Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Primož Golle

Podpora za delo z EKG odčitki in OpenEHR

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: dr. Andrej Brodnik

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Stroški zdravstvene oskrbe v sodobnem svetu strmo naraščajo in eden glavnih razlogov je tudi, na srečo, dvig kakovosti oskrbe. Izkaže se, da je od kakovosti oskrbe odvisno izboljšanje stanja posameznika. V ta namen je bila razvita slpošna platforma eOskrba. Platformo želimo nadgraditi z oskrbo kardio postoperativno oskrbo.

Pripravite knjižnico za podporo OpenEHR zapisa podatkov in se pri tem posvetite predvsem standardiziranem shranjevanju EKG zapisa ter možnosti njegove predstavitve preko portala eOskrba.

Zahvaljujem se staršem za podporo med študijem ter prof. dr. Andreju Brodniku za pomoč pri realizaciji diplomske naloge.

Kazalo

Povzetek

Abstract

1 Uvod **1**

1.1 Cilji diplomske naloge 1

1.2 Elektronska zdravstvena kartoteka 2

1.3 Elektrokardiografija 2

1.4 Računalništvo v oblaku in v megli 4

2 Tehnologije **7**

2.1 Merilnik Savvy 7

2.1.1 Opis naprave 7

2.1.2 Datoteke merilnika Savvy 10

2.1.3 Pridobivanje podatkov iz datoteke S2 10

2.1.4 Drugi merilniki EKG 15

2.2 Portal eOskrba 15

2.2.1 Tehnologije 16

2.3 OpenEHR 17

2.3.1 Uvod 17

2.3.2 Opredelitev OpenEHR 18

2.3.3 Podroben opis pojmov 18

2.3.4 Shranjevanje in dostopanje do podatkov 26

2.3.5 Primerljive rešitve 26

2.4 DICOM datoteka 28
2.4.1 Standard DICOM 28
2.4.2 Datoteka DICOM EKG 28

3 Knjižnica DicomECG **37**
3.1 Upravljanje z datoteko DICOM 38
3.1.1 Dostopanje do podatkov 38
3.1.2 Posodobitev in vnos podatkov 40
3.2 Delo z datotekami 46
3.2.1 Dostop do predlog 48
3.3 Grafični prikaz odčitka 49
3.3.1 Opis grafičnega prikaza 49
3.3.2 Opis predloge 49
3.3.3 Tvorjenje grafičnega prikaza 50
3.3.4 Dodajanje opomb 52
3.4 Delo z arhetipi 52
3.4.1 Tvorjenje datoteke ADL 54
3.4.2 Branje datoteke ADL 55

4 Zaključek **57**
4.1 Možnost izboljšav 57
4.2 Implementacija knjižnice DicomECG v sistem eOskrba 58

Literatura **78**

Seznam uporabljenih kratic

kratica	angleški opis	slovenski opis
AD	Analog Digital	pretovrnik analognih v digitane vrednosti.
ADL	Archetype definition language	jezik namenjen definiciji arhetipa
BPM	Business Process Management	tehnologija namenjena nadziranju poslovnih procesov.
cADL	Constraint Definition Language	jezik, s katerim definiramo definicijo arhetipa v datoteki ADL.
CCOW	Clinical Context Object Workgroup	specifikacija interoperabilnosti za vizualno integracijo aplikacij
CDA	Clinical Document Arhitecture	model izmenjevanju zdravstvenih dokumentov
CSS	Cascading Style Sheets	preprost slogovni jezik namenjen definiciji prezentacije spletne strani.
DCM	DICOM	DICOM datotekea, namenjena shranjevanju zdravstvenih podatkov.
DICOM	Digital Imaging and Communications in Medicine	standard namenjen shranjevanju in izmenjevanju zdravstvenih podatkov.

EKG	Electrocardiography	proces merjenja električnih impulzov srca
EZK	Electronic Health Record	elektrnoska zdravstvena kartoteka
FHIR	Fast Healthcare Interoperability Resources	standard namenjen izmenjavi elektronskih zdravstvenih datotek.
GGTS	Groovy Grails Tool Suite	razvojno okolje namenjeno razvijanju Groovy Grails aplikacij.
HL7	Health Level 7	skupek standardov namenjen izmenjavi in delu s kliničnimi in administrativnimi podatki
HTML	Hyper Text Markup Language	označevalni jezik namenjen izdelavi spletnih strani.
HTTP	Hyper Text Transfer Protocol	protokol namenjen prenosu podatkov prek spleta.
IoT	Internet of Things	prepoznavanje fizičnih predmetov na medrežju.
JAR	Java Archive	datoteka, ki predstavlja programsko knjižnico v programskem okolju Java
JSON	JavaScript Object Notation	notacija, ki omogoča strukturiran zapis podatkov
LDAP	Lightweight Directory Access Protocol	protokol za poizvedovanje in spreminjanje imeniških storitev.
mV	miliVolts	merska enota mili Volti, namenjena predstavitvi naptosti.
OAuth		protokol ki omogoča varovano avtorizacijo uporabnikov.

RM	Reference model	referenčni model
STS	Spring Tool Suite	razvojno okolje namenjeno razvijanju Spring aplikacij.
XML	Extensible Markup Language	računalniški jezik namenjen opisovanju strukturiranih podatkov pri prenosu in izmenjavi podatkov med uporabniki

Povzetek

Naslov: Podpora za delo z EKG odčitki in OpenEHR

Avtor: Primož Golle

V diplomskem delu je predstavljena knjižnica, namenjena shranjevanju in prikazovanju EKG-odčitkov. Na začetku dela so opisane osnove elektrokardiografije ter računalništva v megli. Sledi opis merilnika Savvy in portala eOskrba, na katerem bo implementirano končno delo diplomskega dela. V nadaljevanju se dotaknemo tehnologije OpenEHR in opišemo arhetip, namenjen shranjevanju EKG-odčitka. Največji del diplomskega dela je namenjen opisu in razlagi datoteke Dicom ter knjižnice DicomECG. Tu opišemo funkcionalnosti datoteke, strukturo in prikažemo osnovne primere uporabe metod, ki jih ponuja knjižnica. Na koncu predstavimo možne posodobitve sistema eOskrba in smiselne nadgradnje knjižnice v nadaljnjem razvoju.

Ključne besede: shranjevanje odčitka EKG, Savvy, EKG, Dicom, openEHR .

Abstract

Title: Support for manipulating ECG and OpenEHR

Author: Primož Golle

The diploma thesis presents a library for storing and displaying ECG readings. At the beginning, basics of electrocardiography and fog computing are explained. In the second chapter of diploma thesis are described technologies relevant to the field of work. First section presents portable ECG measurement device Savvy and its files, followed by short presentation of platform eOskrba, on which the final product will be implemented. Chapter ends with detailed description of technology OpenEHR and its archetypes, used to store clinical data. Majority of thesis is used to describe and explain structure and functionality of Dicom files and DicomECG library. In addition to the definition explanation, basic usage examples of library classes and methods are provided. To sum everything up, possible updates of platform eOskrba and practical library upgrades are listed.

Keywords: saving ECG measurements, Savvy, EKG, Dicom, openEHR.

Poglavje 1

Uvod

V zadnjih letih smo priča hitremu staranju prebivalstva v Sloveniji in svetu. Družba se bo morala prilagoditi na vseh področjih, če bo želela, da bo življenje podobno današnjemu.

Na področju zdravstva se bomo soočili s težavo dražje zdravstvene oskrbe, daljših čakalnih vrst za preglede in operativnih posegov, pomankanja prostora v bolnišnicah ter vedno bolj kronično bolne populacije zaradi nezdravega načina življenja.

Za to, da pacientom ne bi bilo treba prihajati k zdravnikom na preglede, se na trgu že pojavljajo razni merilniki življenjskih znakov. Ti pacientu omogočajo enostavnejše in samodejno pošiljanje meritev zdravnikom, ki imajo pregled nad njihovim zdravstvenim stanjem.

1.1 Cilji diplomske naloge

Cilj diplomskega dela je shraniti meritev EKG z merilnika Savvy v sistem, ki omogoča storitve e-zdravstva.

Na že obstoječi platformi eOskrba je cilj podpreti klinično pot pooperativne oskrbe srčnega bolnika. EKG-odčitke bolnika pridobimo s pomočjo merilnika Savvy, ki omogoča opazovanje bolnika tudi, ko ta ni v oskrbi bolnice. Rezultate meritev shranimo v datoteko DICOM, ki jo poleg drugih

kliničnih podatkov hranimo v elektronski zdravstveni kartoteki tehnologije OpenEHR. Da bi omogočili zgoraj opisani tok podatkov, moramo narediti knjižnico, ki bo podpirala shranjevanje Savvy odčitka v datoteko DICOM, tvorjenje elektronske zdravstvene kartoteke ter prikazovanje shranjenih EKG-odčitkov.

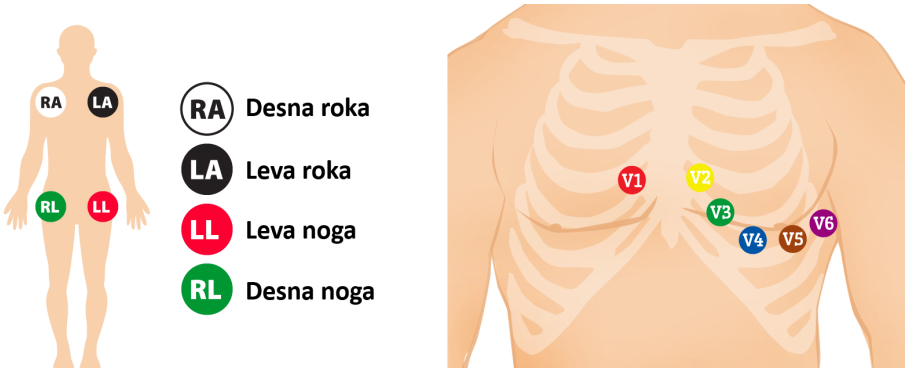
1.2 Elektronska zdravstvena kartoteka

Elektronska zdravstvena kartoteka (EHR) [7] je digitalna različica zdravstvenega kartona pacienta. Omogoča hranjenje zdravstvenih podatkov (izvidi, diagnoze, zdravila, ki jih pacient prejema, radiološke slike ...). Ti podatki so dostopni vsem pooblaščenim osebam v želenem trenutku. Centralizirano hranjene kartoteke omogoča hitrejše, učinkovitejše in bolj koordinirano delo med zdravstvenimi institucijami ter zanesljivejše diagnoze zdravnikov.

1.3 Elektrokardiografija

Elektrokardiografija (EKG) ([8, 27, 6]) je proces merjenja električnih impulzov srca v časovnem obdobju s pomočjo elektrod, ki so natančno nameščene na bolnika. Elektrode na koži zaznajo male električne spremembe, ki so rezultat elektrofizičnega vzorca srčne mišice med vsakim srčnim utripom. Elektrokardiografija je v medicini prisotna vsakodnevno prav zaradi pomembne funkcije srca v telesu. Zdravniki se za izvajanje meritev EKG odločijo zaradi mnogih zdravstvenih težav. Nekatere izmed njih so:

- bolezni srca in ožilja,
- sum na srčni napad,
- sumljiva srčna embolija,
- preskušanje hipertrofične karidiomopatije,
- perioperativno in postoperativno spremljanje srca bolnika.



Slika 1.1: Prikaz postavitev EKG elektrod. [6]

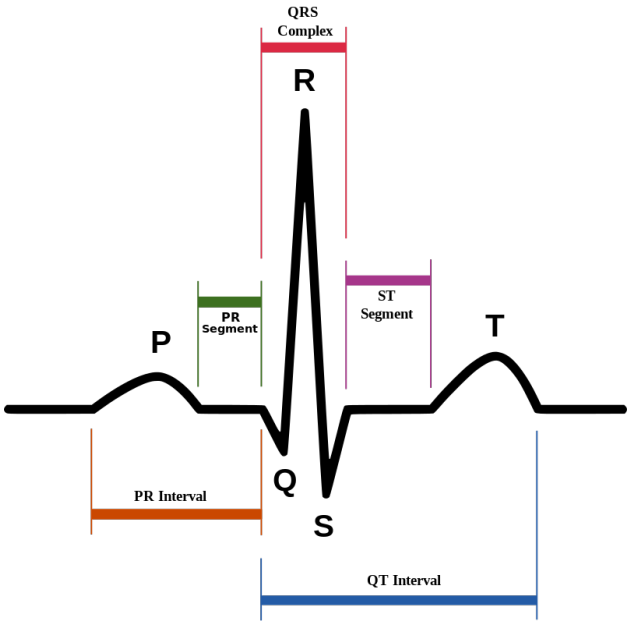
Val	Opis	Trajanje
P	Posledica širjenja impulza iz desnega v levi preddvor	80ms
QRS	Skupina zobcev Q, R in S predstavlja depolarizacijo v levi prekat	Od 80ms do 120ms
T	Repolarizacija prekatov	160ms

Tabela 1.1: Opis elektrokardiografskih valov

Pri običajnem merjenju delovanju srčne mišice se na telo pacienta pritrdi devet elektrod (glej Sliko 1.1). Šest elektrod je obsrčnih, kar pomeni, da so nameščene na sprednji in levi strani prsnega koša. Ostale tri so pritrjene na levo in desno zapestje ter na levi gleženj. Merilec za vsak srčni udarec prejme dvanajst sklopov podatkov. Sprva dobi podatke iz dipolnih odvodov (relacija med dvema elektrodama, na okončinah človeka), nato pa še iz enopolnih odvodov (podatki ene same elektrode, nameščene ob srcu pacienta).

Vzorec EKG temelji na teoriji elektromagnetike. Običajen srčni impulz je sestavljen iz treh elektrokardiografskih valov (P, QRS in T). Četrti val U se nahaja med valoma T in P, zato je običajno neviden. Opis kardiografskih valov lahko najdemo v Tabeli 1.1, grafični prikaz pa lahko razberemo s Slike 1.2.

Za prikaz meritev se uporablja elektrokardiogram. Vodoravna os prikazuje časovno odvisnost v sekundah (s), navpična pa vrednost meritev. Vrednosti meritev so zapisane v napetosti, in sicer v milivoltih (mV). Osnovna



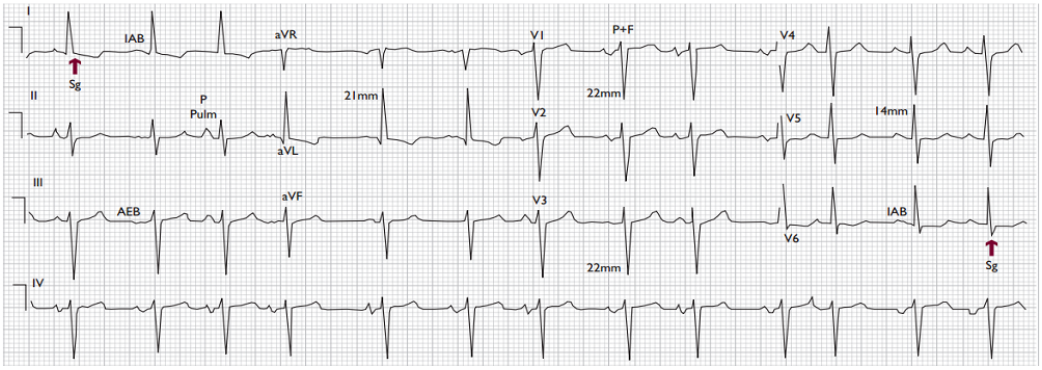
Slika 1.2: Prikaz elektrokardiografskih valov [27]

črta oz. vrednost grafa je enaka 0 mV. Poznamo jo tudi po imenu izoelektrična črta. Prikaz meritve EKG s pomočjo elektrokardiograma najdemo na Sliki 1.3.

1.4 Računalništvo v oblaku in v megli

V današnjem času si skorajda ne moremo predstavljati naprave, ki ne bi zbirala podatkov ter jih tako ali drugače uporabljala. Naj gre za predstavitev podatkov končnemu uporabniku ali za pošiljanje statističnih podatkov razvijalcem, ki jih nato uporabljajo za spremljanje stanja naprave in nadaljnji razvoj. Vsi ti podatki naprav se morajo nekje hraniti.

V zadnjih dveh letih naj bi naprave ustvarile kar 90 % vseh podatkov, s katerimi razpolaga svet [16]. Če je verjeti napovedim strokovnjakov, naj bi se število naprav, priključenih v internet, v naslednjih treh letih povečalo za štirikrat, in sicer na približno trideset milijard. Ker razvoj čipov ter hitrosti internetne povezave ne tekmuje s hitrostjo večanja števila elektronskih naprav

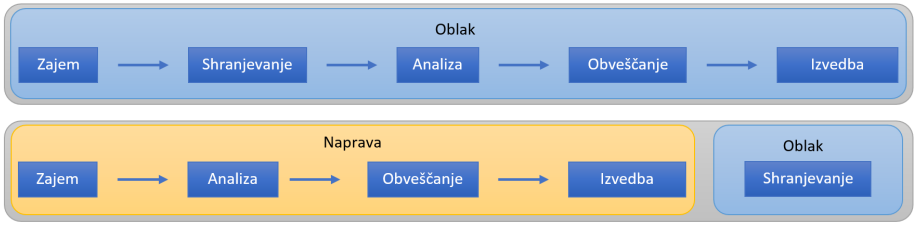


Slika 1.3: Prikaz elektrokardiograma [26]

v omrežju, je treba ubrati nov pristop shranjevanja podatkov v oblaku.

Računalništvo v oblaku deluje po naslednjem principu: procesna enota v napravi prejme podatke iz različnih senzorjev in jih posreduje v oblak. Ti podatki se shranijo in začnejo procesirati na istem strežniku. Z obdelanimi podatki nato delamo razne raziskave, obveščamo končne uporabnike, ustrezno ukrepamo in podobno. Problem nastane pri velikem številu podatkov, ki jih ne potrebujemo za trajno hrambo, počasnem procesiranju podatkov in potrebi po vedno več razpoložljivih virih.

V ta namen se je razvila arhitektura v megli [16]. Senzor v napravi pošilja podatke v krmilno enoto. Ta v realnem času spremlja in ureja podatke. Ko so podatki filtrirani in nad njimi opravljene želene operacije, se shranijo na strežnik v oblaku. S tem pridobimo porazdeljeno infrastrukturo za obdelavo in shranjevanje podatkov, večjo razširljivost in zanesljivost sistema ter znižanje stroškov. V IoT lahko najdemo računalništvo v megli. Na Sliki 1.4 vidimo razliko v arhitekturi med računalništvom v oblaku in v megli [16]. Vsi procesi pri računalništvu v oblaku se izvedejo na strani strežnika, medtem ko se pri računalništvu v megli vsi logični procesi izvedejo na strani naprave, shranjevanje podatkov pa v oblaku (na strežniku).



Slika 1.4: Primer strukture računalništva v oblaku in megli

Poglavje 2

Tehnologije

2.1 Merilnik Savvy

V tem poglavju se seznanimo z merilnikom EKG Savvy, njegovim delovanjem in pretvorbo podatkov iz datoteke S2 v obliko, ki nam omogoča nadaljnje delo. Usvojimo osnove elektrokardiografije, načine merjenja in zakaj se uporablja v medicini. Na koncu poglavja spoznamo še druge merilnike, ki jih lahko najdemo na trgu.

2.1.1 Opis naprave

Savvy [22] je medicinski pripomoček za spremljanje srčnega ritma. Merilnik Savvy je produkt razvoja Instituta Jožefa Štefana (IJS) s poslovnimi partnerji. Savvy EKG je prenosna medicinska naprava, namenjena natančnim meritvam srčnega ritma, kar omogoča odkrivanje morebitnih odstopanj od normalnega delovanja srca. Videz naprave je prikazan na Sliki 2.1.

Poleg senzorja za zaznavanje srčnega utripa naprava vsebuje še senzorje za vlažnost, zračni tlak, temperaturo, gibanje in orientacijo v prostoru. Dodatni merilci omogočajo ambientalno inteligenco, ki omogoča določanje pogojev, v katerih so meritve nastale. Skupaj z dodatnimi podatki nam Savvy omogoča zajem visokokakovostnega EKG-signala, ki omogoča identifikacijo srčnih napak in morebitne aritmije.

Slika 2.1: Merilnik Savvy EKG. [23]

Naprava je sestavljena iz merilne enote in standardnih samolepljivih elektrod. Materiali, iz katerih je naprava zgrajena, so biokompatibilni, kar zagotavlja koži prijazen občutek. CE certifikat vgrajenih materialov in celovite zasnove zagotavlja kakovosten izdelek.

Vgrajena baterija omogoča neprestano delovanje do treh dni. Zaradi izredno lahke in majhne zasnove (tehta le 21 g ter sega 10 cm v dolžino) je EKG-merilec nemoteč ter primeren za nošenje ob vsaki priložnosti. Poleg majhne oblike naprava za delovanje ne potrebuje nobenih kablov, stikal ali nastavitev, kar omogoča enostavno vzdrževanje in uporabo. Primerna je za individualno uporabo najrazličnejših ljudi. Na Inštitutu Jožef Štefan opisujejo, da je naprava namenjena [22]:

- ljudem s kroničnimi boleznimi,
- rekreativnim in profesionalnim športnikom,
- poslovnežem,
- ljudem s prebolelimi kapmi oziroma srčnimi infarkti,
- ljudem, ki si želijo informativno spremljati delovanje srca,
- ljudem in zdravnikom, ki ugotavljajo kako zdravila vplivajo na srčni ritem.

Sistem ni namenjen:

- ljudem, ki potrebujejo hospitalizacijo,
- ljudem z morebitnimi smrtno nevarnimi aritmijam,
- ljudem z alergijo na gel ali lepilo na merilnih elektrodah.

IJS je za prikaz meritev razvil mobilno aplikacijo MobECG, ki vsebuje naslednje funkcionalnosti:

- shranjevanje meritev in analize dodatnih senzorjev,
- grafični prikaz podatkov,
- osnovne analize (npr. izračun srčnega utripa),
- zahtevnejše analize (npr. iskanje in klasifikacija ekstrasistol),
- označevanje uporabnikove zaznave na določenem časovnem intervalu meritve,
- opozarjanje uporabnika na morebitne napake oz. preseganje uporabnikovih mejnih vrednosti.

Za nemoteno delovanje mobilne aplikacije, ta potrebuje sistem z naslednjimi tehnologijami:

- Bluetooth 4.0,
- operacijski sistem Android (4.0–6.0),
- vsaj 2 GB prostora.

Poleg zgoraj naštetih funkcionalnosti sistem omogoča prenos meritev v oblak. Namizna aplikacija VisECG za operacijski sistem Windows omogoča pregled in izpis poročila za medicinsko osebje. Natisnjeno poročilo lahko tako posredujemo osebnemu zdravniku in tako zagotovimo dodaten vir informacij o stanju srca.

2.1.2 Datoteke merilnika Savvy

Merilnik Savvy po končanem merjenju shrani podatke v datoteko formata S2, ki je bil namensko razvit za merilnik. Datoteka vsebuje metapodatke merilnika kot tudi podatke meritve. Podatke srčnega delovanja zazna senzor, ki shrani vrednost z desetimi biti. Če želimo razumeti vsebino datoteke, potrebujemo knjižnico, ki nam omogoča branje te datoteke.

Knjižnica LibS2Java je javanska knjižnica, ki nam omogoča uporabo datoteke S2. Uporabljamo jo lahko tako v javanskem programu kot tudi v programskem paketu MATLAB. Postopek branja podatkov je podrobneje opisan v razdelku 2.1.3.

2.1.3 Pridobivanje podatkov iz datoteke S2

Opis razreda s2

Razred S2 je namenjen pridobivanju in shranjevanju vseh podatkov, zapisanih v datoteki S2. Ta za delovanje potrebuje omenjeno knjižnico LibS2Java.

```
import si.ijs.e6.*;
```

Razred za inicializacijo potrebuje lokacijo ter ime datoteke S2, časovno obdobje, med katerim pregledujemo podatke, ter velikost objekta (vrstica 1). Po inicializaciji razreda moramo poklicati funkcijo initFile() (vrstica 2).

```
1: s2 s2 = new s2("lokacija/S2datotke", "datoteka.s2", 0, 9*24*3600e9
   (6h));
2: savvy.initFile();
```

Funkcija initFile (razložena v psevdokodi Metoda initFile) preveri, ali je inicializiran objekt S2 veljaven (vrstici 1 in 6), ter deklarira strukture, potrebne za shranjevanje podatkov:

- `strukturo MeasurementData` (namenjena shranjevanju podatkov iz datoteke, vrstica 5),

- `spremenljivka nanos1` (začetna časovna točka branja datoteke, vrstica 7),
- `spremenljivka nanos2` (končna časovna točko branja datoteke, vrstica 8).

Metoda initFile

```
1:  if !s2.error then
2:     File directory = new File(inputDirectory);
3:     measurementData = new MeasurementData(S2);
4:     loadStatus = s2.load(directory, inputFile);
5:     loadStatus.addReadLineCallback(measurementData);
6:     if loadStatus.isOK() then
7:        nanos1 = (long)(fromTimeNanos);
8:        nanos2 = Math.min((long)(fromTimeNanos + blockSizeNanos),
    (long)(toTimeNanos));
9:        collectDataTimeIntervals();
10:    else
11:       System.out.println('S2 object initialization failed');
12:    end if
13: end if
```

V vrstici 9 sledi klic funkcije collectDataTimeIntervals, ki pridobi podatke iz datoteke. Opisana je v psevdokodi Metoda collectDataTimeIntervals. Med branjem datoteke trenutno pridobljene podatke (vrstice 9, 10 in 11) shranjujemo v strukture (vrstice 14, 17 in 20), ki jih inicializiramo v konstruktorju razreda S2:

- `data` (struktura, namenjena shranjevanju vrednosti srčnega ritma),
- `timestamps` (struktura, namenjena shranjevanju časovnih vrednosti zajema),
- `counters` (struktura, namenjena shranjevanju števca zajema).

Shranjevanje podatkov se zaključi v primeru napake (vrstica 5) ali ob dosegu konca datoteke (vrstici 25 in 28). Poleg treh zgoraj naštetih struktur struktura MeasurementData vsebuje še tri skupine podatkov:

- `Comments` (struktura, namenjena shranjevanju komentarjev naprave),
- `MetaData` (struktura, namenjena shranjevanju meta podatkov naprave),
- `SpecialMessages` (struktura, namenjena shranjevanju posebnih sporočil).

Te pridobimo pri inicializaciji strukture (glej psevdokodo Metoda initFile, vrstica 5).

Metoda collectDataTimeIntervals

```
1:  while true do
2:    measurementData.clearStreamPackets();
3:    measurementData.setTimeIntervalInNanos(nanos1, nanos2);
4:    boolean everythingOk = loadStatus.readAllLinesAndProcess();
5:    if !everythingOk then
6:      System.out.println("Nekaj je narobe:"+ s2.getNotes());
7:    else
8:      if Arrays.binarySearch(measurementData.getAvailable-
    Streams(), 0) >= 0 then
9:        float[] tempData = measurementData.getStreamSamplesFor-
    Matlab(0);
10:       long[] tempTS = measurementData.getStreamTimestampsFor-
    Matlab(0);
11:       long[] tempCounters = measurementData.getStreamCounters-
    ForMatlab(0);
12:     end if
13:     for long tmp : tempData do
14:       data.add(tmp);
```

```
15:       end for
16:       for long tmp : tempTS do
17:          timestamps.add(tmp);
18:       end for
19:       for long tmp : tempCounters do
20:          counters.add(tmp);
21:       end for
22:       tempData = null;
23:       readUpToNanos = measurementData.getLastReadNano-
    Timestamp();
24:    end if
25:    if readUpToNanos < nanos2 then
26:       break;
27:    end if
28:    if nanos2 == toTimeNanos then
29:       break;
30:    end if
31:    nanos1 = nanos2;
32:    nanos2 = Math.min((long) (nanos2 + blockSizeNanos), (long)
    (toTimeNanos));
33: end while
```

Uporaba razreda s2

Če želimo pridobiti podatke datoteke S2, moramo v glavnem programu najprej inicializirati razred S2. Sledi klic funkcije initFile, s katero pridobimo podatke podane datoteke.

```
1: s2 savvy = new s2("path/to/file", šampleFile.s2", 0, 9*24*3600e9 (6h),
   0);
2: savvy.initFile();
```

Do osnovnih treh podatkov (vrednosti meritve kanala srčnega ritma, časovne oznake ter števca) dostopamo neposredno.

```
1: ArrayList<Float> data = savvy.data;
2: ArrayList<Long> timestamps = savvy.timestamps;
3: ArrayList<Long> counters = savvy.counters;
```

Do podatkov, shranjenih znotraj strukture MeasurementData, dostopamo prek naslednji metod:

- `getComments()`,
- `getMetaDataOfDevice()`,
- `getSpecialMessages()`.

Vsaka struktura vsebuje metode, ki omogočajo dostop do specifičnih informacij. Primer za pridobitev prvega sporočila naprave je predstavljen v nadaljevanju.

```
1: ArrayList<Comment> comments = savvy.getComments();
2: comments.get(0).getComment();
3: comments.get(0).getTimestamp();
```

2.1.4 Drugi merilniki EKG

Na tržišču najdemo tudi druge merilnike EKG, ki ponujajo podobno uporabniško izkušnjo kot Savvy. Vse več podjetij se zaveda problemov kroničnih bolezni ter prezasedenosti bolnišnic, zato je industrija omenjenih sistemov v porastu.

Merilnik AliveCor [2] se je na ameriškem tržišču pojavil že leta 2013. Merilnik je zasnovan za hitro in enostavno merjenje srčnega utripa ter EKG. Sistem meri podatke prek dveh senzorjev, na katere prislonimo prstne blazinice. V nekaj sekundah imamo na zaslonu mobilne naprave prikaz trenutne meritve v realnem času. Na ameriškem trgu je cena izdelka 100 $.

Qardiocore [21] je višje cenovni izdelek, ki je namenjen merjenju telesne temperature, srčnega utripa in EKG. Namenjen je tako ljudem, ki si želijo samo spremljanja srca, kot tudi športnikom, saj je zaradi načina merjenja primeren za rabo ob najrazličnejših aktivnostih. Vodoodporni, moderno oblikovani pasni merilec pošilja podatke na mobilno napravo, na kateri lahko spremljamo meritve. Imamo tudi možnost izvoza podatkov in pošiljanja meritev osebnemu zdravniku.

Tako kot Savvy oba sistema uporabljata enovodni sistem merjenja EKG.

2.2 Portal eOskrba

eOskrba je platforma, ki se razvija z namenom kliničnega ovrednotenja inovativne zdravstvene intervencije, ki za podporo zdravstvene oskrbe uporablja sodobne informacijsko-komunikacijske tehnologije.

Spletna aplikacija eOskrba uporabniku omogoča shranjevanje in izmenjavajo elektronskih zdravstvenih kartotek. Za ta namen uporablja tehnologijo openEHR. Sestavljena je iz štirih podportalov: eAstma, eDiabetes, eHujšanje in eŠport.

2.2.1 Tehnologije

Okolje

Platforma je nameščena na operacijski sistem Ubuntu 10.04 LTS. Za pravilno delovanje aplikacije je na sistemu treba zagnati še nekaj programskih komponent.

LDAP [11] je protokol, namenjen poizvedovanju in spreminjanju imeniških storitev. Imenik vsebuje predmete, ki so organizirani na logičen in hierarhični način. V sistemu eOskrba se uporablja za identifikacijo, avtentikacijo in avtorizacijo uporabnikov aplikacije. Za nadzor in upravljanje protokola je na operacijski sistem nameščen program phpLDAPadmin. Je spletna aplikacija, napisana v jeziku PHP.

Java Na sistemu je nameščen objektno usmerjen programski jezik Java 1.6.0 [19].

Apache Server Apache [3] je odprto kodni spletni strežnik. Spletni strežnik je namenjen za prikaz strani, ki služi kot vstopna točka v aplikacijo eOskrba. Za uporabo varne povezave HTTPS, je nameščen modul SSL, ki služi za nadzor certifikatov. Aplikacijski strežnik, ki ga uporabljamo za delovanje aplikacije, se imenuje Apache Tomcat.

PostgreSQL eOskrba shranjuje podatke v odprtokodno objektno-relacijsko podatkovno bazo PostgreSQL [20].

Activiti

Activiti [1] je odprtokodna javanska tehnologija, ki omogoča zagon, nadziranje in ukinitev procesov. Deluje prek standarda BPMN 2.0. Komunikacija poteka prek tehnologije REST. V aplikaciji eOskrba se uporablja za izvajanje tistih poslovnih vsebin, ki jih je mogoče predstaviti s procesi.

Razvojno okolje Za razvijanje aplikacije uporabljamo razvojni okolji: Groovy/Grails Tool Suite [24] in Spring Tool Suite [25].

2.3 OpenEHR

V tem poglavju bomo spoznali tehnologijo OpenEHR ([17, 18]). Seznanimo se z načinom delovanja tehnologije in postopkom shranjevanja elektronske zdravstvene datoteke. Omenimo tudi druge rešitve na tržišču.

2.3.1 Uvod

Z razvojem računalništva se je tako kot v drugih gospodarskih panogah tudi v zdravstvu pojavila potreba po informacijskih in komunikacijskih tehnologijah, ki bi olajšale in posodobile postopke hranjenja in obdelovanja podatkov. V ta namen so se začeli razvijati informacijski sistemi, ki lahko zdravstvene podatke shranijo v elektronske zdravstvene kartoteke.

Med razvojem so strokovnjaki naleteli na več težav:

- pomanjkljivi vnosi podatkov,
- izmerljivost,
- zastarelost sistema,
- hitro spreminjanje standardov.

Ker razvijalci programske opreme ne poznajo podrobnosti domenskih specifik na področju zdravstva, je ključnega pomena za pravilno delovanje informacijskega sistema vključitev strokovnjakov, ki podajo specifično znanje programerjem ter s tem poskrbijo za pravilno delovanje sistema.

2.3.2 Opredelitev OpenEHR

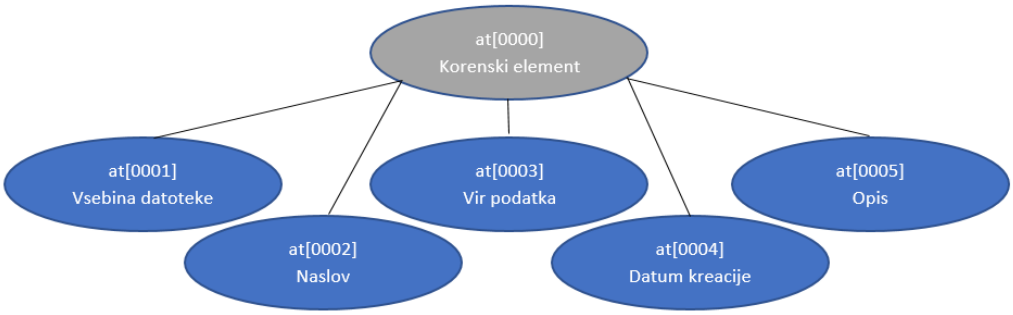
Dvonivojska ureditev sistema OpenEHR se deli na referenčni model (RM) ter na arhetipe in predloge. Skozi model predstavimo logično strukturo EHR, arhetipi in predloge so zadolženi za definicijo specifičnih podatkovnih struktur na področju zdravstva ter obrazcev za delo s podatki. Ker je vsaka meritev zdravstvenega stanja različna, potrebujemo za vsako specifično definicijo podatkovne strukture in obrazca. Orodje ADL Workbench omogoča strokovnjakom s področja zdravstvene domene tvorjenje specifičnih arhetipov ter obrazcev, ki jih nato v nadaljnjem razvoju uporabijo programerji. Ker EHR uporabljajo v različnih ustanovah, je za komunikacijo ključnega pomena izrecno določanje izrazov in kliničnih konceptov.

Primer shranjevanja meritve:
V EHR bi radi shranili meritev srčnega ritma oziroma EKG. Zdravnik opravi na pacientu meritev, ki se zapiše v temu namenjen arhetip prek obrazca. Pred shranjevanjem vrednosti se preveri, ali vnosni podatki ustrezajo zahtevanim omejitvam. Skupek shranjenih arhetipov prek predlog lahko združimo v nov arhetip, ki nam predstavlja nek zdravstveni koncept. Predloge in arhetipi so shranjeni v RM, urejene v hierarhični oz. drevesni strukturi. Drevesna struktura omogoča vnos in dostop do podatkov prek vozlišč.

2.3.3 Podroben opis pojmov

Podatkovna struktura drevo

Hierarhična podatkovna struktura drevo je sestavljena iz vozlišč in listov. Uporabljamo jo, ko želimo imeti povezano strukturo podatkov. Elementi so povezani v razmerju starš-otrok, pri katerem ima običajno oče dva otroka. Primer strukture podatkovnega drevesa je prikazan s Sliko 2.2.



Slika 2.2: Podatkovna ureditev arhetipa openEHR-EHR-CLUSTER.multimedia.v1 v obliki drevesa

Referenčni model OpenEHR

RM predstavlja prvo raven sistema. Sestavljen je iz nespreminjajočih se razredov, s katerimi določa strukturo EHR. V grobem ga delimo na domeno, vzorce in jedro, s katerimi zagotavljajo definicijo, identifikacijo, dostop do podatkov ter verzioniranje podatkov. Zaradi svoje generične strukture nam omogoča tvorjenje najrazličnejših modelov.

Arhetip

Arhetip je definicija strukture za enoten klinični koncept, ki je definiran po vzpostavljeni terminologiji. Enotna terminologija nam omogoča nedvoumno komunikacijo med uporabniki. Podatki, zapisani v drevesni strukturi, so opredeljeni široko z minimalnimi omejitvami referenčnega modela, kar omogoča večkratno implementacijo v širokem naboru kliničnih scenarijev. Za dolgo in stabilno življenjsko dobo je ključnega pomena, da so oblikovani čim bolj idealno v sodelovanju z domenskimi strokovnjaki. Podatek je zapisan v obliki jezika ADL. Datoteka arhetipa, namenjena za zapis podatkov EKG, je prikazana v Dodatku B.

Predloge

OpenEHR predloge so formalno definirani koncepti za določen arhetip v nekem kontekstu. Omogočajo nam združevanje arhetipov in tvorjenje novih skupin. Največkrat jih srečamo kot vnosne obrazce ali kot poročila. ADL-datoteka, ki predstavlja predloge, združuje oz. zapiše nov arhetip oz. predlogo z izjemami. Tako lahko za določen arhetip kakšen podatek izpustimo, v spletni vnosni obrazec zapišemo vnaprej določeno vrednost ter omejimo možnost vnosa za določeno informacijo itd. Da razumemo, kako so med seboj povezani arhetipi ter predloge v drevesni strukturi, se uporablja XML-datoteka .oet. Vsebuje podatke o identifikaciji in imenu predloge, opisa in definicije. Informacije v definiciji datoteke povedo vsak vključen arhetip ter zagotovijo informacijo o lokaciji v drevesni strukturi.

Predloga **device.oet** arhetipa openEHR-EHR-CLUSTER.multimedia.v1 je dosegljiva v razdelku DODATEK D.

ADL-datoteka

Datoteka ADL je namenjena definiranju strukture arhetipa. Sestavljena je iz treh glavnih delov:

- glava datoteke,
- definicja arhetipa,
- identifikatorji ali terminologija arhetipa.

Spodaj so prikazani deli datoteke, ki predstavlja arhetip EKG. Celotna datoteka je dosegljiva v Dodatku B.

Glava datoteke: Glava datoteke: glava datoteke, opisana v psevdokodi Glava datoteke openEHR-EHR-CLUSTER.multimedia.v1, je sestavljena iz identifikacije ter predela, v katerem so zapisane dodatne informacije arhetipa ter razvijalca. Identifikacijski predel (vrstica 2) nam pove različico programskega jezika „openEHR-EHR“, sistemski referenčni model „CLUSTER“,

razred „.multimedia“ in njegovo različico „v1“, v katerem je bila datoteka ustvarjena.

V predelih „concept“ (vrstica 3), „language“ (vrstica 5) in „description“ (vrstica 7) so zapisane informacije o avtorju arhetipa, jezik, v katerem je arhetip napisan, opis in namen uporabe ter kontaktni podatki.

Predel „description“ (vrstica 7) nam nudi informacije o avtorju ter opiše namen, uporabo ter ključne besede arhetipa. Opis je zapisan v vseh jezikih, ki so definirani v datoteki. Informacije predela se lahko uporabi za iskanje ter definiranje indeksov v imeniku standarda. Oddelek je v definiciji datoteke neobvezen, vendar se uporablja ob vsaki praktični uporabi arhetipa.

Glava datoteke `openEHR-EHR-CLUSTER.multimedia.v1`

```
1: archetype (adl_version=1.4)
2:     openEHR-EHR-CLUSTER.multimedia.v1
3: concept
4:     [at0000]                                    // Multimedijski vir
5: language
6:     original_language = <[ISO_639-1::en]>
7: description
8:     original_author = <
9:         ['name'] = <'Heather Leslie'>
10:        ['organisation'] = <"Ocean Informatics'>
11:        ['email'] = <'heather.leslie@oceaninformatics.com'>
12:        ['date'] = <'2014-10-22'>
13:    >
14:    details = <
15:        ['slo'] = <
```

```
16:         language = <[ISO_639-1::slo]>
17:         purpose = <'Namenjneno zapisovanju metapodatkov o
    podrobnostih multimedijski predstavitvi klinicnih podatkov, ki niso
    zajeti samo z uporabo vrste podatkov Multimedia.'>
18:         use = <'Uporabite za snemanje dodatnih metapodatkov o
    vec predstavnostni predstavitvi klinicnih podatkov, ki niso zajeti samo
    z uporabo vrste podatkov Multimedia.'>
19:         keywords = <'slika', 'zvok', 'tekst', 'video', 'aplikacija'>
20:         copyright = <'openEHR Foundation'>
21:       >
22:     >
23:     lifecycle_state = <'AuthorDraft'>
24:     other_contributors = <'Grahame Grieve, Health Intersections,
    Australia', 'Ian McNicoll, freshEHR, UK'>
25:     other_details = <
26:       ['current_contact'] = <'Heather Leslie, Ocean Informatics,
    heather.leslie@oceaninformatics.com'>
27:       ['MD5-CAM-1.0.1'] = <'626AFF94BCD3048F45914AD130D54262'>
28:     >
```

Definicja arhetipa: v psevdokodi Definicija arhetipa se nahaja formalna opredelitev arhetipa. Napisana je v jeziku cADL. Z omejitvami nad referenčnim modelom lahko vzpostavimo posamezne klinične koncepte. Vsi objekti, ki se nahajajo v datoteki, potrebujejo identifikator, ki je že vnaprej določen. Tako se znebimo nekonsistentnosti ter možnih nesporazumov pri delu s podatki.

Primer objekta in njegovega identifikatorja

```
1: ELEMENT[at0002]
```

Podatki arhetipa openEHR-EHR-CLUSTER.multimedia.v1 so zapisani

v elementu „CLUSTER“ (vrstica 2). Cluster struktura nam omgoča zapis večjega števila infromacij za obravnavani zapis - v našem primeru datoteko. Objekt „CLUSTER[at0000]“ (vrstica 2) vsebuje vse podatke, katere hranimo. Posamezne informacije arhetipa najdemo v posameznem „ELEMENET“ objektu, kateri poleg identifikatorja in komentarja (vrstica 4), vsebuje tudi pričakovan podatkovni tip podatka (vrstica 7).

Definicija arhetipa openEHR-EHR-CLUSTER.multimedia.v1

```
1:  definition
2:    CLUSTER[at0000] matches{                    // Multimedijski vir
3:      items cardinality matches 1..*; unordered matches {
4:        ELEMENT[at0001] occurrences matches 0..1 matches { // Multi-
    medijski vir
5:          value matches {
6:            DV_MULTIMEDIA matches {
7:              items media_type matches {[openEHR::]}
8:            }
9:          }
10:       }
11:     ELEMENT[at0002] occurrences matches 0..1 matches{     // Naslov
12:         value matches{
13:           DV_TEXT matches {*}
14:         }
15:       }
16:       ELEMENT[at0003] occurrences matches 0..1 matches{      // Vir
17:       value matches{
18:         DV_TEXT matches {*}
19:       }
20:     }
```

```
21:     ELEMENT[at0004] occurrences matches 0..1 matches{    // Datum
    kreacije
22:         value matches{
23:             DV_DATE_TIME matches {*}
24:         }
25:     }
26:     ELEMENT[at0005] occurrences matches 0..1 matches{      // Opis
27:         value matches{
28:             DV_TEXT matches {*}
29:         }
30:     }
31: }
```

V predelu datoteke „ontology“ so opisani vsi izrazi, ki jih datoteka uporablja kot identifikatorje (glej psevdokodo Identifikatorji ali terminologija arhetipa). Omogoča nam boljše razumevanje podatkovne strukture ter nudi dodatne informacije. Izrazi so opisani v vsakem jeziku, ki ga arhetip podpira (vrstica 3). Poleg imena (vrstica 6) identifikatorja (vrstica 5) nam struktura omogoča zapis opisa elementa (vrstica 7).

Zunanje ontologije omogočajo povezavo interne kode arhetipa z zunanjimi sistemi. To omogoča uporabnikom, ki uporabljajo enako terminologijo, nedvoumno razumevanje kode ter možnosti urejanja kliničnih konceptov.

Identifikatorji ali terminologija arhetipa

```
1: ontology
2:     term_definitions = <
3:         ['slo'] = <
4:             items = <
```

```
5:          ['at0000'] = <
6:            text = <'Multimedijski vir'>
7:            description = <'Podrobnosti o multimedijski predstavitvi
    klinicnih podatkov.' >
8:          ['at0001'] = <
9:            text = <'Multimedijski vir'>
10:           description = <'Multimedijska predstavitev klinicnega
    opazovanja ali iskanja.' >
11:         ['at0002'] = <
12:           text = <'Naslov'>
13:           description = <'Ime vecpredstavnostnega vira.'>
14:         ['at0003'] = <
15:           text = <'Vir'>
16:           description = <'Vir vecpredstavnostnega vira.'>
17:         ['at0004'] = <
18:           text = <'Datum tvorjenja'>
19:           description = <'Datum in cas, kdaj je bil ustvarjen zapis.' >
20:         ['at0005'] = <
21:           text = <'Opis'>
22:           description = <'Opis vecpredstavnostnega vira.' >
23:       >
24:     >
25:   >
26: >
```

2.3.4 Shranjevanje in dostopanje do podatkov

Branje podatkov

Ker so predloge in arhetipi zapisani v obliki podatkovne strukture drevesa, je dostop do podatkov enostaven. Za branje želenega podatka moramo navesti pot od korena drevesa do vozlišča, kjer se vrednost nahaja, ter identifikator.

Branje podatkov iz arhetipa je razloženo v poglavju 3.4.2.

Vnos podatkov

Proces za vnos podatkov je podoben. V predlogo arhetipa se naložijo želeni podatki ter shranijo v začasno podatkovno strukturo. Nad vnesenimi podatki se nato izvrši preverjanje, ki zagotovi, da se bodo v arhetip vnesli klinično pravilni vnosi. Po uspešni potrditvi podatkov se predloga in arhetip shranita v novo vozlišče. Novo kreirano vozlišče se shrani v drevesno strukturo ter zapiše v obliko ADL ali XML.

Podroben opis kreiranja datoteke ter vnos podatkov sta razložena v poglavju 3.4.1.

2.3.5 Primerljive rešitve

MLHIM [15] je odprtokodni standard, namenjen izvajanju semantične interoperabilnosti v zdravstvu. Temelji na večnivojskem modeliranju. Tako kot OpenEHR MLHIM vsebuje referenčni model ter CCD-omejitve, ki nam oblikujejo domenske modele. Za razliko od OpenEHR ima enostavnejšo specifikacijo ter bolj preprost jezik, ki opisuje domeno. MLHIM uporablja za zapis arhetipa kar jezik XML, kar poenostavi razumevanje in pohitri razvijanje, saj za razvoj ni potrebno učenje novega specifičnega jezika.

HL7 [14] je skupek standardov, namenjen prenosu kliničnih in administrativnih podatkov med različnimi aplikacijami, ki se pojavljajo v zdravstvenih institucijah. Prožni standardi, smernice in metodologije, ki so določene

s strani HL7, omogočajo izmenjavo in obdelavo podatkov na enostaven in nedvoumen način. Glavni standardi, ki so vključeni v tehnologijo, so:

- Version 3 Messaging Standard – določa interoperabilnost za zdravstvene transakcije,
- standard CDA – model, namenjen izmenjevanju zdravstvenih dokumentov,
- standard CCD – specifikacija izmenjevanja zdravstvenih kartotek na podlagi standarda CDA za trg Združenih držav Amerike,
- standard SPL – publikacija informacij o zdravilu,
- standard CCOD – specifikacija interoperabilnosti za vizualno integracijo aplikacij.

HL7 tako podatke prenaša v dokumentu standarda CDA. Ker ne podpira arhetipov in referenčnih modelov, se podatki hranijo kar v XML-datotekah.

FHIR [13] je standard, ki vsebuje lastnosti standardov HL7 v2, HL7 v3 in CDA, hkrati pa se osredotoča na enostavno implementacijo in izkoriščanje najnovejših standardov. Rešitve standarda so sestavljene iz modularnih komponent, imenovanih *„resources“*. Ponuja veliko izboljšav trenutnih standardov:

- veliko število knjižnic,
- osnova za spletne standarde XML, JSON, HTTP, OAuth,
- RESTful arhitektura,
- obsežna in razumljivo napisana dokumentacija.

Glavni izziv EHR-standardov – spremenljivost procesov, standard FHIR rešuje z ogrodjem, ki dopolnjuje že napisane *„resource“*. Omogoča tudi človeško enostavno in razumljivo predstavitev podatkov z uporabo jezika HTML.

Standard FHIR bi lahko vključili znotraj OpenEHR, vendar je ta še v procesu testiranja.

2.4 DICOM datoteka

Za shranjevanje informacij v zdravstvu se uporabljajo različni formati datotek. Najbolj poznani so formati DICOM [5], HL7 [14] in ISO [10]. Za shranjevanje odčitkov smo se odločili uporabiti format DICOM, saj je v praksi najbolj uporabljen, dolgotrajen razvoj standarda pa omogoča dodajanje podatkov in razširitev datotek.

2.4.1 Standard DICOM

Standard DICOM zagotavlja potrebna orodja za diagnostično natančne predstavitve in procesiranje medicinskih podatkov ter komunikacijo in izmenjavo podatkov med uporabniki.

2.4.2 Datoteka DICOM EKG

DICOM svoje podatke zapisuje v datoteki s končnico dcm. Ker lahko z DICOM standardom shranimo najrazličnejše medicinske podatke (od meritev ultrazvoka, radioloških slik, odčitkov EKG ...), mora biti struktura datoteke spremenljiva. Dinamičnost podatkovne zbirke omogoča uporaba vnaprej pripravljenih DICOM podatkovnih objektov in elementov. Skupek gradnikov, zapisanih v datoteki, predstavlja zapis kliničnega podatka.

Elementi so v hierarhični strukturi razvrščeni po datoteki. Vsak element je sestavljen iz devetih atributov. Za delo s standardom je treba poznati naslednje:

- **oznaka elementa (*tag*)** je enolična osem-mestna številka, sestavljena iz dveh sklopov. Prvi sklop štirih številk predstavlja skupino v katero spada element, drugi sklop pa identifikacijsko številko elementa,

- **VR (*value representation*)** je dvomestna oznaka, ki definira tip shranjenega podatka,
- **tip (*type*)** je numerična vrednost, ki definira, ali je element opcijski ali obvezen,
- **VM (*value multiplicity*)** je numerična vrednost, ki predstavlja število podatkov v elementu,
- **ključna beseda (*keyword*)** opisuje oznako elementa.

Za pretvorbo med formatoma dcm in xml razvijalci uporabljajo različne knjižnice. Ker se programerji knjižnic odločijo, katere atribute elementa bodo prikazali v datoteki xml, se zgodi, da je izpis človeku berljive datoteke med knjižnicami drugačen [4].

Predloga DICOM EKG

Predloga za shranjevanje EKG-odčitkov je sestavljena iz treh glavnih delov [12].

Metapodatki datoteke, odčitka ter pacienta V prvem delu se nahajajo metapodatki datoteke, meritve ter pacienta. Spodaj je primer zapisa ID-odčitka (vrstica 1), imena merilne naprave (vrstica 2) ter podatkov o pacientu (vrstica 3).

```
1: <attr tag='00080050' vr='SH' keyword='AccessionNumber'>
   20161004085623</attr>
2: <attr tag='00080070' vr='LO' keyword='Manufacturer'> IJS - Savvy
   </attr>
3: <attr tag='00100000' vr='PN' keyword='PatientName'> Primoz Golle
   </attr>
```

Drugi del datoteke je opisan v elementu *„WaveformAnnotationSequence“*. Tip *„SQ“* ali dolgo *„Sequence“* pomeni, da element vsebuje enega ali več vnosov. Znotraj elementa so shranjeni zapisi opomb, ki se razdeljeni v dve skupini.

```
1: <attr tag='004008EA' vr='SQ' keyword=' WaveformAnnotation-
   Sequence '>
```

Prva skupina opomb je namenjena shranjevanju obdelanih informacij srčnega ritma odčitka. V psevdokodi Podatki meritve srčnega odčitka je prikazana struktura za shranjevanje podatka *„PR intervala“*. Podatek je definiran z dvema elementoma tip *„SQ“* (definirana v vrsticah 3 in 11) ter tremi elementi (vrstice 19, 20 in 21). Element *„WaveformAnnotationSequence“* (vrstica 3) predstavlja enoto, v kateri je opomba izražena, *„ConceptNameCodeSequence“* (vrstica 11) pa opisuje, kaj celoten objekt predstavlja (definiran v vrstici 1). Elementi *„ReferencedWaveformChannels“* (vrstica 19), *„AnnotationGroubNumber“* (vrstica 20) in *„NumericValue“* (vrstica 21) definirajo, na kateri kanal se podatki nanašajo, skupino anotacije ter opis opombe.

Podatki meritve srčnega odčitka

```
1: <attr tag='004008EA' vr='SQ' keyword='WaveformAnnotation-
   Sequence'>
2:   <item>
3:     <attr tag='004008EA' vr='SQ' keyword='Measurement-
   UnitsCodeSequence'>
4:       <item>
5:         <attr tag='00080100' vr='SH' keyword='CodeValue'>
   ms </attr>
6:         <attr tag='00080102' vr='SH' keyword='Coding-
   SchemeDesignator'> UCUM </attr>
```

```
7:            <attr tag='00080103' vr='SH' keyword='CodingScheme-
   Version'> 1.4 </attr>
8:            <attr tag='00080104' vr='LO' keyword='CodeMeaning'>
   milliseconds </attr>
9:          </item>
10:       </attr>
11:       <attr tag='0040A043' vr='SQ' keyword='ConceptName-
   CodeSequence'>
12:         <item>
13:           <attr tag='00080100' vr='SH' keyword='CodeValue'>
   5.13.5-7 </attr>
14:           <attr tag='00080102' vr='SH' keyword='CodingScheme-
   Designator'> SCPECG </attr>
15:           <attr tag='00080103' vr='SH' keyword='CodingScheme-
   Version'> 1.3 </attr>
16:           <attr tag='00080104' vr='LO' keyword='CodeMeaning'>
   PR Interval </attr>
17:         </item>
18:       </attr>
19:       <attr tag='0040A0B0' vr='US' keyword='Referenced-
   WaveformChannels'> 1 </attr>
20:       <attr tag='0040A180' vr='US' keyword='AnnotationGroup-
   Number'> 1 </attr>
21:       <attr tag='0040A30A' vr='DS' keyword='NumericValue'>
   - </attr>
22:     </item>
23: </attr>
```

Opombe V drugi skupini anotacij (glej psevdokodo Opombe zdravnika) so shranjene opombe, ki so bile vnesene prek grafičnega vmesnika s strani

zdravnika. Podrobneje opisano dodajanje opomb najdemo v razdelku 3.3.4.

V gradniku *„ConceptNameCodeSequence“* (vrstica 3) je pri vseh opombah tega tipa opis *„GraphAnnotation“* (vrstica 5). Sledijo še trije elementi, s katerimi so shranjene: skupina opombe (vrstica 8), točka v grafu odčitka, za katero je opomba zabeležena (vrstica 9), ter samo besedilo opombe (vrstica 10).

Opombe zdravnika

```
1: <attr tag='004008EA' vr='SQ' keyword='WaveformAnnotation-
   Sequence'>
2:    <item>
3:       <attr tag='0040A043' vr='SQ' keyword='ConceptNameCode-
   Sequence'>
4:       <item>
5:          <attr tag='00080104' vr='LO' keyword='CodeMeaning'>
   GraphAnnotation </attr>
6:       </item>
7:    </attr>
8:    <attr tag='0040A180' vr='US' keyword='AnnotationGroupNumber
   '> 2 </attr>
9:    <attr tag='0040A132' vr='UL' keyword='ReferencedSamplePositions
   '> 231 </attr>
10:   <attr tag='00700006' vr='ST' keyword='UnformattedTextValue '>
   Premalo kisika v krvi. </attr>
11: </item>
12: </attr>
```

Vrednosti EKG-odčitka V tretjem delu predloge se nahaja struktura, namenjena shranjevanju vrednosti odčitka naprave. Ker je elementov, ki definirajo zapis krivulje, preveč, bomo razložili le tiste, ki so pomembni za pravilno shranjevanje in prikaz podatkov.

V začetku strukture je v vrstici 1 z elementom *„NumberofWaveformChannels“* definirano, koliko kanalni merilnik je avtor meritve. To je pomembno, saj moramo definirati vsak kanal posebej. Pove nam tudi, kako moramo zapisati ter brati vrednosti elementa *„WaveformData“*. Element *„NumberofWaveformChannels“* (vrstica 2) definira, koliko posnetkov je zabeležil vsak kanal, *„SamplingFrequency“* (vrstica 33) pa frekvenco beleženja. Iz obeh elementov lahko izračunamo čas opravljanja odčitka.

```
1: <attr tag=' 003A005 ' vr=' US ' keyword='NumberofWaveform-
   Channels'> 1 </attr>
2: <attr tag=' 003A0010 ' vr=' UL ' keyword='NumberofWaveform-
   Samples'> 1008 </attr>
3: <attr tag=' 003A001A ' vr=' DS ' keyword='SamplingFrequency'>
   125 </attr>
```

Sledi definicija kanalov. Struktura je podobna tisti za shranjevanje opomb, le da so elementi različni. Predstavljena je v psevdokodi Shranjevanje EKG odčitka. Ker standard DICOM shranjuje vrednosti odčitka v binarni obliki, moramo nekje definirati, v kakšni obliki so shranjeni izmerjeni podatki. Element *„WaveformBitsStored“* (vrstica 20) definira, s koliko biti je shranjena vrednost posameznega odčitka srčnega delovanja v enoti, definirani znotraj elementa *„ChannelSensitivityUnitsSequence“* (vrstica 11). V našem primeru shranjujemo vrednosti v 16-bitni obliki, kar pomeni, da uporabljamo podatkovno strukturo short. Ker s shortom ni mogoče zapisati decimalnih števil, moramo imeti nekje definiran faktor množenja, s katerim dobimo vrednost meritve v enoti mV. Ta je zapisan v elementu *„ChannelSensitivity“* (vrstica 19).

Shranjevanje EKG odčitka

```
1: <attr tag='003A0200' vr='SQ' keyword='ChannelDefinitionSequence'>
2: <item>
3:   <attr tag='003A0208' vr='SQ' keyword='ChannelSourceSequence'>
4:   <item>
5:     <attr tag='00080100' vr='SH' keyword='CodeValue' >
   5.6.3-9-1</attr>
6:     <attr tag='00080102' vr='SH' keyword='CodingSchemeDesi-
   gnator' >SCPECG</attr>
7:     <attr tag=' 00080103' vr='SH' keyword='CodingScheme-
   Versions'>1.3</attr>
8:     <attr tag=' 00080104' vr='LO' keyword='CodeMeaning'>
   Kanal 1</attr>
9:   </item>
10:  </attr>
11:  <attr tag=' 003A0211 ' vr=' SQ' keyword=' ChannelSensitivityUnits-
   Sequence' >
12:  <item>
13:    <attr tag='00080100' vr='SH' keyword='CodeValue'> uV
   </attr>
14:    <attr tag='00080102' vr='SH' keyword='CodingSchemeDesign-
   ators'>UCUM</attr>
15:    <attr tag='00080103' vr='SH' keyword='CodingSchemeVersion'
   >1.4</attr>
16:    <attr tag='00080104' vr='LO' keyword='CodeMeaning'>
   microvolt</attr>
17:    </item>
```

```
18:    </attr>
19:    <attr tag='003A0210' vr='DS' keyword='ChannelSensitivity'>
       0.00594367</attr>
20:    <attr tag='003A021A' vr='US' keyword= WaveformBitsStored'>
       16</attr>
21:    .
22:    .
23:    .
24:    </item>
25: </attr>
```

Predloga je zaključena z elemetom *„WaveformData“*, ki hrani podatke senzorjev (glej psevdokodi Gradnik WaveformData1 in Gradnik WaveformData 2). Kot že zgoraj zapisano, so podatki v elementu zapisani v 16-bitni obliki. V našem primeru so podatki shranjeni v obliki:

```
[16b_vrednost_1], [16b_vrednost_2], ..., [16b_vrednost_n]
```

V primeru, ko želimo shraniti oz. brati odčitek večkanalnega merilnika, so podatki v naslednji obliki:

```
[kanal_1_vrednost_1], [kanal_2_vrednost_1], ..., [kanal_n_vrednost_1],
[kanal_1_vrednost_2], [kanal_2_vrednost_2], ..., [kanal_n_vrednost_2],
[kanal_1_vrednost_m], [kanal_2_vrednost_m], ..., [kanal_n_vrednost_m]
```

Knjižnice, ki omogočajo pretvorbo datoteke DMC v format xml, lahko te podatke izpišejo na dva načina. Prvi način prikazuje vrednosti, zapisane v obliki podatkovnega razreda Integer.

Gradnik WaveformData 1

```
1: <attr tag='54001010' VR='OW' keyword='WaveformData'>
   521/519 ... 540</attr>
```

Drugi način podatkov ne prikaže, temveč prikaže lokacijo v datoteki, kjer se ti nahajajo, ter velikost.

Gradnik WaveformData 2

```
1: <attr tag=' 54001010 ' VR=' OW ' keyword=' WaveformData '>
2:    <BulkData uri=' file:/filePath/ecgFile.dmc?offset=21764&amp;
   length=1008'/>
3: </attr>
```

Celotna predloga **template.dmc** je dosegljiva v razdelku DODATEK A.

Poglavje 3

Knjižnica DicomECG

Javanska knjižnica DicomECG je namenjena shranjevanju in prikazovanju EKG-odčitkov merilnika Savvy v formatu Dicom.

Knjižnica za delo uporablja več razredov:

- `DicomFile`: dostop do podatkov datoteke Dicom,
- `DicomFileUpdate`: posodobitev datotekeDicom,
- `FileManager`: delo z datotekami,
- `DEVICEArcheSrv`: delo z arhetipi,
- `GetTemplates`: dostop do predlog.

Knjižnica DicomECG vsebuje tudi knjižnice zunanjih razvijalcev. Za obdelavo podatkov, povezanih z datoteko DICOM, se uporablja knjižnico dcm4che2-2.0.29 [9]. Dostop ter obdelavo podatkov datoteke S2 omogoča knjižnica S2Savvy.

Opis knjižnice je razdeljen na več razdelkov. V prvem razdelku 3.1 se seznanimo, kako je datoteka Dicom zgrajena. Naučimo se tvoriti in posodabljati nove Dicom objekte in elemente v sodelovanju s knjižnico S2Savvy. Prikazano je tudi, kako lahko dostopamo do informacij, zapisanih v datoteki.

Naslednji razdelek 3.2 razloži, kako iz ustvarjenega Dicom objekta tvorimo novo datoteko dcm ter jo po želji pretvorimo v notacijo xml. Razložena je tudi uporaba predlog. Grafični prikaz odčitka je pojasnjen v razdelku 3.3. Opisan je postopek, kako iz datoteke dcm tvorimo predstavitveno datoteko odčitka. Opis knjižnice zaključimo z razdelkom 3.4, ki nam pojasni, kako kreirati in brati datoteke ADL.

3.1 Upravljanje z datoteko DICOM

Podatke, pridobljene iz merilnika Savvy, s pomočjo dveh razredov `DicomFile` in `DicomFileUpdate` shranimo v datoteko Dicom. Poleg tvorjenja in posodabljanja datoteke nam razreda omogočata tudi branje podatkov iz že prej definiranih datotek dcm.

3.1.1 Dostopanje do podatkov

Razred, ki razvijalcu omogoča dostop do podatkov datoteke dcm, se imenuje `DicomFile`. Za delovanje potrebuje uvoz nekaterih razredov knjižnice dcm4che2, ki je namenjena za delo z datotekami DICOM.

```
1: import org.dcm4che2.data.DicomObject
2: import org.dcm4che2.data.DicomElement
3: import org.dcm4che2.data.Tag
4: import org.dcm4che2.data.VR
5: import org.dcm4che2.data.DicomInputStream
```

Razred ima na voljo dva konstruktorja, prek katerih lahko inicializiramo parametre razreda (opisana v psevdokodi Konstruktor razreda dicomFile). Prvi (vrstica 6) sprejme kot argument lokacijo datoteke dcm, s katero želimo delati. Drugi (vrstica 14) prejme kot parameter objekt razreda `DicomFile`. Tega uporabimo, če želimo kreirati novo datoteko dcm s pomočjo predloge. Predloge so podrobneje opisane v razdelku 3.2.

Konstruktor razreda dicomFile

```
1: public class dicomFile {
2:     private File inputFile;
3:     String filePath;
4:     DicomObject dicom;
5:
6:     public dicomFile(String inputFile) throws IOException{
7:         this.inputFile = new File(inputFile);
8:         this.filePath = inputFile;
9:         DicomInputStream dis = new DicomInputStream(this.inputFile);
10:        this.dicom = dis.readDicomObject();
11:        dis.close();
12:    }
13:
14:    public dicomFile(dicom dicomObj) {
15:        this.inputFile = dicomObj.inputFile;
16:        this.filePath = dicomObj.filePath;
17:        this.dicom = dicomObj.dicom;
18:    }
19: }
```

Dicom datoteka je sestavljena iz objektov, ki jih sestavljajo elementi. Informacije zapisa pridobimo iz gradnikov datoteke, do katerih dostopamo z naslednjima metodama:

- `getDicomObject(String searchField, DicomObject parentObject)`,
- `getDicomElement(String searchField, DicomObject parentObject)`.

Metodi od uporabnika zahtevata poznavanje strukture datoteke dcm, saj za delovanje potrebujeta ime iskanega gradnika „String searchField“ ter

objekt „DicomObject parentObject“, v katerem se ta nahaja. V primeru, ko razvijalec programske opreme ni seznanjen s strukturo datoteke dcm, je na voljo nekaj pomožnih metod.

V primeru, ko želimo pridobiti podatke posameznega kanala odčitka EKG-meritve, uporabimo metodo `getWaveFormChannelElement(int chanel, String searchField)`. Kot argumente prejme kanal meritve „int chanel“ ter ime elementa „String searchField“, v katerem se nahaja želeni podatek. Metoda `getValueFormWaveAnnotationObjectByName(DicomObject obj, String category)` vrne vrednost elementa „String category“, ki se nahaja znotraj podanega objekta „DicomObject obj“. Funkcija `getWaveFormAnnotationObjectByName(String element)` vrne objekt gradnika *„WaveformAnnotationSequence“* z oznako „String element“.

3.1.2 Posodobitev in vnos podatkov

Razred, ki razvijalcu omogoča posodabljanje datoteke dcm, se imenuje `DicomFileUpdate`. Za delovanje potrebuje uvoz nekaterih razredov knjižnice, namenjene za delo z dcm datotekami dcm4che2.

```
1: import org.dcm4che2.data.BasicDicomObject
2: import org.dcm4che2.data.DicomObject
3: import org.dcm4che2.data.Tag
4: import org.dcm4che2.data.VR
```

Konstruktor razreda (glej psevdokodo Konstruktor razreda DicomFileUpdate) zahteva vnos objekta razreda Dicom (vrstica 6). Za prepis podatkov, shranjenih v podanem objektu Dicom, moramo definirati strukturo knjižnice dcm3che2 DicomObject (vrstica 7), ki nam omogoča urejanje gradnikov datoteke. Atribut „lokacijaDatoteke“ razreda hrani pot datoteke na pomnilniku (vrstica 8).

Konstruktor razreda DicomFileUpdate

```
1: public class DicomFileUpdate{
2:     DicomObject dicomObj;
3:     String lokacijaDatoteke;
4:     Dicom dicom;
5:
6:     public DicomFileUpdate(Dicom el){
7:         this.dicomObj = el.dicom;
8:         this.lokacijaDatoteke = el.lokacijaDatoteke;
9:         this.dicom = el;
10:    }
11: }
```

Če želimo nove podatke shraniti v datoteko dcm, jih moramo zapisati v strukturo DicomObject. Metoda, ki nam omogoča ustvarjanje teh struktur, je zapisana v psevdokodi Metoda createNewObject. Metoda prejme kot argumente (vrstica 1): ime objekta („String searchField“), podatek, ki ga želimo shraniti („String value“), podatkovni tip objekta („VR vrValue“) ter objekt, ki bo starš tvorjenemu gradniku datoteke („DicomObject parentObject“). Metoda na začetku procesa preveri, ali je starševski objekt podan (vrstica 1). V primeru, ko starševski objekt ni definiran, bo postal novi objekt samostojni gradnik. V nadaljevanju pridobimo številčno vrednost imena objekta (vrstica 5), ki jo poleg podatkovnega tipa in podatka zapišemo v novi objekt (vrstica 7). Metoda se zaključi v vrstici 8 z dodajanjem tvorjenega objekta staršu.

Metoda createNewObject

```
1: createNewObject(String searchField, String value, VR vrValue, Dicom-
   Object parentObject) {
2:     if(parentObject == null)
```

```
3:      parentObject = dicomFile;
4:
5:    int searchFieldTag = Tag.toTag(searchField);
6:    DicomObject newObject = new BasicDicomObject();
7:    newObject.putString(searchFieldTag, vrValue, value);
8:    parentObject.add(newObject.get(searchFieldTag));
9: }
```

Če želimo uporabljati zgornjo funkcijo, moramo poznati strukturo datoteke dcm ter imena in tipe objektov. V primeru, da razvijalec ni poznavalec datotek dcm, lahko uporabi pomožne metode. Psevdokoda Metoda setPatientName predstavlja pomožno funkcijo, namenjeno vnosu imena pacienta.

Metoda setPatientName

```
1: public void setPatientName(String name) {
2:    createNewObject("PatientName", name, VR.PN, null);   // VR.PN =
   Podatkovni tip VR.PersonName
3: }
```

Dodajanje opomb oziroma točkovnih opisov EKG-krivulje poteka prek metode `createAnnotation(int position, String description)`, opisane v psevdokodi Metoda createAnnotation. Metoda prejme kot vhodna argumenta opis („String description“) ter vrednost („int position“), na kateri se ta nahaja (vrstica 1).

Metoda v vrstici 3 preveri pozicijo nove opombe. Vrednost pozicije primerja s številom vrednosti EKG-odčitka ter jo v primeru nepravilnosti nastavi na zadnjo vrednost odčitka (vrstica 4). Skozi proces shranjevanja opomb v Dicom objekt se kreirajo trije objekti (vrstice 5, 6 in 7). Objekt „parent“ (vrstica 5) hrani vse gradnike, v katerih so zapisani podatki. Objekt „codeMeaning“ (vrstica 7) predstavlja vrsto podatka, ki ga dodajamo v datoteko dcm (vrstica 9). Gradnik „object“ (vrstica 6) vsebuje elemente, ki

hranijo: opis (vrstica 11), tip anotacije (vrstica 12), pozicijo anotacije (vrstica 13) in vrsto opombe (vrstica 14). Med vrsticami 15 in 19 objekt „parent“ pridobi podatke drugih dveh gradnikov. Metoda se zaključi v vrstici 20, ko tvorjeni objekt zapišemo na ustrezno mesto v Dicom objekt datoteke.

Metoda createAnnotation

```
1: public void createAnnotation(int position, String description){
2:    int getNrSamples = Integer.parseInt(
   dicom.getElementValue(dicom.getNumberOfSample()));
3:    if(position> getNrSamples)
4:       Integer.toString(position) = getNrSamples-1;
5:    DicomObject parent = new BasicDicomObject();
6:    DicomObject object = new BasicDicomObject();
7:    DicomObject codeMeaning = new BasicDicomObject();
8:    object.putSequence(Tag.toTag(ČonceptNameCodeSequence"));
9:    codeMeaning.putString(Tag.toTag(ČodeMeaning"), VR.SH,
   "GraphAnnotation");
10:   object.get(Tag.toTag(ČonceptNameCodeSequence") ).addDicom-
   Object( codeMeaning );
11:   object.putString(Tag.toTag("UnformattedTextValue"), VR.ST,
   description);
12:   object.putString(Tag.toTag("AnnotationGroupNumber"), VR.US,
   "2");
13:   object.putString(Tag.toTag("ReferencedSamplePositions"),
   VR.UL, position);
14:   object.putString(Tag.toTag("TemporalRangeType"), VR.CS,
   "POINT");
15:   parent.add(object.get(Tag.toTag("UnformattedTextValue")));
16:   parent.add(object.get(Tag.toTag("AnnotationGroupNumber")));
17:   parent.add(object.get(Tag.toTag("ReferencedSamplePositions")));
18:   parent.add(object.get(Tag.toTag("TemporalRangeType")));
```

```
19:    parent.add(object.get(Tag.toTag(ČonceptNameCodeSequence")));
20:    dicom.dicom.get( Tag.toTag("WaveformAnnotationSequence") ).
   addDicomObject( parent );
21: }
```

Vnos ali posodobitev podatkov krivulje EKG zapisa dodajamo prek metode `updateECGRawData`, prikazane v psevdokodi Metoda updateECGRawData. Kot je razvidno v 1. vrstici, prejme funkcija datum („String StudyDate“) in čas nastanka odčitka meritve („String StudyTime“) ter vrednosti odčitka „ArrayList<Float> values“. Podatki krivulje izhajajo iz merilnika Savvy, ki analogne podatke senzorja pretvori v digitalno obliko s pomočjo 10-bitnega pretvornika. Če želimo zapisati vrednosti meritve v datoteko dcm, moramo odčitke pretvoriti v binarno obliko, saj Dicom objekt, namenjen shranjevanju teh podatkov, omogoča shranjevanje le 16 bitnih podatkov. „Float“ vrednosti moramo zato pretvoriti v podatkovno obliko „short“, saj ta shrani podatke na pomnilnik s 16 biti (vrstica 5). Začasno spremenljivko nato shranimo v binarno obliko (vrstici 6 in 8). Pri tem moramo paziti, kakšen tip binarnega zapisa je definiran v datoteki dcm. Podatki krivulje so v našem primeru zapisani v obliki tankega konca (najmanj pomemben bajt zapišemo na najmanjši naslov v pomnilniku). Ko dobimo podatke krivulje v binarni obliki, jih zapišemo v začasni objekt „newObject“. Poleg podatkov krivulje (vrstica 14) moramo shraniti tudi število vrednosti odčitka (vrstica 15), frekvenco zajema podatkov naprave, s kolikimi biti je zapisana vrednost (8 ali 16), število kanalov odčitka ter obliko zapisa (tanki ali debeli konec). Ker za zapis EKG-odčitka uporabljamo predlogo datoteke dcm prilagojeno merilniku Savvy, nam razen vrednosti krivulje ter števila odčitkov ni treba definirati drugih podatkov. Posodobimo tudi identifikacijsko številko datoteke, ki je sestavljena iz datuma in časa zajema odčitka (vrstica 18) ter datuma in časa zajema (vrstici 19 in 20).

Metoda updateECGRawData

```
1: public void updateECGRawData(ArrayList<Float> values, String
   StudyDate, String StudyTime){
2:    byte[] biteValues = new byte[values.size()*2];
3:    int byte_i = 0;
4:    for (Float doubleVal : values) {
5:       short tmp_short_value = (short) (Math.round(doubleVal));
6:       biteValues[byte_i] = (byte)(tmp_short_value & 0xff);
7:       byte_i ++;
8:       biteValues[byte_i] = (byte)(tmp_short_value » 8 & 0xff);
9:       byte_i ++;
10:   }
11:
12:   DicomObject object = dicom.getWaveformSequenceObject();
13:   DicomObject newObject = new BasicDicomObject();
14:   newObject.putBytes(Tag.toTag("WaveformData"),VR.OW,
   biteValues);
15:   newObject.putInt(Tag.toTag("NumberOfWaveformSamples"),
   VR.UL,values.size());
16:   object.add(newObject.get(Tag.toTag("WaveformData")));
17:   object.add(newObject.get(Tag.toTag("NumberOfWaveform-
   Samples")));
18:   SetId(StudyDate.replaceAll(-",)+StudyTime.replaceAll(":",));
19:   SetPatientStudyDate(StudyDate.replaceAll(-",));
20:   SetPatientStudyTime(StudyTime.replaceAll(":",));
21: }
```

Ko smo vnesli vse želene podatke v Dicom objekt, uporabimo funkcijo `updateFile()`, ki posodobi datoteko dcm (glej psevdokodo Metoda updateFile).

Metoda updateFile

```
1: public void updateFile(String dicomFilename, String FilePath) {
2:     FileConverter converter = new FileConverter();
3:     converter.createDCMfile(dicomFilename, FilePath);
4: }
```

3.2 Delo z datotekami

Po zaključenem vnosu podatkov odčitka v Dicom podatkovno strukturo sledi tvorjenje datoteke. Razred `FileManager` omogoča zapis podatkov v datoteko dcm. Metode razreda poleg tvorjenja novih datotek omogočajo tvorjenje novih objektov, shranjevanje novih podatkov ter pretvorbo med formati. Razred za pravilno delovanje potrebuje uvoz naslednjih razredov knjižnice dcm4che2:

```
1: import org.dcm4che2.data.DicomObject
2: import org.dcm4che2.io.DicomOutputStream
3: import org.dcm4che2.tool.dcm2xml.Dcm2Xml
4: import org.dcm4che2.tool.xml2dcm.Xml2Dcm
```

Psevdokoda Metoda createXMLfromDCM prikazuje metodo `createXMLfromDCM`, ki omogoča pretvorbo datoteke Dicom v človeško berljiv format XML.

Metoda createXMLfromDCM

```
1: public void createXMLfromDCM(String inputFile, String exportFile){
2:     Dcm2Xml converter = new Dcm2Xml();
3:     converter.convert(new File(inputFile), new File(exportFile));
4: }
```

Psevdokoda Metoda createXMLfromDCM prikazuje metodo `createDCMfromXML`, ki omogoča pretvorbo datoteke iz formata xml v obliko dcm.

Metoda createXMLfromDCM

```
1: public void createDCMfromXML(String inputFile, String exportFile){
2:     Xml2Dcm.main(new String[] {'-x', inputFile, '-o', exportFile});
3: }
```

Metoda `createDCMFile`, predstavljena v psevdokodi Metoda createDCMFile, zapiše podatke, shranjene v strukturi DicomObject, v datoteko dcm na lokacijo v pomnilniku, ki ja podana kot drugi argument funkcije.

Metoda createDCMFile

```
1: public void createDCMfile(DicomObject obj, String exportPath) {
2:     File f = new File(exportPath);
3:     FileOutputStream fos = new FileOutputStream(f);
4:     BufferedOutputStream bos = new BufferedOutputStream(fos);
5:     DicomOutputStream dos = new DicomOutputStream(bos);
6:     dos.close();
7: }
```

Metoda `createDCMFileFromTemplate`, predstavljena v psevdokodi Metoda createDCMFileFromTemplate, ustvari datoteko dcm (vrstica 4) iz predloge, ki jo prejme iz funkcije `getDCMTemplate()` razreda `GetTemplates` (vrstica 3). Metodo uporabimo, ko želimo pridobiti osnovno datoteko dcm, namenjeno zapisu EKG-odčitka. Razred `GetTemplates` je podrobneje opisan v razdelku 3.2.1.

Metoda createDCMFileFromTemplate

```
1: public void createDCMfromTemplate(String exportPath){
2:     getTemplates templates = new getTemplates();
3:     dicom obj = new dicom(templates.getDCMTemplate().getPath());
4:     createDCMfile(obj.dicom, exportPath);
5: }
```

Metoda `createDCMObjectFromTemplate`, predstavljena v psevdokodi Metoda createDCMObjectFromTemplate, ustvari Dicom objekt iz Dicom predloge (vrstica 3). Tvorjen objekt uporabljamo pri delu z Dicom objekti in elementi. Metodo uporabimo, ko želimo pridobiti osnovno strukturiran Dicom objekt celotne datoteke, namenjen shranjevanju EKG-odčitka. Razred `GetTemplates` je podrobneje opisan v razdelku 3.2.1.

Metoda createDCMObjectFromTemplate

```
1: public dicom createDCMObjectFromTemplate(){
2:     getTemplates templates = new getTemplates();
3:     return new dicom(templates.getDCMTemplate().getPath());
4: }
```

3.2.1 Dostop do predlog

Razred `GetTemplates` uporabniku omogoča dostop do datotek, ki so shranjene v knjižnici DicomECG. Datoteki omogočata preprostejše razvijanje, saj vsebujeta osnovo datoteke dcm za hranjenje EKG-odčitka ter predlogo za grafični prikaz meritve. V razredu najdemo dve metodi: `getDCM-Template()` in `getVisualizatorTemplate()`. Obe metodi vračata spremenljivko razreda `URL`, prek katere lahko dostopamo do predloge.

Metodo `getDCMTemplate()` uporabljamo pri zapisovanju EKG-odčitkov v datoteko dcm. Delo s predlogo razvijalcu olajša delo, saj ni treba ročno generirati vseh potrebnih Dicom objektov, temveč moramo samo zamenjati vrednosti elementov datoteke. Predloga je datoteka tipa dcm.

Metodo `getVisualizatorTemplate()` uporabljamo v metodi `createECG-Visualization()`, ki tvori datoteko z grafičnim prikazom shranjenih podatkov EKG-odčitka. Predloga je podrobneje opisana v razdelku 3.3.2.

3.3 Grafični prikaz odčitka

Ko so podatki meritve uspešno shranjeni v datoteki Dicom, jih lahko s pomočjo knjižnice tudi prikažemo.

Za prikaz podatkov meritve potrebujemo datoteko dcm. Trenutno knjižnica omogoča prikaz krivulje srčnega ritma enega kanala, opomb ter prikaz osnovnih podatkov meritve EKG.

3.3.1 Opis grafičnega prikaza

Grafični prikaz meritve EKG, viden na Sliki 3.1, prikazuje podatke v dveh območjih. Prvi (na sliki označen z oranžno barvo) je namenjen prikazu podatkov pacienta in meritve, medtem ko drugi predel (označen z modro barvo) vsebuje predstavitev krivulje odčitka EKG ter anotacij. Vrednosti meritve delovanja srca so predstavljene v enoti mV. Raznorazni točkovni opisi grafa (na sliki označeni z zeleno barvo) so prikazani z navpično rdečo črto, nad njo pa se nahaja besedilo, ki opisuje, kaj prikazuje opomba na grafu.

3.3.2 Opis predloge

Pri tvorjenju grafičnega prikaza odčitka si pomagamo z datoteko ECGVisualization_template.html. Ta nam omogoča enostavno izrisovanje odčitka. Elemente, namenjene predstavitvi podatkov, napolnimo s podatki iz datoteke. Postopek je podrobneje opisan v razdelku 3.3.3.

HTML-datoteka je sestavljena iz treh glavnih delov. HTML-del predloge definira elemente spletne strani in njihovo postavitev. Za predstavitev elementov skrbi notacija CSS, ki se nahaja v glavi HTML-definicije. Skripte, ki poskrbijo za izris grafa, opomb ter prikaz vmesnika za dodajanje opisov, so napisane v obliki JavaScript knjižnice jQuery. Vtičnik, s katerim izrisujemo graf, se imenuje Chart.js.

V Dodatku C se nahaja celotna programska koda datoteke ECGVisualization_template.html.

Slika 3.1: Grafični prikaz EKG odčitka.

3.3.3 Tvorjenje grafičnega prikaza

Metoda `createECGVisualization(String path)`, prikazana v psevdokodi Metoda createECGVisualization, omogoča tvorjenje grafičnega prikaza. Kot argument prejme lokacijo, ki določa, kje se bo nahajal novo tvorjeni prikaz. Postopek se začne z naložitvijo predloge v predpomnilnik v obliki podatkovne strukture String (od 2 do 4 vrstice).

Metoda `getDataForVisualization()` (vrstica 5) pridobi podatke iz datoteke dcm s pomočjo metod, opisanih v razdelku 3.1.1. Podatki, ki jih uporabimo pri kreiranju grafičnega prikaza, so:

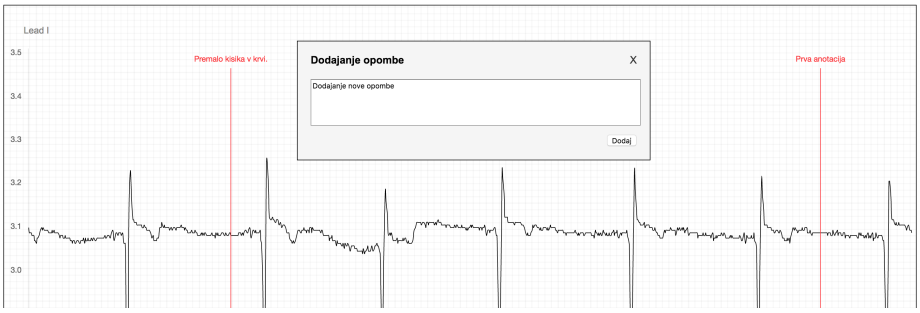
- frekvenca zajema meritve in število odčitkov,
- vrednosti EKG-krivulje,
- opombe,
- podatki pacienta,

- obdelani podatki meritve.

V primeru, ko iskane informacije ni zapisane v datoteki, se uporabi simbol „-“. Vrednosti se nato zapišejo v datoteko tako, da zamenjamo imena spremenljivk z vrednostjo iz slovarja „values“. Psevdokoda prikazuje zapis podatkov med vrsticama 6 in 14. V 16. vrstici se datoteka zapiše na pomnilnik.

Metoda createECGVisualuzation

```
1: public void createECGVisualuzation(String path){
2:    getTemplates templates = new getTemplates();
3:    File template = new File(templates.getVisualizatorTemplate().
   getPath());
4:    String htmlString = FileUtils.readFileToString(template);
5:    Map<String, String> values = getDataForVisualization();
6:    htmlString = htmlString.replace("$leadneame",
   values.get("leadneame"));
7:    .
8:    .
9:    .
10:   htmlString = htmlString.replace("$MeasurementTime",
   values.get("MeasurementTime"));
11:   htmlString = htmlString.replace("$frequency",
   values.get("frequency"));
12:   htmlString = htmlString.replace("$ECGRawdata",
   values.get("ECGRawdata"));
13:   htmlString = htmlString.replace("$ECGlabels",
   values.get("ECGlabels"));
14:   htmlString = htmlString.replace("$annotations",
   values.get("annotations"));
15:   File newHtmlFile = new File(locationOfVisulazation);
16:   FileUtils.writeStringToFile(newHtmlFile, htmlString);
   }
```



Slika 3.2: Obrazec za dodajanje opomb.

3.3.4 Dodajanje opomb

Dodajanje opomb je omogočeno prek enostavnega uporabniškega vmesnika, prikazanega na Sliki 3.2. S klikom na točkovno vrednost v grafu se odpre obrazec, v katerega vpišemo opombo. Gumb »Dodaj« je vezan na funkcijo v predlogi, ki preveri vnosno besedilo ter pokliče metodo `createAnnotaition(int position, String description)` razreda `dicomUpdateFile`.

Ker knjižnica še ni implementirana v okolje eOskrba, funkcionalnost v celoti še ne deluje. Ob implementaciji knjižnice v platformo bo treba prilagoditi predlogo tako, da bo iz Javascript funkcije možen klic na metodo v kontrolnik, ki bo povezovala klik na gumb »Dodaj« ter metodo v knjižnici `createAnnotaition`.

3.4 Delo z arhetipi

Za shranjevanje meritev v format, primeren za hranjenje podatkov v elektronski zdravstveni kartoteki, uporabimo razred `DEVICEArcheSrv`. Ta razširja razrede knjižnice platforme eOskrba, ki že ima implementirano funkcionalnost dela z arhetipi. Po končanem procesu pridobimo datoteko tipa adl ali xml, ki je primerna za hranjenje v OpenEHR. Razreda, ki omogočata delo z arhetipi, sta `ArchetypeManager` in `ArchetypeService`. Razreda skupaj razvijalcu omogočata naslednje funkcionalnosti:

- dodajanje arhetipa,
- dodajanje predloge arhetipu,
- razčlenitev datotek ADL,
- razčlenitev datotek OET,
- validacijo podatkov,
- zapis podatkov v XML-notacijo,
- branje podatkov iz XML-notacije.

Razred `DEVICEArcheSrv` razširja razred `ArchetypeService`. V psevdokodi Konstruktor razreda DEVICEArcheSrv je prikazan konstruktor. Pri incializaciji se pokliče razširjena metoda `loadAdlAndTemplateFiles()` (vrstica 3), ki določi objektu arhetip (vrstica 5) ter ustrezno predlogo (vrstica 6). Za vsak novi arhetip, ki ga želimo hraniti, moramo ponoviti postopek v tem razdelku.

Konstruktor razreda DEVICEArcheSrv

```
1: public class DEVICEArcheSrv extends ArchetypeService {
2:   @Override
3:   public boolean loadAdlAndTemplateFiles() {
4:     try {
5:       archetypeManager.addArchetype("openEHR-
   EHR.OBSERVATION.ecg.v0.adl");
6:       archetypeManager.setTemplate("ECG.oet");
7:     } catch (Exception e) {
8:       e.printStackTrace();
9:       return false;
```

```
10:         }
11:         return true;
12:     }
13: }
```

Ob generiranju novega arhetipa moramo posodobiti tudi JAR archetype-api-1.2. Dodati mu moramo novi ADL- ter OET-datoteki. To storimo v ukazni vrstici z ukazom:

```
1: jar uf archetype-api-1.2.jar /archetypes/ADLdatoteka.adl
2: jar uf archetype-api-1.2.jar /archetypes/OETdatoteka.oet
```

3.4.1 Tvorjenje datoteke ADL

Če želimo tvoriti novo datoteko ADL, moramo najprej incilizirati razred, ki je zadolžen za tvorjenje določenega tipa arhetipa. Postopek je opisan v razdelku 3.4. V našem primeru je to `DEVICEArcheSrv`, saj želimo ustvariti arhetip, ki bo shranjeval odčitek EKG.

Primer najdemo v psevdokodi Tvorjenje datoteke ADL. Struktura ArhceDataPath (vrstica 3) predstavlja pot v drevesni strukturi arhetipa, kjer želimo shraniti določen podatek. Vse poti so shranjene kot ključi v slovarju „HashMap<ArcheDataPath, Object>“. Drugi parameter slovarja shranjuje vrednost vozlišča (vrstica 2). Po končanem polnjenju drevesne struktutre arhetipa pokličemo metodi `initInput()` (vrstica 5) ter `setAndValidataData(vnosniPodatki)` (vrstica 6), ki vneseta slovar prek predloge v arhetip. Preverita tudi, ali so vsi podatki prisotni ter skladni z zahtevanimi omejitvami. V primeru, da so podatki napačni, funkcija `setAndValidataData` vrne napako.

Po uspešnem tvorjenju arhetipa lahko pridobimo referenčni objekt (vrstica 7) ali tvorimo XML datoteko z metodo `saveRmObjectToXML(path)` (vrstica 8).

Tvorjenje datoteke ADL

```
1: DEVICEArcheSrv arhetip = new DEVICEArcheSrv();
2: Map<ArcheDataPath, Object> vnosniPodatki = new HashMap
   <ArcheDataPath, Object>();
3: ArcheDataPath komentar = new ArcheDataPath
   ("/data[at0001]/events[at0002]/data[at0003]/items[at0064]/value");
4: testInput.put(komentar, "Komentar");
5: arhetip.initInput();
6: arhetip.setAndValidataData(vnosniPodatki);
7: arhetip.getRMObject();
8: arhetip.saveRmObjectToXML("izhodnaDatoteka.xml");
```

3.4.2 Branje datoteke ADL

Branje arhetipa poteka na zelo podoben način kot vnos. Po inicializaciji razreda preverimo, ali je referenčni model že definiran. Če ni, moramo vnesti podatke. Spodaj so navedene metode, ki jih uporabljamo za dostop do shranjenih informacij:

- `getRMObject()` (vrne referenčni model arhetipa),
- `getInputsMap()` (vrne struktutro v kateri so zapisane poti ter vrednosti podatkov),
- `getValue(String potDoVozlisca, String vrednostVozlisca)` vrne vrednost posameznega vozlišča.

Spodaj je prikazan primer, kako dostopamo do podatka „at0064“ (komentar), ki smo ga vnesli v psevdokodi Tvorjenje datoteke ADL v 4. vrstici.

```
1: DEVICEArcheSrv arhetip = new DEVICEArcheSrv();
2: arhetip.getValue("/data/events/data/items", "at0064");
```

Poglavje 4

Zaključek

Cilj diplomskega dela je bil pripraviti knjižnico, ki omogoča shranjevanje podatkov in vizualizacijo meritev EKG-odčitka. Končni izdelek diplomskega dela je knjižnica, ki omogoča branje podatkov iz datoteke S2, shranjevanje pridobljenih informacij v datoteki DCM ter ADL, branje in predstavitev podatkov datoteke DCM.

4.1 Možnost izboljšav

Nadgradnja knjižnice DicomECG

Knjižnica DicomECG ima še veliko možnosti za nadgradnjo na področju shranjevanja, izrisa ter obdelovanja podatkov meritev. Smiselno bo posodobiti knjižnico, da bo omogočala shranjevanje EKG-podatkov dvanajstih kanalnih merilnih naprav. Pri prikazovanju bi lahko posodobili knjižnico tako, da bi omogočala določanje in prikaz opomb v določenem časovnem obdobju, nudila izbiro izrisanih podatkov ter shranjevanje prikaza v formatu PDF. Nad podatki odčitkov bi knjižnica lahko vršila dodatne analize. Tako bi lahko razvijalec določil časovna območja posameznih valov ter zaznal nepravilnosti v srčnem ritmu pacienta.

4.2 Implementacija knjižnice DicomECG v sistem eOskrba

V nadgrajeni platformi bo dodana knjižnica DicomECG, ki bo omogočala podporo klinične poti postoperativne oskrbe po kardiološkem posegu.

DODATEK A

Primer predloge Dicom EKG datoteke

```
<?xml version="1.0" encoding="UTF-8"?><dicom>
<attr tag="00020000" vr="UL" len="4">176</attr>
<attr tag="00020001" vr="OB" len="2">00\01</attr>
<attr tag="00020002" vr="UI" len="30">1.2.840.10008.5.1.4.1.1.9.1.1</attr>
<attr tag="00020003" vr="UI" len="44">
    1.3.6.1.4.1.20029.40.20130125105919.5407.1.1</attr>
<attr tag="00020010" vr="UI" len="20">1.2.840.10008.1.2.1</attr>
<attr tag="00020012" vr="UI" len="18">1.3.76.13.1.1.1.1</attr>
<attr tag="00020013" vr="SH" len="10">AISDWAM40</attr>
<attr tag="00080005" vr="CS" len="10">ISO_IR 100</attr>
<attr tag="00080012" vr="DA" len="8">20130125</attr>
<attr tag="00080013" vr="TM" len="6">095427</attr>
<attr tag="00080016" vr="UI" len="30">1.2.840.10008.5.1.4.1.1.9.1.1</attr>
<attr tag="00080018" vr="UI" len="44">
    1.3.6.1.4.1.20029.40.20130125105919.5407.1.1</attr>
<attr tag="00080020" vr="DA" len="8">20060114</attr>
<attr tag="00080023" vr="DA" len="8">20130125</attr>
<attr tag="0008002A" vr="DT" len="14">20130125105919</attr>
<attr tag="00080030" vr="TM" len="6">105919</attr>
<attr tag="00080033" vr="TM" len="6">105919</attr>
<attr tag="00080050" vr="SH" len="14">03028041970546</attr>
<attr tag="00080060" vr="CS" len="4">ECG</attr>
<attr tag="00080070" vr="LO" len="12">IJS - Savvy</attr>
<attr tag="00080090" vr="PN" len="4">2721</attr>
<attr tag="00081010" vr="SH" len="4">1,0</attr>
<attr tag="00081030" vr="LO" len="4">ECG</attr>
<attr tag="00081060" vr="PN" len="0"/>
<attr tag="00081070" vr="PN" len="0"/>
<attr tag="00100010" vr="PN" len="6">Primoz</attr>
<attr tag="00100020" vr="LO" len="6">642341</attr>
<attr tag="00100030" vr="DA" len="8">19941202</attr>
<attr tag="00100040" vr="CS" len="2">M</attr>
<attr tag="00101000" vr="LO" len="0"/>
```

```
<attr tag="00101010" vr="AS" len="4">022Y</attr>
<attr tag="00101020" vr="DS" len="0"/>
<attr tag="00101030" vr="DS" len="0"/>
<attr tag="00101040" vr="LO" len="0"/>
<attr tag="00181000" vr="LO" len="0"/>
<attr tag="00181020" vr="LO" len="6">0.0.0</attr>
<attr tag="0020000D" vr="UI" len="42">
    1.3.76.13.65829.2.20130125082826.1072139.2</attr>
<attr tag="0020000E" vr="UI" len="42">
    1.3.6.1.4.1.20029.40.20130125105919.5407.1</attr>
<attr tag="00200010" vr="SH" len="2">1</attr>
<attr tag="00200011" vr="IS" len="0"/>
<attr tag="00200013" vr="IS" len="2">1</attr>
<attr tag="00200060" vr="CS" len="0"/>
<attr tag="00321030" vr="LO" len="0"/>
<attr tag="00321032" vr="PN" len="4">2721</attr>
<attr tag="00380010" vr="LO" len="8">13002689</attr>
<attr tag="00380300" vr="LO" len="2">52</attr>
<attr tag="00380400" vr="LO" len="0"/>
<attr tag="00384000" vr="LT" len="0"/>
<attr tag="00400555" vr="SQ" len="-1">
<item off="976" len="-1">
<attr tag="0040A040" vr="CS" len="4">CODE</attr>
<attr tag="0040A043" vr="SQ" len="-1">
<item off="1008" len="-1">
<attr tag="00080100" vr="SH" len="10">5.4.5-33-1</attr>
<attr tag="00080102" vr="SH" len="6">SCPECG</attr>
<attr tag="00080103" vr="SH" len="4">1.3</attr>
<attr tag="00080104" vr="LO" len="20">Electrode Placement</attr>
</item>
</attr>
</item>
</attr>
<attr tag="00401002" vr="LO" len="0"/>
<attr tag="0040B020" vr="SQ" len="-1">
<item off="1140" len="-1">
<attr tag="0040A0B0" vr="US" len="4">1\0</attr>
<attr tag="0040A180" vr="US" len="2">0</attr>
<attr tag="00700006" vr="ST" len="4">ECG</attr>
</item>
<item off="1190" len="-1">
<attr tag="004008EA" vr="SQ" len="-1">
<item off="1210" len="-1">
<attr tag="00080100" vr="SH" len="2">ms</attr>
<attr tag="00080102" vr="SH" len="4">UCUM</attr>
<attr tag="00080103" vr="SH" len="4">1.4</attr>
<attr tag="00080104" vr="LO" len="12">milliseconds</attr>
</item>
</attr>
<attr tag="0040A043" vr="SQ" len="-1">
```

```
<item off="1300" len="-1">
<attr tag="00080100" vr="SH" len="8">5.13.5-7</attr>
<attr tag="00080102" vr="SH" len="6">SCPECG</attr>
<attr tag="00080103" vr="SH" len="4">1.3</attr>
<attr tag="00080104" vr="LO" len="12">PR Interval</attr>
</item>
</attr>
<attr tag="0040A0B0" vr="US" len="4">1\0</attr>
<attr tag="0040A180" vr="US" len="2">1</attr>
<attr tag="0040A30A" vr="DS" len="2">-</attr>
</item>
<item off="1426" len="-1">
<attr tag="004008EA" vr="SQ" len="-1">
<item off="1446" len="-1">
<attr tag="00080100" vr="SH" len="2">ms</attr>
<attr tag="00080102" vr="SH" len="4">UCUM</attr>
<attr tag="00080103" vr="SH" len="4">1.4</attr>
<attr tag="00080104" vr="LO" len="12">milliseconds</attr>
</item>
</attr>
<attr tag="0040A043" vr="SQ" len="-1">
<item off="1536" len="-1">
<attr tag="00080100" vr="SH" len="8">5.13.5-9</attr>
<attr tag="00080102" vr="SH" len="6">SCPECG</attr>
<attr tag="00080103" vr="SH" len="4">1.3</attr>
<attr tag="00080104" vr="LO" len="12">QRS Duration</attr>
</item>
</attr>
<attr tag="0040A0B0" vr="US" len="4">1\0</attr>
<attr tag="0040A180" vr="US" len="2">1</attr>
<attr tag="0040A30A" vr="DS" len="2">-</attr>
</item>
<item off="1662" len="-1">
<attr tag="004008EA" vr="SQ" len="-1">
<item off="1682" len="-1">
<attr tag="00080100" vr="SH" len="2">ms</attr>
<attr tag="00080102" vr="SH" len="4">UCUM</attr>
<attr tag="00080103" vr="SH" len="4">1.4</attr>
<attr tag="00080104" vr="LO" len="12">milliseconds</attr>
</item>
</attr>
<attr tag="0040A043" vr="SQ" len="-1">
<item off="1772" len="-1">
<attr tag="00080100" vr="SH" len="10">5.13.5-11</attr>
<attr tag="00080102" vr="SH" len="6">SCPECG</attr>
<attr tag="00080103" vr="SH" len="4">1.3</attr>
<attr tag="00080104" vr="LO" len="12">QT Interval</attr>
</item>
</attr>
<attr tag="0040A0B0" vr="US" len="4">1\0</attr>
```

```
<attr tag="0040A180" vr="US" len="2">1</attr>
<attr tag="0040A30A" vr="DS" len="2">-</attr>
</item>
<item off="1900" len="-1">
<attr tag="004008EA" vr="SQ" len="-1">
<item off="1920" len="-1">
<attr tag="00080100" vr="SH" len="2">ms</attr>
<attr tag="00080102" vr="SH" len="4">UCUM</attr>
<attr tag="00080103" vr="SH" len="4">1.4</attr>
<attr tag="00080104" vr="LO" len="12">milliseconds</attr>
</item>
</attr>
<attr tag="0040A043" vr="SQ" len="-1">
<item off="2010" len="-1">
<attr tag="00080100" vr="SH" len="10">5.10.2.5-5</attr>
<attr tag="00080102" vr="SH" len="6">SCPECG</attr>
<attr tag="00080103" vr="SH" len="4">1.3</attr>
<attr tag="00080104" vr="LO" len="12">QTc Interval</attr>
</item>
</attr>
<attr tag="0040A0B0" vr="US" len="4">1\0</attr>
<attr tag="0040A180" vr="US" len="2">1</attr>
<attr tag="0040A30A" vr="DS" len="2">-</attr>
</item>
<item off="2138" len="-1">
<attr tag="004008EA" vr="SQ" len="-1">
<item off="2158" len="-1">
<attr tag="00080100" vr="SH" len="4">deg</attr>
<attr tag="00080102" vr="SH" len="4">UCUM</attr>
<attr tag="00080103" vr="SH" len="4">1.4</attr>
<attr tag="00080104" vr="LO" len="8">degrees</attr>
</item>
</attr>
<attr tag="0040A043" vr="SQ" len="-1">
<item off="2246" len="-1">
<attr tag="00080100" vr="SH" len="10">5.10.3-11</attr>
<attr tag="00080102" vr="SH" len="6">SCPECG</attr>
<attr tag="00080103" vr="SH" len="4">1.3</attr>
<attr tag="00080104" vr="LO" len="6">P Axis</attr>
</item>
</attr>
<attr tag="0040A0B0" vr="US" len="4">1\0</attr>
<attr tag="0040A180" vr="US" len="2">1</attr>
<attr tag="0040A30A" vr="DS" len="2">-</attr>
</item>
<item off="2368" len="-1">
<attr tag="004008EA" vr="SQ" len="-1">
<item off="2388" len="-1">
<attr tag="00080100" vr="SH" len="4">deg</attr>
<attr tag="00080102" vr="SH" len="4">UCUM</attr>
```

```
<attr tag="00080103" vr="SH" len="4">1.4</attr>
<attr tag="00080104" vr="LO" len="8">degrees</attr>
</item>
</attr>
<attr tag="0040A043" vr="SQ" len="-1">
<item off="2476" len="-1">
<attr tag="00080100" vr="SH" len="10">5.10.3-13</attr>
<attr tag="00080102" vr="SH" len="6">SCPECG</attr>
<attr tag="00080103" vr="SH" len="4">1.3</attr>
<attr tag="00080104" vr="LO" len="8">QRS Axis</attr>
</item>
</attr>
<attr tag="0040A0B0" vr="US" len="4">1\0</attr>
<attr tag="0040A180" vr="US" len="2">1</attr>
<attr tag="0040A30A" vr="DS" len="2">-</attr>
</item>
<item off="2600" len="-1">
<attr tag="004008EA" vr="SQ" len="-1">
<item off="2620" len="-1">
<attr tag="00080100" vr="SH" len="4">deg</attr>
<attr tag="00080102" vr="SH" len="4">UCUM</attr>
<attr tag="00080103" vr="SH" len="4">1.4</attr>
<attr tag="00080104" vr="LO" len="8">degrees</attr>
</item>
</attr>
<attr tag="0040A043" vr="SQ" len="-1">
<item off="2708" len="-1">
<attr tag="00080100" vr="SH" len="10">5.10.3-15</attr>
<attr tag="00080102" vr="SH" len="6">SCPECG</attr>
<attr tag="00080103" vr="SH" len="4">1.3</attr>
<attr tag="00080104" vr="LO" len="6">T Axis</attr>
</item>
</attr>
<attr tag="0040A0B0" vr="US" len="4">1\0</attr>
<attr tag="0040A180" vr="US" len="2">1</attr>
<attr tag="0040A30A" vr="DS" len="2">-</attr>
</item>
<item off="2830" len="-1">
<attr tag="0040A043" vr="SQ" len="-1">
<item off="2850" len="-1">
<attr tag="00080104" vr="SH" len="16">GraphAnnotation</attr>
</item>
</attr>
<attr tag="0040A132" vr="UL" len="4">903</attr>
<attr tag="0040A180" vr="US" len="2">2</attr>
<attr tag="00700006" vr="ST" len="14">Prva anotacija</attr>
</item>
<item off="2950" len="-1">
<attr tag="0040A043" vr="SQ" len="-1">
<item off="2970" len="-1">
```

```
<attr tag="00080104" vr="SH" len="16">GraphAnnotation</attr>
</item>
</attr>
<attr tag="0040A132" vr="UL" len="4">231</attr>
<attr tag="0040A180" vr="US" len="2">2</attr>
<attr tag="00700006" vr="ST" len="22">Premalo kisika v krvi.</attr>
</item>
</attr>
<attr tag="54000100" vr="SQ" len="-1">
<item off="3098" len="-1">
<attr tag="00181068" vr="DS" len="2">0</attr>
<attr tag="00181069" vr="DS" len="2">0</attr>
<attr tag="003A0004" vr="CS" len="8">ORIGINAL</attr>
<attr tag="003A0005" vr="US" len="2">1</attr>
<attr tag="003A0010" vr="UL" len="4">1008</attr>
<attr tag="003A001A" vr="DS" len="4">125</attr>
<attr tag="003A0020" vr="SH" len="4">ECG</attr>
<attr tag="003A0200" vr="SQ" len="-1">
<item off="3200" len="-1">
<attr tag="003A0208" vr="SQ" len="-1">
<item off="3220" len="-1">
<attr tag="00080100" vr="SH" len="10">5.6.3-9-1</attr>
<attr tag="00080102" vr="SH" len="6">SCPECG</attr>
<attr tag="00080103" vr="SH" len="4">1.3</attr>
<attr tag="00080104" vr="LO" len="6">Lead I</attr>
</item>
</attr>
<attr tag="003A0210" vr="DS" len="10">0.00594367</attr>
<attr tag="003A0211" vr="SQ" len="-1">
<item off="3332" len="-1">
<attr tag="00080100" vr="SH" len="2">uV</attr>
<attr tag="00080102" vr="SH" len="4">UCUM</attr>
<attr tag="00080103" vr="SH" len="4">1.4</attr>
<attr tag="00080104" vr="LO" len="10">microvolt</attr>
</item>
</attr>
<attr tag="003A0212" vr="DS" len="2">1</attr>
<attr tag="003A0213" vr="DS" len="2">0</attr>
<attr tag="003A0215" vr="DS" len="2">0</attr>
<attr tag="003A021A" vr="US" len="2">16</attr>
<attr tag="003A0220" vr="DS" len="6">0.050</attr>
<attr tag="003A0221" vr="DS" len="4">300</attr>
<attr tag="003A0222" vr="DS" len="2">0</attr>
</item>
</attr>
<attr tag="54001004" vr="US" len="2">16</attr>
<attr tag="54001006" vr="CS" len="2">SS</attr>
<attr tag="54001010" vr="OW" len="2016">
    521\519\519\519\518\518\518\516\516\515\517\517\518\519\520\521\521...
</attr>
```

```
</item>
</attr>
<attr tag="70011131" vr="CS" len="0"/>
<attr tag="70011132" vr="CS" len="0"/>
<attr tag="70011153" vr="AE" len="6">DW_AM</attr>
</dicom>
```

DODATEK B

Primer arhetipa openEHR-EHR-CLUSTER.multimedia.v1

```
archetype (adl_version=1.4)
        openEHR-EHR-CLUSTER.multimedia.v1

concept
        [at0000]        -- Multimedijski vir
language
        original_language = <[ISO_639-1::en]>
description
        original_author = <
                ["name"] = <"Heather Leslie">
                ["organisation"] = <"Ocean Informatics">
                ["email"] = <"heather.leslie@oceaninformatics.com">
                ["date"] = <"2014-10-22">
        >
        details = <
                ["en"] = <
                        language = <[ISO_639-1::en]>
                        purpose = <"To record metadats details about a
                            multimedia representation of the clinical data
                            that is not captured using the Multimedia data
                            type alone.">
                        use = <"Use to record additional metadata about a
                            multimedia representation of clincal data that
                            is not captured using the Multimedia data type
                            alone.">
                        keywords = <"image", "audio", "text", "video", "
                            application">
                        misuse = <"">
                        copyright = <"openEHR Foundation">
                >
        ["slo"] = <
                        language = <[ISO_639-1::slo]>
```

```
                        purpose = <"Namenjneno␣zapisovanju␣metapodatkov␣o␣
                            podrobnostih␣multimedijski␣predstavitvi␣
                            klinicnih␣podatkov,␣ki␣niso␣zajeti␣samo␣z␣
                            uporabo␣vrste␣podatkov␣Multimedia.">
                        use = <"Uporabite␣za␣snemanje␣dodatnih␣metapodatkov␣
                            o␣vecpredstavnostni␣predstavitvi␣klinicnih␣
                            podatkov,␣ki␣niso␣zajeti␣samo␣z␣uporabo␣vrste␣
                            podatkov␣Multimedia.">
                        keywords = <"slika", "zvok", "tekst", "video", "
                            aplikacija">
                        misuse = <"">
                        copyright = <"openEHR␣Foundation">
                >
        >
        lifecycle_state = <"AuthorDraft">
        other_contributors = <"Grahame␣Grieve,␣Health␣Intersections,␣
            Australia", "Ian␣McNicoll,␣freshEHR,␣UK">
        other_details = <
                ["current_contact"] = <"Heather␣Leslie,␣Ocean␣Informatics,␣
                    heather.leslie@oceaninformatics.com">
                ["MD5-CAM-1.0.1"] = <"626AFF94BCD3048F45914AD130D54262">
        >

definition
        CLUSTER[at0000] matches {       -- Multimedijski vir
                items cardinality matches {1..*; unordered} matches {
                        ELEMENT[at0001] occurrences matches {0..1} matches {
                                -- Multimedijski vir
                                value matches {
                                        DV_MULTIMEDIA matches {
                                                media_type matches {[
                                                    openEHR::]}
                                        }
                                }
                        }
                        ELEMENT[at0002] occurrences matches {0..1} matches {
                                -- Naslov
                                value matches {
                                        DV_TEXT matches {*}
                                }
                        }
                        ELEMENT[at0005] occurrences matches {0..1} matches {
                                -- Opis
                                value matches {
                                        DV_TEXT matches {*}
                                }
                        }
                        ELEMENT[at0003] occurrences matches {0..1} matches {
                                -- Vir
                                value matches {
```

```
                                        DV_TEXT matches {*}
                                }
                        }
                        ELEMENT[at0004] occurrences matches {0..1} matches {
                                -- Datum kracije
                                value matches {
                                        DV_DATE_TIME matches {*}
                                }
                        }
                }
        }


ontology
        term_definitions = <
                ["slo"] = <
                        items = <
                                ["at0000"] = <
                                        text = <"Multimedijski vir">
                                        description = <"Podrobnosti o
                                            multimedijski predstavitvi
                                            klinicnih podatkov.">
                                >
                                ["at0001"] = <
                                        text = <"Multimedijski vir">
                                        description = <"Multimedijska
                                            predstavitev klinicnega
                                            opazovanja ali iskanja.">
                                >
                                ["at0002"] = <
                                        text = <"Naslov">
                                        description = <"Ime
                                            vecpredstavnostnega vira.">
                                >
                                ["at0003"] = <
                                        text = <"Vir">
                                        description = <"Vir
                                            vecpredstavnostnega vira.">
                                >
                                ["at0004"] = <
                                        text = <"Datum kreacije">
                                        description = <"Datum in cas, kdaj
                                            je bil ustvarjen zapis.">
                                >
                                ["at0005"] = <
                                        text = <"Opis">
                                        description = <"Opis
                                            vecpredstavnostnega vira.">
                                >
                        >
```

>

>

DODATEK C

Primer datoteke ECGVisualization_template.html

```
<!doctype html>

<html lang="en">
<head>
  <meta charset="utf-8">
  <title>ECG Savvy measurement visualization</title>
  <meta name="description" content="The␣HTML5␣Herald">
  <meta name="author" content="SitePoint">
  <script type="text/javascript" src="http://code.jquery.com/jquery-1.7.1.
      min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.1.5/Chart.
      bundle.min.js"></script>

  <style>
      body{
          font-family: Arial, Helvetica, sans-serif;
          margin: 20px;
      }
      #main_data{
          margin-left: auto;margin-right: auto;
          font-size: 14px;
      }

      #main_data_table tr td{
          width: 10vw;
      }

      #main_data_table tr td:nth-child(3){
          padding-left: 70px;
      }
      #main_data_table tr td:nth-child(5){
          padding-left: 70px;
      }
```

```
  #ecgGraph{
      border: 1px solid black;
      margin-top: 50px;
      height: auto;
      min-height: 200px;
      background-color: transparent;
    background-position: -6px;
    background-image:
linear-gradient(0deg, transparent 9%,
        rgba(0, 0, 0, .04) 10%, rgba(0, 0, 0, .04) 12%, transparent 13%,
             transparent 29%,
        rgba(0, 0, 0, .04) 30%, rgba(0, 0, 0, .04) 31%, transparent 32%,
             transparent 49%,
        rgba(0, 0, 0, .04) 50%, rgba(0, 0, 0, .04) 51%, transparent 52%,
             transparent 69%,
        rgba(0, 0, 0, .04) 70%, rgba(0, 0, 0, .04) 71%, transparent 72%,
             transparent 89%,
        rgba(0, 0, 0, .04) 90%, rgba(0, 0, 0, .04) 91%, transparent 92%,
             transparent),
linear-gradient(90deg, transparent 9%,
        rgba(0, 0, 0, .04) 10%, rgba(0, 0, 0, .04) 12%, transparent 13%,
             transparent 29%,
        rgba(0, 0, 0, .04) 30%, rgba(0, 0, 0, .04) 31%, transparent 32%,
             transparent 49%,
        rgba(0, 0, 0, .04) 50%, rgba(0, 0, 0, .04) 51%, transparent 52%,
             transparent 69%,
        rgba(0, 0, 0, .04) 70%, rgba(0, 0, 0, .04) 71%, transparent 72%,
             transparent 89%,
        rgba(0, 0, 0, .04) 90%, rgba(0, 0, 0, .04) 91%, transparent 92%,
             transparent);

height:100%;
background-size:50px 50px;
      padding-top:30px;
  }

  .leadname{
      color:grey;
      font-size: 14px;
      padding-left: 30px;
      margin-bottom: 20px;
  }
  #ecgcanvas{
      width:auto;
      overflow-x: scroll;
  }

  #input_window{
      position: absolute;
```

```
            width:500px;
            height: auto;
            border:1px solid black;
            top: 30%;
            left: calc(50% - 250px);
            background-color: whitesmoke;
            padding: 20px;
            display: none;
        }
        #input_window textarea{
            margin-top: 20px;
            margin-bottom: 10px;
            width:100%;
            resize: none;
        }

        #input_window input{
            float:right;
            font-size: 15px;
        }
        #close_annotation_btn{
            color: #333;
        }
        #close_annotation_btn:hover{
            cursor: pointer;
        }
    </style>
</head>
<body>
    <div id="input_window">
        <div id="close_annotation_btn" style="float:right">X</div>
        <b>Dodajanje opombe</b>
        <input id="point" hidden/>
        <br>
        <textarea id="annotation_comment" rows="5" cols="50"></textarea>
        <br>
        <input type="button" value="Dodaj" id="add_annotation_btn" />
    </div>
    <h3>Vizualizacija ECG meritve #$ecgID</h3>
    <div id="main_data">
        <table id=main_data_table>
            <tbody>
                <tr>
                    <td>Patient:</td>
                    <td>$patientNameSurname</td>
                    <td>P-R-T Axis:</td>
                    <td>$prtAxis</td>
                    <td>Study date:</td>
                    <td>$studyDate</td>
                </tr>
```

```
                <tr>
                    <td>Patient id:</td>
                    <td>$patientId</td>
                    <td>PR Interval:</td>
                    <td>$prInterval</td>
                    <td>Total time:</td>
                    <td>$MeasurementTime</td>
                </tr>
                <tr>
                    <td>Patient sex:</td>
                    <td>$patientSex</td>
                    <td>QRS Duration:</td>
                    <td>$qrsDuration</td>
                    <td>Frequency of samples:</td>
                    <td>$frequency</td>
                </tr>
                <tr>
                    <td>Patient birthdate:</td>
                    <td>$patientBirthDate</td>
                    <td>QT/QTc:</td>
                    <td>$qtQtc</td>
                </tr>
            </tbody>
        </table>
    </div>

    <div id="ecgGraph">
        <div class="leadname">$leadneame</div>
      <canvas id="ecgcanvas"></canvas>
    </div>


  <script>

Chart.defaults.global.tooltips = false;
Chart.defaults.global.legend = false;
Chart.defaults.global.animation.duration = 0;

var config = {
  data: {
    labels: $ECGlabels,
    datasets: [{
      type: "line",
      data: $ECGRawdata,
      pointRadius: 0,
      fill:false,
      borderColor: "rgba(0,␣0,␣0,␣1)",
        borderWidth: "1"
    }]
  },
```

```
  options: {
    scales: {
      xAxes: [{
        gridLines: {
          color: "rgba(0, 0, 0, 0)",
        },

      }],
      yAxes: [{
        type: "linear",
        ticks: {
          min: 2.5,
          max: 3.5
        },
        gridLines: {
          display:false
        }
      }]
    }
  },
//lineAtIndex: ['2&lffsdf', '1000&lffsdf']
lineAtIndex: $annotations
};

const verticalLinePlugin = {
    getLinePosition: function (chart, pointIndex) {
      const meta = chart.getDatasetMeta(0); // first dataset is used to
          discover X coordinate of a point
      const data = meta.data;
      return data[pointIndex]._model.x;
  },
  renderVerticalLine: function (chartInstance, pointIndex) {
      const lineLeftOffset = this.getLinePosition(chartInstance, pointIndex.
          split('&')[0]);
      const scale = chartInstance.scales['y-axis-0'];
      const context = chartInstance.chart.ctx;

      // render vertical line
      context.beginPath();
      context.strokeStyle = '#FF3333';
      context.moveTo(lineLeftOffset, scale.top + 30);
      context.lineTo(lineLeftOffset, scale.bottom);
      context.stroke();

      // write label
      context.fillStyle = "#FF3333";
      context.textAlign = 'center';
      context.fillText(pointIndex.split('&')[1], lineLeftOffset, (scale.top
          + 20) );
  },
```

```
  afterDatasetsDraw: function (chart, easing) {
      if (chart.config.lineAtIndex) {
          chart.config.lineAtIndex.forEach(pointIndex => this.
              renderVerticalLine(chart, pointIndex));
      }
  }
  };

  Chart.plugins.register(verticalLinePlugin);

var chart_context = document.getElementById("ecgcanvas");
var chart = new Chart(chart_context, config);


 $(document).ready( function () {
            $("#ecgcanvas").click(
            function(evt){
                var activePoints = chart.getElementsAtEvent(evt);
                var pozicija_tocke = activePoints[0]._index;
                $("#point").val(pozicija_tocke);
                $("#input_window").css('display', 'inline');
            }
            );
     $("#add_annotation_btn").click( function(){
          var pozicija_tocke = $("#point").val();
          var komentar = $("#annotation_comment").val();

          if(komentar != ""){
              $("#input_window").css('display', 'none');
              $("#annotation_comment").css('border', '1px solid grey');

              //klic funkcije ki dodata annotation v dmc datoteko
              //addAnotation(pozicija_tocke, komentar);
          }
          else{
              $("#annotation_comment").css('border', '1px solid red');
          }
      });
      $("#close_annotation_btn").click(function(){
            $("#input_window").css('display', 'none');
     });
 });

    </script>
</body>
</html>
```

DODATEK D

Primer predloge arhetipa openEHR-EHR-CLUSTER.multimedia.v1

```
<?xml version="1.0"?>
<template xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
    http://www.w3.org/2001/XMLSchema" xmlns="openEHR/v1/Template">
  <id>68f711c8-cdaf-449d-bcbe-dfbec5ec4c24</id>
  <name>Merilna naprava</name>
  <description>
    <lifecycle_state>Initial</lifecycle_state>
    <details>
      <purpose>Merilna naprava predloga</purpose>
      <use />
      <misuse />
    </details>
    <other_details>
      <item>
        <key>Multimedijski vir</key>
        <value />
      </item>
      <item>
        <key>Naslov</key>
        <value />
      </item>
      <item>
        <key>Vir</key>
        <value />
      </item>
      <item>
        <key>Datum kreacije</key>
        <value />
      </item>
    </other_details>
  </description>
  <definition xsi:type="ITEM_TREE" archetype_id="openEHR-EHR-CLUSTER.
      multimedia.v1" concept_name="Merilna␣naprava␣predloga">
```

```
  </definition>
</template>
```

Literatura

[1] Activiti. Activiti. Dosegljivo: https://www.activiti.org/docs. [Dostopano: 20. 8. 2017].

[2] Alivecor. Alivecor, 2017. Dosegljivo: https://www.alivecor.com/. [Dostopano: 16. 7. 2017].

[3] Apache. Apache server. Dosegljivo: https://httpd.apache.org/. [Dostopano: 20. 8. 2017].

[4] National Electrical Manufacturers Association. Dicom imaging and communications in medicine (dicom) part5: Data structures and encoding, 2004. Dosegljivo: http://dicom.nema.org/dicom/2004/04_05PU.PDF. [Dostopano: 16. 9. 2017].

[5] National Electrical Manufacturers Association. Dicom imaging and communications in medicine part1: Introduction and overview, 2004. Dosegljivo: http://dicom.nema.org/dicom/2004/04_01PU.PDF. [Dostopano: 16. 9. 2017].

[6] Cablesandsensors.com. 12-lead-ecg-placement-guide-with-illustrations. Dosegljivo: https://www.cablesandsensors.com/pages/12-lead-ecg-placement-guide-with-illustrations. [Dostopano: 25. 8. 2017].

[7] Jerome H. Carter. *Electronic Health Records: A Guide for Clinicians and Administrators*. ACP Press, 2008.

[8] Patrick Davey. *ECG at a Glance*. Wiley-Blackwell, 2008.

[9] dcm4che.org. Knjižnica dcm4che2-2.0.29, 2015. Dosegljivo: `http://www.dcm4che.org/jira/browse/DCM/fixforversion/10984/`. [Dostopano: 20. 8. 2017].

[10] International Organization for Standardization. Iso health standard. Dosegljivo: `https://www.iso.org/`. [Dostopano: 24. 8. 2017].

[11] PHP Group. Lightweight directory access protocol introduction, 2001. Dosegljivo: `http://php.net/manual/en/intro.ldap.php`. [Dostopano: 20. 8. 2017].

[12] Innolitics. Dicom browser, 2017. Dosegljivo: `https://dicom.innolitics.com/`. [Dostopano: 16. 9. 2017].

[13] Health Level Seven INTERNATIONAL. Fast healthcare interoperability resources, 2011. Dosegljivo: `https://www.hl7.org/fhir/`. [Dostopano: 25. 8. 2017].

[14] Health Level Seven INTERNATIONAL. Health level 7, 2011. Dosegljivo: `http://www.hl7.org/`. [Dostopano: 24. 8. 2017].

[15] Jeremy Gibbons Wendy MacCaull. *Foundations of Health Information Engineering and Systems: Third International Symposium*. Springer, 2014.

[16] Amy Nordrum. Popular internet of things forecast of 50 billion devices by 2020 is outdated, 2016. Dosegljivo :`https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated`. [Dostopano: 11. 8. 2017].

[17] openEHR Foundation. Openehr. Dosegljivo: `http://www.openehr.org`. [Dostopano: 13. 9. 2017].

[18] openEHR Foundation. openehr arhitecture arhitecture overview, 2015. Dosegljivo: https://github.com/openEHR/specifications/blob/master/publishing/architecture/overview.pdf. [Dostopano: 13. 9. 2017].

[19] Orace. Java 1.6. Dosegljivo: http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase6-419409.html. [Dostopano: 20. 8. 2017].

[20] PostgreSQL. Postgresql. Dosegljivo: https://www.postgresql.org/. [Dostopano: 20. 8. 2017].

[21] Qardio. Qardiocore wearable electrocardiogram. Dosegljivo: https://www.getqardio.com/qardiocore-wearable-ecg-ekg-monitor-iphone/. [Dostopano: 16. 7. 2017].

[22] SAVING. Savvy ekg. Dosegljivo: http://www.savvy.si/. [Dostopano: 16. 7. 2017].

[23] SAVING. Savvy ekg slika naprave. Dosegljivo: https://i.ytimg.com/vi/x9X8wJZo5_0/maxresdefault.jpg. [Dostopano: 16. 7. 2017].

[24] Pivotal Software. Groovy/grails tool suite. Dosegljivo: https://spring.io/tools/ggts. [Dostopano: 20. 8. 2017].

[25] Pivotal Software. Spring tool suite. Dosegljivo: https://spring.io/tools/sts/all. [Dostopano: 20. 8. 2017].

[26] David H Spodick. Electrocardiogram. Dosegljivo: https://www.radcliffecardiology.com/imaging-diagnostics/electrocardiogram. [Dostopano: 25. 8. 2017].

[27] Hurst JW Walker HK, Hall WD. *Clinical Methods: The History, Physical, and Laboratory Examinations. 3rd edition.* Butterworths, 1990.