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 voon 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 15. 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 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.

1.3. 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.

1.3.1. Implementierungsdetails

In diesem Abschnitt wird kurz auf einige Details der Umsetzung des Unisens-Formates eingegangen. Unisens 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 nur um die Definition der Schnittstellen der einzelnen Klassen. Die eigentliche Umsetzung der Funktionalität ist in der zweiten Datei abgespeichert und die Klassennamen sind durch den Suffix „Impl“ erweitert. Im Folgenden soll sich der Begriff Basisimplementierung auf diese funktionelle Umsetzung beziehen. 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 Basisimplementierung zurückgegriffen werden. Eine Übersicht der Klassenstruktur und der von außen ersichtlichen Attribute ist in Abbildung 1 dargestellt. Die unterstützten Signalarten sind auch in der Klassenstruktur erkennbar:

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

Tabelle 1.: Signalarten und die dazugehörigen 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, errechnet 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.

Durch einen Fehler in der Basisimplementierung 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.

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.

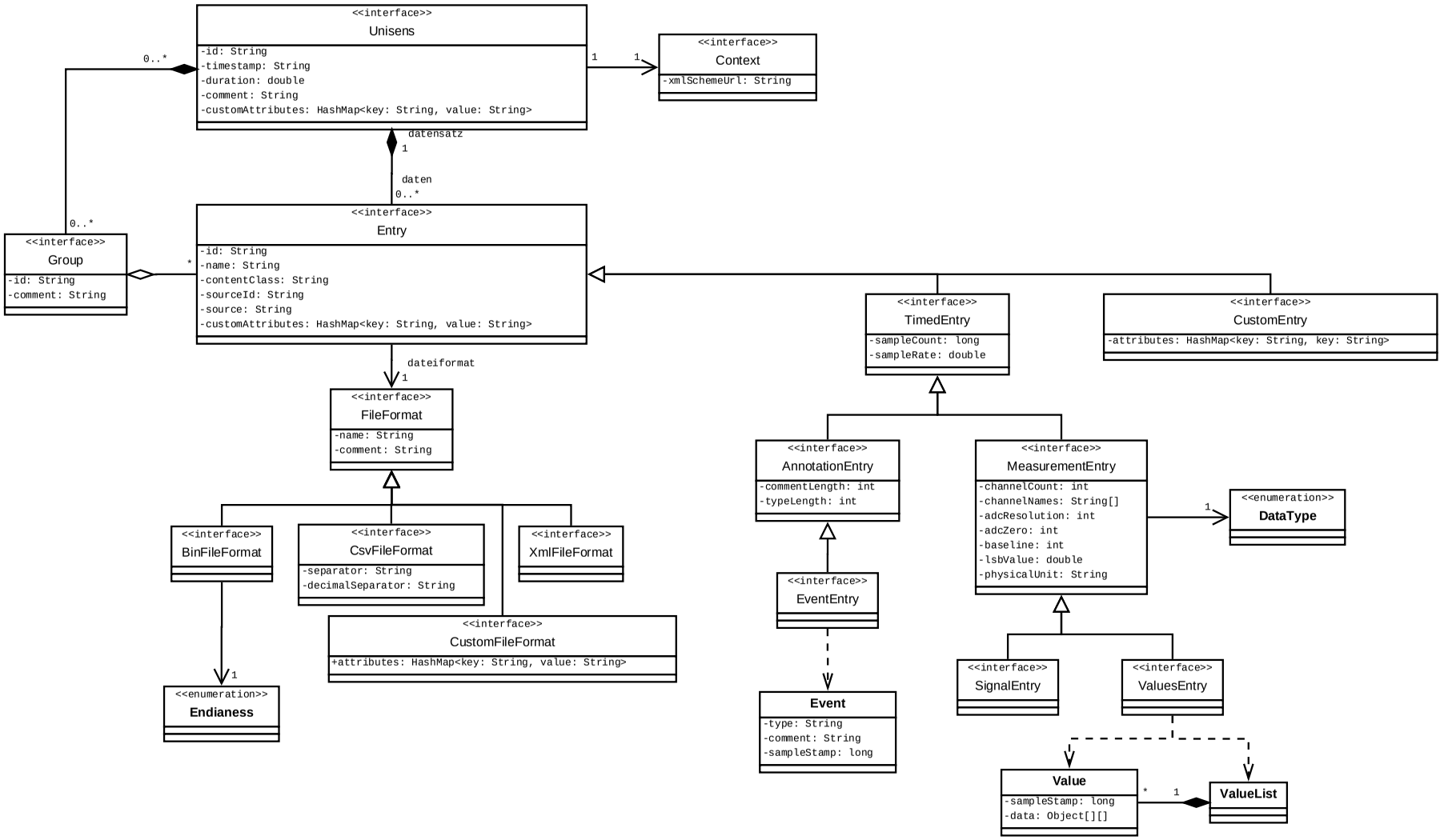

Abbildung 1.: Klassenübersicht der von Unisens definierten Schnittstellen

2. Präzisierung der Aufgabenstellung

2.1. Anwendungsfälle

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. Zum Beispiel 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

2.3. Testszenarien

3. Entwurf

3.1. Konzept

3.2. Datenbehandlung

3.2.1. Datenstruktur

3.2.2. Dateibehandlung

3.2.3. Speichermanagement

3.3. Benutzerführung

3.3.1. GUI

3.3.2. 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] 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