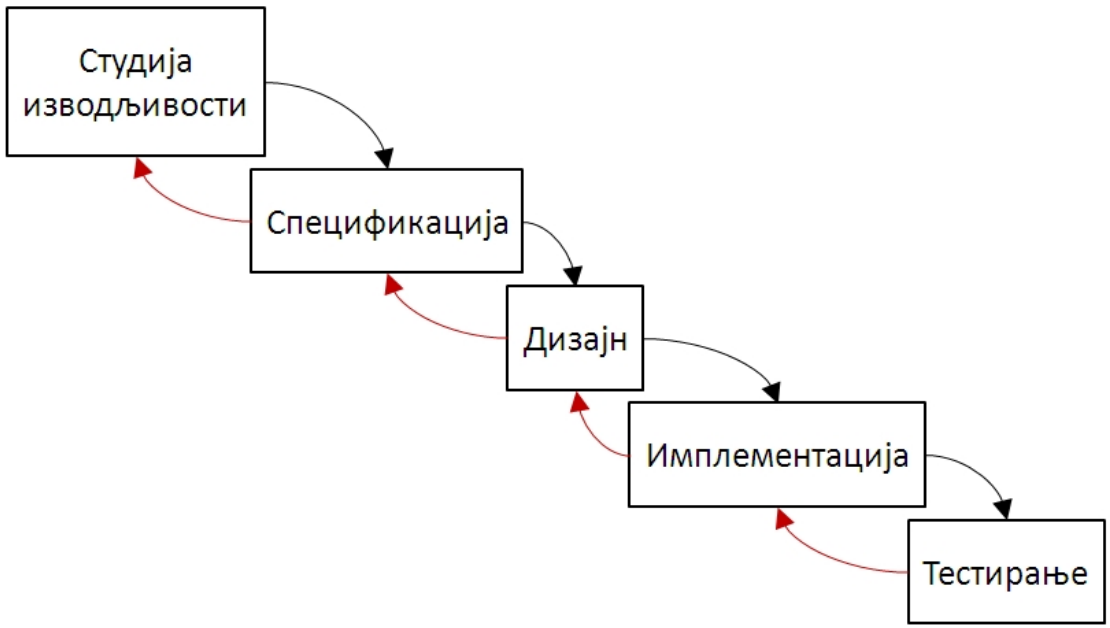
Основни секвенцијални модел (1956.)

- Основни секвенцијални модел предвиђа само један пролаз кроз фазе.
- Свака фаза укључује поступке валидације и верификације:
  - дизајн се проверава у односу на спецификацију, имплементација у односу на дизајн, итд.



Унапређени секвенцијални модел — повратне везе (1)

- Валидација и верификација на нивоу фазе нису довољне, јер проблеми понекад нису уочљиви док се не пређе у наредну фазу.
- Зато се уводе повратне везе између узастопних фаза.

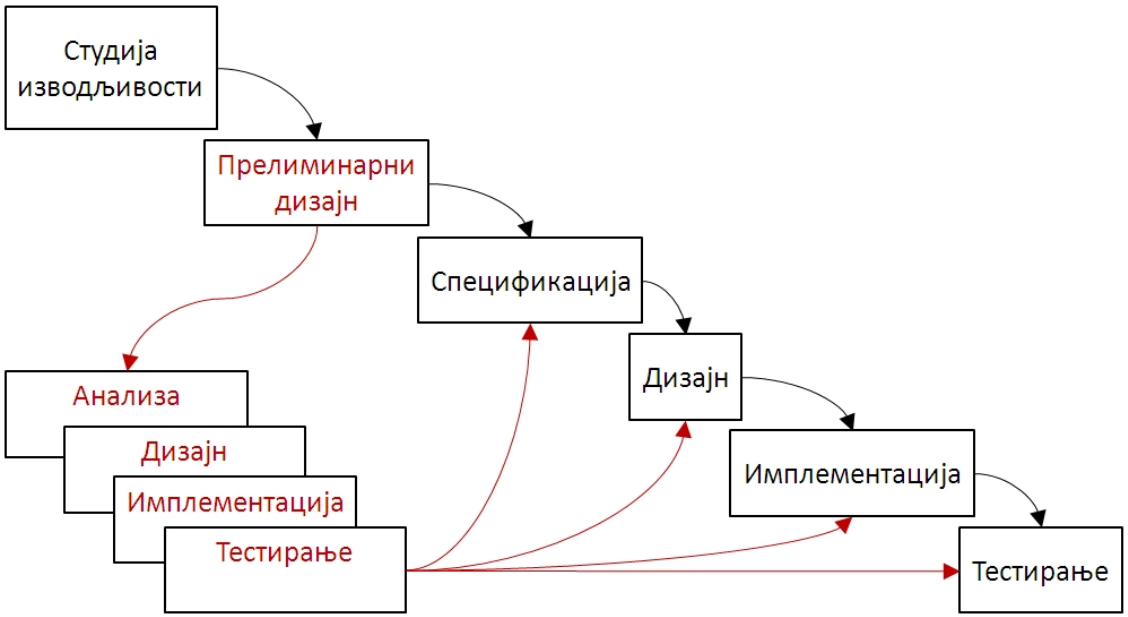


Унапређени секвенцијални модел — повратне везе (2)

- Повратне везе преко више фаза нису предвиђене (иако их неки аутори предлажу):
  - јер би дозвољавале скупе преправке у процесу развоја (ово желимо да минимизујемо), и
  - не би биле доследне идеји секвенцијалног развоја.
- Непожељни сценарио је да се грешка која је направљена у раној фази развоја открије тек у касној фази. Отклањање оваквих грешака може да услови преправку великог обима (и цене).

Унапређени секвенцијални модел — прототип

- Уводи се додатни корак (енг. „*build it twice*“) у ком се развија прототип, и који се извршава паралелно са фазама спецификације и дизајнирања софтвера.
- Овај корак смањује ризик развоја неадекватног софтвера, у случајевима кад дизајнер није сигуран како да развије софтвер.



Примарни недостатак секвенцијалног модела

- Модел захтева потпуно разрађену документацију као критеријум за утврђивање комплетности фаза спецификације и дизајна (енг. „*document-driven*“).
- Овај захтев је адекватан за класе софтвера за које је могуће детаљно дефинисати захтеве унапред, попут компајлера и сигурносних система. За ове класе, секвенцијални модел је обично и најприкладнији, и омогућава развој робустног софтвера.
- За многе друге класе софтвера није. Нпр., захтевање разрађене спецификације корисничких интерфејса није оправдано у ситуацијама када дизајнер нема комплетно разумевање тих интерфејса, и може да доведе до дизајнирања и писања неупотребљивог кôда.

Пример неразумевања интерфејса

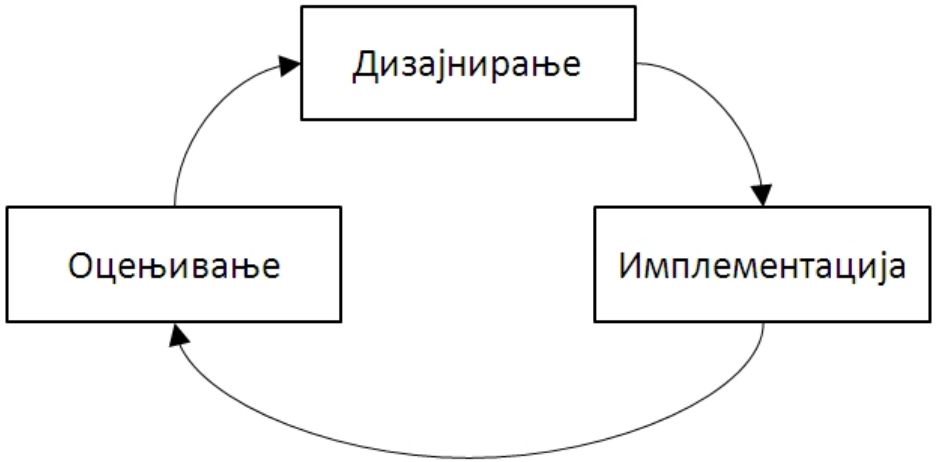
- Претпоставимо да развијамо дијалошки систем.
  - Како се одређује речник израза и фраза које корисници могу да употребе?
  - Како се дефинише граматика по чијим правилима корисници могу да формирају своје дијалошке чинове?
  - Како се предвиђају могући проблеми у комуникацији?
  - Како се дефинишу дијалошке стратегије за превазилажење проблема?
  - Како се моделује дијалог?
- Одговори на оваква питања не смеју да се базирају само на интуицији корисника или дизајнера (више о овоме на неком од следећих предавања).

Уопштено

- Зашто секвенцијални модел није адекватан за развој корисничких интерфејса?
- Развој корисничких интерфејса је инхерентно ризичан.
  - Дизајнер није корисник — интерфејс не треба да се прави по мери дизајнера.
  - Корисник често не зна унапред шта жели од интерфејса.
- У секвенцијалном моделу, корисник се појављује само на два места: на почетку (приликом дефинисања корисничких захтева) и на крају (приликом тестирања прихватљивости софтвера).
  - Грешка у дизајну интерфејса се уочава тек на крају.
- Грешке у интерфејсу често условљавају знатне измене у спецификацији и дизајну.
  - Пажљиво написани и тестирани кôд постаје неупотребљив.

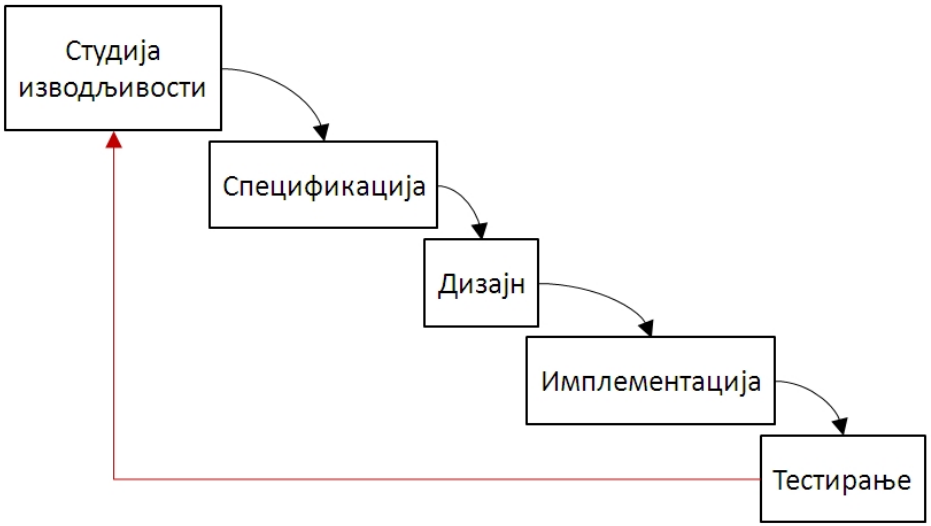
Итеративни дизајн

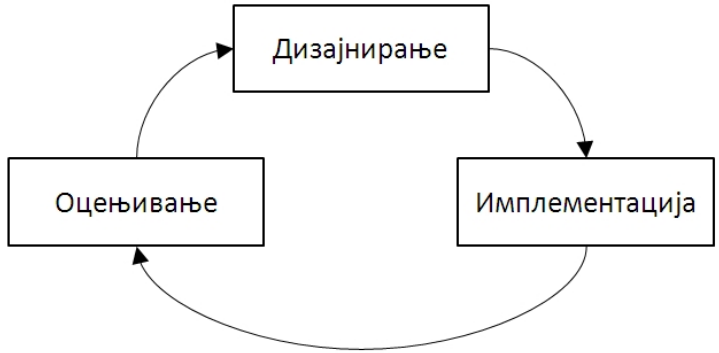
- Итеративни дизајн нуди начин за минимизацију инхерентно присутног ризика у развоју софтвера.
- Софтвер се итеративно побољшава кроз циклусе који укључују три основне фазе:
  - осмишљавање софтвера (дизајнирање),
  - реализацију софтвера (имплеменатирање),
  - тестирање софтвера (оцењивање).



Итеративни дизајн и секвенцијални модел

- Да ли је итеративни дизајн еквивалентан најгорем случају коришћења секвенцијалног модела — када се прођу све фазе, и тек на крају открије грешка, па се онда крене из почетка?





Погрешна употреба итеративног дизајна

- Одговор: НЕ.
- Међутим, немали број комерцијалних софтвера је развијен примењујући идеју итеративног дизајна погрешно:
  - примени се секвенцијални модел, произведе се софтвер и пусти на тржиште;
  - када се корисници жале на недостатке, следећа верзија софтвера се развија на исти начин.
- Не треба очекивати да ће корисници који плаћају софтвер:
  - дугорочно прихватити да раде оцењивање софтвера уместо фирме,
  - купити следећу верзију софтвера.

Спирални модел (1)

- Спирални модел је пример исправне примене итеративног дизајна (енг. „*risk-driven*“).

Спирални модел (2)

- Развој се одвија кроз неколико итерација.
- Радијус се може интерпретирати на два начина:
  - кумулативна цена развоја до датог тренутка,
  - степен у ком прототип одговара жељеном софтверском производу.
- То значи да су рани прототипи јефтини, и да само у малом степену одговарају жељеном производу.

Минимизација ризика

- Ризик је највећи у раним итерацијама, јер тада знамо најмање о специфичним проблемима у развоју софтвера.
  - Зато се најмање труда улаже у ране прототипе.
  - Рани прототипи се не праве да би се користили, већ да бисмо научили о специфичним проблемима.
  - Уколико постоји више опција, може се паралелно направити и оценити више прототипа.
- Након неколико итерација смо научили довољно да избегнемо критичне дизајнерске грешке.
  - Сада правимо прототип који ћемо да користимо.
  - Овај прототип се даље оцењује и побољшава кроз наредне итерације.
- Квалитет прототипа расте са бројем итерација.
  - Само се зрела верзија прототипа прослеђује корисницима.

Литература

- Boehm, B.W., A spiral model of software development and enhancement, *Computer* 21 (5), pp. 61-72, 1998.
(Само до поглавља „Using the spiral model“.)

Хвала на пажњи

- Питања су добродошла.