Technische Universität Dresden

Fakultät für Elektrotechnik und Informationstechnik

Insitut für Biomedizinische Technik

Manuskript Diplomarbeit

Thema: Entwicklung von Methoden zur Analyse und Aufbereitung biomedizinischer Messdaten

Vorgelegt von: Enrico Grunitz

Betreuer: Dr.-Ing. Sebastian Zaunseder
Dipl.-Ing. Fernando Andreotti

Verantwortlicher Hochschullehrer: Prof. Dr.-Ing. habil. Hagen Malberg

Tag der Einreichung: XX. MONAT 2012

Selbständigkeitserklärung

Mit meiner Unterschrift versichere ich, dass ich die von mir am heutigen Tag eingereichte Diplomarbeit zum Thema

Entwicklung von Methoden zur Analyse und Aufbereitung biomedizinischer Messdaten

vollkommen selbständig und nur unter zuhilfenahme der angegebenen Quellen und Hilfsmittel erstellt habe. Zitate fremder Quellen sind als solche gekennzeichnet.

Dresden, den 17. August 2012

Inhaltsverzeichnis

Abkürzungsverzeichnis

GUI grafische Benutzeroberfläche

LGPL *GNU Lesser General Public License*

1. Einleitung

1.1. Motivation

Ergebnisse automatisierter Biosignalverarbeitungsmethoden werden aus mehreren Gründen oftmals manuell nachbearbeitet. So erfordert die Entwicklung neuer Methoden häufig eine Verifikation der Ergebnisse und eine eventuelle Korrektur der automatisch generierten Ausgabe. Zusätzlich ist eine schnelle visuelle Überprüfung von Ergebnissen, um einen ersten Eindruck über den Effekt einer Änderung an einer Methode zu bekommen, ein Mittel, das in der Entwicklungphase genutzt wird. Daher besteht eine Notwendigkeit eines Werkzeugs, welches die Visualisierung übernimmt und den Entwickler bei der Editierung von Messdaten und Ergebnissen der Signalverarbeitung unterstützt.

Ein solches Werkzeug kann durch die Definition und Festlegung von Ein- und Ausgabeformaten zu einer Vereinheitlichung von Datenformaten führen. Durch die Bereitstellung eines solchen Werkzeugs für Dritte kann auch die methodische Grundlage für die Kooperation verschiedener Institutionen geschaffen werden. Um solche Kooperationen zu unterstützen sollte es, aufgrund der unterschiedlichen Voraussetzungen, wenig spezialsierte Anforderungen an seine Umgebung stellen.

1.2. Zielstellung

Das Ziel dieser Arbeit ist ein Programm zu konzipieren und umzusetzen, das unterschiedliche (Bio-) Signale grafisch darstellt und dem Nutzer die Möglichkeit bietet, Zeitpunkte und -intervalle innerhalb des Signalverlaufs zu markieren und mit Kommentaren zu versehen. Hierbei soll insbesonderer die gleichzeitige Darstellung mehrerer Signale unterschiedlicher Natur und Ausprägung unterstützt werden. Die Erstellung und Bearbeitung von Markierungen soll leicht verständlich aus der grafische Benutzeroberfläche (GUI) heraus geschehen. Zudem soll eine Grundlage geschaffen werden, paralell aufgenommene Signale in einem Datensatz zu vereinen. Zusätzlich soll eine zukünftige Erweiterung der Funktionalität ermöglicht und unterstützt werden. Daher ist eine klare Gliederung der Einzelkomponenten gefordert und die Dokumentation des Quelltextes sowie der

einzelnen Programmteile fundamentaler Bestandteil der Aufgabenstellung. Neben der Programmiererdokumentation soll auch eine seperate Dokumentation für die Benutzer des Programms zur Verfügung gestellt werden.

2. Vorbetrachtungen

2.1. Anwendungsfälle

Um eine Anforderungsliste an die zu erstellende Software ausarbeiten zu können, soll zunächst eine Liste von Anwendungsfällen ausgearbeitet werden. Aus der Liste häufiger Arbeitsabläufe und typischer Aufgabenstellungen wird ersichtlich, was der Nutzer von der Software erwartet. Mithilfe dieser Erwartungen können im Anschluss die Anforderungen an das Programm formuliert und festgelegt werden.

Der Anwender möchte ...

a) einen Datensatzes laden. Dieser Datensatz umfasst mehrere (Bio-) Signale die sowohl mit einer konstanten Abtastrate erfasst wurden als auch Signale die nicht zu äquidistanten Zeitpunkten abgetastet wurden.

b) einen geladenen Datensatz mit allen Änderungen speichern. Hierbei sollen auch Einstellungen gespeichert werden, die die optische Präsentation wiederspiegeln.

c) sich Informationen zu dem geladenen Datensatz und seinen beinhalteten Signalen anzeigen lassen und verändern.

d) bestimmte Signale des Datensatzes auswählen und sich diese in ihrem Verlauf anzeigen lassen (Signalansicht). Hierbei möchte er Bildschirmgröße der einzelnen Ansichten verändern.

e) die Signalansicht bezüglich der Zeit- und der Amplitudenachse vergrößern und verkleinern können (Zoomen). Entlang der Zeitachse möchte er sie verschieben können (Scrollen). Signaleverläufe die parallel aufgenommen wurden, sollen auch zusammen gescrollt werden.

f) in einer Signalansicht mehrere Signale mit denselben Achsen darstellen lassen. Beispielsweise um ein Roh- und ein verarbeitetes Signal miteinander vergleichen zu können.

g) einen Amplitudenbereich eines Signals optisch hervorheben.

h) einzelne Zeitpunkte im Signalverlauf mit einer Markierung versehen und kommentieren. Diese Markierung kann sowohl für ein bestimmtes Signal gelten, aber auch für alle Signale des Datensatzes.

i) einen Zeitabschnitt markieren. Die Markierung der Abschnitte soll analog zur Markierung von Zeitpunkten erfolgen.

j) die Markierungen verändern (zeitlich verschieben, umbennen) oder löschen.

k) Markierungen gemeinsam mit dem Datensatz aber auch unabhängig vom Datensatz abspeichern.

2.2. Anforderungen

Aus den oben genannten Anwendungsfällen ergeben sich die folgenden Anforderungen an das Programm. Das Programm ...

A) muss eine grafische Benutzeroberfläche besitzen.

B) muss ein Datensatzformat unterstützen, das äquidistant und nicht äquidistante abgetastete Signale speichern kann.

C) soll in der Lage sein, Daten aus einem Datensatz zu laden. Dem Nutzer muss es ermöglicht werden, diese Signaldaten aus einer Übersicht auszuwählen und in Diagrammen darstellen zu lassen. Ferner muss der Benutzer eine Möglichkeit haben, sich allgemeine Informationen des Datensatzes anzeigen zu lassen.

D) muss in der Lage sein, die Signalverläufe sowohl einzeln in einem Diagramm daszustellen, aber auch mehrere verschieden Signalverläufe in ein und demselben Diagramm zu visualisieren. Diese Signalansichten sollten in ihrer Darstellungsgröße durch den Nutzer veränderbar sein.

E) soll dem Benutzer ermöglichen, seine Signalansicht frei „bewegen“ zu können. Es muss eine Vergrößerung und Verkleinerung bezüglich der Abszissen- und der Ordinatenachse unterstützen. Zusätzlich ist die Fähigkeit des Verschiebens der Ansicht gefordert. Dabei sollen mehrere Diagramme gleichzeitig Verschoben werden können.

F) muss in der Lage sein einen Amplitudenbreich ein oder mehrerer Signalansichten optisch hervorzuheben.

G) soll dem Nutzer ein Werkzeug zur Verfügung stellen, das ihm erlaubt Datenpunkte zu annotieren. Diese Annotationen sollen optisch in den Signalansichten ersichtlich sein und mit

einem Kommentar versehen werden können. Neben der Annotation einzelner Datenpunkte, soll auch die Markierung von Signalbereichen unterstützt werden. Ferner ist gefordert, dass vorhandene Annotationen veränderbar sind.

H) muss Änderungen an den Signalen selbst und den Annotationen speichern können. Annotationen müssen unabh"ngig von anderen Signalen gespeichert werden können. Insbesondere dürfen Annotationen sich nicht verändern, wenn sich das Ursprungssignal verändert oder nicht mehr vorhanden ist.

I) soll interne Einstellungen abspeichern und von einer Sitzung zur nächsten übernehmen. Optionen bezüglich der Darstellung von Signalen sollen in dem Datensatz mit abgespeichert werden können.

Signalverarbeitungsteil fehlt

2.3. Testszenarien

3. Programmentwurf

3.1. Datenbehandlung

vorhandene Datensatzformate

wahl von unisens begruenden

3.1.1. Unisens

Das vom Forschungszentrum Informatik und Institut für Technik der Informationsverarbeitung der Universität Karlsruhe entwickelte Datenformat Unisens dient der Speicherung und der Dokumentation von Sensordaten. Unisens ist konzipiert, Daten verschiedener Sensoren innerhalb eines Datensatzes zu speichern. Ein Datensatz ist im Dateisystem durch ein eigenes Verzeichnis und eine Headerdatei `unisens.xml` hinterlegt. In der Headerdatei werden alle Informationen über die Bestandteile des Datensatzes, deren Formatierung und ihre semantischen Zusammenhänge gespeichert. Messwerte eines Sensors werden üblicherweise in einer Datendatei innerhalb des Datensatzverzeichnisses abgespeichert. Eine solche Datendatei wird als *Entry* in dem Datensatz bezeichnet. Alle Metainformationen zu den Sensordaten werden in der Headerdatei abgspeichert, so dass die Datendateien selbst immer nur die reinen Messdaten enthalten. Als mögliche Sensordaten werden sowohl kontinuierlich abgetastete Signale als auch ereignisorientierte Daten unterstützt. Unisens unterscheidet zwischen vier Arten von Daten:

Signale *(Signal)*

Signale sind äquidistant abgetastete, numerische Messdaten. Sie zeichnen sich durch eine beliebige aber konstante Abtastrate und Abtastauflösung aus. Zudem können Signale aus mehreren Kanälen bestehen, die alle in ein und derselben Datei abgespeichert werden. Alle Kanäle desselben Signals haben auch dieselbe Abtastrate und -auflösung.

Ereignisse *(Event)*

Ereignisse sind diskrete Zeitpunkte die mit einer textlichen Beschreibung versehen sind. (z.B.

Triggersignale) Sie zeichnen sich durch einen Zeitstempel und einer kurzen Beschreibung aus. Optional können noch Kommentare zu einem Ereignis hinzugefügt werden.

Einzelwerte *(Value)*

Einzelwerte sind eine Kombination der beiden oben genannten Datenarten. Sie beinhalten numerische Werte die zu bestimmten Zeitpunkten aufgenommen wurden. Mit Einzelwerten ist es möglich Daten zu speichern, die nicht in festen Zeitintervallen gemessen werden.

Proprietäre Daten *(Custom data)*

Mit dieser Art können anwendungsspezifisch Daten gespeichert werden, die durch die drei oben genannten Arten nicht erfasst werden können. So können beispielsweise schematische Darstellungen des Messaufbaus als Bilddateien oder Patienteakten in Form von Textdateien dem Datensatz hinzugefügt werden.

Eine detailiertere Beschreibung des Formates kann der offiziellen Dokumentation [4] entnommen werden.

3.1.1.1. Details der Implementierung

In diesem Abschnitt wird kurz auf einige Details der Umsetzung des Unisens-Formates eingegangen. Das Unisens-Paket ist in Java implementiert und wird unter der GNU Lesser General Public License *(LGPL)* zur Verfügung gestellt. Die bereit gestellte Bibliothek ist auf zwei Einzeldateien aufgeteilt: `org.unisens.jar` und `org.unisens.ri.jar`. Bei der ersten Datei handelt es sich um die Definiton des Unisensformates und seiner Bestandteile als Javalassenstruktur. Diese Definition erfolgt hauptsächlich als *Interface*klassen und legt die Schnittstellen zwischen den einzelnen Bestandteilen fest. Eine Übersicht der Klassenstruktur und der von außen ersichtlichen Attribute ist in Abbildung 1 auf Seite 13 dargestellt. Die vom Unisensformat unterstützten Signalarten sind auch in der Klassenstruktur erkennbar:

Signale	`SignalEntry`
Ereignisse	`EventEntry`
Einzelwerte	`ValuesEntry`
Proprietäre Daten	`CustomEntry`

Tabelle 1.: Signalarten und ihre repräsentierenden Klassen

Aufgrund der Ableitung der Klassen `EventEntry` und `ValuesEntry` von `TimedEntry` ist ersichtlich, dass die Zeitpunkte von Ereignisdaten und Einzelwertdaten über eine virtuelle Abtastrate bestimmt werden. Der Zeitpunkt eines jeden *Event*- oder *Value*-Eintrags ist als ganzzahlige Samplenummer dieser Abtastrate gespeichert. Die Zeit eines Ereignisses, relativ zum Messbeginn, er-

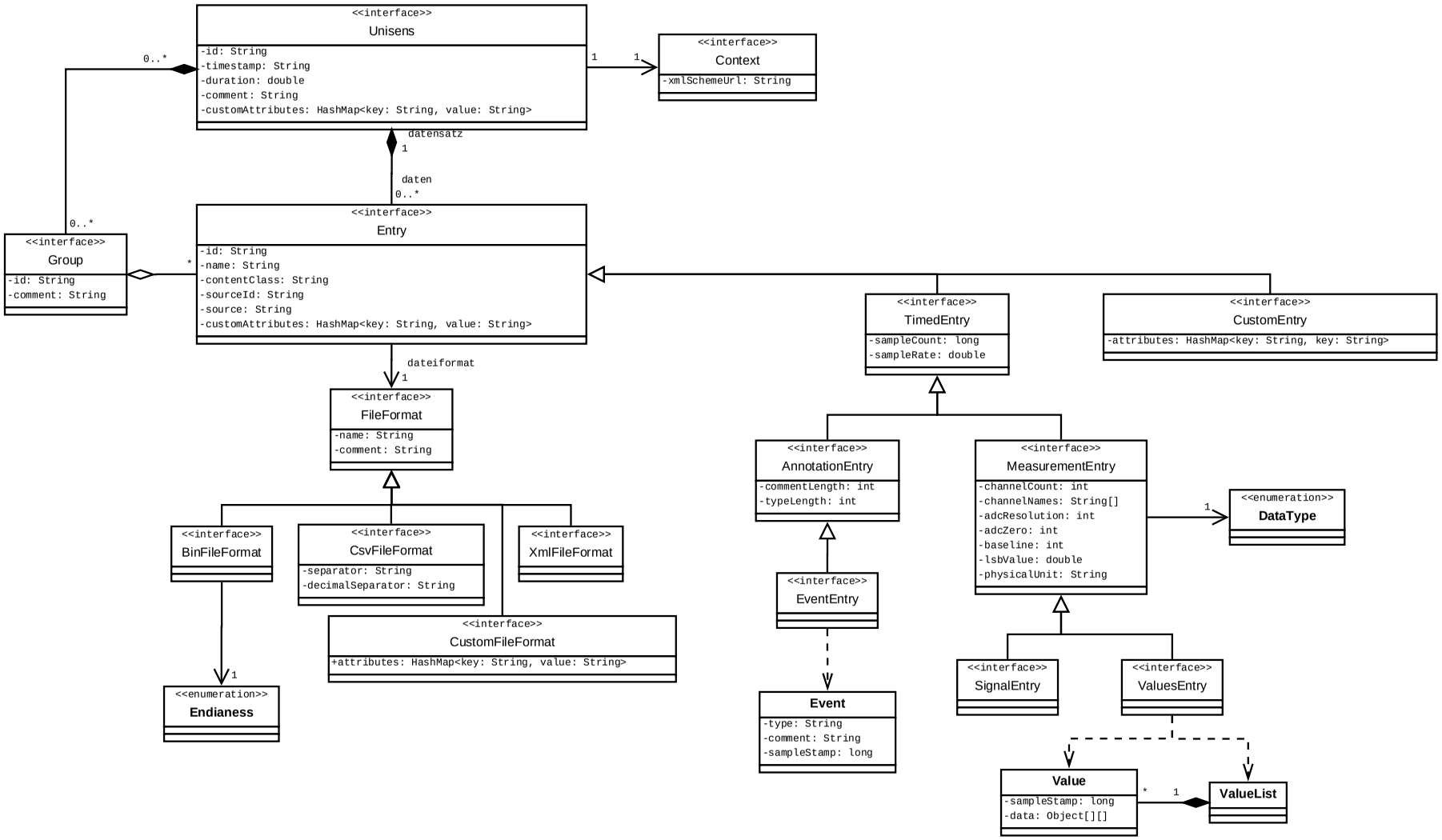

Abbildung 1.: Klassenübersicht der von Unisens definierten Schnittstellen

rechnet sich somit $Zeitpunkt = \frac{Samplenummer}{Abtastrate}$. Möchte man die Möglichkeit Ereignisse für jeden beliebigen Datenpunkt eines Datensatzes zuordnen zu können, dann muss die virtuelle Abtastrate als das kleinste gemeinsame Vielfache aller vorhandenen Abtastraten gewählt werden.

Die Schnittstellendefinition des Unisensformats stellt nur Methoden zum Lesen und Anhängen von Datenpunkten an den Datensatz bereit. Somit wird ein Einfügen, Löschen oder Verändern von Datenpunkten innerhalb eines Dateneintrags nicht unterstützt. Sollen diese Funktionen vorhanden sein, so muss diese Funktionalität selbst implementiert werden.

Die eigentliche Umsetzung der Funktionalität ist in der zweiten Datei abgespeichert. Im Folgenden soll sich der Begriff Referenzimplementierung auf diese funktionelle Umsetzung beziehen. Die Klassen der Referenzimplementierung bestehen aus den Klassennamen der Schnittstellendefinition und dem Suffix „Impl“ (z.B. Objekte die den Datensatz darstellen haben die Klasse `UnisensImpl`). Wenn man schon vorhandene Unisensdatensätze benutzen möchte reicht es aus, die Schnittstellendefinition zu kennen und zu nutzen. Sollen hingegen konkret Objekte erstellt werden, muss auf die Referenzimplementierung zurückgegriffen werden.

Durch einen Fehler in der Referenzimplementierung kann es vorkommen, dass beim Laden eines vorhandenen Unisensdatensatzes in dem Gruppen definiert sind eine `NullPointerException` auftritt. Insbesondere tritt dieser Fehler auf, wenn innerhalb der Headerdatei der Gruppeneintrag nicht hinter den Dateneinträgen steht.

3.1.2. Interne Datenstruktur

Kapselung von unisens in wrapperklassen -> erweiterung und vereinfachung der funktionalität

datenübergabe an visualisierung mithilfe der controller

3.1.3. Dateibehandlung

3.2. Benutzerführung

3.2.1. Grafische Buntzeroberfläche

3.2.2. Datenvisualisierung

4. Realisierung

5. Ergebnisse

NOTIZ: Speicherplatzbedarf bei speicherung in int16 und double

5.1. Erfüllung der Anforderungen

5.2. Evaluation der Nutzeroberfläche

6. Diskussion

6.1. Bewertung der Evaluation

6.2. Ausblick

6.3. Grenzen

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

[1] Chlebek, P. : *User Interface-orentierte Softwarearchitektur.* Friedrich Vieweg & Sohn Verlagsgesellschaft mbH, 2006

[2] Cooper, A. ; Reimann, R. ; Cronin, D. : *About Face 3. The Essentials of Interaction Design.* Wiley Publishing, Inc., 2007

[3] Cooper, A. ; Reimann, R. ; Cronin, D. : *About Face. Interface und Interaction Design.* Hüthig Jehle Rehm GmbH, 2010. – Übersetzung der amerikanischen Originalausgabe [2]

[4] Ottenbacher, J. ; Kirst, M. : *Unisens 2.0 Dokumentation*, Februar 2010. `http://unisens.org/downloads/documentation/Unisens-Dokumentation.pdf`

[5] Rupp, C. ; Queins, S. ; Zengler, B. : *UML 2 glasklar.* Carl Hanser Verlag, 2007

A. UML Dokumentation

B. Daten CD

Inhalt

./Diplomarbeit elektronische Form dieser Diplomarbeit

./Diplomarbeit/src LaTeX-Quelltext dieser Diplomarbeit

./Programm Quellcode des in dieser Arbeit umgesetzten Programms

./Literatur gesammelte Literatur