

Univerzitet u Kragujevcu
Fakultet inženjerskih
nauka



Seminarski rad iz predmeta:
Softverski Inženjering

Tema seminarskog rada:
Prepoznavanje tumora mozga

Student:
Aleksandar Mihajlović 586/2015

Predmetni profesor:
Dr. Nenad Filipović

Kragujevac 2018.

SADRŽAJ

POSTAVKA PROJEKTOG ZADATAKA.....	3
IDEJA SEMINARSKOG RADA.....	4
REALIZACIJA PROJEKTOG ZADATAKA	5
OBJAŠNJENJE KODA.....	6
Klasa Lekar.....	6
Klasa pathFinder.....	6
Klasa Analyze.....	7
Klasa TumorNaMozgu	8
UML DIAGRAMI.....	11
Dijagram klasa (Class Diagram)	11
Dijagram aktivnosti (Activity Diagram)	12
Dijagram sekvenci (Sequence Diagram)	13
Dijagram stanja (State Machine Diagram)	13
Dijagram slučajeva korišćenja (Use Case diagram)	15
LITERATURA	16

POSTAVKA PROJEKTOG ZADATAKA

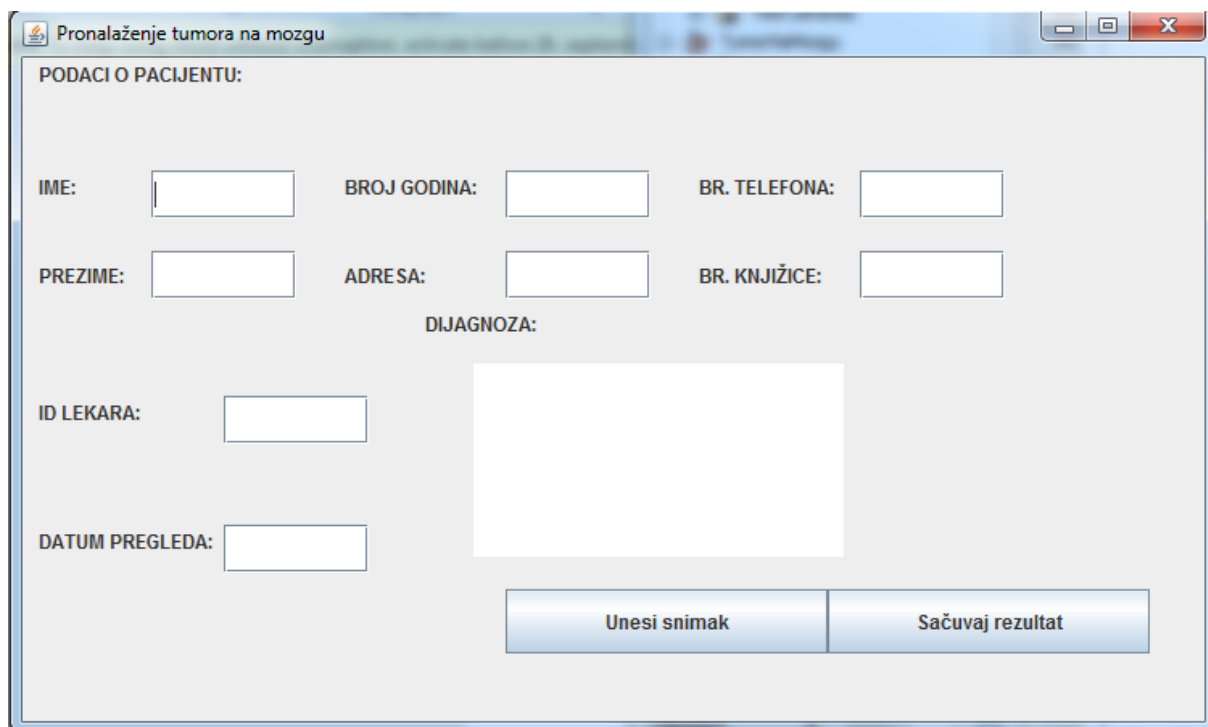
Napraviti aplikaciju koja obavlja prepoznavanje kontura sa 2D snimaka DICOM formata medicinskih slika. Potrebno je uraditi učitavanje DICOM formata i na osnovu zadatog thresholda izvršiti prepoznavanje tumora mozga. Kao rezultat dobijaju se nove slike sa segmentisanim tkivom od interesa. Preporučuje se razvojno okruženje programskog jezika Visual C++ 6.0 ili Visual Studio (verzije sve do 2015).

IDEJA SEMINARSKOG RADA

Ideja seminarskog rada je napraviti aplikaciju koja će prepoznaje tumor na mozgu, ukoliko ga ima, uz pomoć DICOM snimka mozga. Aplikacija je realizovana korišćenjem programskog jezika Java u programu Netbeans IDE 8.2. Nakon pokretanja aplikacije, otvara se prozor koji služi lekaru za interakciju sa programom. Lekar može uneti sve podatke o pacijentu, nakon čega je potrebno da učitava željeni snimak. Nakon učitavanja lekar dobija sliku sa kojom će mu biti obeležen mogući tumor plavom bojom. Lekar dalje može popuniti dijagnozu nakon otvorene slike. Posle popunjavanja izveštaja, lekar može sačuvati isti, gde će dobiti pdf fajl o popunjenim podacima kao i sliku sa naznačenim tumorom na mozgu, ako postoji.

REALIZACIJA PROJEKTOG ZADATAKA

Kao što je već naznačeno u prethodnom poglavlju, aplikacija je pisana u programskom jeziku java. Sadrži četiri klase: *TumorNaMozgu*, *Lekar*, *pathFinder*, *Analyze*. Prilikom pokretanja aplikacije, otvoriće se korisnički interfejs koji koristi lekar.



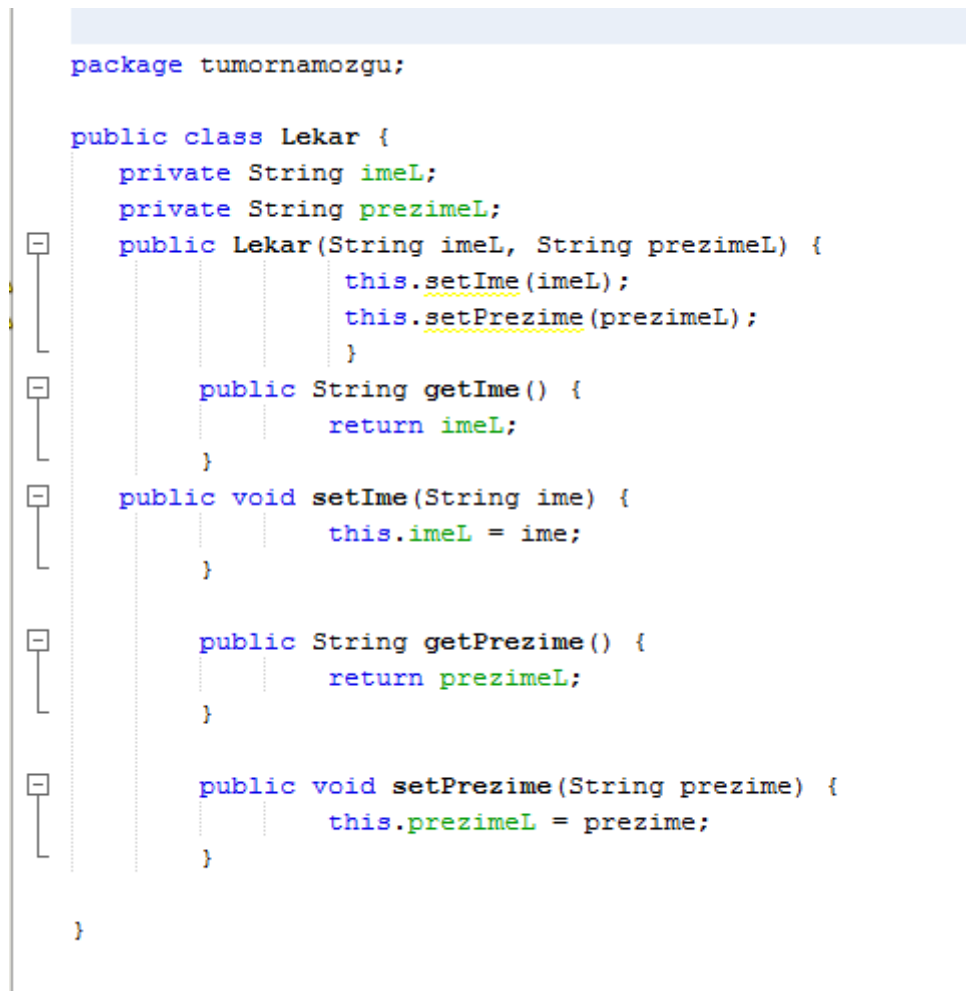
Slika 1 – Grafički korisnički interfejs

Lekar može popuniti prazna polja potrebnim podacima, nakon čega može klikom na dugme Unesi snimak, uneti željeni snimak. Nakon odabira snimka on će biti analiziran i prikazan zajedno sa originalnim snimkom. Lekar nakon pregleda snimaka može utvrditi dijagnozu, upisati je u potrebno polje i klikom na dugme Sačuvaj rezultat, sačuvati rezultat u pdf formatu, kao i snimak sa naznačenim tumorom. Korisnički interfejs je urađen uz pomoć biblioteke i funkcija *JavaXSwing-a*.

OBJAŠNJENJE KODA

Klasa Lekar

Ova klasa služi samo za prikaz i čuvanje podataka o lekaru koji koristi aplikaciju.



```
package tumornamozgu;

public class Lekar {
    private String imeL;
    private String prezimeL;
    public Lekar(String imeL, String prezimeL) {
        this.setIme(imeL);
        this.setPrezime(prezimeL);
    }

    public String getIme() {
        return imeL;
    }

    public void setIme(String ime) {
        this.imeL = ime;
    }

    public String getPrezime() {
        return prezimeL;
    }

    public void setPrezime(String prezime) {
        this.prezimeL = prezime;
    }
}
```

Slika 2 – Klasa Lekar

Klasa koristi 4 metode, prva metoda **getIme()**, vraća ime lekara nakon njenog pozivanja. Druga metoda **setIme()**, postavlja ime lekara, treća metoda **getPrezime()** vraća prezime lekara i četvrta metoda **setPrezime()** postavlja prezime lakara. Ova klasa je povezana sa klasom TumorNaMozgu i taj deo će biti objašnjen u nastavku.

Klasa pathFinder

Ova klasa služi za detekciju snimka koji lekar želi da bude analiziran radi pronalaženja tumora na mozgu. Klasa koristi biblioteku javax.swing. Metoda **pathFinder()**, vraća putanju snimka koji je lekar izabrao. Nakon izabranog fajla putanja u string formatu će biti sačuvana unutar promenljive path. Kod ove klase prikazan je ispod.

```

package tumornamozgu;

import javax.swing.JButton;
import javax.swing.JFileChooser;

public class pathFinder {

    String path;

    public pathFinder() {

        JButton open = new JButton();
        JFileChooser file = new JFileChooser();
        file.setCurrentDirectory(new java.io.File("."));
        file.setDialogTitle("Izaberi datoteku");

        if(file.showOpenDialog(open) == JFileChooser.APPROVE_OPTION) {
        }
        path = file.getSelectedFile().getAbsolutePath();

    }

}

```

Slika 3 – Klasa pathFinder

Klasa Analyze

Ova klasa služi za analizu i prepoznavanje tumora na mozgu. U njoj se koristi biblioteka *ij* (ImageJ API). Pomoću te biblioteke otvara se snimak koji je lekar izabrao i unutar programa je našteloan Treshold koji će biti korišćen za detekciju tumora, funkcijom ***IJ.setThreshold***. Zatim se funkcijom *IJ.run*, startuje analiza i u nju se upisuju parametri za tu analizu.

Parametar *size* uzima vrednosti od 0 do beskonačnosti. U tom parametru se definiše veličina polja koje će funkcija tražiti, one čestice koje su izvan ovog polja, biće ignorisane.

Parametar *pixel circularity* čestice veličine koje su izvan oblasti definisane u ovom parametru će takođe biti ignorisane. Parametar uzima vrednosti od 0 do 1.

Parametar *show* sadrži više opcija, tačnije to je drop-down meni, kojim se definiše na koji način će biti detektovan tumor na mozgu. Konkretno u ovom projektu izabran je OverlayMasks koji označava česticu na koju se sumnja da je tumor plavom bojom i numeriš je, ako se sumnja na više čestica biće više numeracija.

Parametar *exclude* služi za ignorisanje čestica koje su uz ivice slike.

Postoje i mnogi drugi parametri unutar funkcije, ali nisu korišćeni u ovom projektu.

Slika ovog dela koda nalazi se ispod.

```

package tumornamozgu;

import ij.IJ;
import ij.ImagePlus;
import ij.io.Opener;

public class Analyze {
    public Analyze() {
    }
    public void analyzing(String ulaz) {
        Opener o = new Opener();
        ImagePlus slika = o.openImage(ulaz);
        slika.show();

        IJ.setThreshold(146.0, 262.0);

        IJ.run("Analyze Particles...", "size=150-Infinity pixel circularity=0.20-1.00 show=[Overlay Masks] exclude ");
    }
}

```

Slika 4 – Klasa Analyze

Klasa TumorNaMozgu

Ova klasa je glavna klasa. Na početku klase se uzimaju podaci o lekaru kreiranjem objekta lekar. Zatim sledi deo sa grafičkim korisničkim interfejskom i tu su uglavnom funkcije za kreiranje tastera, polja za unos teksta, naslova, okvira i ostalo. Takođe se i definišu veličine istih i njihove pozicije na ekranu. Funkcija **b.addActionListener()** se izvršava kada lekar pritisne dugme **Unesi snimak**. Unutar te funkcije se nalaze objekti i pozivi na klase *Analyze* i *pathFinder*. Kao i poziv za analizu slike. Funkcija **b1.addActionListener()** se izvršava kada lekar pritisne dugme **Sačuvaj rezultat**. Unutar te funkcije se nalaze metode koje koriste biblioteku *org.pdfbox* koja služi za kreiranje pdf fajlova i unošenje podataka u njega. Osim snimanja potrebnog pdf fajla ova funkcija takođe i čuva snimak označenog tumora na mozgu funkcijom *IJ.save()*. Kod je prikazan u nastavku.

```

package tumornamozgu;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.awt.Color;
import java.io.IOException;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.filechooser.FileNameExtensionFilter;
import ij.IJ;
import ij.ImagePlus;
import ij.io.Opener;
import java.awt.FlowLayout;
import java.awt.Image;
import static java.awt.PageAttributes.MediaType.C;
import javax.swing.ImageIcon;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import org.pdfbox.exceptions.COSVisitorException;
import org.pdfbox.pdmodel.PDDocument;
import org.pdfbox.pdmodel.PDPage;
import org.pdfbox.pdmodel.edit.PDPageContentStream;
import org.pdfbox.pdmodel.font.PDType1Font;
import org.pdfbox.pdmodel.graphics.xobject.PDJpeg;
import org.pdfbox.pdmodel.graphics.xobject.PDXObjectImage;

```

Slika 5 – Klasa TumorNaMozgu-importi


```

public class TumorNaMozgu extends JFrame {

    public TumorNaMozgu() {
        Lekar lekar = new Lekar("Aleksandar", "Mihajlovic");
        JButton b = new JButton("Unesi snimak");
        b.setBounds(300, 330, 200, 40);
        JButton b1 = new JButton("Sačuvaj rezultat");
        b1.setBounds(500, 330, 200, 40);
        JTextField t1 = new JTextField();
        JTextField t2 = new JTextField();
        JTextField t3 = new JTextField();
        JTextArea t4 = new JTextArea();
        JTextField t5 = new JTextField();
        JTextField t6 = new JTextField();
        JTextField t7 = new JTextField();
        JTextField t8 = new JTextField();
        JTextField t9 = new JTextField();
        JLabel l = new JLabel();
        l.setBounds(400, 20, 470, 350);
        JLabel pac = new JLabel("<html>PODACI O PACIJENTU:");
        pac.setBounds(10, -140, 300, 300);
        JLabel ime = new JLabel("<html>IME:");
        ime.setBounds(10, -70, 300, 300);
        t1.setBounds(80, 70, 90, 30);
        JLabel prezime = new JLabel("<html>PREZIME:");
        prezime.setBounds(10, -15, 300, 300);
        t2.setBounds(80, 120, 90, 30);
        JLabel bg = new JLabel("<html>BROJ GODINA:");
    }
}

```

Slika 6 – Klasa TumorNaMozgu- deo koda realizacije GUI-a

```

        b.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent izaberi) {

                pathFinder b1 = new pathFinder();
                Analyze o1 = new Analyze();
                Opener o = new Opener();
                ImagePlus slika2 = o.openImage(b1.path);
                slika2.show();
                o1.analyzing(b1.path);

            }
        });

```

Slika 7 – Klasa TumorNaMozgu – kod za dugme Učitaj snimak

Sl

```
b1.addActionListener(new ActionListener() {  
public void actionPerformed(ActionEvent snimi) {  
    String x1=t1.getText();  
    String x2=t2.getText();  
    String x3=t3.getText();  
    String x4=t4.getText();  
    String x5=t5.getText();  
    String x6=t6.getText();  
    String x7=t7.getText();  
    String x8=t8.getText();  
    String x9=t9.getText();  
  
    try{  
        PDDocument doc = new PDDocument();  
        PDPage page = new PDPage();  
        doc.addPage(page);  
        PDPageContentStream content = new PDPageContentStream(doc, page);  
        content.beginText();  
        content.setFont(PDType1Font.HELVETICA, 26);  
        content.moveTextPositionByAmount(220, 750);  
        content.drawString("IZVESTAJ LEKARA");  
        content.endText();  
        content.beginText();  
        content.setFont(PDType1Font.HELVETICA, 20);  
        content.moveTextPositionByAmount(80, 700);
```

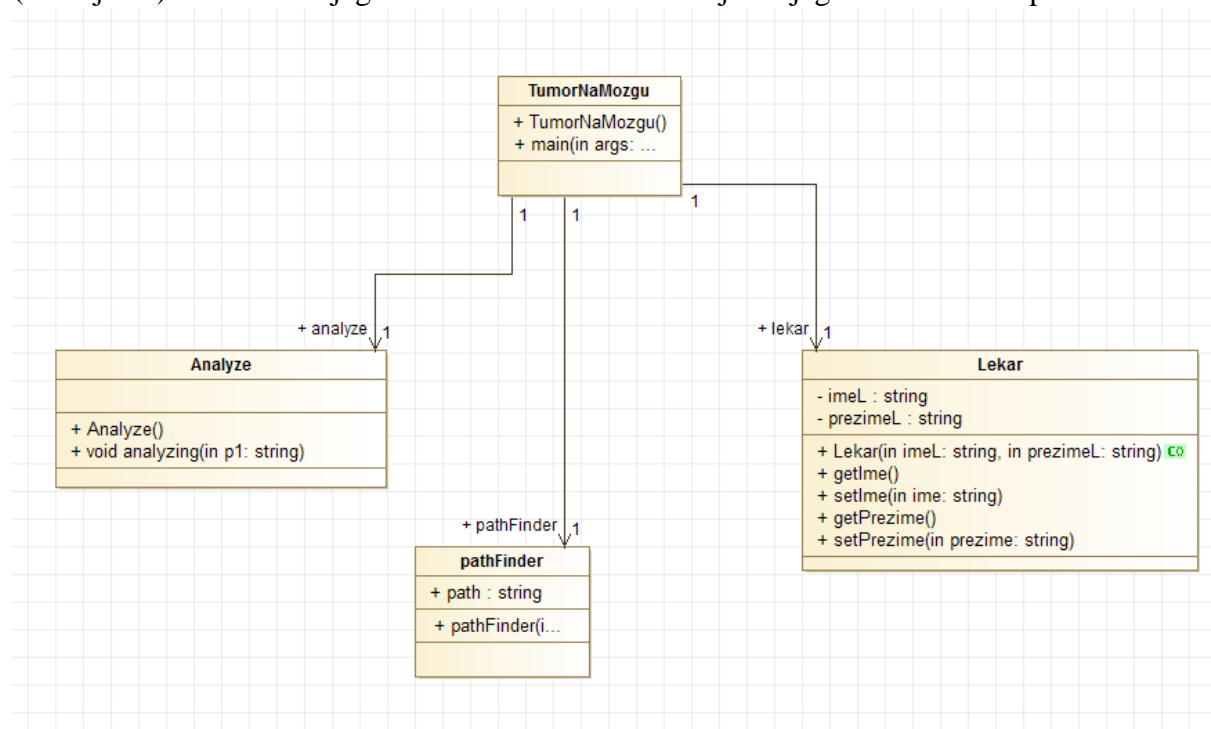
Slika 8 – Klasa TumorNaMozgu- kod za dugme Sačuvaj rezultat

UML DIAGRAMI

Undefined Modeling Language je standardni grafički jezik za modelovanje objektno-orijentisanog softvera. Osnovni element UML-a su dijagrami, elementi i relacije. Dijagrami mogu biti dijagrami ponašanja i dijagrami struktura.

Dijagram klasa (Class Diagram)

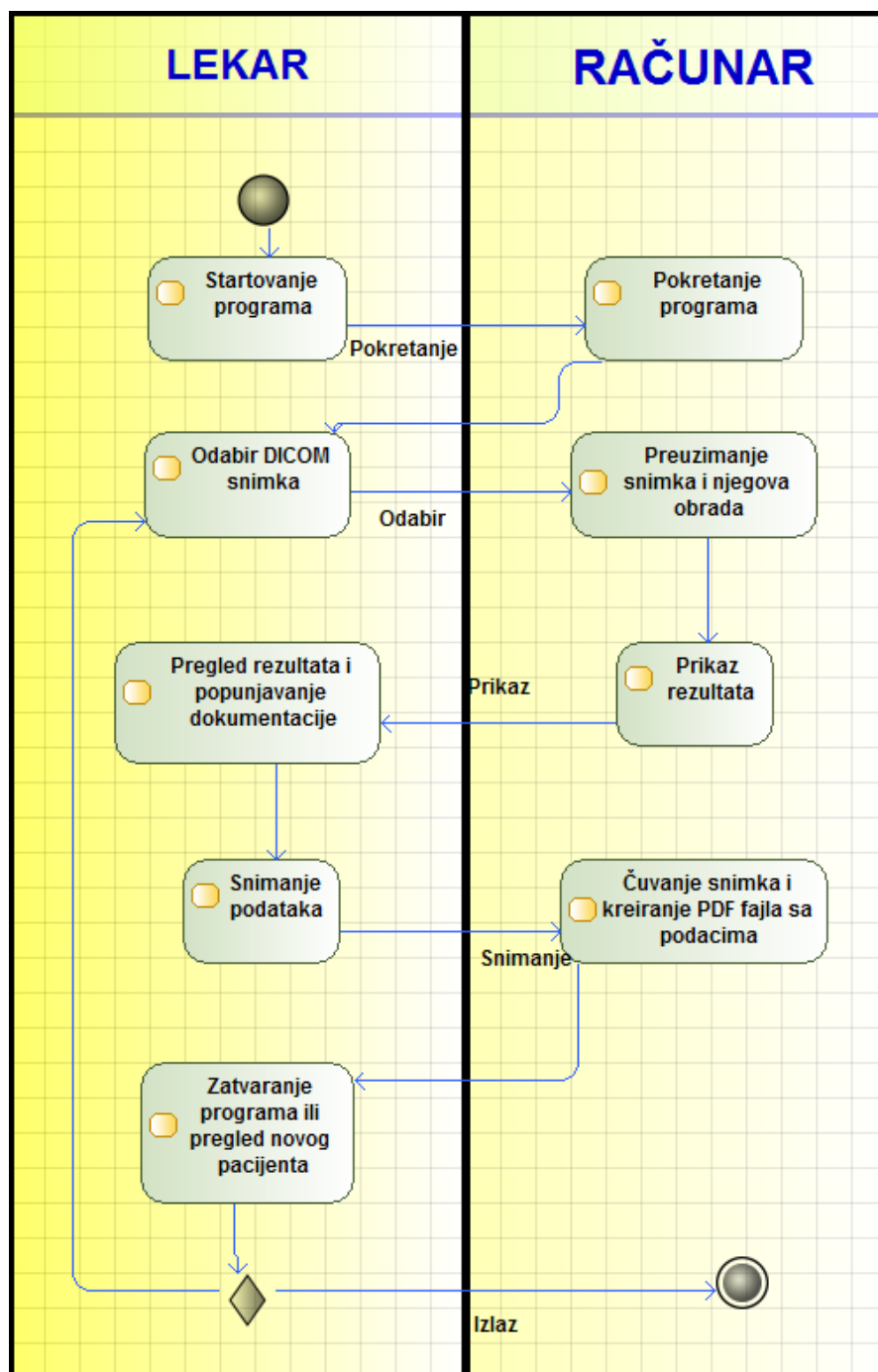
Dijagram klasa prikazuje skup klasa, interfejsa, saradnji, i drugih stvari strukture, povezanim relacijama. Dijagram klasa je graf obrazovan od temena (stvari) povezanih granama (relacijama). Elementi dijagrama klasa su stvari i relacije. Dijagram se nalazi ispod.



Slika 9 – Dijagram klasa

Dijagram aktivnosti (Activity Diagram)

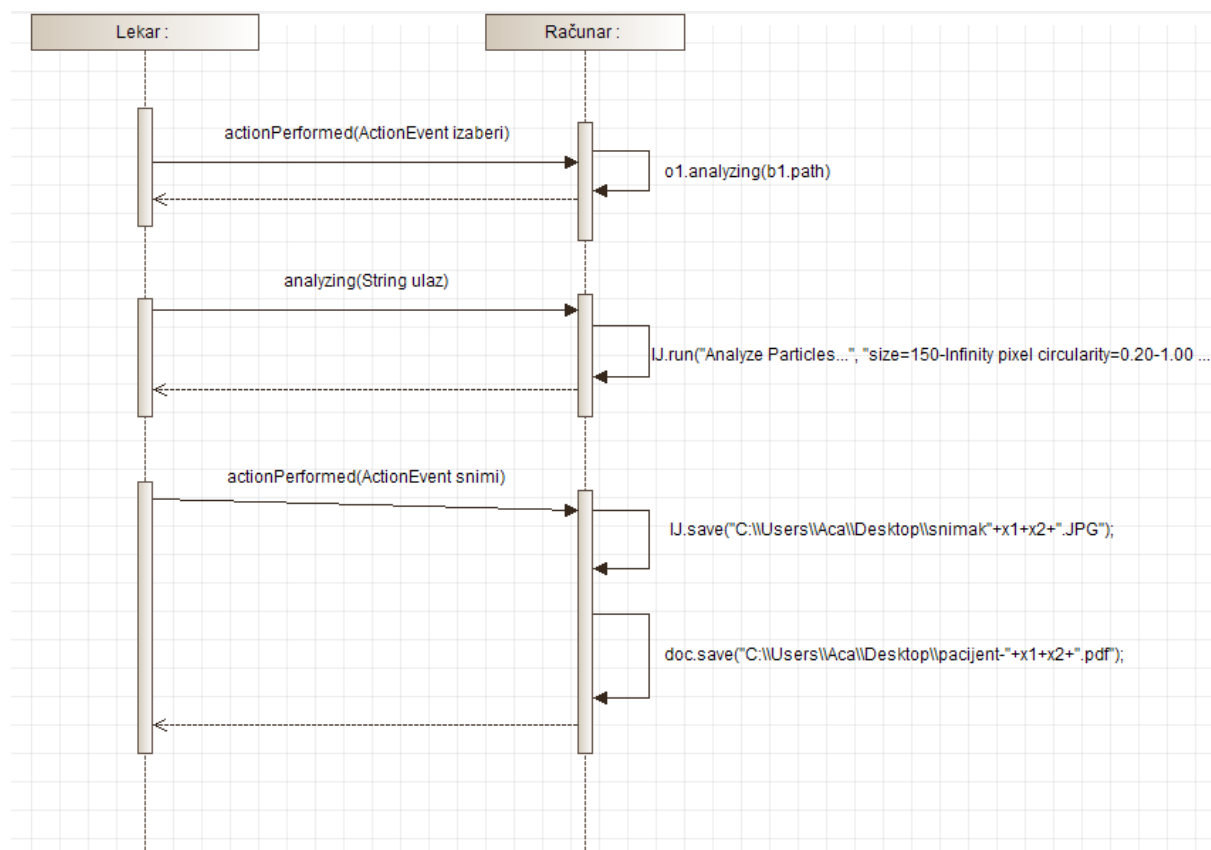
Dijagrami aktivnosti su namenjeni modeliranju dinamičkih aspekata (ponašanja) sistema. Dijagram aktivnosti prikazuje tok aktivnosti koju izvršavaju objekti i eventualno tok objekata između koraka aktivnosti.



Slika 10 – Dijagram aktivnosti

Dijagram sekvenci (Sequence Diagram)

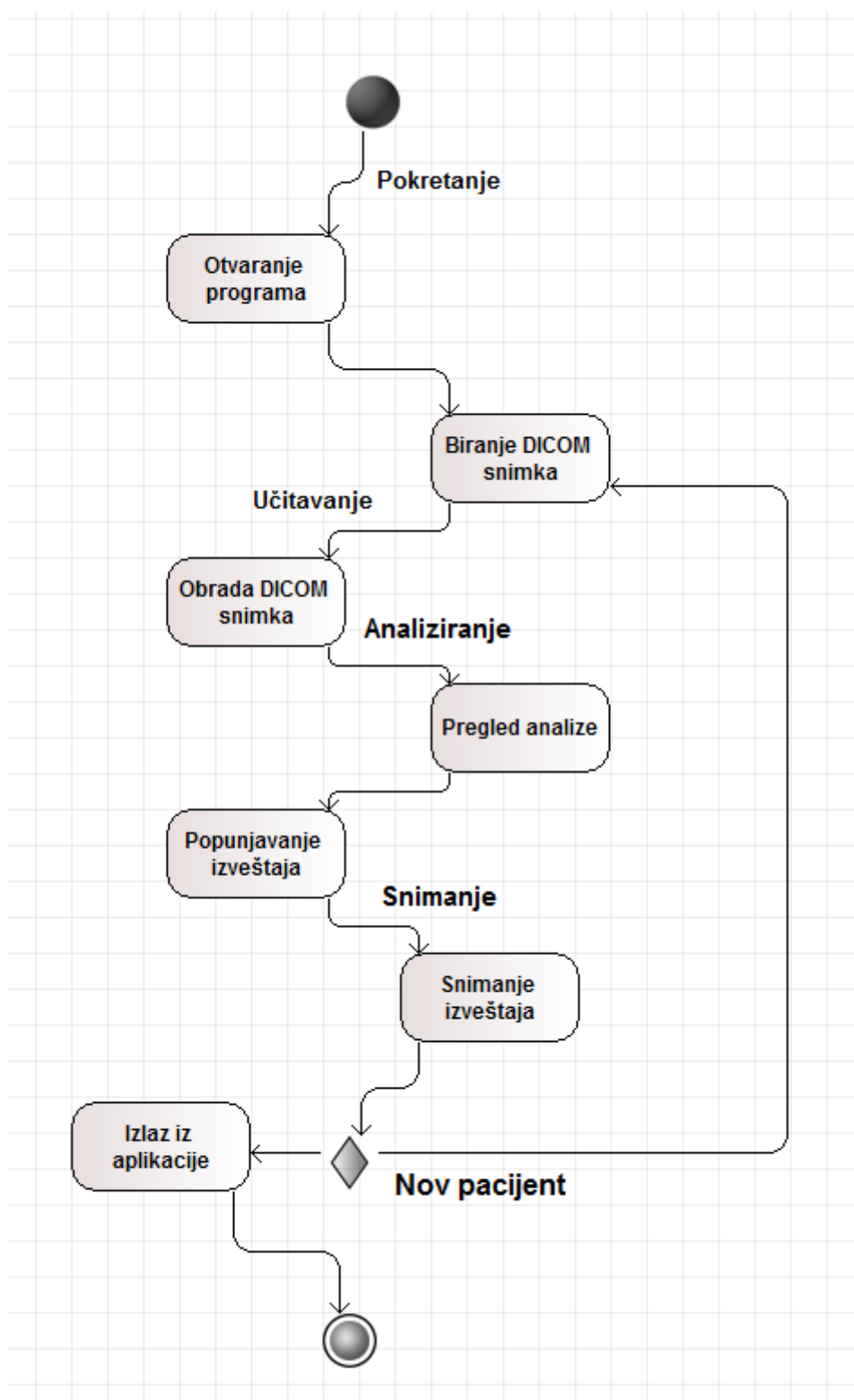
Dijagram sekvenci prikazuje komunikaciju između skupa objekata, koja se ostvaruje porukama koje objekti međusobno razmenjuju u cilju ostvarivanja očekivanog ponašanja. Detaljno opisuje kako se operacije izvode – koje poruke se šalju i kada. Sadrži dve dimenzije vertikalnu (označava vreme) i horizontalnu (označava objekte). Dijagram je prikazan na slici ispod.



Slika 11 – Dijagram sekvenci

Dijagram stanja (State Machine Diagram)

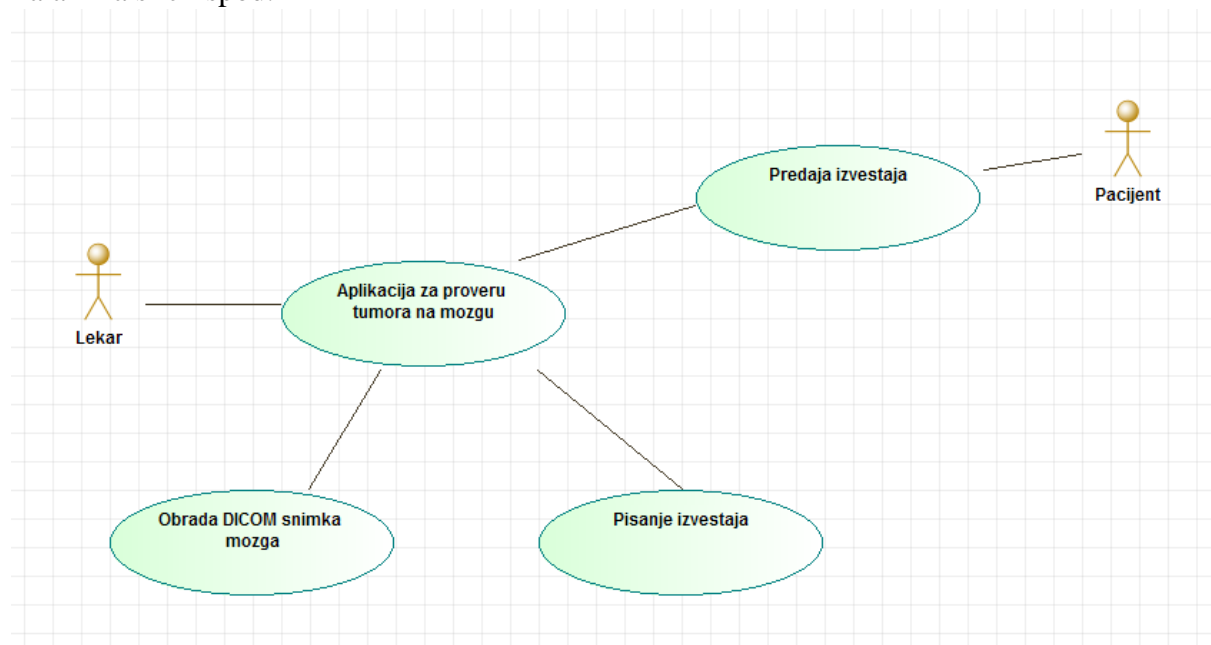
Dijagram stanja prikazuje ponašanje koje specificira sekvence stanja kroz koja prolazi i modelira ponašanje nekog entiteta ili protokol interakcije. Dijagram stanja je graf koji prikazuje automat stanja, čvorovi su stanja a grane su prelazi. Dijagram se nalazi na slici ispod.



Slika 12 -Dijagram stanja

Dijagram slučajeve korišćenja (Use Case diagram)

Dijagram slučajeve korišćenja (use-case) prikazuje skup slučajeve korišćenja i aktera. Tipično se koristi da specificira neku funkcionalnost i ponašanje nekog subjekta (npr. Projektovanog sistema). Elementi dijagrama su slučajevi korišćenja, akteri, relacije i paketi. Dijagram se nalazi na slici ispod.



Slika 13 – Dijagram slučajeve korišćenja

LITERATURA

- Sajt za biblioteku korišćenu za analiziranje čestica:
<https://imagej.nih.gov/ij/docs/guide/146-30.html>
- Moodle portal nastavnog predmeta Softverski inženjering:
<http://moodle.mfkg.rs/course/view.php?id=524>
- Korišćeni sajt za informacije o načinu kreiranja pdf-a:
<https://pdfbox.apache.org/>
- Sajt koje je korišćen za dopunjenje znanja o Javi:
<https://www.javaworld.com/article/2076075/learn-java/core-java-learn-java-from-the-ground-up.html>