Izvještaj o inspekciji koda

VVS – Verifikacija i validacija softvera

Emin Begić, Aldin Velić, Zana Beljuri, Mirnes Fehrić

Novembar 2025

1. Uvod

U sklopu laboratorijske vježbe iz predmeta *VVS – Verifikacija i validacija softvera* izvršena je formalna inspekcija koda nad projektom *TeamTaskManager*. Inspekcija je sprovedena kroz Azure DevOps koristeći:

- Pull Request (PR) mehanizam,
- komentare na konkretne linije koda,
- kreiranje Work Item / Issue kartica za pronađene defekte.

Pri pregledu su korištene generičke checkliste za inspekciju koda (Structure, Variables, Defensive Programming itd.), kao i skala ozbiljnosti grešaka.

Ciljevi inspekcije bili su:

- identifikovati greške u implementaciji,
- klasificirati defekte prema vrsti i ozbiljnosti,
- dokumentovati defekte kroz Issue kartice,
- uraditi Pareto analizu uzroka i defekata.

2. Proces inspekcije

Moderator inspekcije bio je **Emin Begić**, dok su ostali članovi tima učestvovali kao inspektori.

Proces inspekcije tekao je kroz sljedeće korake:

1. Kreiran je poseban branch `inspekcija`.
2. Na branch je dodan kod tima čiji se projekat pregledao.
3. Napravljen je Pull Request ka branchu `main`.
4. Svaki inspektor je, uz dodijeljene checkliste, pregledao dio koda i ostavljao komentare u PR-u.

5. Za odabrane komentare kreirane su pripadajuće Issue kartice (Work Items tipa Bug/Issue).

6. Svi prikupljeni defekti sumirani su i analizirani putem Pareto analize.

3. Identifikovani defekti

Tokom inspekcije pronađeno je ukupno **11 defekata**, od kojih je za svaki kreirana odgovarajuća Issue kartica u Azure DevOps-u. Klasifikacija je rađena prema checklist grupama:

- **Structure**
  - Duplicirani kod u `TaskStatisticsService` (ponavljanje logike u više metoda).
  - Duplirani `using` u `TaskStatisticsService`.
- **Variables**
  - Neiskorišteno privatno polje `_taskCommenting` u `TaskCommentingService`.
  - Neiskorišten alias `CustomerModel` u `MemberService`.
  - Greška u kucanju u parametru `memeberId` u konstruktoru `TaskMember`.
- **Defensive Programming**
  - `TaskService` dozvoljava kreiranje sa `null` Task objektom.
  - `GetTasks()` vraća direktnu internu listu u `TaskManagementService` (narušena zaštita strukture podataka).
  - Nedovoljna validacija u `AddComment` (ne provjerava se `author` parametar).
  - Javne liste u `TaskManagment` (`_tasks`, `_members`, `_taskAssignments`) omogućavaju nekontrolisane izmjene izvan servisa.
  - Konstruktor `Task` ne validira parametre (naziv, opis, dodijeljeni član).
  - Javno polje `hashLozinka` u `Member` klasi, koje predstavlja sigurnosni rizik.

Sume po kategorijama su:

Kategorija defekta	**Broj grešaka**
Defensive Programming	6
Variables	3
Structure	2
Ostalo	0
Ukupno	**11**

4. Pareto analiza

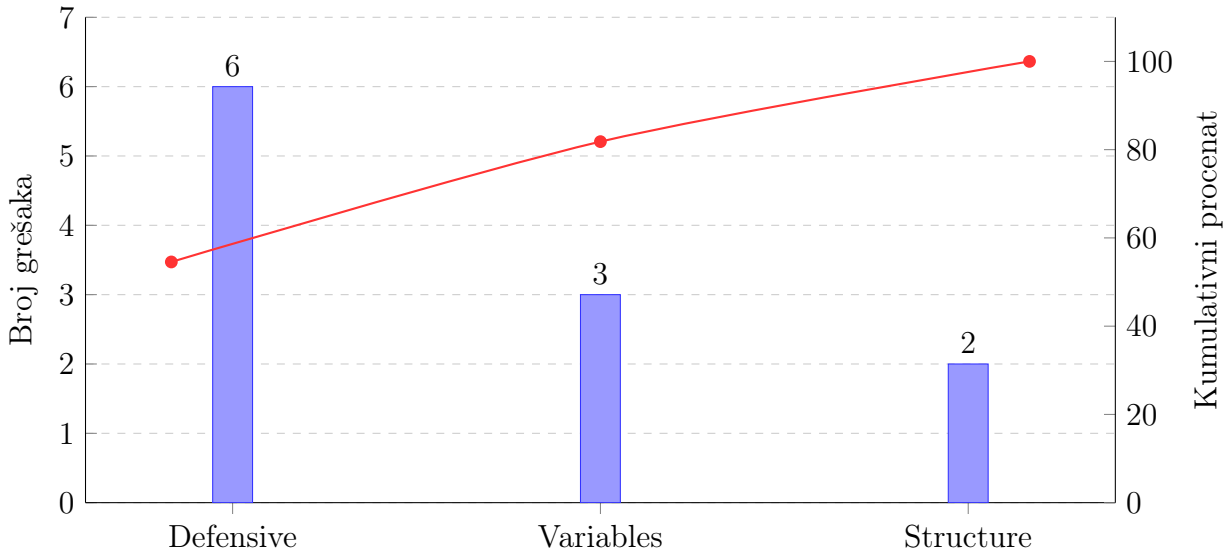
Pareto analiza pokazuje koje kategorije defekata najviše doprinose ukupnom broju grešaka i gdje bi poboljšanja procesa razvoja imala najveći efekat.

4.1. Tabela Pareto analize

Kategorija	Broj	Procenat	Kumulativno
Defensive Programming	6	54.55%	54.55%
Variables	3	27.27%	81.82%
Structure	2	18.18%	100%

4.2. Grafički prikaz (Pareto dijagram)

Na slici 1 prikazan je Pareto dijagram sa prikazom broja grešaka po kategoriji i kumulativnom linijom procenata.



Slika 1: Pareto dijagram za identifikovane defekte

5. Sažetak najvažnijih uočenih problema

Ključni zaključci na osnovu uočenih defekata su:

- **Dominantna kategorija su greške iz oblasti Defensive Programming (54.55%).** Najviše problema nastaje zbog nedovoljne validacije ulaznih podataka (`TaskService`, `Task` konstruktor, `AddComment`), javnih struktura i polja (`hashLozinka`, public liste u `TaskManagment`) i izlaganja internih struktura podataka.
- **Greške u radu sa varijablama (27.27%).** Neiskorištena polja (`_taskCommenting`, alias `CustomerModel`) i greške u kucanju imena dovode do zbunjujućeg i manje održivog koda.

- **Strukturni problemi (18.18%).** Duplicirani kod i duplirani `using` direktoriji ukazuju na potrebu za dodatnim refaktoringom i čišćenjem projekta.

6. Zaključak

Inspekcija koda pokazala je da najveći broj defekata dolazi iz oblasti:

- **Defensive Programming** (validacija ulaza, zaštita podataka, enkapsulacija struktura),
- **Variables** (imenovanje, inicijalizacija i korištenje varijabli),
- **Structure** (organizacija koda, duplirani i nepotrebni dijelovi).

Fokusiranjem na poboljšanje defenzivnog programiranja (provjere `null` vrijednosti, zaštita osjetljivih podataka, kontrola pristupa strukturama podataka), kao i na čišćenje i refaktoring koda (uklanjanje mrtvog koda, dupliranih dijelova i neiskorištenih varijabli), kvalitet softvera se može značajno unaprijediti.

Ovaj proces inspekcije je ujedno poslužio kao praktična demonstracija tehnika verifikacije i validacije softvera u realnom timskom okruženju uz korištenje Azure DevOps alata.