UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

TOBIAS BRIGNOL PETRY

API de DICOM para Sistemas de Telemedicina de Eletrocardiografia

Trabalho de Graduação.

Prof. Dr. Cirano Iochpe
Orientador

Porto Alegre, julho de 2010.

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus, que me deu a vida e os dons para realizar tudo aquilo que Ele quis permitir. Agradeço também a todos que contribuiram para a minha formação pessoal e acadêmica, e que compartilham comigo nesta vida:

- Aos meus pais, Jairo e Lígia, que batalharam para que eu tivesse condições de chegar até a faculdade e valores de uma vida digna e honrada;
- Aos meus irmãos, Jesus e Giovani, que foram exemplos na minha criação, e de dedicação na vida acadêmica e profissional;
- Aos meus familiares, que estiveram presentes nos momentos de alegria e sofrimento, me suportaram, e ajudaram a formar minha personalidade;
- Aos meus amigos Marcos e Inglacir, que pela longa convivência e amizade são como uma extensão da própria família;
- Aos professores que tanto me ensinaram ao longo da caminhada, com lições de conhecimento e de caráter.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ACR	American College of Radiology
API	Application Programming Interface
DICOM	Digital Imaging and Communications in Medicine
ECG	Eletrocardiograma
NEMA	National Electrical Manufacturers Association
PACS	Picture Archiving and Communication System
PRAV	Projetos em Áudio e Vídeo
SOP	Service-Object Pair
UFRGS	Universidade Federal do Rio Grande do Sul
UID	Unique Identifier
USB	Universal Serial Bus
VR	Value Representation

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

As doenças do coração estão atualmente entre as maiores causas de morte no mundo. O exame de eletrocardiograma é um procedimento simples e não invasivo que permite detectar anomalias no funcionamento do coração de um paciente. A possibilidade de manter, em uma mesma base de dados, o histórico de eletrocardiogramas do paciente, independente do equipamento em que eles foram realizados, facilita o acompanhamento médico e o diagnóstico de enfermidades.

Este trabalho apresenta uma análise do modelo de um eletrocardiograma e do padrão de exames médicos DICOM. Com base nesta análise, se realizou o projeto de desenvolvimento de uma API que permite a conversão de exames realizados em dois aparelhos, com protocolos diferentes, para o formato de arquivo do DICOM. A API pode ser usada também para a leitura de arquivos neste formato, de forma a permitir o armazenamento destes exames em um banco de dados.

A aplicação desenvolvida foi testada utilizando um visualizador conhecido de formas de onda em DICOM, dois aparelhos eletrocardiógrafos, e um sistema de telemedicina com armazenamento de exames em um servidor central.

Palavras-Chave: API, doenças do coração, DICOM, ECG, eletrocardiograma, padrão de exames médicos, telemedicina.

A DICOM API for Electrocardiography Telemedicine Systems

ABSTRACT

Heart diseases are among the world's leading causes of death nowadays. An electrocardiogram is a simple, non-invasive procedure that can unveil a patient's heart condition. Storing a patient's history of electrocardiograms in a single database, regardless of the acquisition equipment, can improve medical supervision and the diagnosis of illnesses.

This document presents an analysis of the basics of an electrocardiogram and the DICOM medical standard. An API for conversion of data acquired from two electrocardiographs with different protocols to the DICOM file format was designed and developed based on that analysis. This API also supports reading a file in the named format, allowing storage in a database.

The developed application was tested using a well-known waveform DICOM viewer, two electrocardiographs, and a telemedicine system with examination data stored in a central server.

Keywords: API, heart diseases, DICOM, ECG, electrocardiogram, medical standard, telemedicine.

1 INTRODUÇÃO

As doenças cardiovasculares, em suas diversas formas, estão entre as mais frequentes causas de morte no mundo. De acordo com o estudo Global Burden of Disease (OMS, 2008), baseado em dados coletados em 2004, 12,2% das mortes naquele ano foram causadas por cardiopatia isquêmica, ocupando o primeiro lugar na tabela de causas de óbito. Em segundo lugar, aparecem as doenças cerebrovasculares, com 9,7% das ocorrências. A cardiopatia hipertensiva ainda aparece na décima terceira posição da tabela, com 1,7% dos óbitos.

Levando em conta as estatísticas apresentadas, observa-se a importância do acompanhamento constante do funcionamento da atividade cardiovascular de cada indíviduo e de seu histórico e evolução de doenças e cardiopatias.

Não existe um padrão comum para armazenamento de exames de eletrocardiograma. Essa situação dificulta o estudo do histórico de um paciente, já que a variedade de formatos proprietários para armazenamento não possibilita que os exames sejam colocados em uma base de dados unificada.

O objetivo deste trabalho é disponibilizar uma API que receba os dados de exames de diversos aparelhos de ECG (eletrocardiograma), em diversos formatos, e permita armazená-los em um mesmo padrão, de forma a proporcionar a interoperabilidade entre os diferentes equipamentos e sistemas.

Existem algumas propostas de padronização para ECGs. Por exemplo, o projeto OpenECG dá apoio ao formato SCP-ECG (OPENECG, 2010); a Health Level Seven desenvolveu o padrão aECG (BROWN, 2005); a organização National Electrical Manufacturers Association ajudou a desenvolver e hoje detém os direitos autorais do DICOM (NEMA PS 3.1, 2010). O padrão DICOM é amplamente usado (HILBEL et al., 2007), bem aceito e adotado pela indústria e um dos padrões de dispositivos médicos mais exitosos na atualidade (MORTARA, 2007), principalmente para modalidades de exame que necessitam armazenar imagens. Entretanto, essa grande aceitação do padrão não abrange ainda a modalidade de eletrocardiogramas, cujas informações consistem, tipicamente, em sinais de ondas, armazenados como uma série de amostras de voltagens obtidas durante um determinado intervalo de tempo junto a um paciente.

Por isso, o DICOM foi usado como padrão de interoperabilidade neste trabalho. Além de ser o padrão universal para armazenamento e transferência de imagens nos servidores PACS (Picture Archiving and Communication System), ele suporta o armazenamento de ondas desde 2000 (DICOM STANDARDS COMITTEE, 2000), com módulos específicos para exames de eletrocardiograma. Com isso, os sistemas das diversas instituições de saúde que usam DICOM poderiam armazenar os diversos exames de pacientes, independente de qual dos equipamentos gerou o exame.

1.1 Objetivo

Esta seção apresenta uma descrição geral do objetivo deste trabalho, e o detalhamento dos passos que foram realizados para alcançá-lo.

1.1.1 Objetivo Geral

Este trabalho é direcionado a escolher um padrão para exames médicos e desenvolver uma API que permita converter os dados de eletrocardiogramas para este padrão, permitindo interoperabilidade entre diferentes equipamentos e sistemas. A implementação deverá ser flexível, de forma que a futura integração de novos eletrocardiógrafos de outros fabricantes seja possível.

1.1.2 Objetivos Específicos

O objetivo geral pode ser dividido em uma série de tarefas menores, mais específicas:

- Estudar e descrever, de maneira básica, um eletrocardiograma;
- Estudar e descrever o padrão DICOM;
- Com base nas etapas anteriores, projetar e desenvolver uma API contendo dois módulos principais: o primeiro módulo tem por funcão permitir a geração de um arquivo DICOM diretamente a partir de um conjunto de dados, que pode ser originário de um eletrocardiograma em outro formato. O segundo módulo possibilita a leitura de um arquivo DICOM para obtenção de seu conjunto de dados. Com base nisso, armazenar um eletrocardiograma neste codificado em um banco de dados relacional;
- Testar e validar a API, utilizando exames de eletrocardiograma gerados por equipamentos de dois fabricantes (Ecafix Funbec ECG 12S, da Transform, e BT 3/6, da Corscience) como dados de entrada do primeiro módulo. Os formatos de exame de cada equipamento são proprietários, e constam em seus manuais de instrução. Os arquivos gerados serão visualizados na ferramenta ECG Viewer (PIXELMED PUBLISHING, 2004), que exibe os gráficos de onda e a estrutura de elementos de dados para eletrocardiogramas em DICOM. O teste do módulo de abertura de arquivos DICOM será realizado com um modelo de exame fornecido pela empresa Excel Medical (EXCEL MEDICAL, 2004), e o armazenamento será feito no sistema de banco de dados do iCare Tele-Ecg, desenvolvido pela empresa i9Access em parceria com o laboratório do PRAV.

1.2 Estrutura do Trabalho

Além do capítulo de introdução, este documento apresenta as etapas de pesquisa e desenvolvimento na estrutura apresentada a seguir.

No capítulo 2, ECG, são apresentados a definição e os princípios básicos de um eletrocardiograma, como a identificação das ondas mais comuns e a apresentação de algumas características básicas dos aparelhos utilizados neste trabalho.

O capítulo 3, DICOM, é uma breve apresentação do padrão e da forma como os dados são modelados, especialmente para o salvamento em disco. No final do capítulo,

o conjunto de dados de um exame de eletrocardiograma em DICOM é mostrado em maior destaque.

A descrição das classes em Java para representar o modelo de dados apresentado na parte sobre DICOM aparece no capítulo 4, Implementação da API, juntamente com exemplos do código fonte.

O capítulo 5, Validação, apresenta os testes realizados na API em um sistema de telemedicina, descrevendo os dados de entrada e saída e mostrando imagens dos exames gerados.

No capítulo 6 são apresentadas as considerações finais do trabalho. Por último, as referências bibliográficas e o Apêndice com a descrição dos elementos de dados usados no trabalho.

1.3 Trabalhos Correlatos

Em 2004, a ferramenta ECG Viewer foi inscrita pela PixelMed Publishing em um concurso de programação, organizado pela OpenECG, para aplicações que fossem capazes de testar, converter, exibir, armazenar ou manipular de outra forma dados no formato SCP-ECG (PIXELMED PUBLISHING). Na verdade, esta aplicação era uma extensão de um conjunto de ferramentas de código aberto para manipulação de DICOM, desenvolvido pela própria PixelMed, para que também pudesse suportar a visualização de exames nestes dois formatos. É possível utilizar as classes desta aplicação (em Java) também apenas como uma API para manipulação de dados em DICOM. Entretanto, ela não possui integração com nenhum aparelho de ECG específico para obtenção de dados de exames.

Existe um trabalho bastante impressionante na área de interoperabilidade entre diferentes padrões de ECG (VAN ETTINGER et al., 2008). A ferramenta proposta, chamada pelos autores apenas de “ECG Toolkit”, pode ser usada como uma biblioteca de conversão entre os padrões aECG, SCP-ECG e DICOM. Além disso, é capaz de armazenar os exames em um PACS, exibi-los e imprimi-los. Assim como o ECG Viewer, não tem interação direta com formatos diferentes de eletrocardiógrafos. Seus testes foram realizados com exames armazenados ou obtidos de equipamentos de ECG da Mortara Instrument, cujos modelos mais novos operam em DICOM.

2 ECG

O eletrocardiograma é um exame que analisa a atividade elétrica do coração do paciente ao longo do tempo, através de eletrodos colocados em vários lugares de sua pele (YANOWITZ, 2010). A medida da voltagem entre os pares de eletrodos é útil para indicar o ritmo do coração, e indispensável para detectar e diagnosticar eventuais lesões no miocárdio (VAN MIEGHEM et al., 2004).

Os sinais trocados entre esses pares de eletrodos são chamados de derivações (leads). Para um eletrocardiograma de doze derivações, usa-se dez eletrodos, sendo um em cada braço, um em cada perna, e os outros seis em posições pré-estabelecidas no tórax. Devido à relação existente entre os sinais, é possível calcular o valor de uma derivação a partir dos valores de outras. Dessa maneira, é possível que um aparelho envie o sinal de medição de algumas derivações para um computador, e outras sejam calculadas por software. Um aparelho eletrocardiógrafo pode enviar mais dados além das derivações, como indicação de uso de algum filtro, ganho e frequência cardíaca, por exemplo, além de dados peculiares a cada protocolo.

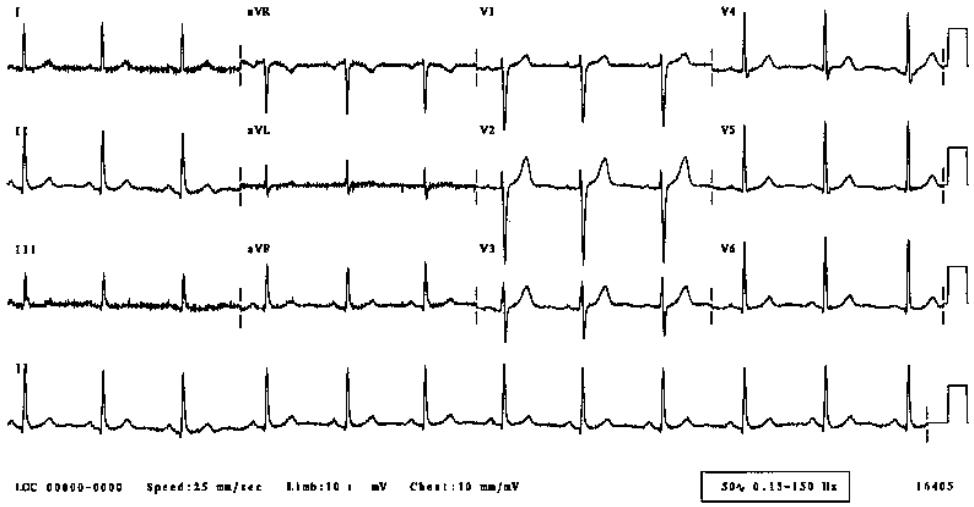

Figura 2.1: Eletrocardiograma de doze derivações (ECG LIBRARY, 2010)

O documento que especifica as formas de onda de ECG no padrão DICOM já foi publicado há algum tempo (DICOM STANDARDS COMMITEE, 2000), mas foi só em 2006 que um fabricante de eletrocardiógrafo anunciou a produção de aparelhos adotando este padrão (HILBEL et al., 2007). Portanto, ainda há muitos aparelhos em circulação que usam seus formatos proprietários.

2.1 Descrição das derivações

As derivações são classificadas em dois tipos: unipolares e bipolares. As derivações bipolares são aquelas que tem um polo positivo e um negativo, combinando os eletrodos colocados nos dois braços e na perna esquerda do paciente. Por isso, essas três derivações são também chamadas de derivações dos membros (limb leads), e são identificadas por I, II e III.

As demais derivações (unipolares) usam o polo positivo de um eletrodo, contra um polo negativo composto dos sinais dos demais eletrodos. Três delas são obtidas dos mesmos eletrodos que compõe as derivações bipolares. São identificadas por aVR (augmented vector right), aVL (augmented vector left) e aVF (augmented vector foot).

As derivações precordiais são unipolares. Seus polos positivos correspondem a cada eletrodo posicionado no tórax. O polo negativo composto é a soma das derivações I, II e III. São identificadas por V1, V2, V3, V4, V5 e V6.

2.2 Eletrocardiógrafos utilizados

Para este trabalho, foram usados eletrocardiógrafos de dois fabricantes: Transform – linha Ecafix Funbec, modelo ECG 12 (TRANSFORM, 2010) e Corscience, modelo BT 3/6 (CORSCIENCE, 2010). Os dois aparelhos possuem especificações diferentes no que diz respeito à resolução dos dados, à frequência de amostragem, às derivações produzidas e ao protocolo de comunicação. A Tabela 2.1 mostra um comparativo das especificações dos aparelhos, descritos a seguir.

Tabela 2.1: Comparativo dos eletrocardiógrafos.

	TRANSFORM Ecafix Funbec ECG 12S	CORSCIENCE BT 3/6
Derivações	12 originais (I, II, III, aVR, aVL, aVF, V1, V2, V3, V4, V5, V6)	2 originais (II, III), 4 derivadas (I, aVR, aVL, aVF)
Resolução	8 bits (256 valores distintos)	15 bits (32768 valores distintos)
Sensitividade	12,5 μv	2,63 μv
Taxa de amostragem	240 Hz	500 Hz

2.2.1 Ecafix ECG 12S PC

Este eletrocardiógrafo transmite os dados para o computador através de uma conexão USB. Os dados são coletados junto ao paciente através de dez eletrodos, de cujos sinais são produzidas as doze derivações citadas na seção 2.1. A resolução dos dados recebidos é de 8 bits, e a diferença de tensão entre dois sinais com valores consecutivos é de 12,5 microvolts. A cada segundo, são enviadas 240 amostras de cada sinal.

2.2.2 Corscience BT3/6

A comunicação entre esse aparelho e o computador é feita via Bluetooth. Duas derivações são medidas no paciente a partir de quatro eletrodos: Lead II e Lead III. Quatro derivações adicionais são calculadas por software a partir dos sinais recebidos: Lead I, aVR, aVL e aVF. A diferença de tensão entre dois sinais com valores consecutivos é de 2,63 microvolts. A resolução do sinal é de 15 bits. A cada segundo, são enviadas 500 amostras de cada um dos dois sinais.

2.3 Conversão através da API de DICOM

Para gerar os exames corretamente, é necessário que os sinais obtidos dos aparelhos sejam interpretados de acordo com os parâmetros de cada aparelho. Os mesmos devem constar nos elementos de dados correspondentes especificados no padrão DICOM, de forma que se evite a distorção das amostras e a perda de informação.

Para cada aparelho suportado, deve ser implementado um módulo que, recebendo um conjunto de parâmetros – como os próprios sinais, frequência de amostragem e quantidade de amostras, por exemplo – gere um arquivo de exame coerente com os dados de entrada.

3 DICOM

Em 1985, foi publicada a primeira versão do padrão, ainda com o nome ACR/NEMA 300 (NEMA PS 3.1, 2010). Foi o resultado de um esforço conjunto iniciado em 1983 por duas organizações - American College of Radiology (ACR) e National Electrical Manufacturers Association (NEMA) - para facilitar a análise de imagens geradas por aparelhos de tomografia computadorizada e ressonância magnética. Como cada equipamento tratava e codificava as imagens à sua maneira, o trabalho dos radiologistas ficava ainda mais complexo. O padrão vem sendo aprimorado ao longo dos anos e, desde 1993, foi renomeado para DICOM. As especificações de comunicação, manuseio, impressão e armazenamento estão divididas em vários sub-documentos, e são atualizadas separadamente conforme a necessidade. A versão mais recente é datada de março de 2010.

No contexto de uma interface de conformação de dados de um eletrocardiógrafo para DICOM, a parte de maior interesse da especificação diz respeito às estruturas de dados, codificação, dicionário de dados e informações diretamente relacionadas a modelos de formas de ondas.

3.1 Modelo de dados

Um arquivo DICOM é composto por uma série de elementos de dados, ou Data Elements, identificados com um rótulo, ou Data Element Tag. Um conjunto de Data Elements é chamado de Data Set (NEMA PS 3.5). O padrão prevê uma grande variedade de Tags, para diversos tipos de exames, além de dados mais genéricos de identificação (NEMA PS 3.6). Cada Data Element tem um tipo de representação associado, como sequências de caracteres, sequências de bytes, pixels de imagens ou formatos de data, por exemplo. Os tipos de representação são chamados de Value Representation (NEMA PS 3.5), e na versão atual do padrão, são 27 variações. A Tabela 3.1 apresenta a lista dos tipos de representação. Além disso, implementações específicas podem definir seus próprios Data Element Tags para elementos de dados, sem sobreposição com os rótulos padrão.

Tabela 3.1: Tipos de representação (Value Representation).

Cód.	Nome	Descrição de uso
AE	Application Entity	Nome da aplicação
AS	Age String	Idade
AT	Attribute Tag	Rótulo de elemento (Data Element Tag)

CS	Code String	Item enumerado, como texto
DA	Date	Data
DS	Decimal String	Número decimal, como texto
DT	Date Time	Data, horário e fuso horário
FL	Floating Point Single	Valor real com 32 bits
FD	Floating Point Double	Valor real com 64 bits
IS	Integer String	Número inteiro, como texto
LO	Long String	Texto com até 64 caracteres
LT	Long Text	Texto com até 10240 caracteres
OB	Other Byte String	Sequência de valores de um byte
OF	Other Float String	Sequência de valores reais, 32 bits cada
OW	Other Word String	Sequência de valores de dois bytes
PN	Person Name	Nome de pessoa estruturado
SH	Short String	Texto com até 16 caracteres
SL	Signed Long	Valor inteiro com 4 bytes
SQ	Sequence of Items	Sequência de Data Sets aninhados
SS	Signed Short	Valor inteiro com 2 bytes
ST	Short Text	Texto com até 1024 caracteres
TM	Time	Horário
UI	Unique Identifier	Identificador, composto por algarismos e pontos
UL	Unsigned Long	Valor sem sinal com 4 bytes
UN	Unknown	Codificação não conhecida pela aplicação
US	Unsigned Short	Valor sem sinal com 2 bytes
UT	Unlimited Text	Texto com até (2^32) – 2 caracteres

Fonte: NEMA PS 3.5, 2010. p 25-32.

3.1.1 Data Element Tag

É o identificador dos Data Elements dentro de um Data Set. Consiste em um par de números, geralmente representados na base hexadecimal, usando quatro dígitos: o número de grupo - Group Number, e o número de elemento – Element Number. O número de grupo organiza os elementos de dados por critérios de similaridade. Por exemplo, os Data Elements referentes à identificação de um paciente tem número de grupo igual a 16 (hexadecimal 0010), e sua distinção é feita pelo número de elemento. Existem aproximadamente 3000 Data Element Tags definidos no padrão DICOM (NEMA PS 3.6). Para cada rótulo, estão associados um nome, o tipo de representação e a multipicidade de valores permitidos para o elemento de dados com este rótulo.

3.1.2 Data Element

Um elemento de dados, ou Data Element, é identificado univocamente pelo seu rótulo (Tag), dentro de um conjunto de dados. É caracterizado ainda pelo valor de dados propriamente dito (Data Element Value), que está condicionado ao tipo de representação (Value Representation) associado a esse rótulo. Tipicamente, um determinado elemento de dados tem um tipo de representação fixo, mas há exceções – por exemplo, o elemento de dados Waveform Data, identificado pelo rótulo de grupo 5400 e de elemento 1010 (valores em base hexadecimal), aceita dois tipos de representação, dependendo da resolução das amostras: "Other Byte String", para amostras de 8 bits, ou "Other Word String", para amostras de 16 bits (NEMA PS 3.6).

3.1.3 Data Set

Um conjunto de dados, ou Data Set, é composto por diversos Data Elements. Um arquivo DICOM é, tipicamente, um conjunto de dados. Um conjunto de dados não deve conter dois Data Elements com o mesmo rótulo, já que o rótulo é um identificador unívoco. Entretanto, essa regra não se aplica para conjuntos de dados aninhados (NEMA PS 3.5). Os elementos de dados com o tipo de representação "Sequence of Items" possuem por valor uma sequência de itens, onde cada item é, por si só, um Data Set. Logo, elementos de dados de mesmo rótulo (com valores possivelmente distintos) podem aparecer em itens diferentes de uma sequência, ou ainda, no Data Set externo ao Data Element que contém a sequência de itens.

Tabela 3.2: Sintaxes de Transferência do padrão DICOM.

DICOM IMPLICIT VR LITTLE ENDIAN TRANSFER SYNTAX
DICOM LITTLE ENDIAN TRANSFER SYNTAX (EXPLICIT VR)
DICOM BIG ENDIAN TRANSFER SYNTAX (EXPLICIT VR)
TRANSFER SYNTAXES FOR ENCAPSULATION OF ENCODED PIXEL DATA (jpeg/mpeg, lossy/lossless, e variações, totalizando 13 identificadores únicos de sintaxes de transferência)
DICOM DEFLATED LITTLE ENDIAN TRANSFER SYNTAX (EXPLICIT VR)
DICOM JPIP REFERENCED TRANSFER SYNTAX (EXPLICIT VR)
DICOM JPIP REFERENCED DEFLATE TRANSFER SYNTAX (EXPLICIT VR)

3.2 Formato de Arquivo

O formato de salvamento de arquivos de objetos DICOM é dependente da escolha da sintaxe de transferência. O padrão especifica diversas sintaxes de transferência, enumeradas na Tabela 3.2, e a opção arbitrária por alguma delas é importante no contexto de transmissão de dados usando o próprio protocolo DICOM de transmissão, ou na compressão de imagens JPEG. As diferenças fundamentais entre as opções de sintaxe dizem respeito à ordem de transmissão dos bytes de uma informação (Little Endian ou Big Endian) e a informação explícita ou não do tipo de representação de valores de cada elemento (Value Representation).

Este trabalho não abrange a implementação do protocolo de transmissão de dados de DICOM, mas apenas o salvamento em arquivos de formas de onda, sem imagens JPEG. Portanto, foi escolhida e implementada uma sintaxe de transferência, chamada "DICOM Explicit VR Little Endian Transfer Syntax". O tipo de representação de valores é explicitamente informado, o que auxilia a leitura posterior do arquivo gerado, e a ordem dos bytes de um valor é Little Endian. A sintaxe padrão de DICOM é "DICOM Implicit VR Little Endian Transfer Syntax", que não explicita o tipo de representação de valor dos elementos de dados. Porém, a escolha da sintaxe foi influenciada também pelo fato de que o exame externo utilizado como exemplo para a leitura de arquivo (EXCEL MEDICAL, 2004) está codificado com VR explícito.

Um arquivo DICOM é identificado por um prefixo de quatro caracteres (bytes, ASCII) "DICM". Esse prefixo é antecedido por 128 bytes livres, chamados de preâmbulo, que tipicamente têm o valor zero. Entretanto, uma implementação específica pode usar outros valores no preâmbulo se desejado. Após o preâmbulo e o prefixo, uma série de elementos de dados chamados de File Meta-Information identifica informações como a sintaxe de transferência usada, o tipo de informação ou exame que consta no arquivo (através de um identificador único) e um identificador único para essa instância (NEMA PS 3.10).

	Data Element Tag	*Value Representation*	*Value Length*	*Value*
Bytes	*8*	*2 ou 4*	*2 ou 4*	*x*
Ex.:	*(0002, 0001)*	*OB*	*x*	*y*

Figura 3.1: Formato do Data Element.

A seguir, o próprio conjunto de dados desejado é escrito no arquivo, elemento a elemento. De acordo com a sintaxe de transferência escolhida, a estrutura de cada elemento de dados (incluindo aqueles que constam nas meta-informações, File Meta-Information) segue o seguinte padrão básico (NEMA PS 3.5), conforme a figura 3.1:

- Data Element Tag: são 4 bytes. Os dois primeiros são lidos como uma palavra em ordem Little Endian (o primeiro byte é o menos significativo) que indica o número de grupo (Group Number) do elemento de dados. Da mesma maneira, os dois bytes seguintes indicam o número de elemento (Element Number).
- Value Representation: esse campo só está presente nas sintaxes de transferência que prevêem essa informação como explícita. Esse campo pode ter dois ou quatro bytes, dependendo do seu valor. Os dois primeiros bytes são caracteres que identificam o tipo de representação (por exemplo, "US" para "Unsigned Short") e, dependendo desse tipo, podem ser reservados mais dois bytes com valor zero, para uso em futuras versões do padrão DICOM.
- Value Length: esse campo possui um valor sem sinal, que indica quantos bytes serão usados para representar apenas o valor deste Data Element, sem contar os campos anteriores (Tag, Value Representation e Value Length). Ele também é codificado em ordem Little Endian, usando tantos bytes quanto o campo Value Representation tiver usado (ou dois bytes para cada campo, ou quatro bytes para cada). Para alguns tipos de representação de

dados, pode-se usar um valor com todos os bits em 1 para indicar comprimento indefinido – neste caso, o fim do Data Element será indicado por um delimitador.

- Value: o próprio valor do Data Element. O formato e o tamanho variam entre os diferentes Data Elements, mas são coerentes com os campos anteriores.

Dentro de um Data Set, os Data Elements são ordenados pelo seu rótulo, primeiro por ordem crescente de Group Number, e depois por ordem crescente de Element Number. Essa regra de ordenação, por si só, garantiria que os elementos de dados correspondentes a File Meta-Information ficassem no início do arquivo, já que o Group Number para todos esses elementos é o mais baixo utilizado (0x0002). Por isso, é possível enxergar a união entre o Data Set escolhido para ser salvo no arquivo e os Data Elements de File Meta-Information como, ela mesma, um Data Set válido.

3.3 Elementos para um ECG em DICOM

Há elementos de dados que são comuns aos Data Sets de quase todos os tipos de exames, como identificação de paciente, médico, instância do exame e data de realização. Nesta seção são apresentados alguns elementos de dados que caracterizam um exame de eletrocardiograma, segundo a definição de objeto de informação General Electrocardiogram (NEMA PS 3.3, 2010). A descrição mais detalhada de todos os elementos de dados usados se encontra no Apêndice Data Elements usados para General ECG.

A modalidade do equipamento que gera o exame é informada no elemento Modality, que deve ter o valor “ECG”. A presença deste elemento com o referido valor não é suficiente para identificar a modalidade do exame, uma vez que há especializações de eletrocardiograma, como 12-Lead ECG ou Ambulatory Electrocardiogram, por exemplo, que compartilham este mesmo valor. O identificador único associado à classe de exames General ECG é indicado em dois Data Elements: Media Storage SOP Class UID (elemento de meta-informação) e SOP Class UID. Seu valor deve ser 1.2.840.10008.5.1.4.1.1.9.1.2 (NEMA PS 3.4).

O instante em que a acquisição de dados do ECG foi iniciada é indicado no elemento Acquisition DateTime. O momento em que o conteúdo do arquivo é gerado é indicado em dois elementos, chamados Content Date e Content Time.

As informações sobre os canais e as próprias amostras de ondas são todas inseridas dentro do elemento Waveform Sequence. O valor desse elemento é uma sequência de itens, onde cada item é um DataSet que descreve uma série de canais. Esta modalidade de exame permite que haja, no mínimo, um, e no máximo, quatro itens dentro de um Waveform Sequence. Os canais podem ser agrupados dentro de um mesmo item quando a similaridade de suas características assim o permite. As propriedades utilizadas neste trabalho que são compartilhadas por canais inseridos dentro do mesmo item são indicadas nos elementos Waveform Originality (classifica as ondas dos canais como originais, ou derivadas a partir de outras), Number of Waveform Samples (quantidade de amostras) e Sampling Frequency (frequência de amostragem, que deve estar no intervalo aberto entre 200 e 1000 Hz). Os demais elementos do item não interferem na decisão de agrupar ou separar canais. São eles Number of Waveform Channels (número de canais dentro deste item), Waveform Bits Allocated (número de bits alocados – tem

valor fixo “16” para a modalidade General ECG), Waveform Sample Interpretation (representação de dados, que tem valor fixo “SS” – signed 16 bit linear - para essa modalidade), Waveform Padding Value (valor inserido pelo equipamento quando os sinais estão inválidos ou ausentes), Waveform Data (as amostras das ondas, descritas no final dessa seção) e Channel Definition Sequence (elemento cujo valor é uma sequência de itens, onde cada item corresponde a um canal).

Os itens do elemento Channel Definiton Sequence, que identificam um canal cada, definem em seus elementos internos informações como o nome da derivação, a unidade de medida, a sensitividade do canal, a resolução em bits dos valores, o valor máximo e o valor mínimo possível para as amostras.

Canais	*1*	*2*	*3*	*...*	*x*
Amostra 1	bytes 0, 1	bytes 2, 3	bytes 4, 5		bytes 2x-2, 2x-1
Amostra 2	bytes 2x, 2x+1				
Amostra ...					

Figura 3.2: Formato de um elemento Waveform Data.

Um elemento de dados Waveform Data, que representa formas de onda, é uma série de amostras. Cada amostra de uma derivação é representada por dois bytes, e os canais presentes neste item do elemento Waveform Sequence são intercalados, conforme a Figura 3.2. Por exemplo, em um eletrocardiograma de 12 derivações (Lead I, Lead II, Lead III, Lead aVR, Lead aVL, Lead aVF, Lead V1, Lead V2, Lead V3, Lead V4, Lead V5 e Lead V6) em que todos canais constam em um mesmo item, todas as primeiras amostras das derivações aparecem em sequência, na mesma ordem em que os canais são enumerados nos itens do elemento Channel Definiton Sequence. Os 24 bytes seguintes representam a segunda amostra de cada derivação no tempo, e assim por diante.

4 IMPLEMENTAÇÃO DA API

Este capítulo descreve como o modelo de dados especificado no padrão DICOM foi implementado na linguagem de programação Java, de forma a permitir a manipulação por programação dos valores dos elementos de dados e a construção de arquivos DICOM válidos para a modalidade de exames de ECG.

O uso da linguagem Java neste projeto facilita o uso desta API em diversas plataformas. Além disso, simplifica o acoplamento com o sistema iCare Tele-Ecg, também desenvolvido nesta linguagem, e usado nos testes de validação. O código documentado foi produzido com auxílio do ambiente de desenvolvimento integrado NetBeans IDE 6.8 (NetBeans IDE 6.8 Release Information, 2010), e o diagrama de classes foi gerado automaticamente a partir do código-fonte, e depois simplificado, usando a ferramenta de análise e projeto de sistemas orientados a objeto astah UML 6.1.1 (astah* UML, 2010).

O conjunto de dados ou Data Set pode ser facilmente transformado em um arquivo, isto é, uma sequência de bytes. Para cada aparelho de ECG suportado, basta implementar um módulo que traduza o protocolo de comunicação para os parâmetros de entrada, porque a estrutura do Data Set gerado será a mesma.

Para que a API seja estendida para outras modalidades de exame, é necessário criar classes que montem a estrutura dos módulos previstos para elas na parte 3 da especificação do padrão DICOM. Entretanto, as classes de Value Representation, Data Element Tag, Data Element, Data Set e escrita de arquivo poderiam ser reaproveitadas diretamente.

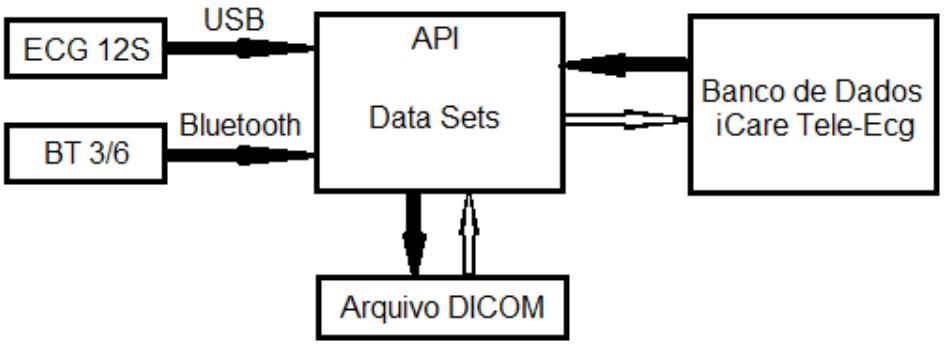

Figura 4.1: Módulos do sistema e fluxo de dados.

Apesar de que essa implementação não tem por objetivo se conformar completamente a qualquer arquivo DICOM, ela é capaz ao menos de ler arquivos binários gerados por ela mesma. Para contornar essa limitação, seria necessário ao

menos implementar todas as sintaxes de transferência possíveis, e todas as variações de codificação inerentes a cada tipo de representação de elementos de dados. Essa tarefa está fora do escopo deste trabalho, e pode ser realizada no futuro. A Figura 4.1 mostra os componentes do sistema em que a API foi inserida, com as setas indicando a direção do fluxo de dados. As setas que estão representadas apenas pelo contorno indicam funcionamento limitado. Isto é, a API não suporta a leitura universal de arquivos DICOM de ECG, nem o armazenamento no banco de dados do iCare Tele-Ecg.

Os protocolos de comunicação dos aparelhos utilizados neste trabalho são de formato proprietário, e não serão descritos neste documento. As informações elementares para a compreensão das características de cada um dos aparelhos constam no capítulo 2, na seção 2.2.

A Figura 4.2 apresenta, em um diagrama de classes, a estrutura básica da API para representar um conjunto de dados em DICOM. O diagrama não enumera todas as subclasses de DataElement, para evitar a poluição visual. Por esse mesmo motivo, a recursividade que advém da possibilidade de aninhar Data Sets (conforme explicado na seção 3.1.3) não é representada explicitamente no diagrama.

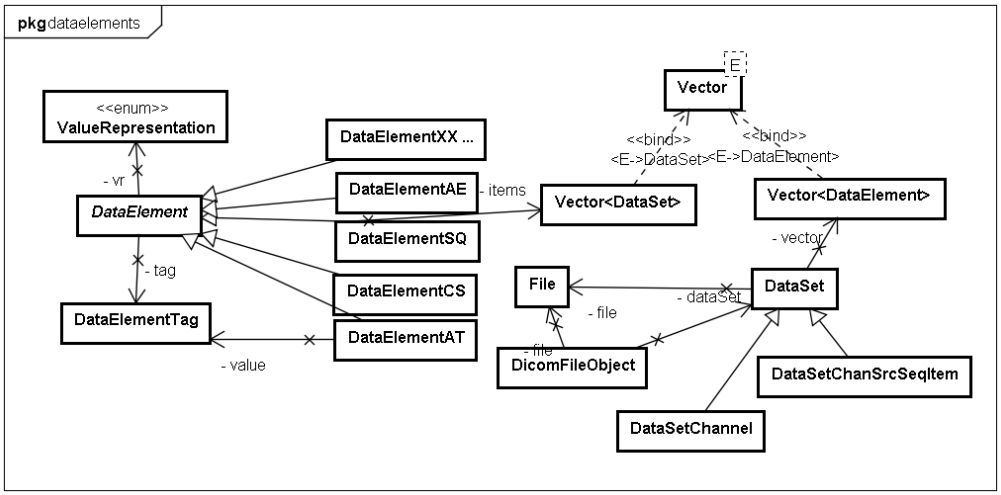

Figura 4.2: Diagrama de classes para o modelo de dados de DICOM.

Nas próximas seções, as principais classes da API serão brevemente descritas. Alguns exemplos de código serão apresentados, com a assinatura dos métodos das classes.

```
1   /**
2    * The ValueRepresentation enumeration defines the byte sequence and the number
3    * of bytes used for the Value Length field for each of the Value Representation
4    * types specified by DICOM.
5    *
6    * @author tbpetry
7    */
8   public enum ValueRepresentation {
9
10      ValueRepresentation(byte[] vrType, int valueLengthFieldLength) {
11          this.vrType = vrType;
12          this.valueLengthFieldLength = valueLengthFieldLength;
13      }
14
15      /**
16       * Returns the byte sequence used in a DICOM file to represent this Value
17       * Representation type in a Data Element.
18       *
19       * @return  the byte sequence.
20       */
21      public byte[] toByteArray();
22
23      /**
24       * Returns how many bytes a "Value Length" field of a Data Element with this
25       * Value Representation type must have in a DICOM file.
26       *
27       * @return  the length of the "Value Length" field.
28       */
29      public int getValueLengthFieldLength();
30
31      /**
32       * Returns the Value Representation type that corresponds to the first two
33       * bytes of the given array.
34       *
35       * @param byteArray  the array of bytes.
36       * @return  a Value Representation enumeration of the corresponding type.
37       * @throws DataSetException if the array of bytes does not match any VR.
38       */
39      public static ValueRepresentation fromByteArray(byte[] byteArray)
40          throws DataSetException;
```

Figura 4.3: Métodos da classe ValueRepresentation.

4.1 ValueRepresentation

É uma enumeração dos tipos de representação da especificação do padrão. Não é declarada diretamente como uma classe, mas sim como Enum (The Java Tutorials, 2010). Define a representação em bytes do identificador de cada tipo para o campo Value Representation, e o tamanho em bytes deste mesmo campo e do campo Value Length. É possível obter a enumeração a partir dos bytes iniciais através de um método, “fromByteArray” (linha 39), para faciliar a criação de um Data Set com origem em um arquivo DICOM, conforme a Figura 4.3.

4.2 DataElementTag

Essa classe mantém as informações que pertencem ao rótulo, ou seja, o número de grupo e o número de elemento. É possível comparar duas instâncias de DataElementTag através de um método (linha 51), de modo a facilitar o ordenamento de elementos de dados a partir do rótulo. O número de grupo e o número de elemento podem ser informados na construção através de Strings, com a representação textual do valor em base hexadecimal (linha 10), ou através dos bytes lidos de um arquivo (linha 17).

A Figura 4.4 apresenta as assinaturas dos métodos da classe, com alguns comentários suprimidos por questão de espaço e clareza.

```
1  public class DataElementTag {
2
3      /**
4       * Creates a new tag with the Group and Element numbers passed as arguments,
5       * given as <code>String</code> objects representing hexadecimal values.
6       *
7       * @param groupString   the Group Number
8       * @param elementString the Element Number
9       */
10     public DataElementTag(String groupString, String elementString);
11
12     /**
13      * Creates a new tag with the byte array passed as argument.
14      *
15      * @param byteArray   a byte array read from a file.
16      */
17     public DataElementTag(byte[] byteArray) throws DataSetException;
18
19     public String getElementString();
20     public void setElementString(String elementString);
21     public String getGroupString();
22     public void setGroupString(String groupString);
23
24     /**
25      * Returns the byte sequence representing this Tag
26      * field, when written to a file. In other words, this method returns the
27      * "Tag" field itself.
28      *
29      * @return  a byte sequence representing the "Tag" field.
30      */
31     public byte[] toByteArray();
32
33     /**
34      * Checks if this Data Element Tag has a Group Number == 0002, meaning that
35      * it refers to a MetaInformation Data Element.
36      *
37      * @return  true if this is a Meta Information Data Element Tag; false
38      * otherwise;
39      */
40     public boolean isMetaInformationTag();
41
42     /**
43      * Compares this Data Element Tag to the given Data Element Tag, so as to
44      * define which one should be written first to a DICOM file.
45      *
46      * @param otherTag  the Data Element Tag to be compared with this one.
47      * @return          a value 0 if both tags have the same Group and Element
48      * numbers; a value less than 0 if this tag has a lower Group Number...
49      * a value greater than 0 if this tag has a higher Group and Element numbers
50      */
51     public int compareTo(DataElementTag otherTag);
52
53 }
```

Figura 4.4: Métodos da classe DataElementTag.

4.3 DataElement

É uma classe abstrata, isto é, não pode ser instanciada. Serve para conter o código da implementação que é comum a todos os tipos de Data Elements. Para cada um dos vinte e sete tipos de representação, há uma classe, chamada DataElementXX (onde "XX" é substituído pelo código de dois caracteres que identifica cada Value Representation), que estende DataElement. Esta classe abstrata define um campo para a associação de um DataElementTag e um ValueRepresentation ao elemento de dados, e mantém um campo com o comprimento, em bytes, do valor do elemento.

Possui métodos para retornar a representação binária do elemento de dados, uma estimativa de quantos bytes esse elemento de dados ocuparia se inserido em um arquivo, e um método abstrato (não implementado, mas obrigatório para as classes que estendem esta) para a transformação apenas do valor do DataElement em uma sequência de bytes (linha 47).

```
1  /**
2   * Extends DataElement to support the INTEGER_STRING Value Representation.
3   *
4   * @author tbpetry
5   */
6  public class DataElementIS extends DataElement {
7
8      /**
9       * Instantiates a Data Element of INTEGER_STRING Value Representation, with
10      * the given tag and value.|
11      *
12      * @param tag   the Data Element Tag associated to this Data Element.
13      * @param value an Integer ({@code -2^31 <= value < 2^31}) representing the
14      * value for this DataElement.
15      */
16     public DataElementIS(DataElementTag tag, Integer value);
17
18     /**
19      * Instantiates a Data Element of INTEGER_STRING Value Representation, from
20      * a byte array read from a file.
21      *
22      * @param value
23      */
24     public DataElementIS(DataElementTag tag, byte[] value);
25
26     public Integer getValue();
27
28     /**
29      * Sets the value of this DataElement to the given argument.
30      *
31      * @param value an Integer ({@code -2^31 <= value < 2^31}) representing the
32      * new value for this DataElement.
33      * @return  true if the value was succesfully set, and is not null/empty;
34      * false if the value was set to null/empty, either intentionally or due to
35      * an invalid argument.
36      */
37     public boolean setValue(Integer value);
38
39     @Override
40     /**
41      * Returns the byte sequence representing the Value
42      * field of this DataElement, when written to a file. In other words, this
43      * method returns the "Value" field itself.
44      *
45      * @return  a byte sequence representing the "Value" field.
46      */
47     public byte[] valueToByteArray()
48
49 }
```

Figura 4.5: Métodos da classe DataElementIS, que estende DataElement.

Todos os tipos de representação são suportados para a geração de arquivos, através de classes que estendem DataElement. Entretanto, alguns tipos de representação podem ser codificados de mais de uma maneira, e nem todas elas foram implementadas neste trabalho. Por isso, há limitações na leitura de arquivos DICOM. A Figura 4.5 mostra a assinatura de DataElementIS, que estende DataElement para o tipo de representação Integer String. A diferença entre as várias classes que estendem DataElement é a maneira como o valor do elemento é manipulado. Apesar de que a maioria das classes trata o valor como um String, a construção de Data Elements com passagem do valor como argumento (linha 16) e o método que define o valor (setValue, linha 37) possuem mecanismos de validação.

Pela especificação do padrão DICOM, o comprimento em bytes de todos os campos de um Data Element deve ser um valor par. Os campos de Tag, Value Representation e Value Length já tem, por definição, valor par de comprimento. Para o campo de valor, pode ser necessário incluir um byte de alinhamento, que varia conforme o tipo de representação (pode ser o valor zero, ou o caractere de espaço em branco, por exemplo). Essa implementação se encarrega de fazer isso automaticamente, de maneira que o programador pode definir valores com comprimento ímpar. O byte de alinhamento será adicionado, se necessário, apenas no valor de retorno do método de obtenção do campo de valor (valueToByteArray, linha 47).

```
1  /**
2   * Extends DataElement to support the SEQUENCE_OF_ITEMS Value Representation.
3   *
4   * @author tbpetry
5   */
6  public class DataElementSQ extends DataElement {
7
8      /**
9       * Instantiates a Data Element of SEQUENCE_OF_ITEMS Value Representation,
10      * with the given tag and items.
11      *
12      * @param tag   the Data Element Tag associated to this Data Element.
13      * @param value variable number of DataSet arguments representing the items
14      * for this DataElement.
15      */
16     public DataElementSQ(DataElementTag tag, DataSet... items);
17
18     /**
19      * Adds a DataSet item to the Sequence inside this Data Element at the
20      * specified index, shifting right existing items if necessary.
21      */
22     public void addItem(int index, DataSet dataSet) throws ArrayIndexOutOfBoundsException;
23
24     public boolean addItem(DataSet dataSet);
25     public boolean removeItem(DataSet dataSet);
26     public DataSet getItem(int index);
27
28     /**
29      * Removes the DataSet item at the given index of the Sequence inside this
30      * Data Element.
31      */
32     public DataSet removeItem(int index) throws ArrayIndexOutOfBoundsException;
33
34     /**
35      * Checks if the given DataSet item exists in the Sequence inside this Data
36      * Element.
37      */
38     public boolean containsItem(DataSet dataSet);
39
40     /**
41      * Returns the number of DataSet items in the Sequence inside this Data
42      * Element.
43      */
44     public int size();
45
46     @Override
47     public byte[] toByteArray();
48
49     @Override
50     public byte[] valueToByteArray();
51
52 }
```

Figura 4.6: Métodos da classe DataElementSQ, que estende DataElement.

Há uma extensão que se diferencia das outras, por não possuir um campo chamado “value” com métodos “getValue” e “setValue”. DataElementSQ é a extensão para o tipo de representação Sequence of Items, cujo valor é, na prática, uma série de Data Sets internos, com uma ordem arbitrária. Os métodos fornecidos por DataElementSQ são mais semelhantes aos de manipulação de vetor, como “addItem” (linhas 22 e 24), “removeItem” (linhas 25 e 32), “containsItem” (linha 38), “getItem” (linha 26) e “size” (linha 44). A Figura 4.6 mostra os principais métodos, em assinatura, para essa classe.

4.4 DataSet

Essencialmente, essa classe encapsula um vetor de Data Elements. A ordem dos elementos é definida com base nos rótulos – ordem crescente de número de grupo, e depois ordem crescente de número de elemento. Não é permitido adicionar dois elementos com o mesmo rótulo (isto é, números de grupo iguais e números de elemento iguais). Os métodos de procura (linha 27) e remoção (linha 39) de Data Elements recebem um rótulo por parâmetro. Assim como DataElement, possui um método que retorna a representação de todos os elementos contidos como uma sequência de bytes (linha 46), e um método que retorna uma estimativa de quantos bytes este conjunto ocuparia se escrito em um arquivo (linha 52).

```
1  /**
2   * A DataSet is a set of Data Elements ordered by their associated Data Element
3   * Tags. A DICOM file can be seen as a Data Set preceded by a header. Each item
4   * inside a Data Element with SEQUENCE_OF_ITEMS Value Representation is also a
5   * Data Set (nested).
6   * <p>
7   * Data Elements are written in a DataSet in ascending order of Group Number,
8   * and then in ascending Element Number. The tag uniquely identifies a Data
9   * Element inside a DICOM DataSet. However, this restriction does not apply to
10  * nested Data Sets. For example, two Data Elements with the same tag can appear
11  * in a Data Set if one Data Element is placed directly in the external DataSet,
12  * and the other one is in an internal DataSet - inside a Data Element with
13  * "SEQUENCE_OF_ITEMS" Value Representation. This rule applies to any number of
14  * Data Elements with the same tag.
15  *
16  * @author tbpetry
17  */
18 public class DataSet {
19
20     public DataSet();
21
22     /**
23      * Tries to find a DataElement inside the DataSet by its tag.
24      * <p>
25      * Note: this method does not search inside any nested Data Sets.
26      */
27     public DataElement findDataElement(DataElementTag tag);
28
29     /**
30      * Adds a DataElement to this DataSet, if there is no DataElement with the
31      * same tag in this DataSet (excluding nested Data Sets).
32      */
33     public boolean addElement(DataElement dataElement);
34
35     /**
36      * Removes the DataElement with the given DataElementTag from the DataSet,
37      * if it can be found.
38      */
39     public boolean removeElement(DataElementTag tag);
40
41     /**
42      * Returns the full byte sequence for all Data Elements
43      * inside this DataSet, ordered by the tags, as it must be written to a
44      * DICOM file.
45      */
46     public byte[] toByteArray();
47
48     /**
49      * Returns an estimate of how many bytes this Data Set would use if written
50      * to a file.
51      */
52     public long getSizeInBytes();
53
54 }
```

Figura 4.7: Métodos da classe DataSet.

A classe DataSet é estendida para facilitar a criação de alguns itens específicos, que são inseridos dentro de elementos do tipo Sequence of Items e usados várias vezes no programa como, por exemplo, a definição dos canais (itens do elemento Channel Definition Sequence), com as classes DataSetChannel e DataSetChanSrcSeqItem. A Figura 4.7 mostra os principais métodos desta classe.

4.5 DataSetException

DataSetException é uma extensão bem simplificada da classe Exception do Java SE (The Java Tutorials, 2010). Foi criada para distinguir as exceções ocorridas na leitura de arquivo ou na criação e manipulação de um Data Set das demais exceções, como entrada e saída, por exemplo. Estas exceções podem ser usadas também quando um arquivo DICOM válido, mas não suportado, é manipulado.

4.6 DicomFileObject

A classe DicomFileObject foi projetada para envolver um DataSet externo, ou seja, aquele que será salvo em um arquivo e que, portanto, deve conter o cabeçalho. Além de adicionar o prefixo "DICM" e permitir a alteração do valor do preâmbulo, esta classe recebe por parâmetro um descritor File, o qual será usado para o salvamento do arquivo. A escrita do arquivo propriamente dita é chamada pelo método "saveFile".

4.7 DicomECGParameters

Essa classe é o ponto de entrada da criação de um DataSet. Por ela, são informados os parâmetros e as próprias amostras de ondas. A Figura 4.8 mostra os parâmetros que são encapsulados por essa classe. A Tabela 4.1 mostra uma descrição desses campos. Os métodos dessa classe são apenas os de recuperação e atribuição dos valores dos campos (também chamados de getters e setters), e não estão apresentados no código para que a visualização seja mais simples.

Tabela 4.1: Descrição dos campos de DicomECGParameters.

acquisitionDateTime	Instante do início do exame
arquivo	Descritor para o arquivo onde o exame será salvo
frequency	Taxa de amostragem
instanceNumber	Número da ocorrência
patientBirthDate	Data de nascimento do paciente
patientId	Código do paciente
patientName	Nome do paciente
patientSex	Sexo
physicianName	Nome do médico
sensorName	Nome do equipamento que gerou o exame
waveformData	Amostras, multiplexadas por canal, codificada como um String de números separados por ponto e vírgula.
originality	Originalidade das ondas
channels	Lista de canais, onde cada canal é uma classe que estende DataSet, com os elementos já definidos internamente

```java
/**
 *
 * @author tbpetry
 */
public final class DicomECGParameters {

    private static DicomECGParameters parameters = new DicomECGParameters();
    private static Date acquisitionDateTime = new Date();
    private static File arquivo;
    private static String frequency = "240";
    private static int instanceNumber = 0;
    private static Date patientBirthdate;
    private static String patientId;
    private static String patientName = "";
    private static String patientSex;
    private static String physicianName = "";
    private static String sensorName;
    private static String waveformData = "";
    private static String originality = "ORIGINAL";
    private static List<DataSetChannel> channels = new ArrayList<DataSetChannel>();

    /* métodos... */

}
```

Figura 4.8: Campos da classe DicomECGParameters.

4.8 ToDicom

A classe ToDicom é a responsável pela estruturação do DataSet de um exame de ECG em DICOM. Nela estão contidos os métodos usados para transformar os parâmetros recebidos em um DataSet pronto para ser salvo em um arquivo, ou para transformar um arquivo lido em um DataSet pronto para ser manipulado.

```java
/**
 * Creates a DataSet with the given parameters class.
 *
 * @param parameters    the parameters class used to generate the DataSet.
 * @return  the resulting DataSet.
 */
public DataSet ecgToDataSet(DicomECGParameters parameters) {
    String sopInstanceUid = getUid(parameters);
    DataSet dataSet = new DataSet();

    // Elementos de Meta Information
    DataElementUL groupLength = new DataElementUL(new DataElementTag("0002", "0000"));
    long groupLengthValue = 0;
    DataElementOB version = new DataElementOB(new DataElementTag("0002", "0001"), new byte[]{0, 1});
    groupLengthValue += 4 + (2 * version.getVr().getValueLengthFieldLength()) + version.getLength();
    /* Trecho de código suprimido */
    groupLength.setValue(groupLengthValue);
    dataSet.addElement(groupLength);
    dataSet.addElement(version);

    // Patient
    // General Study
    // General Series
    // Equipment
    // Waveform Identification
    /* Trechos de código suprimido */

    /* Channel Definition Sequence */
    DataElementSQ channelDefSeq = new DataElementSQ(new DataElementTag("003A", "0200"));
    /* Adding channels */
    for (DataSetChannel channel : parameters.getChannels()) {
        channelDefSeq.addItem(channel);
    }
    /* End of Channel Definiton Sequence */

    DataElementUS wfBitsAllocated = new DataElementUS(new DataElementTag("5400", "1004"), 16);
    DataElementCS wfSampleInterpretation =
            new DataElementCS(new DataElementTag("5400", "1006"), "SS");
    DataElementOW wfData = new DataElementOW(new DataElementTag("5400", "1010"),
            DicomUtils.stringToShortArray(parameters.getWaveformData()));
    /* Trecho de código suprimido */
    wfSequenceItem.addElement(wfBitsAllocated);
    wfSequenceItem.addElement(wfSampleInterpretation);
    wfSequenceItem.addElement(wfData);
    //////
    wfSequence.addItem(wfSequenceItem);
    ////
    dataSet.addElement(wfSequence);

    /* Trecho de código suprimido */

    return dataSet;
};
```

Figura 4.9: Método ecgToDataSet.

Esta classe, em conjunto com a classe DicomECGParameters, forma uma estrutura suficiente para suportar os dois aparelhos de eletrocardiografia usados no trabalho, uma vez que a tradução do protocolo tenha sido feita. Entretanto, por praticidade, foram

projetadas classes que estendem ToDicom, não necessitando da leitura de parâmetros que são fixos para qualquer exame gerado pelo mesmo aparelho. Este é um exemplo do desenvolvimento de um módulo para suportar um novo aparelho para esta API: criar uma classe que estende ToDicom e sobreescreve seu método “ecgToDataSet” com os parâmetros desejados. A Figura 4.9 mostra uma versão reduzida do método “ecgToDataSet”, com alguns elementos de dados suprimidos. Uma lista mais detallhada dos elementos de dados que foram omitidos está no Apêndice.

```
1  public static DataSet byteArrayToDataSet(byte[] byteArray) throws DataSetException {
2      try {
3          Stack<DataElementSQ> sequenceStack = new Stack<DataElementSQ>();
4          Stack<DataSet> dataSetStack = new Stack<DataSet>();
5          dataSetStack.push(new DataSet());
6          /* Trecho sup´rimido... pulo do preâmbulo do arquivo */
7          while (index < byteArray.length) {
8              tag = new DataElementTag(Arrays.copyOfRange(byteArray, index, index + 4));
9              if ((tag.getGroupString().equals("FFFE")) &&
10                     (tag.getElementString().equals("E000"))) { // Item start
11                 dataSetStack.push(new DataSet());
12                 index += 8;
13             } else if ((tag.getGroupString().equals("FFFE")) &&
14                     (tag.getElementString().equals("E00D"))) { // Item end
15                 sequenceStack.peek().addItem(dataSetStack.pop());
16                 index += 8;
17             } else if ((tag.getGroupString().equals("FFFE")) &&
18                             (tag.getElementString().equals("E0DD"))) { // Sequence end
19                 dataSetStack.peek().addElement(sequenceStack.pop());
20                 index += 8;
21             } else {
22                 index += 4;
23                 ValueRepresentation vr = ValueRepresentation.fromByteArray(
24                     Arrays.copyOfRange(byteArray, index, index + 2));
25                 int fieldLength = vr.getValueLengthFieldLength();
26                 index += fieldLength;
27                 long valueLength = DicomUtils.littleEndianByteArrayAsValue(
28                     Arrays.copyOfRange(byteArray, index, index + fieldLength));
29                 index += fieldLength;
30                 switch (vr) {
31                     case APPLICATION_ENTITY:
32                         dataSetStack.peek().addElement(new DataElementAE(
33                             tag, Arrays.copyOfRange(byteArray, index, index + (int) valueLength)));
34                         index += valueLength;
35                         break;
36                     /* todos os Data Elements ... */
37                     case SEQUENCE_OF_ITEMS:
38                         if (valueLength == 0) {
39                             dataSetStack.peek().addElement(new DataElementSQ(tag));
40                         } else {
41                             sequenceStack.push(new DataElementSQ(tag));
42                         }
43                         index += 0;
44                         break;
45                     default:
46                         throw new DataSetException("Unsupported Data Element Type.");
47                 }
48             }
49         }
50         return dataSetStack.pop();
51     } catch /* Tratamento de exceções... */
52 }
```

Figura 4.10: Método byteArrayToDataSet.

O método “byteArrayToDataSet” é responsável por fazer a análise sintática e leitura de um arquivo DICOM. Logo, o argumento do método é um array de bytes, lido de um arquivo. A leitura dos elementos de dados é trivial, com exceção dos elementos DataElementSQ, que incluem os Data Sets aninhados. Para a construção correta da estrutura do DataSet, a leitura dos marcadores de início e fim dos items e dos elementos de sequência insere e remove referências em duas pilhas: uma com os DataSets nos quais os elementos são inseridos, e outra com os Data Elements de sequência de itens dentro dos quais estão os Data Sets. A Figura 4.10 mostra um trecho do código do método “byteArrayToDataSet”.

4.9 DicomUtils

É uma classe que dispõe de constantes e métodos estáticos úteis em diversas partes da API. Possui as expressões regulares e conversores para os elementos de data e hora, funções de manipulação e concatenação de arrays, leitura de valores de tamanho variável em Little Endian, leitura de valores de ponto flutuante codificados em formato binário, e até funções para calcular ondas derivadas de ECG.

5 VALIDAÇÃO

Este capítulo apresenta a validação das funções para as quais a API foi desenvolvida, através de uma descrição mais específica do sistema (já introduzido no início do capítulo 4) e dos testes realizados.

5.1 Sistema

O sistema iCare Tele-Ecg (I9ACCESS, 2010), usado como ambiente para os testes, é voltado para a obtenção, em uma estação remota, de um exame de eletrocardiograma junto a um paciente, com o envio dos dados obtidos no aparelho para um servidor central. A saída gerada pelo eletrocardiógrafo deve ser registrada na estação remota, um computador conectado à Internet, através de uma Applet que é exceutada no navegador.

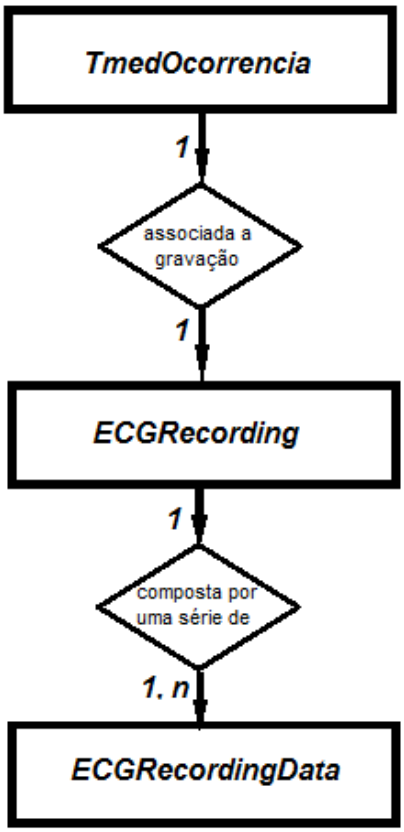

Figura 5.1: Modelo ER para a gravação de exames em banco de dados.

Os exames são armazenados no servidor central em um esquema de banco de dados relacional, cuja representação simplificada no Modelo Entidade Relacionamento é apresentada na Figura 5.1. A entidade TmedOcorrencia se refere a uma consulta ou agendamento, associada a um paciente, a um médico, a um local de atendimento e outras informações, que estão omitidas no diagrama. O relacionamento representado na figura é com uma gravação, que é a entidade ECGRecording. A entidade ECGRecording mantém informações como a taxa de amostragem, a identificação do aparelho que gerou os dados e suas configurações. Ela é relacionada com uma ou mais amostras no tempo. As amostras estão contidas na entidade ECGRecordingData (uma amostra por instância), através dos valores dos sinais das ondas de um determinado instante.

5.2 Testes

A verificação do funcionamento correto da API foi realizada através de uma série de testes de geração de arquivos. Os dados de ECG usados como entrada foram obtidos de exames de exemplo, fornecidos pelos fabricantes dos equipamentos, e por experimentos de exames realizados na empresa i9Access em parceria com o laboratório do PRAV.

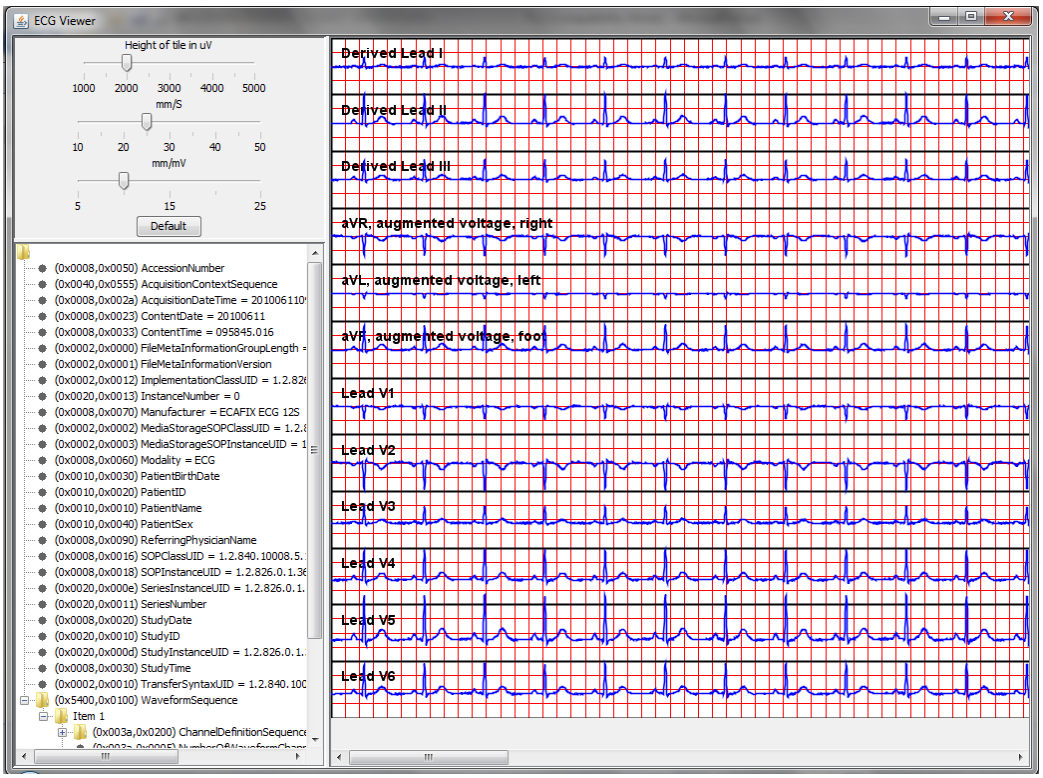

Figura 5.2: Arquivo DICOM gerado a partir de um exame do Ecafix ECG 12S.

Não houve acompanhamento de um profissional da medicina durante a realização dos testes e exames. Entretanto, isso não se configura em um problema, já que a realização de um exame de eletrocardiograma é um procedimento não invasivo e que não oferece riscos. Não há envio de eletricidade do aparelho para o corpo humano e, por consequência, não há risco de choque (DUGDALE et al., 2009). Além disso, o objetivo destes testes não era realizar um diagnóstico de saúde de um paciente, mas apenas verificar que as informações geradas no arquivo eram coerentes com a coleta de amostras do aparelho.

Para a visualização dos arquivos de saída em formato DICOM foi utilizado o software ECG Viewer (PIXELMED PUBLISHING, 2004), capaz de exibir exames de eletrocardiograma nos formatos SCP-ECG e DICOM. Além do desenho de um gráfico das ondas, uma estrutura de árvores com os elementos de dados e seus valores também é mostrada na tela. Não há uma grande variedade de visualizadores de arquivos DICOM que suportem a exibição das formas de onda (VAN ETTINGER et al., 2008). Na figura 5.2, é demonstrado um arquivo DICOM gerado a partir de um exame de exemplo do aparelho ECG 12S, usando a API deste projeto.

Figura 5.3: Exame experimental com BT 3/6, com todas as ondas agrupadas.

Uma das limitações do ECG Viewer é o fato de que ele só exibe os gráficos para o primeiro item do elemento Waveform Sequence de um conjunto de dados DICOM. Isso é um problema para o caso da criação de arquivos de exame com o aparelho Corscience BT 3/6, que gera o sinal original de dois canais de ondas, e os valores de outras quatro ondas são derivadas a partir destes dois sinais. Canais de ondas originais e derivadas não podem ser colocadas em um mesmo item de Waveform Sequence, conforme apresentado na seção 3.3.

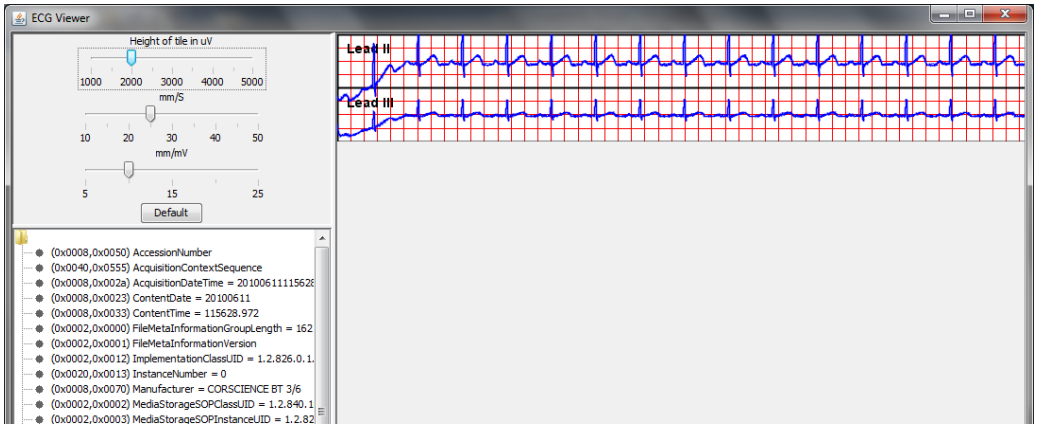

Figura 5.4: Exame modelo do BT 3/6 com ondas originais.

Para contornar essa situação, foram realizados os testes com duas alternativas de solução. A primeira, mais voltada para a facilidade de visualização, foi agrupar todas as ondas em um mesmo item, ainda que a classificação de algumas delas (entre originais ou derivadas) ficasse imprecisa. A Figura 5.3 mostra o trecho de um exame experimental, realizado com um paciente voluntário no dia 9 de junho de 2010, com o aparelho BT 3/6, em que todas as ondas foram agrupadas.

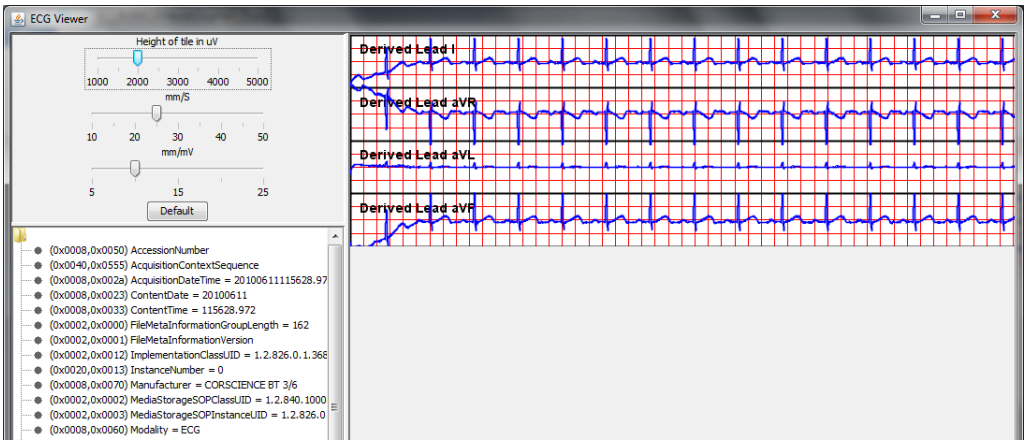

Figura 5.5: Exame modelo do BT 3/6 com ondas derivadas.

A segunda alternativa, mais correta semanticamente, foi manter as ondas originais e derivadas em itens diferentes, e criar dois arquivos DICOM para aquele Data Set, cada um com uma ordem diferente dos itens dentro do elemento de dados Waveform Sequence. A Figura 5.4 mostra as ondas originais do exame de exemplo do aparelho BT 3/6, e a Figura 5.5 mostra as ondas derivadas, do mesmo exame.

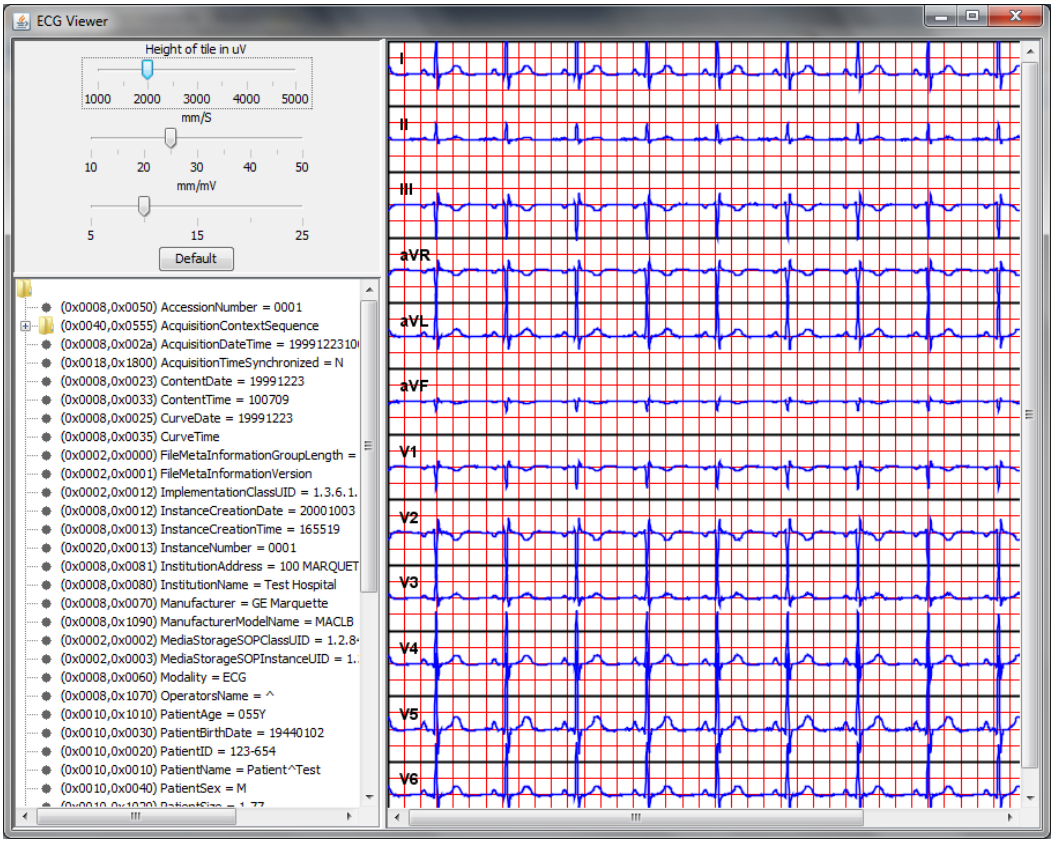

Figura 5.6: Exame da Excel Medical no ECG Viewer.

Para testar a capacidade de abrir arquivos DICOM, foram usados os próprios arquivos gerados pela API e um popular exame de exemplo de um eletrocardiograma de doze derivações, fornecido pela empresa Excel Medical Electronics, Inc. (EXCEL MEDICAL, 2004). Este exame é um raríssimo caso de arquivo DICOM disponível na Web com exemplo de eletrocardiograma, e foi utilizado também na validação do próprio visualizador ECG Viewer (PIXELMED PUBLISHING). A Figura 5.6 mostra este exame no ECG Viewer.

6 CONCLUSÃO

O padrão DICOM, apesar de ser bem difundido em diversas modalidades, não foi adotado como padrão oficial para ECG. Apenas em 2006 que um primeiro fabricante resolveu adotar o padrão DICOM para diagnóstico de eletrocardiografia (HILBEL et al., 2007). Se essa tendência for continuada no futuro por mais fabricantes de ECG, talvez a construção de módulos para converter dados de aparelhos diferentes dentro desta API acabe se restringindo aos equipamentos mais antigos. Mesmo assim, a estrutura da API poderá ser aproveitada para a leitura de arquivos e para a manipulação de Data Sets. Caso a tendência de adoção do padrão diretamente nos aparelhos eletrocardiógrafos não se confirme, esta API poderá ser facilmente estendida para suportá-los, através da implementação da tradução de seus protocolos.

De maneira mais ampla, este trabalho apresenta uma descrição básica do modelo de dados DICOM e de seu formato de arquivo. A API implementada é mais abrangente do que apenas um conversor de ECG, de forma que uma boa parte do trabalho necessário para criar conjuntos de dados para outras modalidades de exames já está pronta.

Novos recursos poderão ser incluídos em trabalho futuro, não apenas para uso direto nas aplicações médicas, mas para servir como uma nova alternativa que pode encorajar o desenvolvimento de aplicações nessa área. Afinal, é notório como há poucas ferramentas conhecidas para testes e visualização de formas de onda em DICOM, e as que existem ainda possuem limitações perceptíveis. As extensões em potencial incluem um visualizador de ondas próprio, o suporte a outras sintaxes de transferência, e a implementação de recursos de diretório e de comunicação que o padrão especifica.

REFERÊNCIAS

BROWN, B. D.; BADILINI, F. **HL7 aECG Implementation Guide**, 2005. Disponível em: <http://www.amps-llc.com/UsefulDocs/aECG_Implementation_Guide.pdf>. Acesso em: junho 2010.

CHANGE VISION, **astah* UML**. Disponível em: <http://astah.change-vision.com/en/product/astah-uml.html>. Acesso em: julho 2010.

CORSCIENCE. **BT3/6, BT12 PC-ECG**. Disponível em: < http://www.corscience.de/fileadmin/Datenblaetter/BT3_6_12_en.pdf>. Acesso em: junho 2010.

DICOM STANDARDS COMMITEE, WORKING GROUP 1 – CARDIAC AND VASCULAR INFORMATION, **SUPPLEMENT 30**: Waveform Interchange. Rosslyn, 2000.

DUGDALE, D. C.; ZIEVE, D. **Electrocardiogram: MedlinePlus Medical Encyclopedia**, U.S. National Library of Medicine, 2009. Disponível em: <http://www.nlm.nih.gov/medlineplus/ency/article/003868.htm>. Acesso em: junho 2010.

ECG LIBRARY, **A normal adult 12-lead ECG**. Disponível em <http://www.ecglibrary.com/norm.html>. Acesso em: junho 2010.

EXCEL MEDICAL ELECTRONICS, INC. **DICOM waveform examples**. 2004. Disponível em: <http://www.excel-medical.com/waveforms/Waveform.htm>. Acesso em: junho 2010.

HILBEL, T. ; BROWN, B. D.; BIE, J. de; LUX, R. L.; KATUS, H. A. **Innovation and Advantage of the DICOM ECG Standard for Viewing, Interchange and Permanent Archiving of the Diagnostic Electrocardiogram**. Durham: Computers in Cardiology, 2007.

I9ACCESS, **I9Access Tecnologia Ltda**. Disponível em: <http://www.i9access.com.br>. Acesso em: julho 2010.

MORTARA, J. L. Choosing DICOM for diagnostic ECG. **Executive Healthcare Management**, New York, Issue 2, November 2007. Disponível em: < http://www.executivehm.com/article/Choosing-DICOM-for-diagnostic-ECG/>. Acesso em: junho 2010.

NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION, **NEMA PS 3-2010**: Digital Imaging and Communications in Medicine (DICOM) – Part 1: Introduction and Overview; Part 3: Information Object Definitions; Part 4: Service Class Specifications; Part 5: Data Structure and Semantics; Part 6: Data Dictionary; Part 10: Media Storage and File Format; Part 16: Content Mapping Resource. Rosslyn, 2010.

OPENECG, **The OpenECG Portal**. Disponível em: <http://www.openecg.net>. Acesso em: junho 2010.

ORACLE CORPORATION, **NetBeans IDE 6.8 Release Information**. Disponível em: <http://netbeans.org/community/releases/68>. Acesso em: julho 2010.

ORACLE CORPORATION, **The Java Tutorials**. Disponível em: <http://java.sun.com/docs/books/tutorial/index.html>. Acesso em: junho 2010.

ORGANIZAÇÃO MUNDIAL DA SAÚDE (OMS), **Global Burden of Disease: 2004 update**, Genebra, 2008. Disponível em: <http://www.who.int/entity/healthinfo/global_burden_disease/GBD_report_2004update_full.pdf>. Acesso em: junho 2010.

PIXELMED PUBLISHING, **ECG Viewer Release Notes**, 2004. Disponível em: < http://www.dclunie.com/pixelmed/software/ECGViewer_ReleaseNotes.pdf>. Acesso em: junho 2010.

TRANSFORM, **Eletrocardiógrafo ECG-12**. Disponível em: < http://www.transform.ind.br/transform_linha-medica_ecg12.html>. Acesso em: junho 2010.

VAN ETTINGER, M. J. B.; LIPTON, J. A.; DE WIJS, M. C. J.; VAN DER PUTTEN, N.; NELWAN, S. P. **An Open Source ECG Toolkit with DICOM**, Roterdã, 2008. Disponível em: <http://www.openecg.net/CinC2008_paper_de_Wijs.pdf>. Acesso em: julho 2010.

VAN MIEGHEM, C.; SABBE, M.; KNOCKAERT, D. **The Clinical Value of the ECG in Noncardiac Conditions**, Roterdã, 2004. Disponível em: < http://chestjournal.chestpubs.org/content/125/4/1561.long>. Acesso em: julho 2010.

YANOWITZ, F. G., **Lesson I: The Standard 12 Lead ECG**. Disponível em: < http://library.med.utah.edu/kw/ecg/ecg_outline/Lesson1/index.html>. Acesso em: julho 2010.

GLOSSÁRIO

Applet – Aplicação escrita na linguagem de programação Java, que pode ser inserida em uma página HTML. O código é executado pela máquina virtual Java do navegador.

Big Endian – Diz respeito ao ordenamento de bytes em um valor composto por vários deles. Os bytes mais significativos são colocados antes dos menos significativos.

Cardiopatia – Termo global para definir diversas doenças do coração.

Data Element – Uma unidade de informação do padrão DICOM, definida no dicionário de dados ou criada por um particular, caracterizada por um Data Element Tag (rótulo), Value Representation (tipo de representação) e um nome. A um elemento de dados pode estar associado um atributo.

Data Element Tag – Rótulo usado para identificar Data Elements, definido por dois valores numéricos (Group Number e Element Number), tipicamente escritos em notação de base hexadecimal.

Data Set – Um conjunto de atributos em DICOM, cada um associado a um Data Element.

Derivação – Sinal gerado por um eletrocardiógrafo, pela medida de diferença de voltagem entre eletrodos conectados a um paciente.

Eletrodo – Condutor elétrico usado para detectar a atividade elétrica do corpo do paciente em um exame de eletrocardiograma.

Little Endian – Diz respeito ao ordenamento de bytes em um valor composto por vários deles. Os bytes menos significativos são colocados antes dos mais significativos.

Preâmbulo – Em DICOM, é o trecho inicial do arquivo, com 128 bytes, que pode ser usado livremente pelas aplicações que escrevem os arquivos. Antecede o prefixo “DICM”.

Taxa de amostragem – Medida da frequência com que as amostras são recebidas, isto é, o número de amostras geradas durante um determinado intervalo de tempo. Tipicamente medida em Hz (Hertz), que definem este intervalo de tempo como um segundo.

Value Representation – Define o tipo e o formato dos dados do valor contido no campo Value de um Data Element.

APÊNDICE < DATA ELEMENTS USADOS PARA GENERAL ECG>

Este apêndice mostra uma lista dos Data Elements definidos no padrão DICOM que são usados neste trabalho. São omitidos aqui alguns elementos de dados que podem ter valor vazio e, por simplificação, são gerados assim no Data Set nessa implementação. Para cada Data Element, é informado o seu nome, seu Tag e uma breve descrição. A descrição mais detalhada se encontra na parte 3 do padrão DICOM.

- File Meta Information Group Length (0002,0000): Número de bytes usados pelos elementos de Meta Informação, sem contar este.
- File Meta Information Version (0002,0001): Versão do cabeçalho.
- Media Storage SOP Class UID (0002,0002): Identificador único da classe SOP (Service Object Pair) – no caso, o tipo de exame.
- Media Storage SOP Instance UID (0002,0003): Identificador único dessa instância do exame.
- Transfer Syntax UID (0002,0010): Identificador único da sintaxe de transferência usada no arquivo.
- Implementation Class UID (0002,0012): Identificador único desta implementação que gerou o arquivo.
- SOP Class UID (0008,0016): Novamente, identificador único da classe SOP (modalidade do exame).
- SOP Instance UID (0008,0018): Novamente, identificador único da instância.
- Content Date (0008,0023): Data de geração dos dados.
- Acquisition DateTime (0008,002A): Instante do início da coleta dos dados.
- Content Time (0008,0033): Hora de geração dos dados.
- Modality (0008,0060): Modalidade do equipamento usado para o exame.
- Manufacturer (0008,0070): Fabricante do equipamento que gerou os dados.
- Referring Physician's Name (0008,0090): Nome do médico.
- Code Value (0008,0100): Código usado para descrever uma unidade de medida ou uma lead.
- Coding Scheme Designator (0008,0102): Esquema de codificação usado.

- Coding Scheme Version (0008,0103): Versão do esquema de codificação.
- Code Meaning (0008,0104): Nome da unidade de medida ou da lead indicada pelo código.
- Patient's Name (0010,0010): Nome do paciente.
- Patient ID (0010,0020): Código do paciente.
- Patient's Birth Date (0010,0030): Data de nascimento.
- Patient's Sex (0010,0040): Sexo.
- Study Instance UID (0020,000D): Identificador único para o estudo.
- Series Instance UID (0020,000E): Identificador único para a série.
- Instance Number (0020,0013): Número de identificação para as ondas.
- Waveform Originality (003A,0004): Indica se as ondas são originais ou derivadas.
- Number of Waveform Channels (003A,0005): Número de canais contidos em um item de agrupamento.
- Number of Waveform Samples (003A,0010): Quantidade de amostras de cada canal de um item de agrupamento.
- Sampling Frequency (003A,001A): Frequência de amostragem (Hz).
- Channel Definition Sequence (003A,0200): Sequência de itens, onde cada item representa um canal.
- Channel Source Sequence (003A,0208): Sequência com um item, que descreve o nome e o código da lead de um canal.
- Channel Sensitivity (003A,0210): Sensitividade do canal.
- Channel Sensitivity Units Sequence (003A,0211): Sequência com um item, que descreve o código e o nome da unidade de medida de um canal.
- Channel Sensitivity Correction Factor (003A,0212): Constante a ser multiplicada para corrigir amostras de acordo com a unidade de medida.
- Channel Baseline (003A,0213): Valor que serve como referência de zero.
- Channel Sample Skew (003A,0215): Número de amostras desde o início da coleta até o primeiro valor válido.
- Waveform Bits Stored (003A,021A): Número de bits significativos de uma amostra.
- Waveform Sequence (5400,0100): Sequência de itens, onde cada item representa um agrupamento de canais.
- Channel Minimum Value (5400,0110): Valor mínimo válido para uma amostra.
- Channel Maximum Value (5400,0112): Valor máximo válido para uma amostra.

- Waveform Bits Allocated (5400,1004): Tamanho de cada amostra, dentro do element Waveform Data.
- Waveform Sample Interpretation (5400,1006): Tipo de representação dos dados das amostras.
- Waveform Data (5400,1010): Valores das amostras.