INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

MESTRADO EM COMPUTAÇÃO E INSTRUMENTAÇÃO MÉDICA

Integração modular em visualizadores médicos e aposição da marcação CE do Dispositivo Médico

MARTINHA ARMADA RODRIGUES ALVES

Outubro de 2013

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Departamento de Física

Integração modular em visualizadores médicos e aposição da marcação CE do Dispositivo Médico

Martinha Armada Rodrigues Alves

Dissertação apresentada no Instituto Superior de Engenharia do Porto para a obtenção do grau de Mestre em Engenharia de Computação e Instrumentação Médica

Orientador:
Eng. Jorge Estrela da Silva
Co-Orientador:
Eng. Filipe Morais

Outubro de 2013

Agradecimentos

Este espaço permite-me agradecer a todos que possibilitaram e ajudaram na realização desta dissertação.

À minha mãe devo um agradecimento muito especial. Ela foi o estímulo do início e uma ajuda constante, concreta e infatigável ao longo destes anos. Ao meu pai e aos meus irmãos agradeço o apoio, amizade e amor que me oferecem dia após dia.

Ao Eng. Jorge Silva, orientador desta tese de Mestrado, expresso os meus sinceros agradecimentos pelo rigor científico, disponibilidade em ensinar e empenho da supervisão.

Ao Eng. Filipe Pires de Morais e ao Eng. Ricardo Pinho pelo acolhimento na Empresa Efficientia - Management, Integration and Consulting, pela sugestão do tema e pela minha introdução nesta área de soluções médicas, assim como ao Serviço de Cardiologia do Centro Hospitalar de Vila Nova de Gaia/Espinho, EPE - Unidade I, pelas orientações.

Ao Eng. Carlos Ramos, Director do Mestrado de Computação e Instrumentação Médica, e aos restantes docentes que me acompanharam nesta nova fase da minha formação todos os conhecimentos transmitidos.

Aos meus amigos nomeadamente à Lia, ao Ricardo, à Matilde e à Andreia que além dos momentos de concentração também me proporcionaram momentos simples e divertidos.

Resumo

Neste documento é feita a descrição detalhada da integração modular de um *script* no *software* OsiriX. O objectivo deste *script* é determinar o diâmetro central da artéria aorta a partir de uma Tomografia Computorizada. Para tal são abordados conceitos relacionados com a temática do processamento de imagem digital, tecnologias associadas, e.g., a norma DICOM e desenvolvimento de *software* .

Como estudo preliminar, são analisados diversos visualizadores de imagens médica, utilizados para investigação ou mesmo comercializados.

Foram realizadas duas implementações distintas do *plugin*. A primeira versão do *plugin* faz a invocação do script de processamento usando o ficheiro de estudo armazenado em disco; a segunda versão faz a passagem de dados através de um bloco de memória partilhada e utiliza o *framework Java Native Interface*.

Por fim, é demonstrado todo o processo de aposição da Marcação CE de um dispositivo médico de classe IIa e obtenção da declaração de conformidade por parte de um Organismo Notificado.

Utilizaram-se os Sistemas Operativos Mac OS X e Linux e as linguagens de programação Java, Objective-C e Python.

Palavras-Chave: Visualizadores de imagens médicas, *Plugin*, DICOM, Dispositivo Médico e Marcação CE.

Abstract

This thesis presents an approach to some concepts related to the theme digital image processing and associated technologies, for example, the DICOM standard. There are presented some of the medical image viewer used for research or to be sold, a detailed description of modular integration into the software of a script-Osirix is also made. The purpose of this script is to determine the core diameter of the aorta from a CT scan.

Two different implementations of the plug-in have been performed. The first version of the plug-in relies on the processing script using the study file stored on disk; The second version makes the transitions from the data through a shared memory block and uses the Java Native Interface framework.

Finally, it is shown the whole process of affixing the CE marking of a medical device class IIa and the attainment of the declaration of conformity by a Notified Body.

Operating Systems as Mac OS X and Linux have been used as well as the programming languages Java, Objective-C and Python.

keywords: Viewers of Medical Images, Plugin, DICOM, Medical Device and CE mark.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

ACR	American College of Radiolagy
AET	Application Entity Title
ANSI	American National Standards Institute
API	Application Programming Interface
AVI	Audio Video Interleave
BMP	Bitmap Image File
DICOM	Digital Imaging and Communications in Medicine
EEE	Espaço Económico Europeu
FDA	Food and Drug Administration
FITS	Flexible Image Transport System
FSF	Free Software Foundation
FTP	File Transfer Protocol
GHTF	Global Harmonization Task Force
GIF	Graphics Interchange Format
HTTP	Hypertext Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
ISO	International Organization for Standardization
JNI	Java Native Interface
JPEG	Joint Photographic Experts Group
JVM	Java Virtual Machine
JWS	Java Web Start
LGPL	Lesser General Public License
LS	LossLess
MB	megabyte
MCM	Medical Content Manager
NEMA	National Equiment Manufactures Association
IP	Internet Protocol
ITK	Insight Segmentation and Registration Toolkit
OSI	Open Systems Interconnection
PACS	Picture archiving and communication system
PET	Positrion Emission Tomography
PGM	Portable GrayMap

PNG	Portable Network Graphics
RFC	Request For Comments
RGBA	Red Green Blue Alpha
RLE	Run-Lenght Encoding
RMN	Ressonância Magnética Nuclear
ROI	Region of Interest
SPECT	Singles Photon Emission Computed Tomography
TC	Tomografia Computorizada
TCP	Transmission Control Protocol
TIFF	Tagged Image File Format
UID	Unique Identifier
URI	Uniform Resource Identifiers
URL	Uniform Resource Locator
US	Ultra-som
VR	Value Representation
VTK	Visualization Toolkit
WADO	Web Access to DICOM Objects
WL	Window Level
WW	Window Width
WWDC	Apples Worldwide Developer Conference
XML	eXtensible Markup Language
2D	Duas Dimensões
3D	Três Dimensões

1

Introdução

As últimas décadas têm sido marcadas por um progresso significativo nas áreas da Engenharia conjuntamente com a Medicina, proeminentemente no desenvolvimento de dispositivos médicos e soluções cada vez mais inovadoras e adaptadas. Esta área de dispositivos conta com aproximadamente 10 000 produtos e a sua indústria é uma relevante entidade empregadora. Em Portugal, a indústria de equipamentos médicos é diminuta, embora as trocas comerciais realizadas neste setor de atividade contribuam para a economia nacional [5].

1.1 Motivação e Objetivos

As Ciências da Computação exploradas ao longo do percurso académico, juntamente com a temática da imagem médica, suscitaram à autora um enorme interesse de descobrir, estudar e contribuir para estas áreas. A possibilidade de integrar funcionalidades num *software* popular na área médica, aprender novas linguagens de programação e o assimilar novos conceitos deram a motivação necessária para o desenvolvimento desta dissertação.

A maioria dos Visualizadores de Imagens Médicas atuais são aplicações complexas, repletas de funcionalidades permitindo cálculos e medições, possibilitando perspetivar anatomicamente o interior do corpo humano de forma minuciosa.

Os objetivos deste projeto são:

- Fazer a integração modular de um filtro, num visualizador que cumpra as expectativas dos utilizadores alvo da empresa;
- Associar um repositório de imagens médicas a um visualizador;

- Examinar os requisitos da Certificação de Dispositivos Médicos para a classe em que o software médico se insere.

Antes de serem colocados no mercado, os dispositivos médicos têm que ser avaliados por uma entidade competente, que é regida por uma rigorosa e metódica legislação. A assimilação de conceitos, assim como a elaboração da documentação técnica dos dispositivos, tem grande importância para o desenvolvimento e comercialização do produto.

Este projeto foi proposto pela empresa *Efficientia - Manegement, Integration and Consulting*, juntamente com o Serviço de Cardiologia do Centro Hospitalar de Vila Nova de Gaia - Unidade I, Hospital Santos Silva.

A empresa distribui estações de trabalho avançadas de visualização e edição de imagem médica, sistemas de arquivo médico digital, impressão de imagem médica DICOM em papel ou a gravação num cd.

Paralelamente a este projecto foi desenvolvido um *script*, que permite fazer medições automáticas do diâmetro da artéria aorta, por outra autora na mesma empresa[4].

O *Plugin* deverá integrar o projeto referido anteriormente e ser instalado num visualizador de imagens com forte implementação, apetecível para a classe médica. Inicialmente, devem ser analisados vários visualizadores desenvolvidos em Sistemas Operativos diferentes, escolhendo dois com forte representatividade no mercado da imagiologia médica;

1.2 Enquadramento

Neste subcapítulo serão abordados conceitos fundamentais neste projeto, tais como, *Plugin*, Dispositivo Médico e Marcação CE.

1.2.1 *Plugin*

Na Engenharia de *Software*, a modularidade de um *software* é considerada um aspeto de qualidade e importância. Caso o *software* não esteja preparado para receber extensões o(s) autor(es), perante as mudanças e as alterações com um tempo limitado, pode(m) ter grandes difculdades para introduzir essas alterações [6].

Em meados do século passado, as aplicações eram programas monolíticos, processuais e só mais tarde surgiu a programação orientada ao objecto. No desenvolvimento de novos aplicações foram usados novos paradigmas para uma melhor estruturação.

Adicionalmente, a comunicação entre módulos tem sido especificado num nível cada vez mais abstrato para reduzir a rigidez do acoplamento e para encorajar a intercambialidade. No entanto, todas estas abordagens de composição são utilizados ao nível de código fonte. Após a compilação, os módulos de uma aplicação são ainda muitas vezes ligados na forma de executáveis monolíticos ou bibliotecas estáticas. Dessa forma, quando um módulo necessita de ser modificado a aplicação tem de ser recompilada[1].

Um *plugin* ou *add-in* é um módulo que permite adicionar novas funcionalidades a uma aplicação após essa aplicação ter sido compilada e implantada num Sistema. Estas extensões permitem, além da modularidade a nível do código fonte, a instalação das mesmas ao nível do código objeto. Os *plugins* podem ser desenvolvidos e compilados independentemente da aplicação principal (ver a figura 1.1)[1, 6, 7].

O glossário da CNET define como *"This term refers to a type of program that tightly integrates with a larger application to add a special capability to it. The larger app must be designed to accept plug-ins, and the software's maker usually publishes a design specification that enables people to write plug-ins for it*

[1].

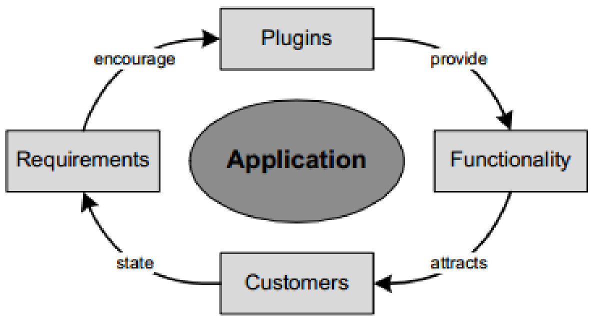


Fig. 1.1: Diagrama explicativo da arquitetura de um *software* que está preparado para receber *plugins* [1]

Esta metodologia pode ser utilizada quando [7]:

- Existe a necessidade de expansão durante a execução, mesmo em *star-up* ;
- Modularização de grandes sistemas para reduzir a complexidade;
- Desenvolvimento independente dos componentes do sistema sem modificar outros módulos ou reconstruir todo o sistema;

[1] *CNET Glossary* acedido em 29/08/2013

- Permitir o desenvolvimento por terceiros com base apenas no conhecimento das interfaces;
- Permitir a fácil implementação de novas funcionalidades e actualizações, após o envio do pedido;
- Curto tempo de arranque com requisitos de hardware baixos, especialmente de memória;
- Criar flexibilidade para servidores que executam software que não deve ser reiniciado frequentemente.

Em meados da década de 70, o editor de texto EDT$^{\text{TM}}$ foi uma das primeiras aplicações com uma arquitetura preparada para receber novos módulos. No final da década de 80 e início da década de 90 começaram a aparecer inúmeras aplicações flexíveis, recetivas a *plugins*, em alguns casos seguindo uma arquitetura aberta que permitia a utilização de *plugins* independentes das aplicações [1].

Atualmente, a extensibilidade através de *plugins* pode ser encontrada em muitas aplicações, tais como: *softwares* gráficos, clientes de *e-mail*, *browsers* da Internet, desenvolvimento de ambientes e *media players*, adicionando filtros, *codecs* ou *rendering engines*. São usados com o objectivo de inserir funcionalidades muito específicas num determinado ponto. A aplicação principal é também um *plugin* base que faz a gestão da estrutura que providencia a localização, o carregamento e a instanciação dos outros. A comunicação é feita através de pontos de extensão. Diferentes autores podem desenvolver *plugins* para a mesma aplicação, podendo não ter acesso ao código fonte de outros *plugins* ou mesmo da aplicação principal sem ser necessário uma compilação geral[6] .

Como exemplo de uma arquitetura estruturada com *plugins* existe a OSGi$^{\text{TM}}$ *Alliance* [2]. Esta organização permite a montagem modular de *software* construído com a tecnologia Java. Este conceito será abordado novamente no subcapítulo 3.2.

Em [6], o autor considera duas arquiteturas para uma aplicação: 1)*traditional plugins* e 2)*pure plugins* .

1) Na arquitetura *traditional plugins* existe a *host application*, à qual todos os *plugins* se podem ligar, contudo todos os *plugins* operam independentemente, como se pode ver esquemáticamente na figura 1.1.

2) Quando se trata de um sistema maior, a arquitetura deve apoiar a extensibilidade dos *plugins* de *plugins*. Na ausência de uma *host application*, *plugins* tornam-se

[2] *The OSGi Architecture* acedido em 06/05/2012

anfitriões de outros *plugins*, fornecendo pontos de extensão bem definidos. Na arquitectura *pure plugins*, todos os módulos são *plugins*. O papel da *host application* é reduzido a um mecanismo de *plugin runtime engine* ou *kernel*.

O OsiriX é uma aplicação com forte implantação no ambiente médico. O *plugin* desenvolvido permite integrar no OsiriX um filtro para medição do diâmetro da aorta baseado na ferramenta Fiji. O processamento desta artéria humana utilizou imagens de uma Angio Tomografia Computorizada Torácica e foi realizado num projeto paralelo no mesmo intervalo temporal e na mesma Empresa, *Efficientia - Manegement, Integration and Consulting* Lda.

Neste trabalho, na integração será utilizada uma arquitetura típica, em que haverá uma *host application* e um *plugin*.

1.2.2 Dispositivo Médico e Marcação CE

Pelo Decreto-Lei n.º 145/2009 de 17 de Junho define-se como **Dispositivo Médico** qualquer instrumento, aparelho, equipamento, *software*, material ou artigo utilizado isoladamente ou em combinação, incluindo o *software* destinado pelo seu fabricante, a ser utilizado especificamente para fins de diagnóstico ou terapêuticos e que seja necessário para o bom funcionamento do dispositivo médico, cujo principal efeito pretendido no corpo humano não seja alcançado por meios farmacológicos, imunológicos ou metabólicos, embora a sua função possa ser apoiada por esses meios, destinado pelo fabricante a ser utilizado em seres humanos para fins de:

- Diagnóstico, prevenção, controlo, tratamento ou atenuação de uma doença;
- Diagnóstico, controlo, tratamento, atenuação ou compensação de uma lesão ou de uma deficiência;
- Estudo, substituição ou alteração da anatomia ou de um processo fisiológico;
- Controlo da concepção [8].

Dispositivo médico activo é qualquer dispositivo médico cujo funcionamento depende de uma fonte de energia eléctrica, ou outra não gerada directamente pelo corpo humano ou pela gravidade, e que actua por conversão dessa energia, não sendo considerados como tal os dispositivos destinados a transmitir energia, substâncias ou outros elementos entre um dispositivo médico activo e o doente, sem qualquer modificação significativa e sendo que o *software*, por si só, é considerado um dispositivo médico activo [8].

Os dispositivos médicos estão divididos em quatro classes de risco (I, IIa, IIb e III) atendendo à vulnerabilidade do corpo humano e aos potenciais riscos decorrentes da concepção técnica e do fabrico [3]. As regras de classificação encontram-se no Anexo IX do mesmo Decreto-Lei e depende de quatro pontos fundamentais:

- A duração do contacto com o corpo humano (temporário, curto prazo e longo prazo);
- A invasibilidade do corpo humano;
- A anatomia afetada pela utilização;
- Os potenciais riscos decorrentes da concepção técnica e do fabrico [8].

O Decreto-Lei n.º 145/2009 de 17 de Junho, atualmente em vigor, transpõe para a ordem jurídica interna a Diretiva n.º 2007/47/CE, do Parlamento Europeu e do Conselho, de 5 de Setembro, que só é válida na Europa. Nos Estados Unidos, quem estabelece as diretivas é a organização *Food and Drug Administration* (FDA) que, face à diferente legislação, pode exigir aos fabricantes dos Estados-membros da Europa outros requisitos para além dos exigidos pela marcação CE. No entanto, existem outros países como Moçambique e Cabo Verde que reconhecem a marcação CE [9].

Devido à diferente legislação que é apresentada entre continentes, e mesmo dentro de continentes, foi criada a *Global Harmonization Task Force* (GHTF). O objetivo da GHTF é promover a convergência das práticas reguladoras relacionadas à garantia da segurança, eficácia/desempenho e qualidade de dispositivos médicos, promovendo a inovação tecnológica e facilitando o comércio internacional. Estes documentos fornecem um modelo para a regulação dos dispositivos médicos que podem então ser adotadas/implementadas pelas autoridades reguladoras nacionais. Esta organização possibilita também o intercâmbio de informações através do qual países com sistemas reguladores de produtos médicos em desenvolvimento podem beneficiar da experiência daqueles com os sistemas existentes e/ou práticas dos membros fundadores da GHTF [4].

A marcação CE é um pré-requisito para colocar no mercado e permitir a livre circulação dos dispositivos médicos no Espaço Económico Europeu (EEE), constituindo uma garantia de que estes produtos estão em conformidade com os requisitos

[3] *infarmed - Classificação de Dispositivos Médicos* acedido em 11/08/2012
[4] *Global Harmonization Task Force* acedido em 07/07/2012

essenciais que lhes são aplicáveis. Esta marcação tem um grafismo próprio e deve estar aposta pelo fabricante de forma legível, visível e indelével em todos os dispositivos médicos, exceto nos feitos por medida ou nos destinados a investigação clínica [8].

O dispositivo deve alcançar as caraterísticas e o desempenho indicados pelo fabricante. As caraterísticas e o desempenho não devem, portanto, sofrer alterações ao longo do seu ciclo de vida ao ponto de comprometer a segurança do doente [5].

1.3 Contribuições

Este projeto possibilita esquematizar e detalhar procedimentos mais adequados para inserção de módulos no Visualizador de Imagens Médicas OsiriX. Facilitará projetos futuros no mesmo visualizador e clarificar alguns conceitos de certificação e licenciamento de dispositivos médicos de classe IIa em Portugal.

1.4 Organização do Documento

A dissertação encontra-se organizada em seis capítulos: Introdução, Conceitos e tecnologias utilizadas, Visualizadores de Imagens Médicas, Integração de Funcionalidades no Visualizador OsiriX, Aposição da Marcação CE e, por fim, Conclusão.

No Capítulo 1 é enquadrado o trabalho, são expostos os objetivos que se pretendem alcançar com esta dissertação e a motivação para a realizar. As contribuições e colaborações explicam de que forma os procedimentos de organização e planeamento serão úteis futuramente e como foram alcançados.

Seguidamente, no Capítulo 2 é feito uma contextualização na Imagiologia abordando alguns conceitos e tecnologias utilizados. Aborda-se sobre o processamento de imagens digitais, as diferentes modalidades das Imagens Digitais, a norma DICOM, desenvolvimento em Fiji, a linguagem Objective-C e a tecnologia *Java Native Interface* (JNI).

No Capítulo 3 são apresentados e analisados os visualizadores de imagens médicas não comerciais e comerciais, destacando as funcionalidades e as mais valias de cada uma. As soluções selecionadas para inserção de um módulo serão apresentadas também neste capítulo com mais detalhe, sendo elas Weasis e OsiriX.

A integração das funcionalidades no OsiriX é apresentada na capítulo 4. É

[5] *infarmed - Perguntas Gerais de Dispositivos Médicos* acedido em 07/07/2012

explorada a ferramenta que se pretende inserir, Fiji, e são mencionadas duas formas de fazer a integração.

Todas as etapas da aposição a Marcação CE e obtenção da declaração de Conformidade do dispositivo médico de Classe IIa, num Organismo Certificado, são descritas no capítulo 5.

Finalmente, no capítulo 6, são apresentadas as conclusões finais da dissertação e apontadas direções a tomar para um trabalho futuro.

2

Conceitos e tecnologias utilizados

Neste capítulo são explicados alguns conceitos de imagiologia que serão úteis para a leitura deste documento.

2.1 Modalidades e Formatos Principais das Imagens Médicas

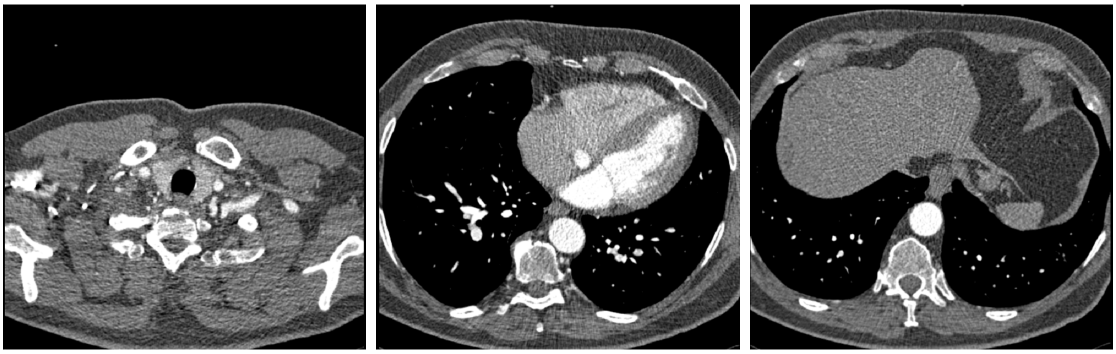
As modalidades médicas podem ser classificadas em duas categorias: anatómicas e funcionais.

As modalidades anatómicas são utilizadas para identificar morfologias, estando presentes neste grupo as modalidades Raio-X, Tomografia Computadorizada (TC), Ressonância Magnética Nuclear (RMN), Ultra-som (US) entre outras.

As modalidades funcionais são aquelas que têm o objetivo de adquirir informação a respeito do metabolismo relacionado a uma anatomia, incluindo Cintilografia, SPECT (*Single Photon Emission Computed Tomography*), PET (*Positron Emission Tomography*), modalidades de medicina nuclear e RMN funcional [10].

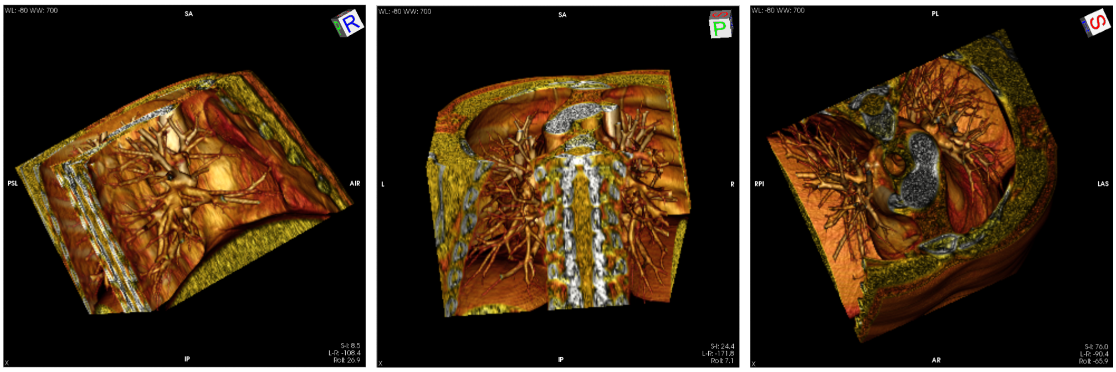
A finalidade das imagens médicas é ajudar o diagnóstico de anomalias e auxiliar, por vezes, tratamentos. Cada modalidade tem objetivos específicos e pode apresentar vantagens e desvantagens em comparação com as restantes. Num tratamento podem-se usar dois estudos do mesmo paciente mas de modalidades diferentes, uma modalidade permite completar as desvantagens da outra [11].

A perspetiva anatómica obtida através de qualquer modalidade referida anteriormente é bidimensional, isto é, são adquiridas inúmeras imagens de cortes/*slices* da parte anatómica do paciente em estudo. Num Raio-X é apenas uma imagem bidimensional, numa TC são inúmeras imagens [12]. Estas imagens ficam guardadas no



(a) (b) (c)

Fig. 2.1: Imagens bidimensionais de uma Tomografia Computadorizada

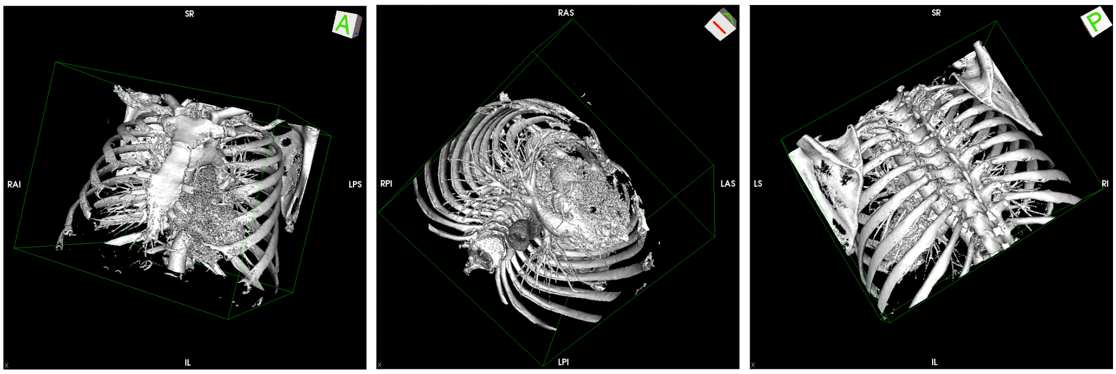


(a) (b) (c)

Fig. 2.2: Imagens tridimensionais do estudo apresentado na Figura 2.1, com aplicação do *Volume Rendering* do *software* OsiriX

formato DICOM [13]. Na figura 2.1 estão representadas três imagens bidimensionais de uma TC.

A visualização tridimensional é possível através de diferentes etapas. Na sobreposição as imagens são justapostas para formarem uma matriz tridimensional de *voxels* que são unidades cúbicas que compõe um objeto tridimensional. Na interpolação são preenchidos espaços vazios, sem informação útil, na matriz; na manipulação são feitas transformações geométricas, por exemplo rotação e finalmente a projeção que possibilita a visualização total do objeto [12, 14]. Nesta perspetiva tridimensional é possível ver o conjunto total dos dados através de um *Volume Rendering*, como se pode ver na figura 2.2 ou apenas uma superfície de um objeto através de *Surface Rendering*, imagem 2.3 [15]. O formato *standard* para imagens tridimensionais, na área médica, é o vtk, *Visualization Toolkit*.



(a) (b) (c)

Fig. 2.3: Imagens tridimensionais do estudo apresentado na Figura 2.1, com aplicação do *Surface Rendering* do *software* OsiriX

2.2 Processamento de Imagens Digitais

Como se viu anteriormente, a imagem digital é uma representação de um objeto físico, que pode ser armazenada, manipulada e interpretada [11]. Matematicamente, essa imagem define-se como um *array* bidimensional de valores que representam a distribuição da intensidade contínua de um sinal espacial. O sinal espacial contínuo é amostrado a intervalos regulares e as intensidades são quantificadas com um número finito de níveis.

A função

$$f\,(m, n) \tag{2.1}$$

traduz a intensidade de um *pixel*. Um *pixel* é a menor unidade de uma imagem e fica localizado na posição m e n ao longo de eixos ortagonais posicionados perpendiculares um ao outro. Na visualização, a imagem é constituída por M linhas e N colunas e tem P níveis de intensidade (níveis de cinza) e os valores variam entre 0 e P-1 [14]. O valor 0 representa o preto e o valor P-1 o valor branco.

O objetivo de definir matematicamente a imagem é a possibilidade de manipular o seu conteúdo a fim de transformá-lo e retirar as informações importantes. Ao vasto conjunto de operações que se pode aplicar numa imagem que está subdividida em *pixels*, ou seja, definido na forma de uma matriz, denomina-se processamento de imagem [11].

Quando se pretende alterar uma imagem, por exemplo a escala de cinzas que compõe a imagem, realçar ou suavizar caraterísticas, é feita uma convolução usando operadores locais designados *Kernels*. Se se considerar o *Kernel*

$$w\left(k,l\right) \tag{2.2}$$

como um *array* de (2K+1 x 2L+1) coeficientes, e o ponto (k,l)=(0,0) por o centro do *Kernel*, a convolução da imagem com o *Kernel* é definido pela seguinte equação:

$$g\left(m,n\right) = w\left(k,l\right) \times f\left(m,n\right) = \sum_{k=-K}^{K} \sum_{l=-L}^{L} w\left(k,l\right) \times f\left(m-k,n-l\right) \tag{2.3}$$

A função g(m,n) é o resultado da convolução, ou seja, é a imagem resultante da aplicação do *Kernel* w(k,l) à imagem $f(m,n)$. Para se obter o resultado, o *Kernel* fica centrado no *pixel* (m,n) e o produto ponto por ponto dos coeficientes do *Kernel* com os *pixels* correspondentes são obtidos. A soma destes produtos será o valor final daquele *pixel*. O resultado completo da imagem é obtido por sucessivas operações em todos os *pixels* da imagem original. O *Kernel* será dependente do objetivo presente para alterar a imagem [14].

O processamento de imagens médicas engloba três níveis diferentes: pré-processamento, processamento e análise. O pré-processamento permite remover dados indesejáveis com uma filtragem inicial, realçar dados importantes e fazer um pré-tratamento da imagem (*image enhancement*). O processamento possibilita identificar formas significativas de uma imagem e extraí-las do fundo, este processo é designado por segmentação. A análise é responsável pela parametrização e reconhecimento do objeto em estudo [11, 16].

2.3 Norma DICOM

A gestão da informação num hospital envolve a transmissão de imagens e dados em redes, implicando a integração de estações de visualização, bases de dados *on-line*, sistemas de gestão de imagens e redes locais permitindo que a informação circule e flua entre serviços e profissionais de saúde. A comunicação entre os diversos equipamentos e todo o *software* envolvido no processo exige uma normalização. Perante esta necessidade o *National Equipament Manufacturers Association* (NEMA) juntamente com o *American College of Radiology* (ACR) cooperaram e criaram um formato padrão, chamado ACR-NEMA (publicado em 1985), propondo aos fabricantes de equipamentos de imagem digital que todos os seus equipamentos e *softwares* o tivessem na sua base. Posteriormente, este evoluiu para a norma DICOM (publicado em 1993) e tornou-se amplamente aceite na área.

A norma DICOM facilitou a interoperabilidade entre dispositivos de declarem a conformidade com a norma[17].

Foi estruturado um documento *"The DICOM Standard"* com múltiplos capítulos, podendo ser consultado a cada atualização e descrevendo todas as potencionalidades desta norma. Atualmente apresenta 18 capítulos, tendo sido o último acrescentado no início deste ano [1] [17].

A norma é específica em diversos aspetos, tais como [18]:

- Para comunicações em Rede, os dispositivos devem seguir um conjunto de critérios declarando a conformidade com a norma;
- A semântica dos comandos e de dados associados devem seguir um padrão para que sejam percetíveis por todos os equipamentos envolvidos, e.g., uma informação enviada pode acionar a execução no dispositivo remoto de funções de receção que irão gerar o reenvio de informações para o dispositivo solicitante;
- A semântica dos serviços de arquivos, formatos de arquivos e informações necessárias são estabelecidas para poder haver uma comunicação *off-line*;
- A estrutura da norma DICOM está preparada para a introdução de novos serviços (*tags* proprietárias), permitindo a evolução dos equipamentos e das aplicações médicas futuras.

2.3.1 Comunicação utilizando o Norma DICOM

A figura 2.4 apresenta o modelo geral de comunicação do padrão que implementa, englobando a parte de rede ***(On-line Communication)*** e a parte de armazenamento ***(Off-line Communication)***. Assim, as implementações DICOM podem trabalhar com independência em relação à camada superior *(DICOM UPPER)*, que fornece a comunicação básica ***(DICOM Basic File Service)*** com a utilização do protocolo TCP/IP. Com isto, há a possibilidade de fazer o acesso aos meios de armazenamento de forma transparente.

Uma classe de informação do objeto ***(Information Object Class)*** descreve o objetivo e os atributos que definem uma dada classe de objetos. Uma classe de

[1] *The DICOM Standard* acedido em 16/07/2012

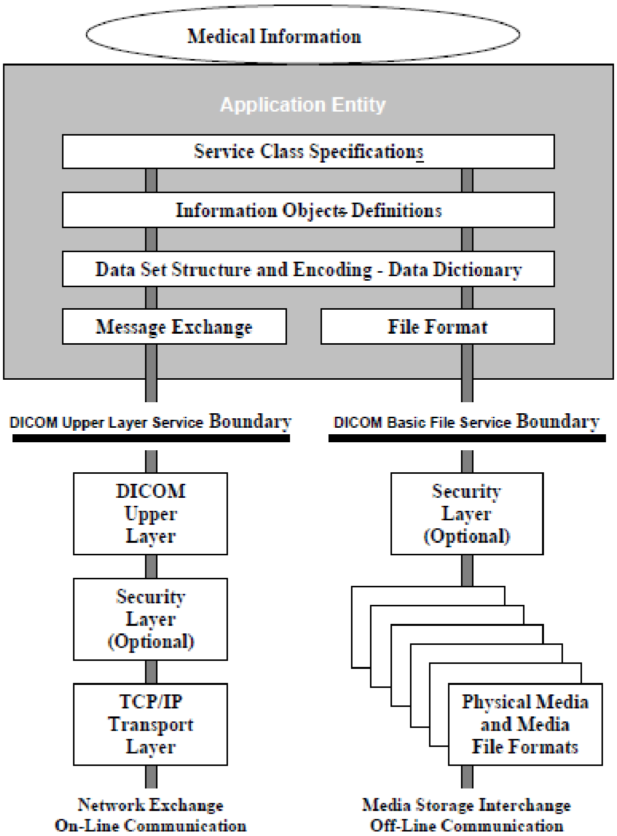


Fig. 2.4: Modelo de comunicação geral da norma DICOM [18]

informação do objeto não inclui os valores para os atributos que compreendem a sua definição. Existem dois tipos de classes de informação do objeto: normalizado e composto. As classes de informação do objeto normalizadas incluem somente aqueles atributos inerentes na entidade *realworld* representada. Por exemplo, a classe de informação do objeto do estudo, que é definida como normalizada, contém a data do estudo e a hora do estudo, isto por que são inerentes a um estudo real. O nome paciente, entretanto, não é um atributo da classe de informação do objeto estudo porque é inerente ao paciente em que o estudo foi executado e não ao próprio estudo.

A definição das informações do objeto (***Information Object Definitions***) especifica um número de classes de informação do objeto que fornecem uma definição abstrata das entidades do mundo real (*realworld*) aplicáveis a uma comunicação das imagens médicas digitais e da informação relacionada (relatórios estruturados, dose da terapia de radiação entre outros).

O serviço de especificação de classes (***Service Class Specifications***) define um número de classes de serviço. Uma classe de serviço associa uma ou mais informações do objeto com um ou mais comandos a serem executados (aplicados) nos respetivos objetos. As especificações da classe de serviço indicam exigências para elementos de comando e como os comandos resultantes são aplicados às informações dos objetos. A norma DICOM também define as caraterísticas compartilhadas por todas as classes de serviço, são elas:

Storage (armazenamento);

Query / Retrieve (localização / busca);

Worklist (lista de pacientes de trabalho);

Backup (cópia).

Para a especificação de classes são ainda definidas as operações e as notificações executadas sobre as informações dos objetos, comandos e protocolos.

A estrutura e semântica dos dados (***Data Structure and Semantics***) especifica como as aplicações DICOM constroem e codificam a informação da série de dados referentes às informações dos objetos e presta serviços de manutenção às classes.

O dicionário de dados (***Data Dictionary***) define o conjunto de todos os elementos dos dados DICOM disponíveis para representar a informação, isto é, estabelece elementos utilizados para os meios permutáveis que codificam as informações e uma lista das representações identificadas que são atribuídos pelo DICOM. Para cada elemento, é especificado: *Tag*, que consiste em um grupo e um número para o elemento - exemplos, nome (nome do paciente), tipo (carateres, inteiro ou outro) e multiplicidade (quantos valores por atributo).

A troca de mensagens (***Message Exchange***) especifica o serviço e o protocolo usados por uma aplicação em um ambiente médico para trocar mensagens sobre os serviços de comunicação definidos e suportados pela norma DICOM. Uma mensagem é composta por uma linha de definição seguida por uma outra linha de dados opcional. Assim, pode-se definir que:

- as operações e as notificações estarão disponíveis para prestar serviços de manutenção às classes definidas;

- as regras para estabelecer e terminar associações fornecerão suporte a comunicação;

- será possível controlar as transações de solicitação e resposta;

- o controle das regras necessárias para transferir as mensagens (*streams*) será realizado.

O suporte à comunicação em rede para troca de mensagens (**Network Communication Support for Message Exchange**) descreve os serviços de comunicação e os protocolos das camadas superiores necessários para suportar, em um ambiente de rede distribuído, a comunicação entre aplicações DICOM.

Estes serviços e protocolos de comunicação asseguram-se de que uma comunicação entre aplicações DICOM seja executada de forma eficiente e coordenada através da rede. Os serviços de comunicação especificados são um subconjunto apropriado dos serviços fornecidos pela especificação OSI (ISO 8822) e do serviço de controle de associação da OSI (ACSE) (ISO 8649).

Os serviços da camada superior permitem que as aplicações estabeleçam conexões, transfiram mensagens e terminem conexões. A camada superior do DICOM é, portanto, utilizado conjuntamente com protocolos do transporte do TCP/IP, que é o principal protocolo de comunicação utilizado em redes locais e na internet.

A forma para armazenamento e formatos de arquivos (***Media Storage and File Format***) especifica o armazenamento de imagens médicas em meios removíveis. A finalidade desta parte é fornecer uma estrutura que permita a compatibilidade das imagens e das respetivas informações associadas com uma grande variedade de formatos de arquivos e meios de armazenamento físicos.

2.3.2 Estrutura da Imagem

No padrão DICOM existem 27 tipos de dados designados por *Value Representation*(VR). Os diferentes tipos de dados permitem encapsular e representar qualquer tipo de dado médico possível, e.g., dados sobre o paciente e o equipamento com que foi feito o exame. A tabela 2.1 apresenta alguns exemplos de tipos de dados e as suas caraterísticas.

Uma imagem médica no formato DICOM tem de seguir uma estrutura padrão para que todos os visualizadores preparados para receber este formato a possam abrir sem qualquer erro. Pode ser constituída por um *header*, opcional, e *data elements*, como se vê de forma esquemática na figura 2.5 [19].

Tab. 2.1: Exemplo de VR

VR	Definição	Tipo de Carateres	Tamanho
AE Apllication Entity	Carateres que definem a entidade de aplicação	Alfanumérico	16 bytes máximo
DT	Data e Hora	'0' - '9', '.'	26 bytes máximo
OW	Texto cuja codificação é especificada na sintaxe de transferência	N/A	Depende da sintaxe de transferência
UID	Identificador Único	'0' - '9'	64 Bytes máximo

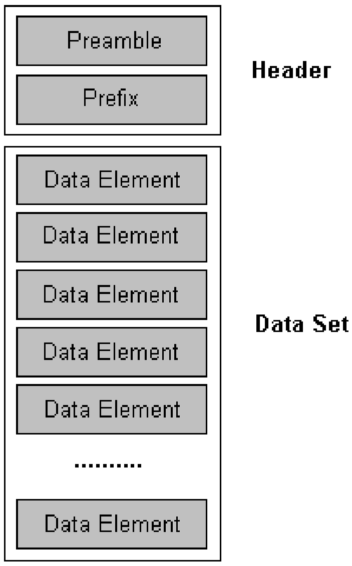


Fig. 2.5: Estrutura Básica de um ficheiro DICOM [13]

Header

O *Header* é constituído por um preâmbulo de 128 *bytes*, utilizado apenas para informações adicionais quanto à compatibilidade dos equipamentos, tamanho da imagem, o número de *bits* por *pixel*, a distância entre cortes do exame e o tipo de equipamento. Caso não seja usado deve ser preenchido com 00H e, nesse caso, pode ser ignorado. É através do *Header* que é possível aceder a uma imagem num repositório.

Após o preâmbulo existe um campo de 4 *bytes* chamado prefixo. Este campo deve conter os caracteres 'DICM', em letras maiúsculas, para indicar o formato do arquivo [19].

O *header* termina com a *tag* (7FE0, 0010), que indica o início das imagens

Tab. 2.2: Exemplo de um *Data Set*

Etiqueta		VR		Tamanho	Valor
Número do Grupo	Número do Elemento	Caracter de 2 bytes estabelecendo a representação do valor (tipo do dado)	Reservado	Tamanho em bytes do campo do valor	
2bytes	2bytes	2bytes	2bytes	4bytes	variável

codificadas.

Data Elements

No *Data Set* os dados da imagem são divididos em *Data Elements*, como se pode ver na imagem 2.5.

Um *Data Set* contém 3 ou 4 campos, como se pode ver na tabela 2.2. A *tag* ou Etiqueta, o VR, o tamanho dos dados e o valor da variável, sendo que o campo VR pode ficar implícito em algumas implementações. A *tag* identifica quais os dados que estarão presentes e a variável será o dado. O número de *tags* pode ser variável.

Cada *tag* é dividida em duas partes, e.g., *tag* (0008, 0008). Os primeiros 2 *bytes* hexadecimais representam o grupo, onde serão agrupados os dados semelhantes. Os dois *bytes* posteriores representam o elemento. A *tag* em questão representa o tipo de imagem, *ImageType*, referente à codificação do *Data set*.

Todas são informações que podem ajudar na representação da imagem, após a última *tag* do *header* e o campo de dados, que indica onde se encontram os dados dos *pixels*, devem ser processadas de acordo com a *tag ImageType*.

A *tag ImageType* indica se a imagem está codificada com compressão, nos formatos JPEG, JPEG2000, JPEG-LS, compressão RLE (*Run-Lenght Encoding*) e ZIP, ou sem compressão, nos formatos TIFF e BMP. Pode também aparecer no formato de vídeo AVI, variando de acordo com a qualidade de imagem desejada pelo utilizador [19].

Na tabela 2.3 são apresentados os campos de algumas *tags*, seguidas pelos seus títulos e descrições das mesmas.

Um corte normalmente ocupa cerca de 524 KB. Considerando que um exame TC possui 500 cortes, a totalidade do exame ocupa cerca de 250 MB. Se no *data elements* consistir uma reconstrução 3D ou mamografia, os requisitos de armazenamento são

Tab. 2.3: Exemplo de uma etiqueta

Grupo	**Elemento**	**Título**	**Descrição**
0002	0010	Transfer Index UID	Define a sintaxe de transferência
0002	0016	AE Fonte	Título de entidade de aplicação
0008	0016	SOP Class UID	Define a classe par serviço-objeto(SOP)
0008	0060	Modalidade	Modalidade de Imagem(MR,CT,RX)
0008	0070	Fabricante	GE,Toshiba,Siemens
0018	0050	Espessura do corte	Parâmetro de aquisição da imagem
7FE0	0010	Pixel Data	Imagem

ainda maiores.

2.3.3 WADO

Atualmente usam-se sistemas de armazenamento/repositório de objetos médicos, como o MCM - *Medical Content Manager* - ou o PACS - *Picture Archiving and Communication System*. Quando um sistema de informação médica pretende procurar num repositório objetos DICOM armazenados anteriormente, é utilizado o WADO - *Web Access to DICOM Objects*. O WADO é um serviço baseado na Internet que faz parte da norma DICOM e vem especificado no capítulo 18 do mesmo.

Este serviço acede a um repositório DICOM através dos protocolos HTTP/HTTPS e usa o DICOM *Unique Identifier* (UID) para identificação de estudos, séries e instâncias [2].

O *Web client*, através de uma conexão HTTP ou HTTPS, constrói o URL para enviar o pedido com o formato estabelecido pelo RFC2396(URI):

<scheme>://<authority><path>?<query>

No <*path*> terão de ir os seguintes parâmetros:

study*UID,*series*UID,*object*UID** [&frameNumber***]

O conjunto de parâmetros transmite ao servidor qual o objeto DICOM que está a ser solicitado e o formato em que deve ser devolvido para o *Web Client*, como se pode verificar na imagem 2.6. O servidor tem muitos recursos que podem ser

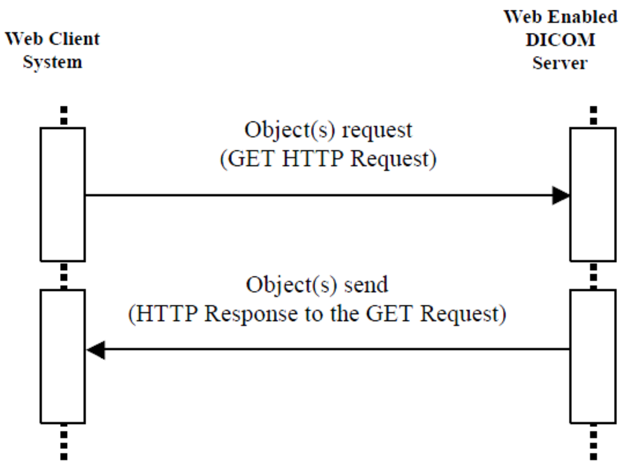


Fig. 2.6: Diagrama de interação [2]

utilizados pelo cliente ao enviar diferentes *querys*.

O servidor recebe o pedido e começa a processar a solicitação. O processo inclui a recuperação do objeto desejado DICOM a partir do repositório e a construção da resposta desejada, podendo ter de extrair alguns dados e enviar a resposta para o cliente.

Com este serviço os visualizadores de imagens médicas podem aceder a imagens num repositório PACS, desde que o PACS suporte o WADO. Como exemplo prático, neste documento será apresentado o visualizador Weasis que utiliza o WADO para se ligar a um PACS através da Internet. O Weasis será explorado no sub-capítulo 3.2.

2.4 Desenvolvimento em Fiji

A linguagem utilizada para o desenvolvimento do filtro em [4] foi o Python, tendo sido utilizados vários filtros do Fiji.

Este *software* é uma distribuição *Open-source* do ImageJ (o software será analisado no capítulo 3) porém possui uma maior quantidade de bibliotecas e *plugins* relevantes para análise biológica. Facilita a prototipagem rápida de *type-agnostic* e algoritmos genéticos. Permite as linguagens de *script* Jython, Clojure, JavaScript, JRuby e BeanShell e são disponibilizados tutoriais explicativos no *site* da aplicação e documentação de suporte.

É possível fazer a ligação do Fiji com o MatLab através do *Plugin* Miji e também com o itk [20].

O *software* encontra-se licenciado pela *General Public Licence* mas os *Plugins* podem ser distribuídos utilizando outra licença, por exemplo, *Apache License*, *Public Domain* ou *Lesser General Public License* entre outras.

Apresenta uma interface simples, como se pode ver na figura 2.7, todavia os seus sub-menus *Process*, *Analyze* e *Plugins* aparesentam dezenas de funcionalidades, *macros* e *Plugins*.

Fig. 2.7: Interface do Fiji

Uma mais valia deste *software* é ser multi-plataforma, podendo assim ser utilizado quer em sistemas Microsoft Windows como Mac OS X. Os ficheiros correspondentes podem ser descarregados na *site* da aplicação [2].

A combinação de avançadas soluções de análise de imagens e a simplicidade de manuseamento tem atraído muitos desenvolvedores a contribuírem para a aplicação Fiji. O projeto Fiji tem aproximadamente dois anos, contudo, é um *software* de processamento muito completo.

2.5 Objective-C

Objective-C é uma linguagem simples de programação orientada ao objeto que corresponde a um superconjunto do ANSI-C. Quando comparada à linguagem C, possui alguns termos e construções adicionais, todavia a utilização de classes e a passagem de mensagens é feito de um modo similar ao *Smalltalk*, uma das primeiras linguagens de programação orientada ao objeto [21].

As vantagens do Objective-C incluem a possibilidade de se carregar as definições das classes e métodos em tempo de execução, os objetos serem instanciados dinamicamente, a possibilidade de se utilizar objetos remotos, a persistência e a existência de protocolos de delegação. Alguns de seus principais problemas são a inexistência de herança múltipla e a inexistência de variáveis de classe [3].

[2] *Fiji Is Just ImageJ* acedido em 11/08/2012
[3] *Mac Developer Library* acedido em 11/08/2012

2.6 JNI - *Java Native Interface*

O *Java Native Interface* é um mecanismo de comunicação de baixo-nível da plataforma Java, que permite a interligação entre Java e código nativo. Este mecanismo invoca uma interface para realizar a comunicação com uma *Java Virtual Machine* (JVM) desde a aplicação nativa. Quando se utiliza o JNI é possível escrever partes da aplicação em código nativo e aceder diretamente do Java ou executar o processo inverso, aceder funções Java diretamente do código nativo.

O JNI fornece um conjunto de funções C ou C++ capazes de interagir com objetos Java. Estas funções são compostas por funções genéricas e outras específicas do programa que está a ser desenvolvido e que são utilizadas pelo código nativo, isto é C ou C++, para possibilitar a interação com o código escrito em Java. Os componentes nativos são compilados de modo que se crie uma biblioteca dinâmica que será importada pela classe Java que estabelece a interface com o código nativo. Desta forma o código escrito em Java poderá interagir com o código escrito em C ou C++.

Do lado Java, passam-se objetos como parâmetros para a função nativa que, por sua vez, pode alterar dados destes objetos, executar os métodos e retornar valores que também podem ser objetos Java. Desta forma, é transparente para o programa que a função chamada é nativa e não Java.

Do lado C ou C++ deve-se implementar as funções a serem utilizadas em Java seguindo a convenção especifica pelo JNI para a escolha dos métodos das funções. Os componentes C ou C++, uma vez em execução, podem utilizar quaisquer recursos, como se fossem um programa *stand-alone* redigido nestas linguagens [22, 23].

O JNI foi utilizado neste trabalho para aceder aos blocos de memória partilhada. Por conseguinte, apenas serão integrados métodos nativos na classe Java, como será explicado na secção 4.4.2.

3

Visualizadores de Imagens Médicas

Atualmente os médicos especialistas em imagiologia recorrem a aplicações para poderem analisar imagens médicas e mais facilmente fazer um diagnóstico de um paciente.

Um visualizador é um *software* que permite a visualização e o processamento de imagens médicas num computador convencional.

3.1 Análise das Soluções Existentes

Entre dezenas de soluções existentes para apoio à visualização e ao diagnóstico foi selecionado um pequeno conjunto para análise tendo em atenção fundamentalmente três factores: ser *Software* Não-Comercial ou Comercial, ser amplamente usado no meio médico/académico e apresentar ferramentas/funcionalidades relevantes. Os visualizadores selecionados são apresentados e descritos seguidamente.

3.1.1 ImageJ

O ImageJ é uma aplicação de processamento e análise de imagem *Open-Source* desenvolvida em linguagem de programação Java. O seu autor é Wayne Rasband e esta ferramenta foi desenvolvida tendo por base o programa NIH Image para *Macintosh*.

Este *software* permite a sua instalação em Sistemas Operativos tais como Windows, Mac OS X e Linux. Corre como uma *online applet* ou como uma aplicação com uma *Java Virtual Machine* 1.5 ou superior.

É possível com este *software* abrir imagens com o formato TIFF, GIF, JPEG, PNG, BMP, DICOM, PGM e FITS, para outros formatos é necessário instalar *plu-*

gins adicionais. Permite visualizar, editar, analisar, processar, guardar e imprimir imagens de 8-bit, 16-bit e 32-bit.

A arquitetura do ImageJ permite carregar módulos de código em *macros*, *plugins* e *scripts*. Para suportar outras linguagens tais como BeanShell, Clojure, Python ou Ruby usa-se o Fiji [24].

O Fiji juntamente com o MBF ImageJ são atualizações do ImageJ, apresentando *plugins* e *macros* mais específicos. O MBF é vocacionado para a microscopia e o Fiji para análise biológica [25, 26]. O Fiji é mencionado no sub-capítulo 2.4.

Como se pode ver pela figura 3.1, a sua interface é pequena e quando se abre um estudo ou se faz um processamento as imagens surgem em novas janelas. Os menus apresentam muitas funcionalidades.

Suporta funções de processamento como manipulação de contraste, *sharpening*, *smoothing*, *edge detection* e *median filtering*. Pode criar histogramas e *plot profiles*, que são gráficos que relacionam a intensidade dos *pixels* com a frequência que a mesma intensidade aparece numa imagem[25].

Nesta aplicação é possível visualizar informação referente à imagem através de um *plugin*, mas não é possível ver uma *Local Database* com os estudos organizados.

A versão analisada do ImageJ foi a 1.46.

3.1.2 DicomWorks

O DicomWorks é um *software* livre desenvolvido por Philippe Puech e Loïc Boussel. Este visualizador permite abrir imagens DICOM, aplicar alguns filtros, fazer medições, adicionar informações de diagnóstico e posteriormente guardá-las com um formato comprimido. Como suporta os protocolos DICOM e FTP permite transferir imagens através da Internet.

Este *software* foi desenvolvido em C++, não suporta a inserção de *plugins* e é direcionado apenas para a plataforma Windows [27].

A sua interface é simples e estruturada como se verifica na figura 3.2.

A versão analisada foi a 1.3.5.

3.1.3 MicroView

O visualizador MicroView é *Open-Source* desenvolvido em C++ e Python e é distribuído pela Empresa GE Health Care diferenciando-se de outras aplicações por

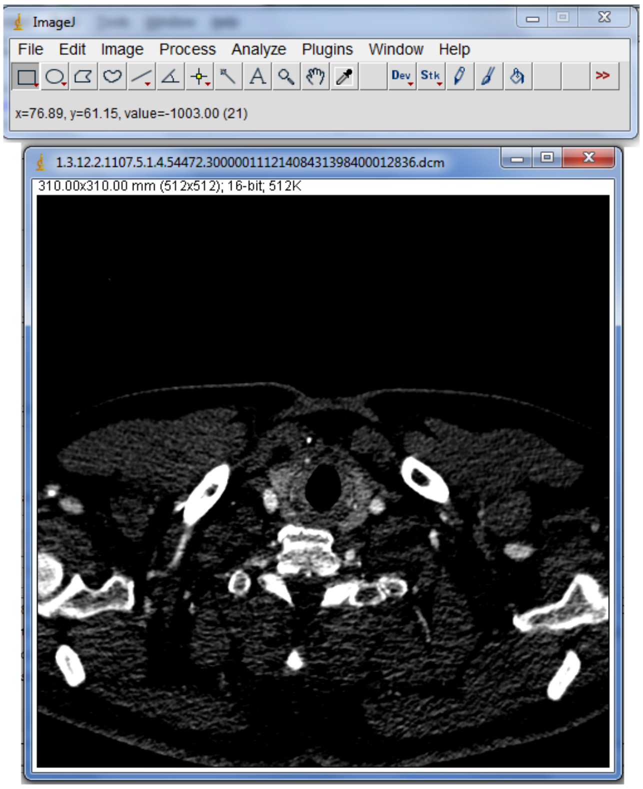


Fig. 3.1: ImageJ com uma imagem DICOM de um Angio-TC

ser muito completo.

Permite carregar imagens com os formatos: DICOM, Raw, Analyze, VTK, HFH, MINC, JPEG, TIFF, BMP e PNG, entre outros. Além da visualização bidimensional ou tridimensional, apresenta melhor perceção dos planos da imagem como se pode ver na figura 3.3. Esta funcionalidade é possível sem recorrer a *plugins* adicionais, ao contrário de outros visualizadores. Permite isolar e guardar uma *Region of Interest*, ROI, de uma imagem e as bibliotecas usadas para tratamento da mesma são o vtk e o itk, também estas *Open-Source*. As medições feitas ou os parâmetros que foram inseridos aquando de um diagnóstico podem ser formatados para uso em folhas de cálculo e *software* de análise de dados. Não é possível na sua interface observar uma árvore organizada com os Estudos e Séries de imagens nem informações do paciente ou do exame, como acontece no ImageJ e no DicomWorks [1] [2].

A versão analisada do MicroView foi a 2.1.2 e é possível instalá-la em Windows, Mac OS X e Linux.

[1] *MicroView 3D Image Viewer and Analysis Tool* acedido em 11/08/2012
[2] *GE MicroView Training* acedido em 11/08/2012

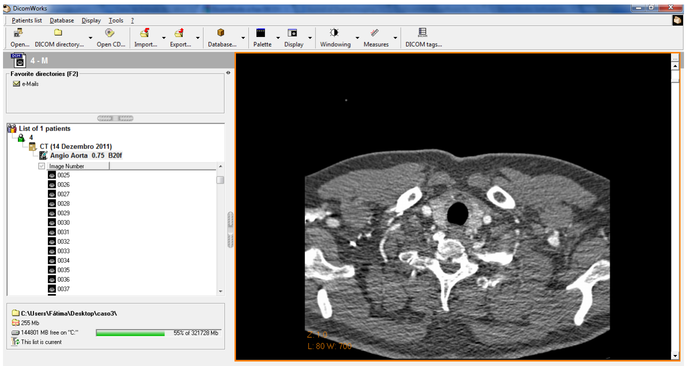


Fig. 3.2: Interface do DicomWorks permitindo visualizar uma imagem DICOM de um Angio-TC. É possível verificar como os Estudos ficam organizados do lado esquerdo da tela

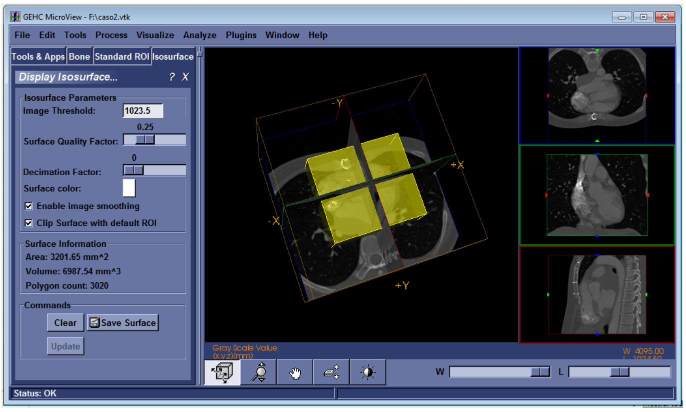


Fig. 3.3: Interface do MicroView, é apresentada uma ROI no centro da imagem delineada a amarelo e no lado direito são apresentados os cortes axial, coronal e sagital

3.1.4 ParaView

O ParaView é um *software* multi-plataforma de visualização e análise podendo ser instalado em Windows, Mac OS X e Linux. O seu projeto foi iniciado em 2000, no entanto só foi tornado público no final de 2002 pelo trabalho conjunto da Kitware Inc. e Los Alamos National Laboratory.

A base de código em Python do Paraview é concebida de tal maneira que todos os seus componentes podem ser reutilizados para desenvolver rapidamente aplicações verticais. Esta flexibilidade permite que os desenvolvedores do Paraview rapidamente projetem aplicativos que possuam uma funcionalidade específica. A interface XML permite que os desenvolvedores adicionem os seus próprios filtros vtk sem escrever qualquer código especial e/ou recompilação.

A sua interface (figura 3.4) e as valências são análogas às do MicroView mas a aplicação de filtros ou de funcionalidades é feito através de uma *pipeline*. Uma *pipeline* possibilita ver todas as fases do processo, por exemplo obter as estruturas moles e as duras em diferentes cores, sendo estas selecionadas pelo utilizador, ou aplicando diferentes valores para a *isosurface* nunca perdendo o resultado anterior.

Refira-se que é possível abrir ficheiros com o formato vtk, EnSight 6 e EnSight Gold e Plot3d entre outros [3].

A versão analisada do ParaView foi a 3.12.0.

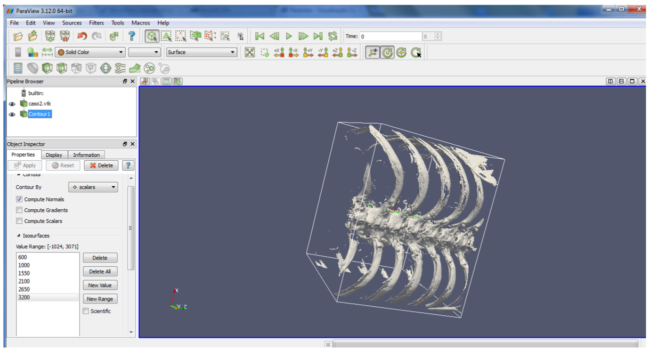


Fig. 3.4: Visualização dos tecidos duros - grade costal - de uma imagem Angio-TC, do lado esquerdo da tela é possível verificar a pipeline e os valores da *isosurface*, Interface do Paraview

3.1.5 MicroDicom

A MicroDicom é um visualizador de imagens médicas livre pouco conhecido e com funcionalidades limitadas. Apresenta as ferramentas básicas de manipulação de imagens DICOM como *Zoom*, *Pan*, Rotação entre outras e foi desenvolvido unicamente para Windows. A interface é simples e intuitiva, como se vê na imagem 3.5, assim como sua utilização, porém os seus recursos visuais são escassos, apresentando somente a visualização 2D de imagens individuais ou em *slices* e uma lista de estudos organizados do lado esquerdo da tela. Permite abrir arquivos de imagens em formato Dicom e JPEG, GIF e BMP permitindo guardá-los como ficheiros de vídeo com formato avi. O seu autor foi Simeon Stoykov e as linguagens de programação utilizadas foram C e C++ [4].

A versão analisada do MicroDicom foi a 0.5.4.

[3] *ParaView Users Guide* acedido em 11/08/2012
[4] *MicroDicom - free DICOM viewer for Windows* acedido em 11/08/2012

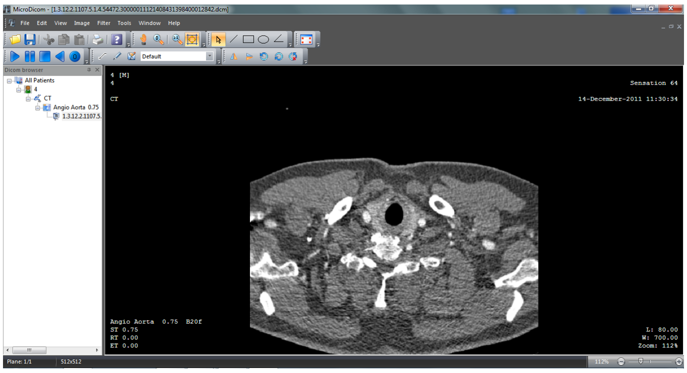


Fig. 3.5: Interface do *software* bastante simples, organizada e intuitiva

3.1.6 iQ-View

O iQ-View é uma aplicação preparada para trabalhar em Windows e com uma grande quantidade de estudos. Este visualizador apresenta duas versões diferentes: Básico, para médicos e técnicos de saúde, e PRO, especialmente para radiologistas.

Permite o processamento e a análise de uma imagem recorrendo a funcionalidades base, tal como qualquer outro visualizador. No entanto, as suas funcionalidades não são práticas pois só algumas são apresentadas numa barra sobreposta à imagem e outras em menus como se pode ver na figura 3.6. Possibilita importar imagens com o formato DICOM, BMP, JPEG, TIFF e RAW e permite também o utilizador optar por uma interface em Português.

Como *software* comercial, possui uma versão disponível por 30 dias para possíveis testes e é comercializado pela empresa IMAGE Information Systems Ltd [5].

A versão analisada foi a 2.8.0.

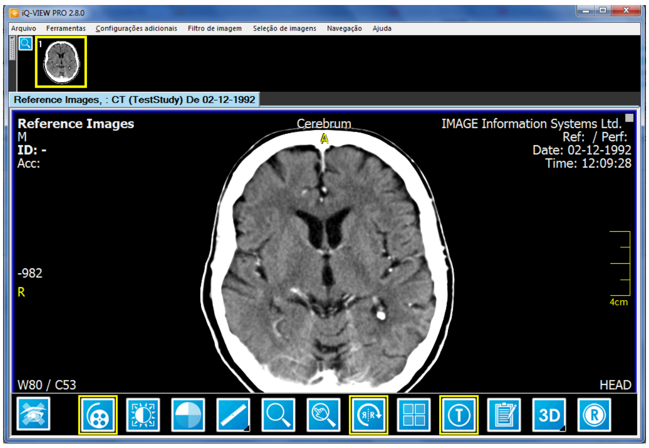


Fig. 3.6: Interface do visualizador iQ-View

[5] IMAGE Information Systems Ltd. acedido em 11/08/2012

3.1.7 eFilm

O eFilm é um visualizador de imagens médicas cuja instalação é possível em Windows, no entanto há também uma versão para *iPad* e *iPhone*. Não é aprovado para imagens mamográficas. É o visualizador com a barra de ferramentas mais completa e personalizável (ver imagem 3.7) e permite configurar combinações de teclas para executar funções da barra de ferramentas. Está preparado para receber ficheiros DICOM, JPEG e TIFF mas permite fazer a função *Rendering* permitindo ver a imagem como volume e não em *slices*.

A versão analisada do eFilm-Workstation foi a 3.4 e é disponibilizada por 30 dias, contudo, há uma licença académica anual por um valor muito mais baixo que a versão clínica [6].

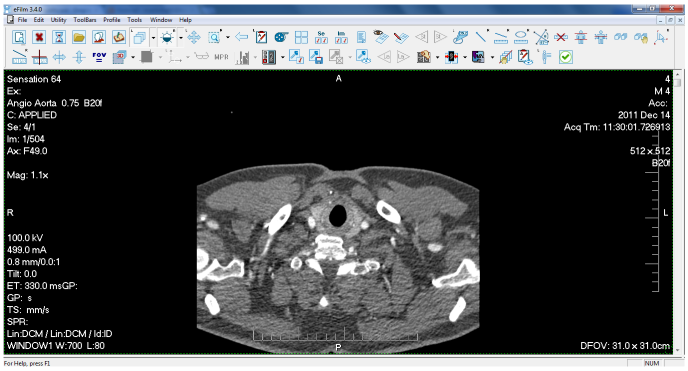


Fig. 3.7: Interface muito completa do eFilm

3.1.8 Dicom Viewer

O Dicom Viewer é um *software* desenvolvido em C++ preparado somente para Windows e cujo autor é Rutger den Boer. Possui uma versão disponível por 60 dias para possíveis testes e é comercializado pela empresa Rubo Medical [7].

O visualizador possui uma interface simples e as suas ferramentas encontram-se no menu superior, não sendo muito acessível, como se verifica imagem 3.8.

A versão analisada do Dicom Viewer foi a 2.0. A licença por um ano é de 750Euros e se for para acesso a 5 utilizadores poderá custar 3000Euro [8].

[6] *User's Guide eFilm Workstation v.3.4* acedido em 11/08/2012
[7] *Rubo Dicom Viewer Help* acedido em 11/08/2012
[8] *Rubo Dicom Viewer* acedido em 11/08/2012

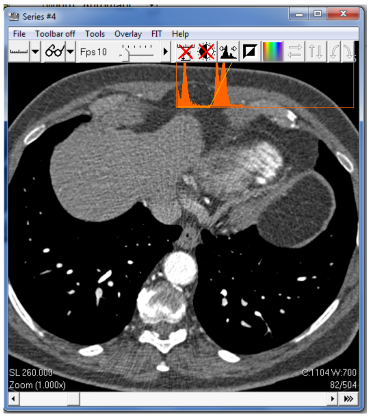


Fig. 3.8: Interface simples e pouco intuitiva

3.2 Weasis

Com a necessidade de distribuir e visualizar imagens radiológicas dentro e fora de um hospital, juntamente com novas tecnologias e protocolos Web foi criado este visualizador multiplataforma.

O projeto do Weasis iniciou-se em 2009 por Nicolas Roduit no Hospital Universitário em Genebra, tendo por base o *software* OsiriX. Neste projeto foi utilizada a linguagem Java, um conjunto de aplicativos e utilidades *Open-Source* em Java para a área médica designadas por *dcm4che* [9] e bibliotecas *Java Advanced Imaging* [10].

A sua interface é simples e organizada, como se pode ver na imagem 3.9. Apresenta ferramentas no lado direito e permite aplicar poucos filtros. O *software* está preparado para a visualização de imagens DICOM, sendo no entanto necessário inserir um módulo adicional para visualização de imagens 3D. Permite também a visualização de vídeo (DICOM Mpeg-2) [28].

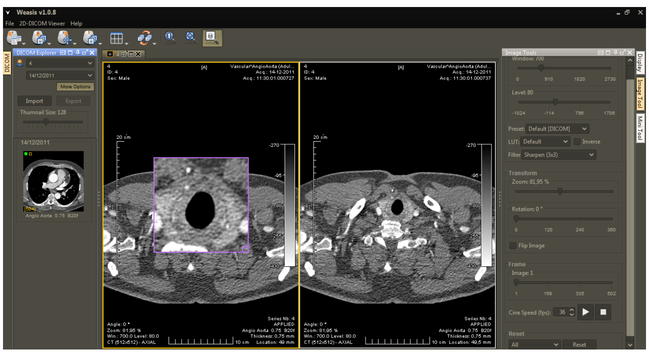


Fig. 3.9: Interface do Weasis

[9] *dcm4che.org - Open Source Clinical Image and Object Management* acedido em 11/08/2012
[10] *Java Media APIs* acedido em 11/08/2012

Os visualizadores que permitem comunicar através do protocolo WADO e sejam *Open-Source* são muito poucos. Este *software* é o primeiro com estas caterísticas e a ser distribuído gratuitamente.

O Weasis pode ser facilmente interligado com um PACS que suporte o protocolo Wado através de um portal Web ou como um consumidor XDS-I num ambiente IHE - *Integrating the Healthcare Enterprise* [3, 28] .

Um serviço XDS-I é uma solução prática para a transferência de imagens DICOM. Diferencia-se do XDS pois as imagens não são guardadas num repositório. Uma imagem DICOM ocupa uma grande largura de banda, só podendo ser guardada no repositório XDS a última imagem requisitada. Por exemplo, um utilizador da Internet que requisite uma imagem, estará e enviar um pedido através da DMZ e a imagem será requerida ao PACS na Intranet do hospital, esta será guardada no repositório até ser requisitada outra novamente [29].

O Weasis permite então visualizar as imagens que estão armazenadas no PACS, tanto numa rede local como na rede global, como se pode ver na imagem 3.10.

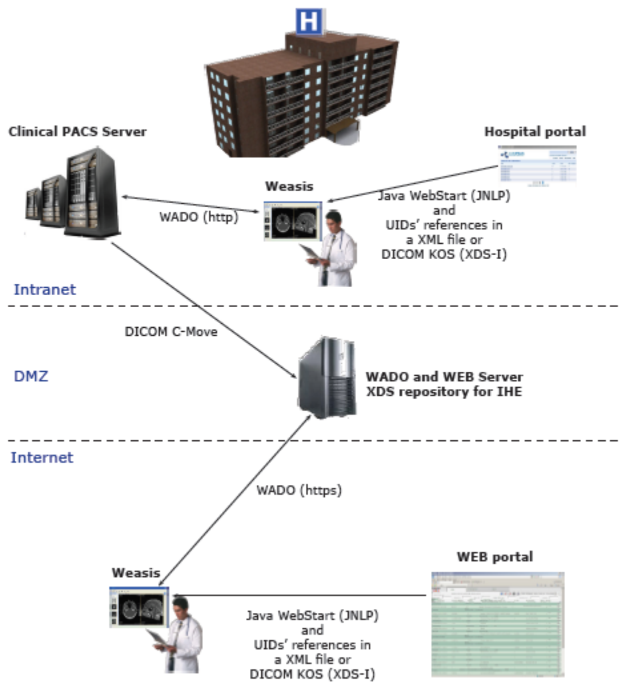


Fig. 3.10: Esquema explicativo da troca de imagens através de uma Intranet e da Internet [3]

O *Java Web Start* (JWS), é instalado automaticamente com as versões atuais do JRE. Quando um utilizador clica numa ligação para um arquivo de lançamento

(*Java Network Lauching Protocol*, JNLP), isto faz com que o *browser* inicie o JWS que se descarrega automaticamente, armazena-se em memória e execute a aplicação baseada em Java. Todo o processo é automático sem a necessidade de interação com o utilizador, com a exceção do clique inicial. As aplicações lançadas com JWS são armazenadas localmente, uma aplicação já descarregada é lançada como uma aplicação que tenha sido instalada anteriormente [28].

3.2.1 Licenciamento

O Weasis é distribuído pela Eclipse Public License e aprovada pela Eclipse Foundation [3].

Neste contexto, *Program* são as contribuições distribuídas em conformidade com esta licença, *Contributor* é a pessoa ou a entidade que distribui o *Program* e *Recipient* qualquer utilizador que pretende usufruir do *Program*. Esta licença permite o *Contributor* fazer a distribuição de uma aplicação com o código de desenvolvimento disponível através de um repositório e informar como se deve obtê-lo e fazer a sua distribuição também numa versão já compilada.

A mais valia desta licença é o livre intercâmbio de partes de código para aplicações similares e permitir que novos *Recipients* criem e anexem *plugins*.

Caso o *Recipent* queira comercializar ou distribuir outra versão do *Program* deve obter outra licença sobre a patente, explicar as novas modificações e será considerado o *Commercial Contributor*. Todas as responsabilidades de manutenção serão do novo *Contributor* [11].

3.2.2 Arquitetura do *Software*

A tecnologia OSGI é um conjunto de especificações para que desenvolvedores Java sigam um modelo de desenvolvimento. Permitem que as aplicações sejam dinâmicas/modulares e que componentos do mesmo *software* "escondam"as suas implementações durante a comunicação através de serviços. A estrutura do código é mais organizada, permite a reutilização de partes, os *bugs* são detetados mais facilmente e o desenvolvimento das aplicações pode ser feito em Eclipse e em Spring [12].

A arquitetura do Weasis está desenvolvida sob o *framework*, nomeadamente com o Apache Felix, que implementa a tecnologia OSGI Service Platform R4 sob a licença Apache. O projeto fica dividivo em sub-projetos e cada um tem um especificação

[11] *Open Source Initiative - Eclipse Public License 1.0* acedido em 11/08/2012
[12] *OSGI Alliance* acedido em 11/08/2012

relacionada, por exemplo, HTTP Service permite as comunicações em rede e o *Maven Bundle Plugin* permite construir um *plugin Maven*, simplificando a forma de o inserir no *software* geral [13]. O *Plugin Maven* está no centro do *isoftware* e é responsável pela gestão de todas as ferramentas utilizadas e pela execução do *framework*[14].

O *download* do código fonte pode ser feito através do repositório *dcm4che.org* [15].

O código fonte está organizado em 9 pastas: *base*, *core*, *dicom*, *distributions*, *framework*, *imageio*, *imageioext*, *weasis-launcher* e *weasis-parent*.

O autor para elaboração do programa utilizou diferentes *packages* e classes. Para criação da interface e para manipulação de gráficos e imagens foi utilizado o *package java.awt* que possui variadas classes para controlar toda a visível do programa. O *package java.imageio* permite a manipulação de imagens como objetos, controlando o processo de leitura da imagem e processar a imagem escrita. As classes mais importantes neste grupo são a *ImageIO*, *ImageReader*, *Imagewritter* e *IIOImage*. O *package org.weasis* possui classes com funções específicas para o Weasis, e o *package org.dcm4che* contém classes de nível baixo que lidam com os dados DICOM, estrutura de dados e codificação. Podem ser usadas por outros visualizadores desenvolvidos em Java.

A linguagem XML permite a marcação de dados que prevê um formato para descrever dados estruturados. Isso facilita declarações mais precisas de conteúdo e resultados mais significativos de busca através de múltiplas plataformas e visualização de dados via Internet.

Na figura 3.11 são apresentadas as classes principais e a forma como se relacionam na arquitetura do Weasis, a sua cor indica funções similares, embora a classe central e de gestão seja a *Weasis Main UI*.

A sua arquitetura permite a extensibilidade através de *plugins*. Estes podem ser por exemplo módulos de compressão de dados e ferramentas específicas de análise de imagens.

3.2.3 Ensaios com o Weasis

O ConQuest é um servidor DICOM que permite armazenar imagens médicas. Este PACS livre, como suporta o protocolo WADO, permite a conexão com visualizadores de imagens que suportem também o WADO. Através do *query* e *retrive* é

[13] *Apache Felix* acedido em 11/08/2012
[14] *Apache Maven Project* acedido em 11/08/2012
[15] *dcm4che.org - Weasis Plug-in Development Guidelines* acedido em 11/08/2012

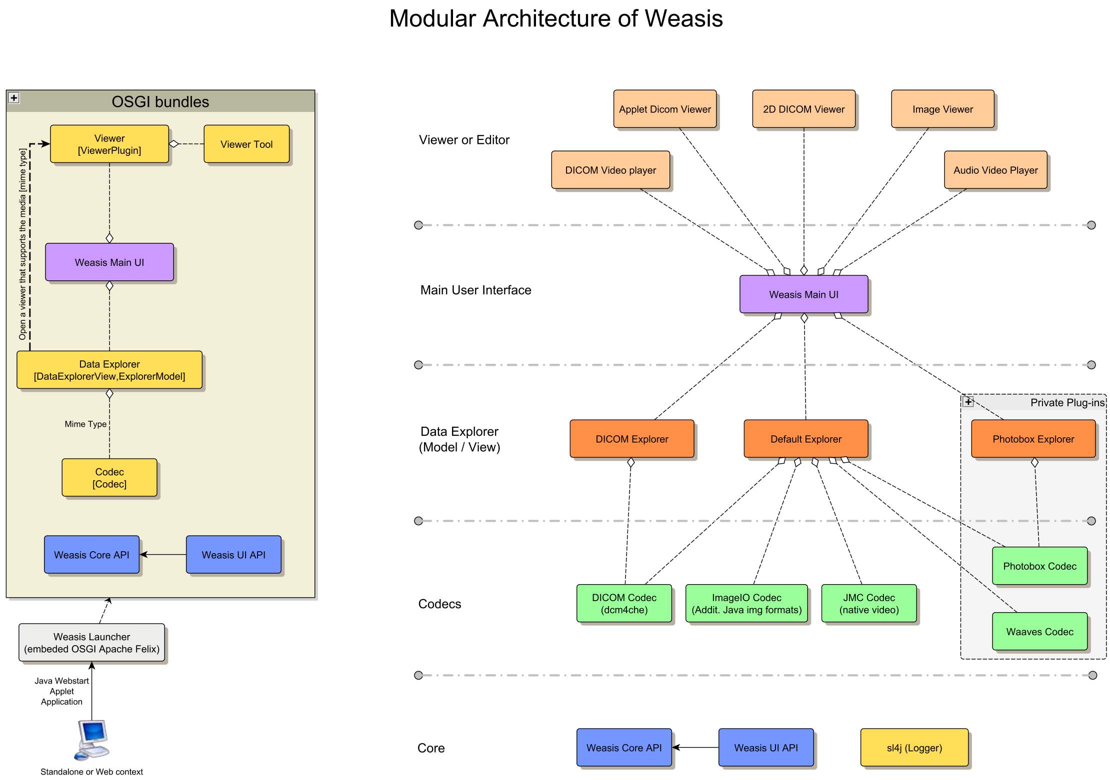


Fig. 3.11: Arquitetura modular do Weasis

possível abrir as imagens guardadas com um visualizador [16].

Neste trabalho foi feita a conexão entre o Weasis e o ConQuest, através do terminal[17].

Para confirmar que a conexão foi feita com sucesso, o estudo requerido ao PACS foi visualizado no Weasis.

Outra forma de conectar o visualizador com o PACS, é através de um *Application Entity Title* (AET). Os três dados que compõem o AET (*String*, porta e ip) permitem o PACS reconhecer/identificar o visualizador, possibilitando uma comunicação DICOM ao PACs. O AET tem de ficar configurado no PACS e deve ser colocado nas propriedades do visualizador. No caso do Conquest o AET é guardado no ficheiro *acrnema.map*, no seu diretório [30, 31]. A imagem 3.12 mostra as configurações feitas no Weasis para que o mesmo possa descarregar os estudos do PACs.

[16] *dcm4che.org Building weasis-pacs-connector* acedido em 03/05/2012
[17] *dcm4che.org Building weasis-pacs-connector* acedido em 29/08/2013

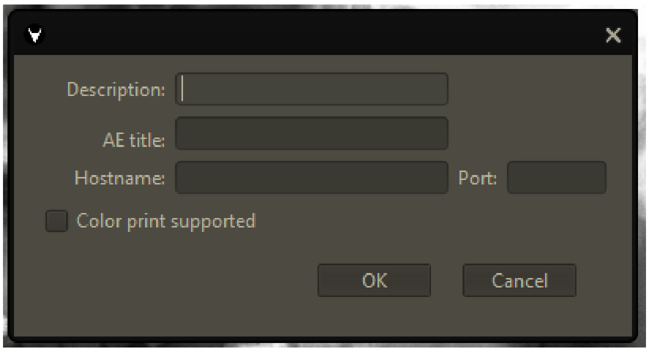


Fig. 3.12: Interface do Weasis que possibilita configurar um AET

3.3 OsiriX

3.3.1 História

O projeto OsiriX iniciou-se em Novembro de 2003 e a sua primeira versão (0.1a) foi desenvolvida por Antoine Rosset, sendo tornada pública em Abril de 2004 com a licença LGPL. Era um visualizador simples, sem funções de pós-processamento e ferramentas de medição [32].

O primeiro artigo sobre o projeto foi publicado em Junho de 2004 no *Journal of Digital Imaging*. Os primeiros *Plugins* foram inseridos em Março de 2005 e permitiam calcular *Surface Rendering* e *Volume Rendering* [33].

Em Junho de 2005, na *Apple's Worldwide Developer Conference* (WWDC) em San Francisco, o projeto OsiriX foi prestigiado com dois prémios da *Apple Design Awards*: *Best Use of Open-Source* e *Best Mac OS X Scientific Computing Solution*.

Em Março de 2009 Antoine Rosset, Joris Heuberger e Osman Ratib criam a fundação OsiriX para promover o desenvolvimento *Open-Source* na Medicina. Esta organização sem fins lucrativos tem por base incentivar os estudantes ao desenvolvimento de soluções *Open-Source* em torno do OsiriX e da imagem digital atribuindo bolsas e prémios [32].

A empresa Pixmeo, criada em 2010, comercializa o OsiriX como uma solução na área da imagem médica certificada pela FDA, no entanto, também possibilita o desenvolvimento *Open-Source* [32], como se verá no capítulo 4.

3.3.2 Licenciamento

A Free Software Foundation (FSF) é uma organização sem fins lucrativos que promove o desenvolvimento e o uso de *software* livre nas áreas da computação [18]. A versão académica do OsiriX é distribuída com a licença GNU Lesser General Public License (LGPL), sendo esta aprovada pela FSF [34].

A LGPL acrescenta restrições ao código-fonte de um *software*, embora não exija restrições a outros *softwares* que utilizem o seu código, desde que este esteja disponível na forma de uma biblioteca. Logo, a inclusão do código desenvolvido sob a LGPL como parte integrante de um *software* só é permitida se o código-fonte for disponibilizado anteriormente [35].

Este processo de licenciamento envolve a adição de dois elementos em cada arquivo do programa: um aviso de *copyright* e uma declaração que permite a cópia, dizendo que o programa é distribuido sobre os termos da LGPL [19].

3.3.3 Arquitetura do Software e Extensibilidade

O OsiriX foi criado com o objetivo de ser uma plataforma comum que facilitasse a interpretação e a gestão de uma grande variedade de dados de imagens dinâmicas, funcionais e moleculares. O *software* teria de ser compatível com imagens DICOM tendo de recebê-las de um equipamento de captura, abri-las, analisá-las, processá-las, guardá-las ou enviá-las [34].

Os autores recorreram a um Sistema Operativo baseado em Unix (Mac OS X, Apple Computer, Cupertino e Calif). Isso deveu-se ao facto deste tipo de sistema operativos apresentar um bom suporte para multiprocessamento, um ambiente *multi-thread* com um *design* apelativo e uma interface de utilização simplificada que corre na quinta geração de microprocessadores de alta performance que está otimizado para aplicações gráficas, por exemplo, processadores PowerPC 970, IBM e New Yorker [34].

A estrutura de desenvolvimento é baseada em ferramentas *Open-Source* tais como Papyrus, DCMTK e DICOM Offis, vtk e itk, PixelMed, OpenGL, Cocoa, XML Expat,Libtiff, Jasper e Libjpeg [34] [20].

As bibliotecas Papyrus, escritas em C, permitem a gestão de ficheiros DICOM[34].

As bibliotecas DCMTK e DICOM Offis permitem a conversão de ficheiros de

[18] *Free Software Foundation* acedido em 11/08/2012
[19] *GNU Operating System* acedido em 11/08/2012
[20] *Online OsiriX Documentation/OsiriX Specifications* acedido em 11/08/2012

imagens DICOM, a comunicação *off-line*, o envio e a receção de imagens através de uma conexão de rede dentro de um serviço PACS. São escritas em C e C++ [21].

As bibliotecas vtk e itk ambas escritas em C++, são disponibilizadas pela empresa Kitware e permitem a segmentação de dados multidimensionais e a sua visualização [22] [23].

As bibliotecas PixelMed, escritas em Java, apoiam as funções da rede DICOM. O suporte OpenGL possibilita uma execução rápida de procedimentos gráficos interativos recorrendo a um melhor proveito do *hardware* [34]. [24].

A linguagem Objective-C é a linguagem primordial para quem desenvolve para a plataforma OS X, consequentemente o OsiriX ter sido desenvolvido em Objective-C. É uma linguagem orientada ao objeto como o C++ e o Java, mas é menos complexa que o C++ e mais rápida que o Java. O compilador usado também foi *Open-Source*: o Compilador GCC, *GNU Compiler Collection*.

O *framework* usado é o Cocoa. Este é um *framework* poderoso e orientado a objetos que permite que sejam criadas interfaces gráficas complexas [36].

Plugins **Distribuídos**

Com a versão académica, o *software* OsiriX permite criar e instalar um *plugin* com um objetivo definido mas também instalar *plugins* distribuídos por outros autores. Pode-se encontrar uma lista de *Plugins Open-Source* e Comerciais do *Software*, no submenu *Plugins/PluginsManager*. É possível ver uma pequena descrição, com as funcionalidades do *Plugin*, a versão, o autor ou a empresa que o distribui, se é *Open-Source* ou comercializado e uma ligação caso se pretenda instalar (ver a figura 3.13) [36]. Estes *plugins* são distribuídos com a licença GPL (General Public License), marcação CE e/ou FDA [25].

Como um exemplo de um *Plugin* distribuído e muito útil, destaca-se o *UMM-Perfusion*. Este *plugin* livre permite avaliar a qualidade da irrigação no miocárdio, com o intuito de detectar se há ou não isquemia deste tecido, através de um estudo de Ressonância Magnética [37]. A sua interface é pequena, no entanto, os resultados são apresentados em forma de gráficos, como se pode ver imagem 3.14.

[21] *DCMTK - DICOM Toolkit* acedido em 11/08/2012
[22] *VTK - Visualization Toolkit* acedido em 11/08/2012
[23] *ITK - Insight Segmentation and Registration Toolkit* acedido em 11/08/2012
[24] *Online OsiriX Documentation/OsiriX Specifications* acedido em 11/08/2012
[25] *Osirix Plugins* acedido em 11/08/2012

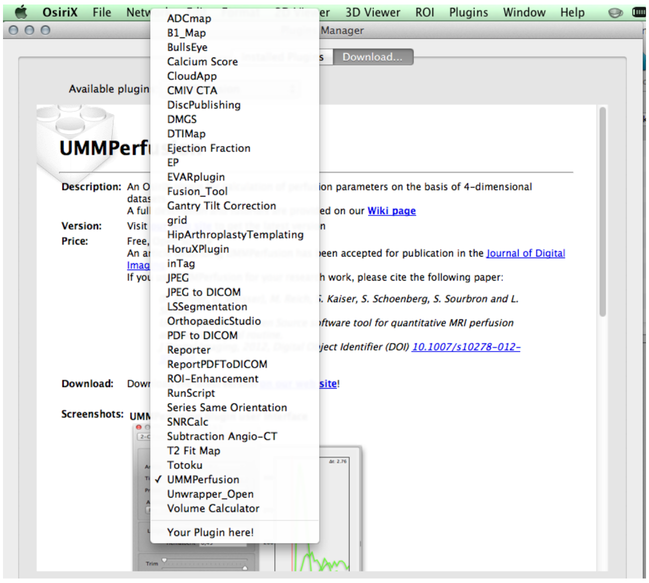


Fig. 3.13: Interface com a lista dos *Plugins* que são possíveis instalar no *Software* e a pequena descrição que os acompanha

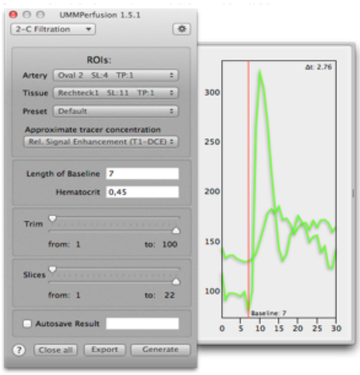


Fig. 3.14: Interface do *Plugin UMMPerfusion*

3.3.4 OsirixMD

A empresa Pixmeo SARL além de disponibilizar uma versão académica comercializa o OsiriXMD.

A versão comercial distingue-se da versão gratuita por possuir a funcionalidade *e-mail*, o manual de utilizador anexado, atualizações gratuitas durante um ano e ser disponibilizado numa extensão 64-bit permitindo uma melhor alocação de memória e tempo de execução de funções mais rápido. É comercializado por 599 U.S.D (484,591Euros) [26].

O OsiriXMD é certificado com a licença FDA, órgão governamental responsável

[26] *Pixmeo - Products* acedido em 11/08/2012

pela certificação dos Estados Unidos da América, e com a marcação CE que possibilita a livre circulação no Espaço Económico Europeu. Há também uma versão para *iPad* e *iPhone* - OsiriX HD - e uma para teste que é *Open-Source* - OsiriX. Esta ultima será usada neste projeto e possui a licença LGPL, sendo esta licença explorada mais à frente [27].

3.4 Análise Comparativa

Para uma melhor comparação das funcionalidades principais dos visualizadores anteriormente apresentados, é apresentada uma tabela comparativa em 3.1.

As aplicações analisadas apresentam as funcionalidades básicas de um visualizador. Não foram mencionadas anteriormente mas também têm a sua importância. O *Zoom* permite o aumento da imagem; o centro e a largura da janela são parâmetros utilizados para controlar facilmente o brilho e o contraste das imagens; as ferramentas de medição permitem determinar distâncias retas, circulares e angulares; a função *Pan* permite o utilizador deslocar a imagem e existem sempre alguns filtros para um pequeno processamento, suavizações, inversão de cores ou realces de superfícies. Todos os visualizadores possuem de alguma forma o controlo destes parâmetros, quer seja através de *scrollbars*, isto é, movimento do rato e um botão pressionado ou combinações de teclas como atalhos. Os visualizadores comerciais são sempre sistemas *multidisplay*, ou seja, permitem especificar se a visualização é realizada apenas considerando um monitor ou vários.

Após a análise da tabela 3.1 é possível observar que a inserção de *plugins* é suportada por quatro visualizadores: o ImageJ, o MicroView, o Weasis e o OsiriX. No entanto, os dois primeiros carecem de suporte para base de dados local. Além disso possuem uma interface menos atrativa para o mercado alvo deste produto.

O Weasis é um projeto com três anos, possui um fórum onde podem ser trocadas dúvidas entre desenvolvedores de *plugins*. A última versão do *software* foi lançada no início do ano 2012, não estando datado o lançamento da próxima versão.

No entanto, a falta de documentação para o desenvolvimento de *plugins* no Weasis levaram a que esta opção fosse rejeitada. Por outro lado, a última opção, o OsiriX é mais completo em termos de documentação e, além disso, é muito mais conhecido na área médica. Depois de alguns ensaios, concluiu-se que o processo de

[27] *OsiriX Imaging Software* acedido em 11/08/2012

Tab. 3.1: Tabela Comparativa de Visualizadores Médicos analisados

Visualizador/ caraterísticas	*Local Database*	Linguagem de desenvolvimento	Sistema Operativo	Inserção de de *Plugins*	Visualização de várias imagens da mesma série	Volume e *Surface Rendering*
ImageJ 1.46	Não	Java	Windows, Mac OS X e Linux	Sim	Não	Sim
DicomWorks 1.3.5	Sim	C++	Windows	Não	Sim	Não
MicroView 2.1.2	Não	C++ e Python	Windows, Mac OS X e Linux	Sim	Sim	Sim
Paraview 3.12.0	Não	Python	Windows, Mac OS X e Linux	Não	Não	Sim
MicroDicom 0.5.4	Sim	C e C++	Windows	Não	Não	Sim
iQ-View 2.8.0	Sim	-	Windows	Não	Sim	Sim
eFilm 3.4	Sim	-	Windows	Não	Sim	Sim
Dicom Viewer 2.0	Sim	C++	Windows	Não	Sim	Não
Weasis	Sim	Java	Windows, Mac OS X e Linux	Sim	Sim	Não
OsiriX 4.1.2	Sim	Objective-C	Mac OS X	Sim	Sim	Sim

desenvolvimento de novos módulos seria mais expedito no OsiriX.

4

Integração de Funcionalidades no Visualizador OsiriX

Neste capítulo serão descritos todos os passos para a integração modular no Visualizador OsiriX. Foram utilizadas as linguagens de programação Objective-C, C e Java e o Xcode para desenvolvimento.

Atente-se, que o objetivo do *plugin* será determinar o diâmetro da artéria aorta a partir de uma tomografia computorizada torácica, usando o algoritmo baseado no *software* Fiji desenvolvido em [4]. Nesta parte do documento serão descritas duas formas de fazer a integração.

4.1 *Script*

O *script* apresenta três fases distintas, sendo elas: pré-processamento, processamento e análise. Na primeira fase são extraídas dimensões das imagens, em x, y e z, a imagem é reduzida e suavizada. Com o processamento pretende-se segmentar a imagem, obter uma linha central da aorta (ver figura 4.1) e obter cortes perpendiculares a esta.

Note-se que em cada filtro está implícito o processo descrito no sub-capítulo 2.2. Na análise são obtidos os diâmetros da aorta em vários locais, ver figura 4.2, e os resultados são apresentados numa tabela ou na própria figura [4].

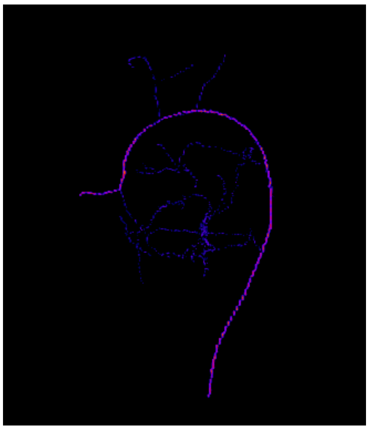


Fig. 4.1: Linha central da aorta [4]

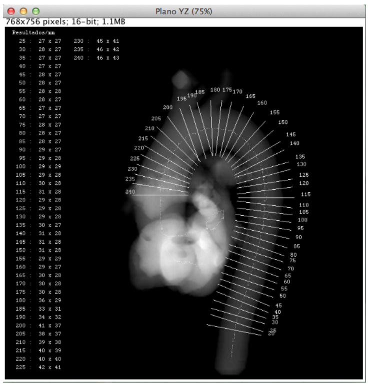


Fig. 4.2: Representação tridimensional da aorta. Os cortes perpendiculares à artéria indicam os locais onde serão medidos os diâmetros [4]

4.2 API do OsiriX

O OsiriX apresenta um estrutura muito complexa, todavia é um *software* que permite a utilização de *plugins* desenvolvidos por terceiros. As classes e os métodos estão disponíveis num repositório[1] indicando a sua utilidade.

A estrutura principal dos dados no OsiriX são listas de ficheiros com imagens DICOM. Estas imagens podem ser selecionadas no início do programa, internamente cada conjunto de imagens consiste num *NSArray* que contém elementos do tipo *DCMPix*.

O OsiriX suporta imagens em tons de cinza ou a cores. Os *pixels* são guardados um após outro pela ordem que constituem a matriz da imagem. Se a imagem for em tons de cinza, cada *pixels* terá um valor da escala CLUTS, ocupará 32 *bit* e o *array* será do tipo *float*. Se a imagem for a cores, cada *pixel* terá um valor da escala RGBA

[1] *OsiriX index page* acedido em 11/08/2012

(*Red Green Blue Alpha*), ocupará 8 *bit* por canal e o *array* será do tipo *unsigned char*. Todavia, uma imagem em tons de cinza pode estar guardada numa escala RGBA[2].

Um tipo de janelas, 2D ou 3D, está associada a um conjunto de imagens.

O *Framework Cocoa* sugere o uso do design *Model-View-Controller*, muitas das funcionalidades do OsiriX estão guardadas em *Viewer-Controller-pairs*, como por exemplo as classes *SRView* e *SRController*, usadas para extrair uma superfície de um conjunto de imagens DICOM. Uma das classes mais importantes é a *Viewer-Controller* que gere as operações que podem ser aplicadas para a representação 2D e cria e inicializa os controladores para a visualização 3D.

As janelas de representação 3D são unicamente identificadas pela sua sintaxe @'SR', por exemplo para a classe *SRController*, e de seguida o apontador para a imagem DICOM, algumas dessas classes são: *MPRView*, *VRView* e *SRView*.

As janelas *Controller* podem ser acedidas pelo método *FindViewer* do *appController* que contém uma lista de *views* ativos por cada conjunto de imagens.

Para cada conjunto de imagens DICOM existe um único *SRView*, este é criado como um atributo do *SRController* correspondente quando o método do *SRViewer ViewerController* é chamado. O *SRController* processa a entrada de utilizador através da interface de usuário contido no ficheiro *SR.nib*. A classe *SRView* é derivada da classe *VTKView*, que é derivado de *VTKCocoaGLView*. Assim, a manipulação da entrada da janela *SRView* é definida essencialmente pelo VTK e não pelo OsiriX. A classe *SRView* contém as ferramentas VTK para calcular a representação da superfície 3D apartir de imagens 2D. Estas classes são usadas no método de *changeActor SRView*.

Como o OsiriX permite a representação de duas superfícies extraídas dos dados 2D a partir de diferentes valores de cinzento, existem duas matrizes dimensionais para cada uma destas classes. Opcionalmente, pode-se aplicar alguns filtros com dados de entrada para melhorar a qualidade da representação. *SRView* também fornece alguns métodos para manipular pontos da superfície, utilizando o *VTKWorlPointPicker*. Estes pontos são representados pela *VTKSphereActors*, as suas posições, raios e as cores, são armazenadas em *NSMutableArrays* separadamente [38].

[2] *Osirix Plugin Basics* acedido em 11/08/2012

4.3 Comunicação entre um *Plugin* e o OsiriX

O *Plugin* criado é considerado uma sub-classe do objeto *PluginFilter*, este objeto está definido no ficheiro *Pluginfilter.m*, na pasta *OsiriX header* e nunca deve ser alterado.

Após a realização de um *plugin* (ver anexo B.2) o OsiriX será lançado e procurará na pasta do *Plugin* o ficheiro com a extensão *.osirixplugin* . A aplicação *host* tentará encontrar a sub-classe na mesma pasta, se o objeto tem a função *filterimage* e a função *initPlugin* o *Plugin* é instalado.

A função *filterimage*, ((long) filterImage:(NSString*) menuName), é a função *main* do *plugin*. O OsiriX usa sempre esta função para chamar um *plugin*. A *String menuName* contém o menu que o autor escolheu para o *Plugin* ficar localizado (ver anexo B.2, ponto 6) [36].

O *Plugin* criado pode virtualizar algumas funções do OsiriX. Alguns objetos serão apresentados seguidamente [36]:

- *PluginFilter*: um *plugin* criado será uma sub-classe deste objeto e que possui algumas funções úteis;
- *ViewerController*: controla as janelas 2D;
- *DCMView*: permite controlar o *display* da imagem;
- *DCMPix*: permite adquirir a informação de um *pixel* da imagem;
- *dicomFile*: permite adquirir a informação da imagem DICOM;
- *ROI*: permite adquirir uma Região de interesse;
- *MyPoint*: descreve um ponto 2D.

4.4 Implementação usando um ficheiro temporário

A criação e inserção do *Plugin* tem várias fases. Neste sub-capítulo serão descritas todas as etapas necessárias para inserir um *script* Fiji, como se podem enviar parâmetros e como se visualizam os resultados no visualizador OsiriX.

Na figura 4.3 é apresentado um diagrama explicativo da interação do OsiriX com o *Plugin* invocando o Fiji.

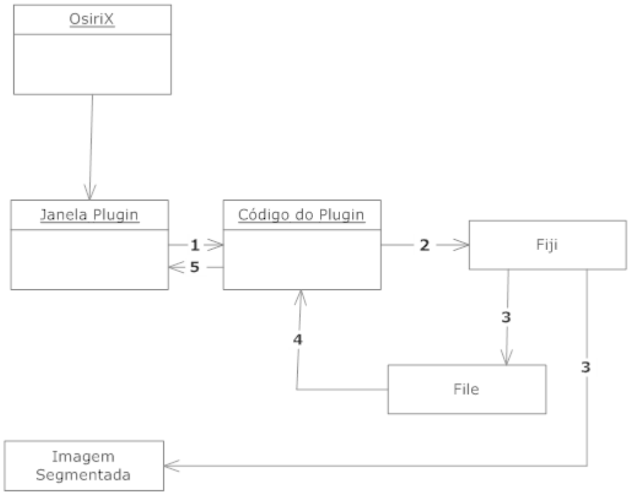


Fig. 4.3: Diagrama explicativo da interação do OsiriX com o *Plugin* que invoca o Fiji. Em 1 são enviados o nome do ficheiro e os parâmetros de *threshold*, em 2 é feita a invocação do *script* para cálculo do diâmetro, em 3 são gerados os resultados, em 4 é feita a leitura dos resultados e em 5 é atualizada a janela com os mesmos.

4.4.1 Criação do projeto

Para criar um projeto no Xcode e torná-lo como um *plugin* no OsiriX é necessário seguir o tutorial apresentado no anexo A.2. O tutorial indica algumas configurações e todos os passos detalhados.

4.4.2 Acesso à imagem

Quando se pretende abrir um estudo no OsiriX a imagem é importada para uma base de dados do próprio *software*. Esta encontra-se em Users/.../Documents/OsiriX Data/DATABASE.noindex e os estudos encontram-se organizados por pastas.

Em Objective-C, o método *sourcefile* do objeto *curPix*, possibilita obter o caminho para o estudo no sistema de ficheiros. O *curPix* é um objeto do tipo *DCMPix*.

As linhas de código correspondentes são:

```
DCMPix *curPix;
...
NSString *file_path = [curPix sourceFile];
```

4.4.3 Criação de um novo *script*

O Fiji reconhece um *script* como argumento de entrada. Com isto, o caminho da imagem tem de ser integrado no *script* para posteriormente o Fiji ser executado a partir do *Plugin*, daí a necessidade anterior de se saber o caminho físico da imagem.

As linhas de código do *script*:

```
IJ.run("Close All");
IJ.run("Image Sequence...", "open()")
```

criam uma janela que permite selecionar a imagem onde o *script* vai ser aplicado, no entanto não existe a necessidade de escolher a imagem e estas linhas foram alteradas para:

```
IJ.run("Close All");
#OPEN_FILE_LINE_FOR_PARSER
path = "open="
IJ.run("Image Sequence...", path)
```

As linhas alteradas vão permitir que o *script* receba um caminho de uma imagem. Este será o caminho obtido através da classe *DCMPix* da API do OsiriX, tal como mencionado na secção 4.5.2.

Foi implementada uma função, em Objective-C, que cria uma nova versão do *script* inicial actualizando o caminho para o estudo desejado.

A função procura linha-a-linha os carateres ”path=”após :

```
#OPEN_FILE_LINE_FOR_PARSER
```

Se a linha do *script* for diferente é enviada sem qualquer alteração para um *array*, se a linha for igual então cria uma nova linha que que indicará o caminho para o estudo (e.g., path = ”open=/imagens/estudo1.dcm”).

4.4.4 Invocação do Fiji

A classe *NSTask* [3]permite executar uma aplicação numa linha de código. Esta classe permitiu executar o Fiji e receber como argumento a linha de comando para executar o *script* alterado.

4.4.5 Parâmetros de *threshold* no *script*

O utilizador pode inserir os valores limite, *Lower* e *Upper*, do filtro *threshold* que constitui o *script*. Os valores são alterados no *script* de uma forma análoga ao caminho da imagem mencionado anteriormente.

[3] *Mac Developer Library - NSTask Class Reference* acedido em 11/08/2012

4.4.6 Visualização dos valores dos diâmetros

A classe *NSPipe* [4] permite guardar a leitura do *output* dos comandos executados na *shell* a partir do código Objective-C. Esta classe foi utilizada para ler o *output* do *script Fiji*. Neste caso, o *output* do *script* são os valores dos diâmetros da aorta. Os resultados são posteriormente mostrados numa *label* do *Plugin*.

4.4.7 Criação da interface

Para se criar uma interface num *plugin* é necessário inicialmente elaborar o código de todos os elementos. De seguida, é necessário criar a interface e os elementos visíveis e só posteriormente fazer a associação entre as duas partes, o código e a interface. Todos os passos realizados estão descritos no Anexo A.3.

O código do *plugin* pode ser consultado no anexo B.1 .

4.4.8 Utilização do *Plugin*

Para instalar o *Plugin* é necessário ter o OsiriX instalado nos Aplicativos e o ficheiro de instalação do *Plugin* que possui a extensão *.osirixplugin* . Para correr o *Plugin* é necessário ter o Fiji instalado nos aplicativos.

Com um duplo clique no *Plugin*, o OsiriX será lançado. Abre-se um estudo da *Local Database* e o *Plugin* será visível no sub-menu *Image Filters*.

A sua interface apresenta unicamente um botão como se pode ver na figura 4.5. Como entrada é utilizado o estudo que foi aberto pelo OsiriX inicialmente. O utilizador pode inserir os valores do *threshold*, *Lower* e *Upper*. Caso não sejam inseridos, por defeito são inseridos no *Plugin* os valores 260 e 650. Quando é pressionado, aparecem as imagens resultantes de todo o processamento associado ao *script*, ver figura 4.5. Quando se fecha a janela do processamento, os diâmetros da aorta, nos locais préviamente estabelecidos [4], são mostrados na interface do *Plugin*, figura 4.6.

[4] *Mac Developer Library - NSPipe Class Reference* acedido em 11/08/2012

Fig. 4.4: Interface do *plugin*, permite que o utilizador insira os valores do filtro *threshold*

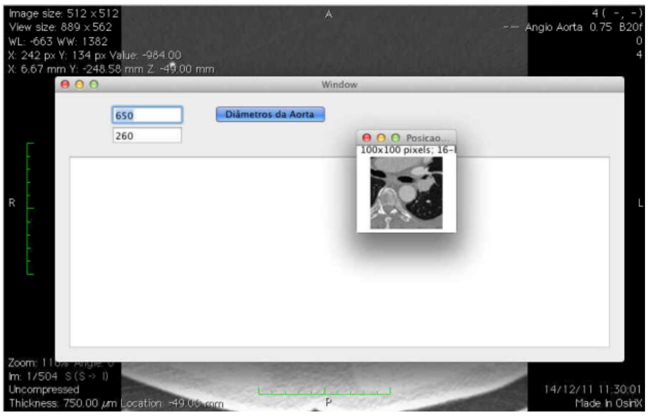


Fig. 4.5: *Plugin* com a imagem final do processamento da aorta

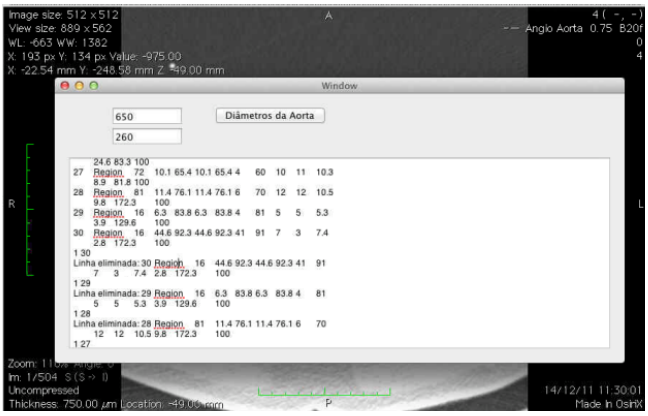


Fig. 4.6: *Plugin* com os resultados do diâmetro da aorta

4.5 Invocação do Fiji com passagem de dados através de memória partilhada

O método anterior ignora eventuais alterações feita às imagens no OsiriX que ainda não tenham sido gravadas, uma vez que acede diretamente às imagens armazenadas no sistema de ficheiros.

De forma a não penalizar o tempo de execução do *script*, pretende-se evitar a escrita de ficheiros temporários. Ou seja, pretende-se que seja possível lançar o *script* sem obrigar o utilizador a gravar eventuais alterações que tivesse feito, e sem criar ficheiros temporários.

Contornando estes pontos, na segunda forma de implementação do *script* Fiji foi utilizado outro método, baseado no *Java Native Interface* (JNI). Para tal, foi necessário converter o código Python do *script* original para Java.

Como foi referido na secção 2.6 o JNI pode ser utilizado de duas formas: 1) Chamar funções nativas (e.g., desenvolvidas em C) a partir do código Java; 2) Executar métodos de classes Java a partir de código nativo, através do lançamento de uma JVM *Invocation Interface*.

Inicialmente, tentou-se usar a segunda abordagem, para passar uma cópia da imagem, ou da região de interesse da imagem, diretamente para a JVM. No entanto, com o decorrer dos testes, observou-se que tal não era suportado no sistema utilizado.

Dessa forma, optou-se pela abordagem de, no OsiriX, criar uma cópia da imagem em memória partilhada e, no Fiji, utilizar um conjunto de métodos nativos que permitem fazer a passagem da imagem na memória partilhada para um *array* em Java. A utilização dos métodos nativos é necessária, pois o Java não oferece nenhuma classe para interagir com este tipo de mecanismo.

A criação do *Plugin* é descrita no sub-capítulo 4.3.1 e é a única fase comum à implementação anterior.

Esta implementação é composta por três blocos principais: *Shared Memory*, JNI e conversão do código Python para Java.

4.5.1 *Shared Memory*

A memória partilhada - *Shared Memory* - permite que vários processos compartilhem a mesma região de memória. Se um processo atualizar a memória partilhada, a mudança é imediatamente visível aos outros processos que partilham a mesma região [39].

A memória partilhada foi útil para guardar uma cópia da imagem num bloco de memória. Foram utilizados algumas funções da norma POSIX (API para sistemas Unix, também utilizada por outros sistemas, tais como o Linux) cuja função se encontra descrita seguidamente.

- *shm_open* - permitem criar/abrir objetos de memória partilhada;
- *shm_unlink* - elimina um objeto de memória partilhada;
- *ftruncate* - define o tamanho do objeto;
- *mmap* - mapeia o objeto no espaço de endereçamento virtual do processo;
- *munmap* - faz o processo contrário ao *mmap* no espaço de endereçamento virtual do processo;

O seguinte extrato de código permite: eliminar o objecto "myshm"que possa existir na memória, criar um novo bloco de memória "myshm", definir o seu tamanho e preenchê-lo com a imagem. A variável *fImageNew* é um apontador para o endereço da imagem alocada em memória.

```
shm_unlink("myshm");
int fd = shm_open("myshm", O_CREAT | O_RDWR, 0777);
if(fd<0){
    perror("shm_open");
    exit(1);
}
if(ftruncate(fd, N*sizeof(float))<0) {
    perror("ftruncate");
}
fImageNew = mmap(NULL, N*sizeof(float), PROT_READ | PROT_WRITE,
MAP_SHARED, fd, 0);
if(fImageNew == NULL){
    perror("mmap");
    exit(1);
}
```

Para que a imagem seja mostrada com a mesma gama do modelo de carregamento do Fiji somou-se 1024 à intensidade de todos os *pixels*. Os *pixels* da imagem da memória partilhada estavam entre [-1024;64512] e passaram para [0;65536]. O WW e a WL foram ajustados para 0 e 500 respetivamente, após consulta dos metadados da imagem. A figura 4.7 mostra através do histograma qual a gama de maior interesse no processamento. Uma vez que o script original estava adaptado para este formato de estudos, não houve preocupação em tornar o *plugin* mais genérico. Este procedimento poderia ter sido automatizado no *plugin* do OsiriX através das linhas:

```
float myWL = [viewerController curWL];
float myWW = [viewerController curWW];
[viewerController setWL: myWL WW: my WW];
```

O valor das duas variáveis teria de ser ajustado consoante os valores do Fiji.

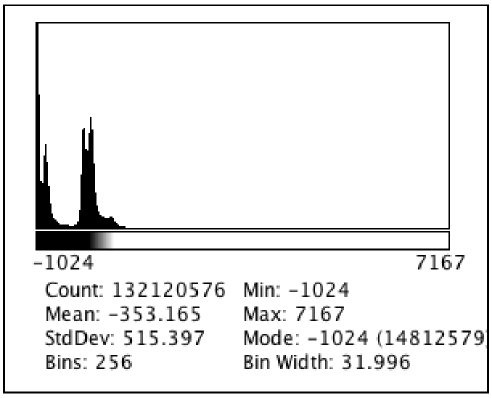


Fig. 4.7: Histograma do estudo obtido com o OsiriX

4.5.2 JNI - *Java Native Interface*

Inicialmente, delineou-se que seria chamada uma *Java Virtual Machine* (JVM) a partir de Objective-C. O código desenvolvido inspirou-se nas instruções fornecidas em [22, 23]. No entanto, durante a fase de prototipagem observou-se que o código desenvolvido não produzia os resultados esperados no sistema utilizado (OS X 10.8.4 e JRE 1.7.0), tendo sido impossível fazer a invocação da JVM desta forma. É de salientar que o mesmo código, com ligeiras alterações, funcionou normalmente num sistema Linux. Não foi encontrada informação adicional sobre este tópico nos repositórios da Apple.

Após esta primeira fase, o JNI foi utilizado neste trabalho para aceder aos blocos de memória partilhada. Por conseguinte, apenas serão integrados métodos nativos na classe Java. Esta integração consiste em seis passos fundamentais[22, 23, 40].

1. Criar a classe que declara o método nativo

Inicialmente cria-se a classe Java e a declaração do método nativo como:

```
%private native void testeInit();
public native float[] testeGetArray();
```

Esta linha de código indica à JVM que a implementação do método *testeGetArray()* não se encontra na classe Java e estará numa biblioteca externa.

2. Compilar o programa

De seguida, compila-se o ficheiro Java para *bytecode*. Uma forma das formas de o fazer é através do compilador *javac* que vem com o SDK. O comando para a compilação é:

```
javac JNI_Java.java
```

3. Criar um ficheiro *header*

Posteriormente, é necessário criar um ficheiro *header*. Este ficheiro irá definir a assinatura da função nativa, utilizando o comando:

```
javah JNI_Java
```

JNI_Java indica o nome do ficheiro Java.

Para que a ligação entre Java e o código nativo seja feita corretamente o JNI define regras de nomenclatura, que devem ser estabelecidas do lado do C. A implementação do método definido anteriormente no ficheiro *header* que é gerado automaticamente ficará:

```
JNIEXPORT void JNICALL Java_JNI_Java_testeGetArray(JNIEnv*env, jobject obj);
```

JNIEXPORT e JNICALL definem as convenções para exportação e a nomenclatura da função de acordo com o sistema operativo. Seguidamente, é apresentado "Java_", o nome da classe e do método. Os pontos ficam substituídos por "_". As funções em C possuem dois parâmetros: JNIEnv*env, jobject obj. A estrutura JNIEnv aponta para uma tabela que contém apontadores para as funções JNI que podem ser chamadas no código nativo, enquanto que o parâmetro *jobject* referencia o objeto Java, permitindo a manipulação de objetos Java a partir de C.

4. Criar o ficheiro C com a implementação do método nativo

Para criar o ficheiro C deve-se ter em atenção que a implementação da função deve ter por base a declaração gerada no ficheiro *header*. Refere-se que o *header* do JNI, que contém as definições necessárias do JNI para ser reconhecido do lado do C e o *header* gerado devem ser incluídos neste ficheiro.

O método seguinte permite adquirir o *array* de *Floats* e ajustar o WW e a WL para 0 e 500 respetivamente.

```
JNIEXPORT jfloatArray JNICALL Java_JNITest_testeGetArray(JNIEnv *env, jobject obj)
{
    int i;
    jfloatArray jdata = (*env)->NewFloatArray(env, N);
    jfloat *data = (*env)->GetFloatArrayElements(env, jdata,0);
    for(i=0;i<N;++i)
    {
        data[i] = fImageNew[i];
        if(data[i]<0)
        data[i]=0;
        if(data[i]>500)
        data[i]=500;
    }
    (*env)->ReleaseFloatArrayElements(env, jdata, data,0);
    return(jdata);
}
```

5. Compilar o código C e gerar a biblioteca dinâmica

É necessário criar uma biblioteca dinâmica que contém o código nativo compilado, no entanto, este passo difere entre sistemas operativos. Em sistemas Microsoft Windows deve ser gerada uma biblioteca .dll, em Linux um *shared object* .so e em Mac OS X .jnilib. Para isso, deve-se criar um ficheiro intermédio .o em que /System/Library/Frameworks/JavaVM.framework/Headers é a localização do JNI no sistema Mac OS X através do comando:

```
cc "-I/System/Library/Frameworks/JavaVM.framework/Versions/Current/Headers"

"-I/System/Library/Frameworks/JavaVM.framework/Versions/A/Headers" -c main.c
```

Para criar o biblioteca com o código nativo é compilado com a extensão .jnilib usa-se o comando:

```
cc -dynamiclib -o libJNI.jnilib main.o
```

6. Invocação do código nativo

A biblioteca gerada deve ser carregada pelo código Java. Para tal, deve ser usada a seguinte instrução:

```
static{
    System.loadLibrary("shm");
}
```

Este último método referido é integrado no início da classe Java.

O código do *plugin* pode ser consultado no anexo B.2 .

4.5.3 Processamento da imagem no Fiji

A imagem guardada na memória partilhada foi obtida através de um *array* de *floats* pelo método nativo. Em cada posição do mesmo está guardado o valor da intensidade de um *pixel*.

Seguidamente, passou-se o *array* para uma *stack* de imagens, com o objetivo de visualizar e processar a imagem, com as dimensões iniciais (512 *pixels* de comprimento e largura em 256 *slices*). Cada posição da *stack* equivalerá a uma *slice* da imagem.

Como a imagem guardada apenas tem informações referentes ao valor da intensidade dos *voxels*, é necessário fazer uma calibração inicial à *stack* para que não haja qualquer erro de processamento. Atribuiu-se a função de calibração utilizada em [4], redefiniu-se o tamanho do *voxel* e converteu-se a unidade de processamento de *pixels* para milímetros. A classe utilizada foi a *Calibration* com as seguintes instruções:

```
Calibration cal = imp.getCalibration();
cal.pixelWidth =  0.60546875;
cal.pixelHeight = 0.60546875;
cal.pixelDepth = 0.5;
cal.setUnit("mm");
```

Os valores de escala usados acima deverão ser obtidos a partir do cabeçalho do ficheiro DICOM e poderão variar de estudo para estudo.

As classes mais importantes no *script* são:.

Tab. 4.1: Diferenças entre código Python e Java

Pontos distintos	Python	Java
Importação de bibliotecas	from ij import IJ	import ij.IJ;
Definição de variáveis	scxx=0.61	float scxx=0.61;
Ciclos	for i in range(3):	for(int i : range(0,3)){}
Criação/instanciação de objetos	imp=IJ.getImage();	ImagePlus imp = IJ.getImage();
Funções *built-in*	range(3)	range(0,3);
Criação de estruturas de dados e métodos associados	append()	add();

- *FloatProcessor*: permite criar uma *slice* de 32bit com as dimensões desejadas a partir de um *array* de *floats*;
- *ImageStack*: permite criar uma *stack* com as *slices* criadas com a *FloatProcessor*;
- *ImagePlus*: permite criar objetos a partir do *ImageStack* permitindo processar a imagem com diferentes *Plugins* da classe *IJ*;
- *AnalyzeSkeleton*: permite criar um esqueleto das estruturas para posteriormente calcular a estrutura mais longa indicando a linha central da aorta;
- *ResultsTable*: permite construir uma tabela com os medidas referentes a cada corte perpendicular à linha central da aorta;

Todas as classes referidas pertencem ao *package ij.jar* do *software* Fiji [41].

Alguns *Plugins* aplicados da classe *IJ* são dependentes da aplicação, isto é, só podem ser utilizados dentro do diretório do mesmo. Com isto, todo o processamento em Java tem de ser compilado e executado no diretório do *software*. O conjunto de ficheiros do JNI necessitam estar no diretório do Fiji.

Ao fazer a conversão de Python para Java foi necessário ter atenção a alguns pontos. É possível ver algumas diferenças entre as duas linguagens na tabela 4.1. O código desenvolvido anteriormente em Python não seguia um modelo estrito de orientação a objetos e usava frequentemente a tipagem dinâmica.

As maiores dificuldades, na conversão do código Python para Java, foram perceber qual o tipo de objeto e qual a estrutura de dados mais adequada a usar. Para auxiliar este processo foi usado sempre o mesmo estudo como referência. Neste estudo, ao usar-se os resultados de [4], observou-se que o segmento central tinha de ser constituído por 775 pontos para posteriormente o processamento obtido ser correto.

O diagrama da figura 4.8 permite uma melhor compreensão da interacção entre o *Plugin* e o OsiriX, nesta implementação utilizando o JNI.

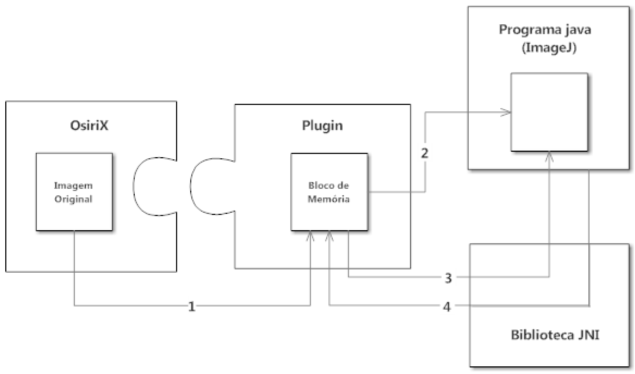


Fig. 4.8: Diagrama explicativo da interação do OsiriX com o *Plugin*. Em 1 é guardada o estudo num bloco de memória partilhada; em 2 é invocado o programa Java; em 3 o estudo é obtido da memória partilhada através dos métodos do JNI para posteriormente fazer o processamento em Java, que está representado em 4.

Envio de resultados para OsiriX através da memória partilhada

4.5.4 Análise Comparativa e Trabalho Futuro

A primeira implementação requereu um menor tempo de conceção, uma vez que não requereu qualquer conversão de linguagens de programação. No entanto, esta versão sofre do potencial problema de enviar para o *plugin* uma imagem diferente daquela que está a ser visualizada. Isto acontecerá caso o utilizador altere a imagem no visualizador e invoque o *plugin* antes de gravar as alterações.

A segunda implementação requer mais recursos da máquina utilizada, nomeadamente da memória RAM, evitando inconsistências entre o que está no disco e na memória RAM. Também diminui os tempos de carregamento do estudo, contudo utiliza um método mais adequado ao tipo de utilização esperada. Para a área médica e para a sua finalidade, a interface com o processamento e com a apresentação dos dados é mais adequada e perceptível. Note-se que, por exemplo, um estudo com 250MB, num disco comum com uma taxa de transferência de 50MB/s, implica um tempo de carregamento de cerca de 5 segundos.

Devido à forma como o *script* Fiji é invocado em cada caso, a sequência de janelas apresentadas é ligeiramente diferente em cada caso, tal como pode ser visto nas figuras 4.6 e 4.9.

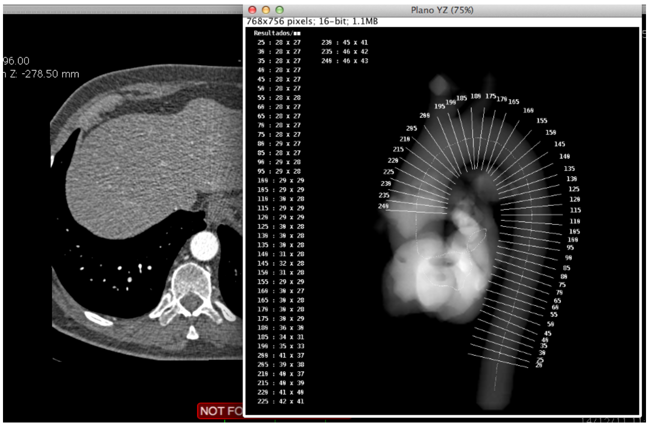
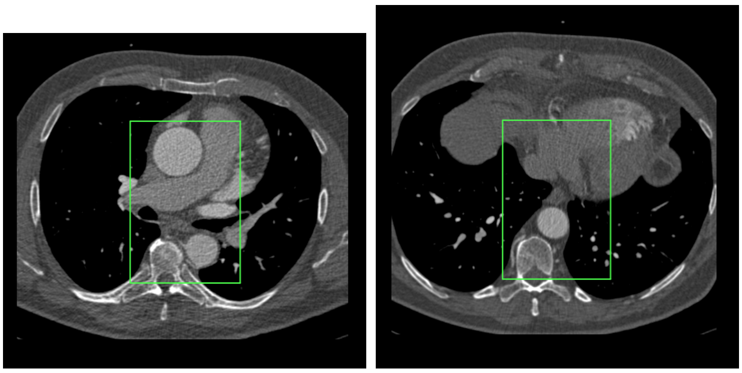


Fig. 4.9: Aorta com os cortes perpendiculares e com a respectiva medição do diâmetro maior e mais pequeno

Perante uma Tomografia Computorizada Torácica existe uma região central, onde está presente a artéria aorta e o coração, com maior importância neste trabalho. Para diminuir o volume de dados na memória partilhada e diminuir o tempo de processamento, em trabalho futuro poderia delinear-se uma região de interesse - ROI - envolvendo estas estruturas. Este procedimento de delineação seria realizado pelo utilizador. A título de exemplo, ao simular um volume demarcado com as dimensões aproximadas de 15 cm de comprimento, 10 cm de largura pelo número total das *slices* da imagem, o volume de dados seria aproximadamente 16.2% da tomografia torácica completa. Este processo seria realizado no OsiriX, sendo que a sus API para *plugins*, disponibiliza funções para identificar ROI indicadas pelo utilizador.

A figura 4.10 apresenta, no estudo total, a delineação da ROI.

Atente-se que, para trabalho futuro a informação da resolução e da escala da imagem deve ser passada através da memória partilhada e a informação de WW e WL deve ser obtida automaticamente.



(a) (b)

Fig. 4.10: *Slices* com a ROI delineada a verde

5

Aposição da Marcação CE

Neste Capítulo serão abordados alguns conceitos sobre Aposição da Marcação CE pelo fabricante, a respetiva legislação nacional e qual o procedimento a adoptar para obter declaração de Conformidade de um Dispositivo médico de Classe IIa. Será preenchida e organizada a documentação para submissão no Organismo Notificado Infarmed.

5.1 Legislação Aplicável

A disciplina jurídica dos dispositivos médicos é regida por um conjunto disperso de normas decorrentes do contínuo progresso técnico e científico e da necessidade de adaptar a legislação nacional às normas da União Europeia.

O Decreto que estabelece as regras, em Portugal, a que devem obedecer a investigação, o fabrico, a comercialização, a entrada em serviço, a vigilância e a publicidade dos dispositivos médicos e respectivos acessórios, é o n.º 145, de 17 de Junho de 2009 [8].

5.2 Organismo Notificado

O Organismo Notificado é uma entidade estabelecida para avaliar e verificar a conformidade dos dispositivos com os requisitos exigidos no Decreto-Lei n.º145, bem como aprovar, emitir e manter os certificados de conformidade com o objetivo de atribuir a marcação CE perante o dispositivo em análise. No nosso país, um dos organismo com tais responsabilidades é o Infarmed, de acordo com os procedimentos previstos na Diretiva 2007/47/CE [8].

O Organismo Notificado tem as seguintes funções [1]:

- Efetuar os procedimentos de Avaliação da Conformidade dos dispositivos médicos no quadro da legislação nacional e comunitária;
- Autorizar a aposição da marcação CE dos dispositivos médicos;
- Emitir os certificados CE de conformidade dos dispositivos médicos;
- Assegurar que o fabricante cumpre corretamente com as obrigações decorrentes do sistema de qualidade aprovado;
- Colaborar com as Autoridades Competentes Nacionais dos Estados Membros;
- Colaborar com os Organismos Notificados dos Estados Membros.

5.3 Classificação do Dispositivo Médico

Pelo Decreto-Lei n.º145/2009, de 17 de Junho, um *Plugin* para um *Software* Médico é um Dispositivo Médico Ativo de Classe IIa. Esta classificação é justificada com as alíneas t) e u) do Capítulo I do Artigo 3.º, Anexo IX, Grupo I, Parte I, 1.6 e Grupo III, Parte IV, Regra n.º16.

As alíneas t) e u) do Capítulo I do Artigo 3.º definem um Dispositivo Médico e um Dispositivo Médico Ativo. No Anexo IX, no Grupo I, Parte I, 1.6, confirma que um *software* por si só é considerado um dispositivo médico ativo.

O Grupo III, Parte IV, Regra n.º16, justifica a classe de risco atribuída a um *Plugin*, os dispositivos especificamente destinados ao registo de imagens radiográficas de diagnóstico pertencem à classe IIa [8].

Como o conceito "registo"' não é esclarecedor, pelo *Manual on Borderline and Classification in the Community Regulatory Framework for Medical Devices*, os PACS que oferecem visualização e pós-processamento enquadram-se na classe IIa [42].

5.4 Dossier Técnico do Dispositivo

O Organismo Notificado Infarmed desenvolveu metodologias e elaborou procedimentos aplicados às atividades de avaliação da conformidade de dispositivos médicos

[1] *infarmed - Autoridade Nacional do Medicamento e Produtos de Saúde* acedido em 07/08/2012

consoante a classe de risco que o dispositivo se insere, com isto, o Infarmed tem o objetivo de auxiliar o requerente a sintetizar o processo de marcação CE [2]. Os documentos estão presentes no Anexo C.

5.4.1 Avaliação da Conformidade

Para Avaliação da Conformidade do Dispositivo foi preenchido um documento baseado no Sistema completo de garantia de qualidade (foi realizado pelo Infarmed e é baseado no Anexo II, exceto ponto 4 do Decreto-Lei n.º145), uma Minuta de requerimento para avaliação da mesma e foi elaborada uma Declaração CE de conformidade e de compromisso.

Documentação para Avaliação da Conformidade - AnexoII

Perante o Decreto-Lei n.º145, é possível fazer a avaliação de conformidade de um dispositivo médico da classe IIa através de três procedimentos diferentes: pelo Anexo II exceto o ponto 4; pelo Anexo VII juntamente com o Anexo V; pelo Anexo VII juntamente com o Anexo VI. Neste caso foi selecionado o Anexo II exceto o ponto 4. Estes Anexos referem-se a Anexos do decreto aplicável.

Este documento tem como objetivo verificar se a empresa tem um sistema de gestão da qualidade bem implementado. Visa informações gerais da empresa, a gestão de risco e a produção.

Supondo que um fabricante pretende certificar um novo dispositivo e já possua o seu Sistema de Qualidade avaliado por já ter certificado produtos anteriormente, apenas deve enviar ao Organismo Notificado a parte da documentação específica relativa ao dispositivo, Minuta de Requerimento para Avaliação de Conformidade, Declaração CE de Conformidade e Declaração de Compromisso e Tabelas de Requisitos. Contudo, o Organismo Notificado pode solicitar novas atualizações relativas ao Sistema de Qualidade anteriormente aprovado.

Minuta de Requerimento para Avaliação de Conformidade

A Minuta de Requerimento solicita o Organismo Notificado para uma Avaliação de Conformidade, indicando principalmente a classe do dispositivo, a regra de classificação, segundo o Decreto-lei n.º145/2009, o procedimento de avaliação da conformidade escolhido, neste caso, Anexo II, e a assinatura do requerente.

[2] *infarmed - Autoridade Nacional do Medicamento e Produtos de Saúde, Procedimentos de avaliação de conformidade pelo ON* acedido em 07/08/2012

A Minuta de Requerimento pode ser consultada no Anexo C.1 e está devidamente preenchida.

Declaração CE de Conformidade e Declaração de Compromisso

Esta declaração é obrigatória e muito importante. O fabricante declara e compromete-se a cumprir com as suas responsabilidades, assinando no final.

Este documento pode ser consultado no Anexo C.2 devidamente preenchido.

5.4.2 Tabela de Requisitos Essenciais

Com a Tabela de Requisitos é possível verificar se as caraterísticas mais específicas do dispositivo médico estão de acordo com a legislação aplicável e devem ser devidamente justificadas. Estes requisitos são relativos à conceção e ao fabrico, tais como, propriedades relativas ao fabrico e condições ambientais, proteção contra radiações, cuidados com uma fonte de energia ou que dela disponham como equipamento, proteção contra riscos elétricos, rotulagem e instruções de utilização.

Este documento pode ser consultado no Anexo C.3 e está devidamente preenchido.

A empresa é certificada pela ISO 9001, esta norma reconhece o esforço da organização em assegurar a conformidade dos seus produtos e/ou serviços, a satisfação dos seus clientes e a melhoria contínua [3]. Para justificar alguns pontos da tabela de requisitos esta norma foi mencionada.

Toda a documentação deverá ser enviada no formato digital e organizada em pastas. A Minuta de Requerimento e a Declaração CE de conformidade e de compromisso, têm de ser enviados em suporte papel, devidamente preenchidos e assinados.

[3] *apcer* acedido em 10/08/2012

6

Conclusão

As soluções médicas, nomeadamente os visualizadores de imagens médicas, auxiliam na observação de estudos, mais precisamente, em algumas patologias focalizadas. Por conseguinte e atendendo ao objectivo deste trabalho, a integração do *script* do *software* Fiji num visualizador médico veio solucionar a determinação manual do diâmetro em vários pontos da artéria aorta.

Um visualizador contém ferramentas que permitem e auxiliam a medição, no entanto, quanto mais automatizado for o processo mais benefícios traz ao profissional de saúde, ou seja, permite diminuir o tempo de obtenção e aumento da precisão do valor do diâmetro.

Existe uma gama abrangente de *softwares*, todavia apenas o OsiriX, o Weasis e o ImageJ destacaram-se pela capacidade de integração de funcionalidades e automatização de processos.

A implementação foi feita no OsiriX. Este *software* é bastante utilizado e reconhecido no meio médico, além de que foi o solicitado pelo serviço de Cardiologia do Hospital Santos Silva. Como projeto de contínuo desenvolvimento e conjuntamente com a sua arquitectura modular, permitiu uma fácil integração do *plugin*. Apresenta uma interface funcional e intuitiva, permitindo fácil acesso ao *plugin*.

A integração utilizando o JNI e um ficheiro temporário foram procedimentos adoptados unicamente para o OsiriX. Todo o processo de integração requereu bastante pesquisa e agregação de novos conhecimentos. As linguagens utilizadas, orientadas ao objeto, conjuntamente com os filtros do Fiji em Java facilitaram todo o processo.

Por último, foi feita a simulação da aposição da marcação CE. Caso o Organismo Notificado emita a Declaração de Conformidade, declarará a conformidade do *plugin* com os requisitos legais, possibilitando a venda no Mercado Europeu, garantindo a qualidade da produção e a valorização do produto.

Bibliografia

[1] S. Wagner, *Heuristic Optimization Software Systems.* PhD thesis, Johannes Kepler Universität Linz, Altenberger Straße 69, 4040 Linz, Österreich, März 2009.

[2] *Digital Imaging and Communications in Medicine (DICOM)*, vol. Part 18: Web Access to DICOM Persistent Objects (WADO). Nacional Electrical Manufactures Association, 2011.

[3] N. Roduit, F. Klumb, D. Badon, A. Geissbuhler, and O. Ratib, "Weasis: a free web-based viewer aimed for telemedicine," June 2010.

[4] L. Silva, "Software de medição automática do diâmetro da aorta torácica em tc," Master's thesis, Instituto Superior de Engenharia do Porto, Novembro 2012.

[5] M. do Céu Ferreira, "A importância da metrologia na saúde." VII - Conferência Estatística e Qualidade na Saúde, 2011.

[6] D. Birsan, *On Plug-ins and Extensible Architectures.* Queue, March 2005.

[7] F. S. Johannes Mayer, Ingo Melzer, "Lightweight plug-in-based application development," *Revised Papers from the International Conference NetObjectDays on Objects, Components, Architectures, Services, and Applications for a Networked World*, pp. 87 – 102, 2002.

[8] *Decreto-Lei n.Âº 145/2009.* Diário da República, 17 de Junho 2009.

[9] D. Rodrigues, "Marcação CE, otimização e caraterização de uma superfície de redução de pressão para doentes acamados," Master's thesis, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, Lisboa, 2007.

[10] A. Maintz and M. A. Viergever, "A survey of medical image registration," *Medical Image Analysis*, vol. 2, pp. 1–36, March 1998.

[11] F. Nunes, "Introdução ao processamento de imagens médicas para auxílio ao diagnóstico - uma visão prática," in *Livro das Jornadas de Atualização em Informática*, ch. 2, pp. 73–126, Julho 2006.

[12] P. Romera, "Implementação de um protótipo de software para reconstrução de modelos 3d a partir de imagens no padrão dicom," Novembro 2001.

[13] O. Pianykh, *Digital Imaging and Communications in Medicine (DICOM)*, p. 417. second edition ed., 2012.

[14] I. Bankman, *Handbook of Medical Imaging: Processing and Analysis.* 2000.

[15] I. Manssour and C. Freitas, "Visualização volumétrica," *RITA*, vol. IX, no. 2, pp. 97–126, 2002.

[16] M. Albuquerque and M. Albuquerque, "Processamento de imagens: Métodos e análises."

[17] "Strategic document," tech. rep., Digital Imaging and Communications in Medicine, 1300 North 17th Street, Suite 1752, April 2012.

[18] *Digital Imaging and Communications in Medicine (DICOM)*, vol. Part 1: Introduction and Overview. National Electrical Manufacturers Association, 2011.

[19] *Digital Imaging and Communications in Medicine (DICOM)*, vol. Part 5: Data Structures and Encoding. Nacional Electrical Manufactures, 2011.

[20] J. S. et al., "Fiji: an open-source platform for biological-image analysis," *FOCUS ON BIOIMAGE INFORMATICS*, vol. 9, pp. 676–682, July 2012.

[21] S. Kochan, *Programming in Objective-C 2.0.* 2011.

[22] S. Liang, *The Java Native Interface, Programmers Guide and Specification.* Addison-Wesley Professional, June 1999.

[23] "Technical note tn2147, JNI development on mac OS X," tech. rep., Apple Inc., July 2011.

[24] T. Ferreira and W. Rasband, "Imagej user guide ij 1.46r," Julho 2012.

[25] T. Collins, "Imagej for microscopy," *Biotechniques*, vol. 43, pp. 25–30, July 2007.

[26] K. Eliceiri and et al., "Biological imaging software tools," *FOCUS ON BIOIMAGE INFORMATICS re*, vol. 9, pp. 692–710, July 2012.

[27] P. Puech, L. Boussel, S. Belfkih, L. Lemaitre, P. Douek, and R. Beuscart, "Dicomworks: Software for reviewing dicom studies and promoting low-cost teleradiology," *Journal of Digital Imaging*, vol. 20, pp. 122–130, Julho 2007.

[28] J. Rivera and V. Lasso, "Plataforma de código abierto para servicios de teleconsulta con soporte de imágenes médicas bajo el estándar dicom," Master's thesis, Universidad del Cauca, Facultad de Ingeniería Electrónica y Telecomunicaciones, Popayán, 2012.

[29] D. Mendelson, P. Bak, and E. Menschik, “Image exchange: Ihe and the evolution,” *RadioGraphics*, vol. 28, pp. 1817–1833, November-December 2008.

[30] M. van Herk and L. Zipj, *Using Conquest on Linux*, 2013.

[31] M. van Herk and L. Zipj, *Conquest DICOM Server*, version 1.4.17 ed., May 2013.

[32] A. Rosset, *OsiriX Manual*. Pixmeo.

[33] A. Rosset, L. Spadola, and O. Ratib, “Osirix: An open-source software for navigating,” *Journal of Digital Imaging*, vol. 17, pp. 205–216, Junho 2004.

[34] A. Rosset, L. Spadola, L. Pysher, and O. Ratib, “Navigating the fifth dimension: Innovative interface for multidimensional multimodality image navigation,” *Informatics in Radiology (infoRAD)*, vol. 26, pp. 299–308, Janeiro-Fevereiro 2006.

[35] C. Filho, “Demanda legais no desenvolvimento de software sob uma visão tecnológica,” Fevereiro 2012.

[36] A. Rosset, *OsiriX Plug-ins Developer Toolkit*, version 1.0 ed., 2004.

[37] F. Zõllner, G. Weisser, M. Reich, S. Kaiser, S. Schoenberg, S. Sourbron, and L. Schad, “Ummperfusion: an open source software tool towards quantitative mri perfusion analysis in clinical routine,” *Journal of Digital Imaging*, vol. 25, July 2012.

[38] O. Mahdi, W. Globke, and P. Huthwohl, “The planning butler an osirix plug-in for complex 3d planning,” April 2006.

[39] M. Kerrisk, *The Linux Programming Interface*. October 2010.

[40] A. Inc., ed., *Java Development Guide for Mac*. October 2010.

[41] W. Burger and M. J. Burge, *Digital Image Processing, An Algoritthmic Introducion Using Java*. first ed.

[42] “Manual on borderline and classification in the community version 1.11,” December 2011.

Apêndice A

Tutoriais - OsiriX

A.1 Instalação e Configurações Iniciais

1. Instalar o Xcode 3.0.0 nos Aplicativos;

2. Instalar o OsiriX 4.1.2, versão para testes. Atualmente a versão disponível é a 5.1.0;

3. Fazer o *download* do código fonte do OsiriX através do Terminal:

   ```
   svn co https://osirix.svn.sourceforge.net/svnroot/osirix osirix
   ```

4. Abrir o projeto OsiriX_Lion.xcodeproject localizado na pasta Osirix/Osirix_Launcher. No ficheiro main.m fazer as seguintes configurações como mostra a figura A.1 e compilar seguidamente;

5. Abrir novamente o Terminal e fazer *download* do código dos *plugins Open-Source* do OsiriX:

   ```
   svn co https://osirix.svn.sourceforge.net/svnroot/osirixplugins osirixplugins
   ```

6. Fazer as configurações em Xcode/Preferences como mostra a figura A.2

A.2 Inserção de um *Plugin*

1. Ir a pasta osirixplugins/_help e descompactar o ficheiro *Osirix Plugin Generator.zip*;

2. Na mesma pasta abrir o *Generator Osirix Plugin*, inserir o nome do *Plugin* e o autor como se vê na imagem A.3;

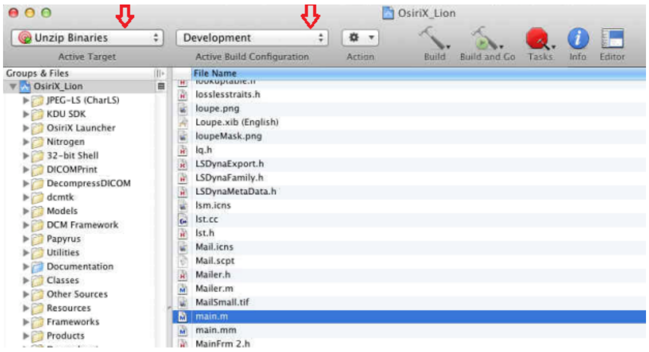


Fig. A.1: Configurações no Xcode para desenvolvimento de *plugins* para o OsiriX

3. Abrir o ficheiro Exemplo1.xcodeproj na pasta osirixplugins/_help/Exemplo1. Será mostrada a interface do Xcode e a árvore de ficheiros do lado esquerdo da interface, ver figura A.4

   Cada ficheiro possui a sua função:

   Exemplo1.lproj: contém a informação para se criar um *plugin* em vários idiomas;

   Exemplo1.pch: projeto específico pré-compilado, não se deve alterar;

   Exemplo1.m e Exemplo1.h: ficheiros principais que contêm as definições e algoritmos;

   Exemplo1.plist e Exemplo1.plist: ficheiros que contém a informações sobre o *Plugin*, nome, versão, ícon entre outros. Devem ser alterados diretamente do Xcode.

   OsiriXHeaders: pasta de ficheiros com os objetos do OsiriX para serem utilizados pelo *Plugin*, nunca deve ser alterada.

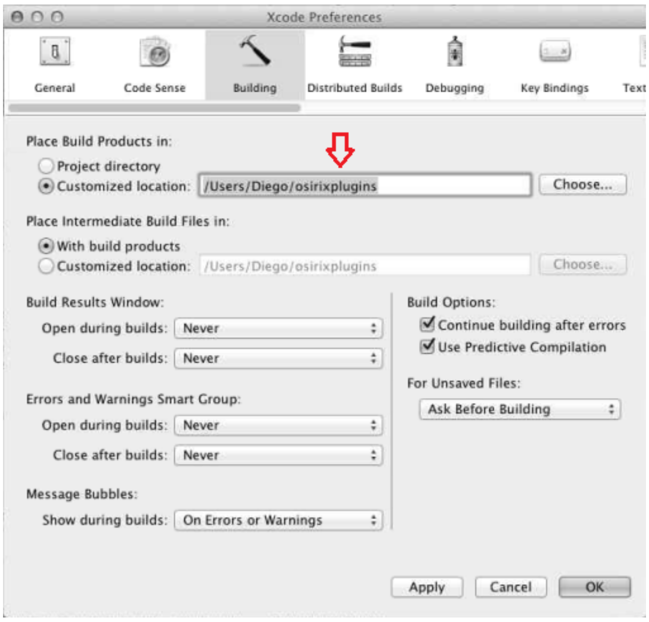
4. Ligar o *Plugin* à aplicação principal em Executables>Add>New Custom Executable como se vê na figura A.5;

5. Compilar o projeto;

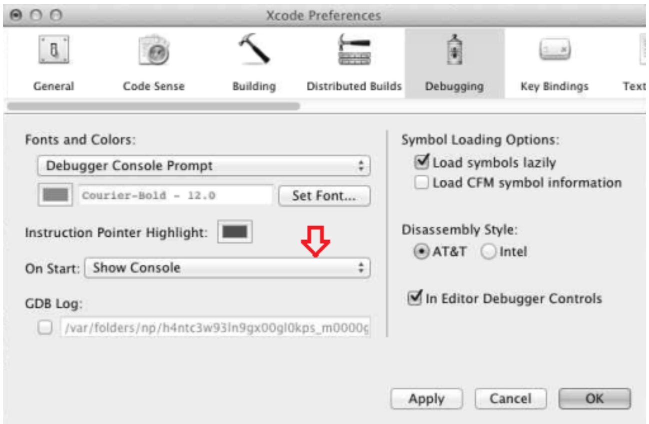
6. Se na compilação aparecer o erro *'There is no SDK at specified SDKROOT path'*, inserir o caminho de instalação do sdk nas informações do projeto, neste caso foi: /Developer/SDKs/MacOSX10.5.sdk. Compilar novamente;

7. Clicar no ficheiro Exemplo1.osirixplugin. Este ficheiro instalará o *Plugin* no OsiriX, a sua cor passará de vermelho para preto indicando que já se encontra instalado e o OsiriX será aberto;

8. Abrir uma imagem, o *plugin* criado aparecerá no Menu *Plugins* e no sub-menu *Image Filters* (ver imagem A.6);

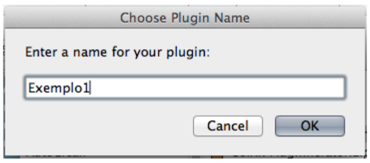


(a)

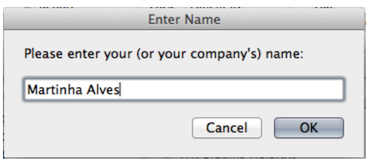


(b)

Fig. A.2: Configurações no Xcode para desenvolvimento de *plugins* para o OsiriX: (a) Separador *Building*; (b) Separador *Debugging*



(a)



(b)

Fig. A.3: O *Generator* permite criar um projeto que será um *plugin* para anexar ao *Plugin*: (a) Janela para inserir o nome do *Plugin*; (b) Janela para inserir o nome do autor

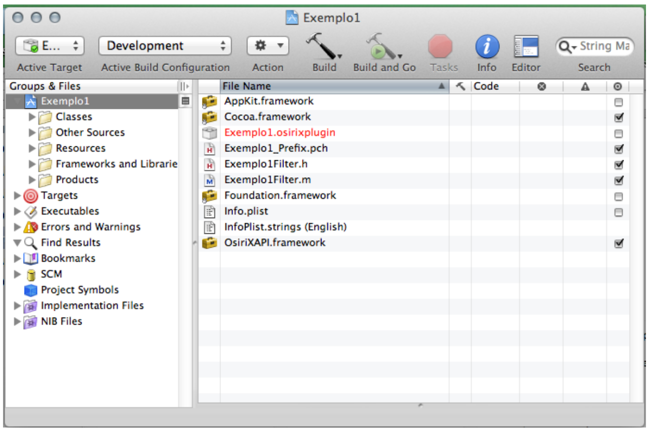
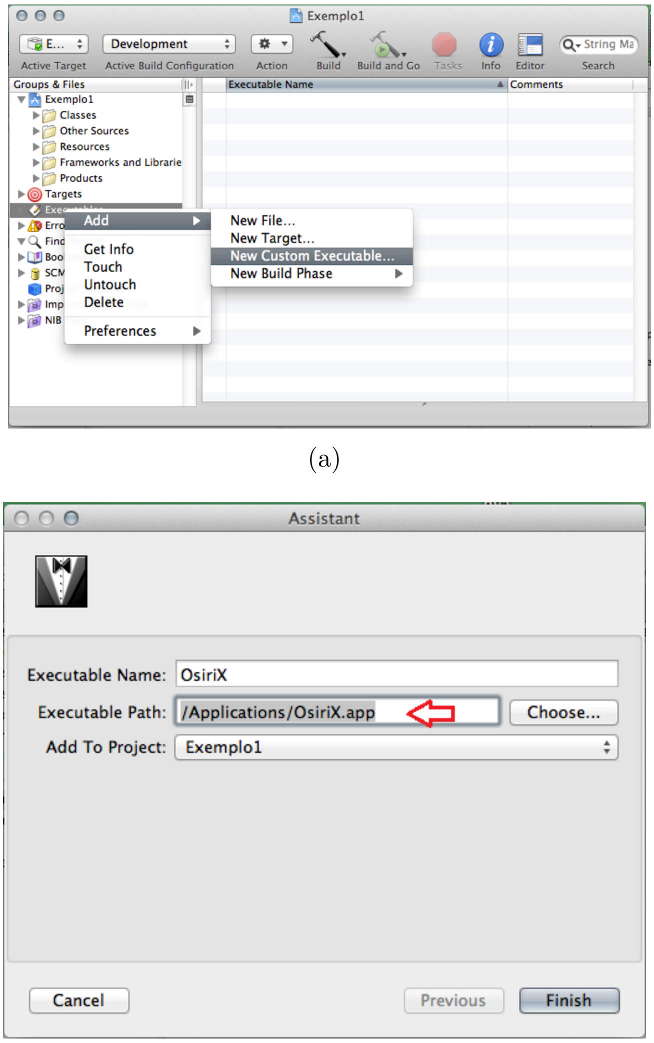


Fig. A.4: Conjunto de ficheiros que constituem um *Plugin*

A.3 Criação de um *Plugin* com Interface

1. Repetir os passos 1 a 4 do tutorial anterior;
2. Declarar a interface, o botão e a ação no ficheiro Exemplo1.h;
3. Criar as interações entre os objetos no ficheiro Exemplo1.m;
   A linha de código:
   ```
   [new2DViewer needsDisplayUpDate];
   ```
   permite atualizar as imagens com os *pixels*
4. Compilar o projeto;
5. Criar a interface em NIB Files> Add> New file> Interface Builder> Cocoa> NIB> Window XIB, figura A.7;
6. Clicar no ficheiro criado e será aberto a Interface *Builder* como se pode ver na imagem A.8;
7. Criar um novo botão e fazer as associações entre ações definidas e os elementos da interface;
8. Compilar o projeto;
9. Clicar no ficheiro Exemplo1.osirixplugin. Este ficheiro instalará o *Plugin* no OsiriX, a sua cor passará de vermelho para preto indicando que já se encontra instalado e o OsiriX será aberto;
10. O *Plugin* criado, interface apresentada na figura A.9, permite duplicar a imagem aberta inicialmente pelo OsiriX e alterar a escala de intensidades dos *pixels*.



(a)

(b)

Fig. A.5: Fazer a ligação entre o *Plugin* e o OsiriX

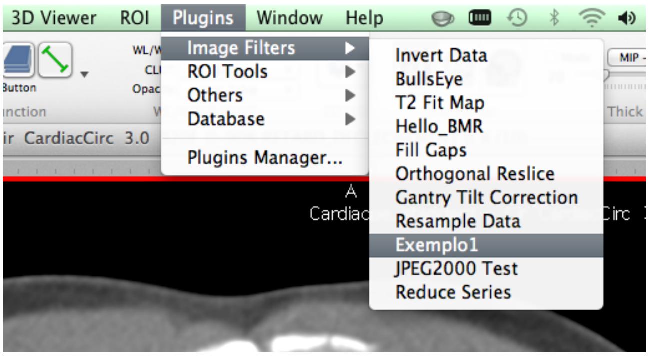


Fig. A.6: Imagem com o submenu onde o *Plugin* fica visível

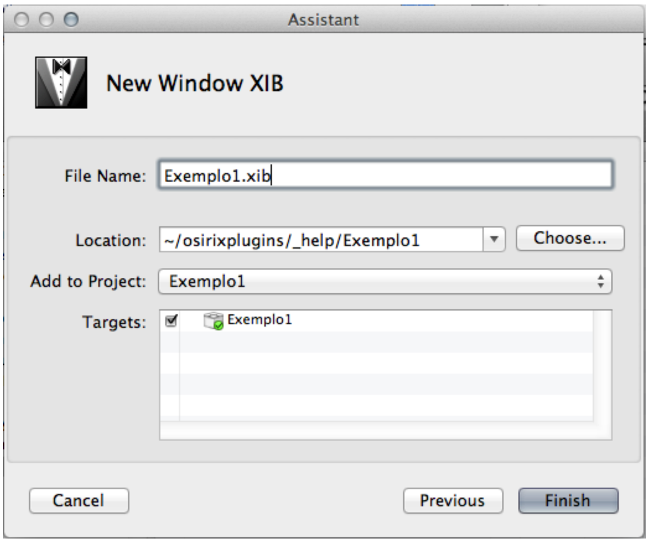


Fig. A.7: Criação do ficheiro xib

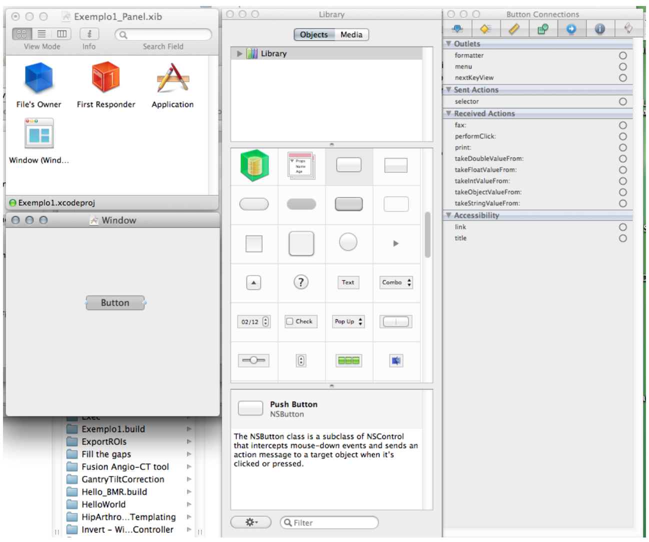


Fig. A.8: Janelas que permitem criar a interface

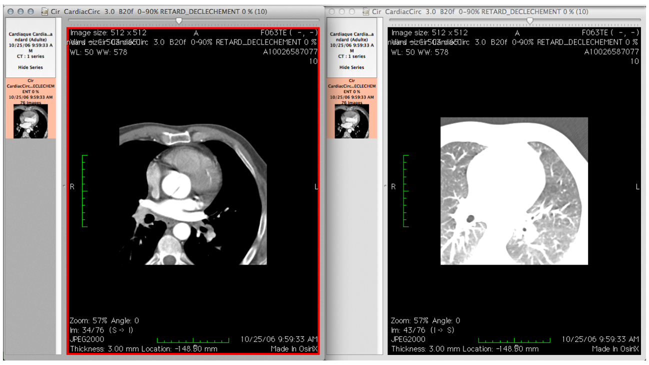


Fig. A.9: O *Plugin* cria uma cópia da imagem original com uma nova escala de cinzas.

Apêndice B

Código dos *Plugins* para medição do diâmetro da aorta

B.1 Invocação usando um ficheiro temporário

```
import "Hello_BMRFilter.h"
@implementation Hello_BMRFilter;
@synthesize lblUpper;
@synthesize lblLower;
@synthesize tvTabela;

- (void) initPlugin
{
}

- (long) filterImage:(NSString*) menuName
{
    NSWindowController *window = [[NSWindowController alloc] initWithWindowNibName:
    @"Hello_BMR_Panel" owner:self];
    [window    showWindow:self];
    return 0;
}

- (IBAction) DuplicateIt: (id)sender;
{
    NSArray    *pixList;
    pixList    = [viewerController pixList];
    DCMPix    *curPix;
    curPix     = [pixList    objectAtIndex:0];
    NSString    *file_path = [curPix sourceFile];
    NSString *entireFileInString = [NSString stringWithContentsOfFile:
    @"/Users/martinha/osirixplugins/_help/Hello_BMR/Nova_aorta_v8.py"];
    NSArray *lines = [entireFileInString componentsSeparatedByString:@"\n"];
    NSString *TAG_OPEN = @"#OPEN_FILE_LINE_FOR_PARSER";
    NSString *TAG_UPPER = @"#UPPER_LINE_FOR_PARSER";
    NSString *TAG_LOWER = @"#LOWER_LINE_FOR_PARSER";
    NSString *Found_TAG = @"";
    NSString *upper_value = @"650";
    NSString *lower_value = @"260";
    upper_value = [lblUpper stringValue];
    lower_value = [lblLower stringValue];
    NSMutableArray *NewFileArray = [NSMutableArray new];
    int shouldIModifieTheLine = 0;
    for (NSString *line in lines) {
```

```
        if ( ([line isEqualToString:TAG_OPEN]) || ([line isEqualToString:TAG_UPPER]) ||
        ([line isEqualToString:TAG_LOWER])){
            shouldIModifieTheLine = 1;
            Found_TAG = line;
            [NewFileArray addObject:line];
        }
        else {
            if (shouldIModifieTheLine == 1) {
                if ([Found_TAG isEqualToString:TAG_OPEN]) {
                    [NewFileArray addObject:[NSString stringWithFormat:@"path = \"open='%@'\"", file_path]];
                    shouldIModifieTheLine = 0;
                }
                else if ([Found_TAG isEqualToString:TAG_UPPER]) {
                    [NewFileArray addObject:[NSString stringWithFormat:@"th_high=%@", upper_value]];
                    shouldIModifieTheLine = 0;
                }
                else if ([Found_TAG isEqualToString:TAG_LOWER]) {
                    [NewFileArray addObject:[NSString stringWithFormat:@"th_low=%@", lower_value]];
                    shouldIModifieTheLine = 0;
                }
            }
            else
            [NewFileArray addObject:line];
        }
    }
    NSString *fullStringToFile = [NewFileArray componentsJoinedByString:@"\n"];
    [fullStringToFile writeToFile:@"/Users/martinha/osirixplugins/_help/Hello_BMR/Nova_aorta_v8.py" atomically:
    YES encoding:NSStringEncodingConversionAllowLossy error:nil];
    NSString *path = @"/Applications/Fiji.app/Contents/MacOS/ImageJ-macosx";
    NSArray *args;
    args = [NSArray arrayWithObjects: @"/Users/martinha/osirixplugins/_help/Hello_BMR/Nova_aorta_v8_2.py", nil];
    NSTask *task = [[NSTask alloc] init];
    [task setLaunchPath: path];
    [task setArguments:args];
    NSPipe *outpipe = [NSPipe pipe];
    [task setStandardInput:[NSPipe pipe]];
    [task setStandardOutput:outpipe];
    [task launch];
    [task waitUntilExit];
    [task release];
    NSData *outData = [[outpipe fileHandleForReading] readDataToEndOfFile];
    NSString *outString = [[[NSString alloc] initWithData:outData encoding:NSUTF8StringEncoding] autorelease];
    [tvTabela insertText:[NSString stringWithFormat:@"=%@", outString]];
}
@end
```

B.2 Invocação do Fiji com passagem de dados através de memória partilhada

```
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include "teste_shmFilter.h"

@implementation teste_shmFilter

- (void) initPlugin
{
}

- (long) filterImage:(NSString*) menuName
{
    int    i, x, zSize;
    float        *fImage, *fImageNew;
    NSArray    *pixList;
    DCMPix    *curPix;
    pixList        = [viewerController pixList];
    zSize        = [pixList count];
    curPix        = [pixList        objectAtIndex:0];
    x        = [curPix pheight] * [curPix pwidth];
    int N = x*zSize;
    shm_unlink("myshm");
    int fd = shm_open("myshm", O_CREAT | O_RDWR, 0777);
    if(fd<0){
        perror("shm_open");
        exit(1);
    }
    if(ftruncate(fd, N*sizeof(float))<0) {
        perror("ftruncate");
    }
    fImageNew = mmap(NULL, N*sizeof(float), PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
    if(fImageNew == NULL){
        perror("mmap");
        exit(1);
    }
    fclose(fd);
    float *ptr_mmap = fImageNew;
    for (i = 0; i < zSize; i++){
        NSLog(@"z=%i",i);
        curPix = [pixList objectAtIndex:i];
        fImage = [curPix fImage];
        x = [curPix pheight] *[curPix pwidth];
       % if(i == 0){
          %  FILE *fp=fopen("/Users/martinha/osirixplugins/_help/teste_shm/image.dat","w");
```

```
            %if(fp == NULL){
               % perror("fwrite");
                %} else {
                %if(fwrite(fImage,sizeof(float),x,fp)!=x)
                %perror("fwrite");
                %else
                %printf("Wrote %d bytes\n",x);
                %fclose(fp);
            %}
        %}
        while ( x-- > 0 )
        {
            *fImageNew = (*fImage);
            fImage++;
            fImageNew++;
        }
    }
    system("/Applications/Fiji.app/launcher.sh");
    munmap(ptr_mmap,N*sizeof(float));
    return(0);
}

@end
```

Apêndice C

Dossier Técnico

C.1 Minuta de Requerimento para Avaliação da Conformidade

Minuta de requerimento para avaliação da conformidade
(Decreto Lei n.º 145/2009 na sua atual redação)

Exmo. Sr.
Presidente do Conselho Diretivo INFARMED I.P.
Organismo Notificado
Parque da Saúde de Lisboa
Avenida do Brasil, nº 53, Pav. 17-A
1749-004 Lisboa
Portugal

Fabricante / Representante legal: *Efficientia Manegement Integration Consulting*
Morada: *Rua Soares dos Reis, 765-3*
Localidade: *Vila Nova de Gaia*
Código Postal: *4400* - *317* *Vila Nova de Gaia*
Telefone: *223 744 830* Fax: *223 744 831*
E-mail: *info* @ *efficientia.pt*
Responsável técnico: *Efficientia*

Dispositivos médicos a avaliar
Categoria: *Software Médico*
Família: *Dispositivo Médico Ativo, Classe IIa*
Identificação: *Plugin 'Cálculo Automático do Diâmetro da Aorta'*
Marca(s) Comercial(is): *Efficientia*

Venho por este meio solicitar ao INFARMED, I.P., como Organismo Notificado, a avaliação de conformidade do(s) dispositivo(s) médico(s) supracitado(s), de classe IIa, de acordo com a(s) regra(s) de classificação *16* (menção da(s) regra(s) aplicada(s)), estabelecida(s) no anexo IX do Decreto-lei n.º 145/2009, na sua atual redação.

A avaliação da conformidade deverá ser efetuada de acordo com o Anexo *II* (procedimento de avaliação da conformidade escolhido) do referido diploma.

Requerente: *Filipe Morais*
Data: ____/_____/_____

____
(Assinatura do Requerente)

NOTA: Este requerimento deverá ser acompanhado do respetivo CD contendo a documentação técnica exigida para avaliação da conformidade do dispositivo e a declaração de compromisso.

C.2 Declaração CE de Conformidade e Declaração de Compromisso

Declaração CE de conformidade e Declaração de compromisso

Dispositivo(s) médico(s): *Plugin para Visualizador de Imagens Médicas*

Categoria(s): *Software Médico*

Família: *Dispositivo Médico Ativo, Classe IIa*

Identificação do(s) dispositivo(s)*: *Plugin 'Cálculo Automático do Diâmetro da Aorta'*

Marca comercial: *Efficientia*

Para vários produtos, anexar à Declaração a lista dos produtos abrangidos.

Fabricante / Representante legal: *Efficientia Manegement Integration Consulting*

Responsável técnico: *Filipe Morais*

Declaro que:

- Não foi apresentado a nenhum outro Organismo Notificado um requerimento equivalente, relativo ao mesmo sistema de qualidade, a que se refere(m) o(s) dispositivo(s) em análise.

- O(s) referido(s) dispositivo(s) médico(s) (~~contêm~~ / não contêm) como parte integrante da sua constituição, uma das substâncias referidas nos nºs 7.4 e 7.5 do Anexo I do Decreto-Lei 145/2009.

- No fabrico do(s) dispositivo(s) (não) se utilizam tecidos de origem animal, tal como referidos na Diretiva nº 2003/32/CE, da Comissão, de 23 de abril e no capítulo VII do Decreto-Lei supra citado.

- O(s) dispositivo(s) cumpre(m) com os requisitos essenciais aplicáveis estabelecidos no anexo I do Decreto-lei supracitado, pelo que não põe(m) em risco a saúde e a segurança dos utilizadores desde que utilizado(s) de acordo com a finalidade para que foi(ram) concebido(s).

Comprometo-me a:

- Notificar o Organismo Notificado Infarmed, caso haja alguma modificação do produto após aprovação.

- A autorizar o Organismo Notificado a efetuar todas as inspeções necessárias e a fornecer-lhe todas as informações que me sejam solicitadas para que o Organismo Notificado possa assegurar-se da aplicação correta do sistema da qualidade aprovado ou a aprovar.

- A cumprir as obrigações decorrentes do sistema de qualidade aprovado, mantendo-o adequado e eficaz, não delegando nenhuma das funções relativas ao sistema de qualidade, tais como o tratamento de reclamações ou vigilância do dispositivo.

- A informar o Organismo Notificado de qualquer projeto de alterações introduzidas no sistema da qualidade aprovado ou da gama de produtos abrangidos.

- A criar e manter atualizado um processo de análise sistemática dos dados adquiridos com os dispositivos na fase de pós-produção e a desenvolver meios adequados de execução das ações corretivas necessárias tendo em conta a natureza e os riscos relacionados com o produto e os incidentes abaixo referidos:
  - qualquer deterioração das características e/ou do funcionamento de um dispositivo, bem como qualquer inadequação da rotulagem ou das instruções respeitantes a um dispositivo que sejam suscetíveis de causar ou ter causado a morte ou degradação grave do estado de saúde de um doente ou utilizador;

  - qualquer motivo de ordem técnica ou médica ligado às características ou ao funcionamento de um dispositivo pelas razões acima definidas que tenha ocasionado a retirada sistemática do mercado dos dispositivos do mesmo tipo;

- A informar a entidade com competência de fiscalização sobre as ocorrências acima referidas, assim que delas tiver conhecimento, bem como o Organismo Notificado.

Data: ____/_____/_____

____

(Assinatura do Responsável técnico)

C.3 Tabela de Requisitos Essenciais

Requisitos essenciais de um dispositivo médico - Diretiva 93/42/CEE na sua atual redação

(Decreto-lei nº 145/2009, na sua atual redação)

	Aplicável	
	Sim 1) Método(s) encontrado(s) para garantir cumprimento (indicar qual(is)) 2) Documento(s) do SGQ relacionado(s)	**Não** 1) Justificação
Grupo I - Requisitos gerais		
1.2.1 A redução, na medida do possível, dos riscos derivados de erros de utilização devido às características ergonómicas do dispositivo ou ao ambiente que está previsto para a utilização do produto (conceção tendo em conta a segurança do doente);	***1) Conceção*** ***2)ISO 9001:2008***	
1.2.2 A consideração dos conhecimentos técnicos, da experiência, da educação e da formação e, se for caso disso, das condições clínicas e físicas dos utilizadores previstos (conceção para utilizadores não profissionais, profissionais, portadores de deficiência ou outros utilizadores);	***1) Conceção*** ***2)ISO 9001:2008***	
2.1 Eliminar ou reduzir os riscos ao mínimo possível (conceção e construção intrinsecamente seguras);	***1) Conceção*** ***2)ISO 9001:2008***	
2.2 Quando apropriado, adotar as medidas de proteção adequadas, incluindo, se necessário, sistemas de alarme para os riscos que não podem ser eliminados;		***1) O software não precisa de medidas de proteção.***
2.3 Informar os utilizadores dos riscos residuais devidos a insuficiências nas medidas de proteção adotadas.		***1) O software não precisa de medidas de proteção.***
3 Os dispositivos devem atingir os níveis de adequação que lhes	***1) Conceção***	

tiverem sido atribuídos pelo fabricante e ser concebidos, fabricados e acondicionados por forma a poderem desempenhar uma ou mais das funções previstas na alínea u) do artigo 3.º do decreto-lei de que o presente anexo é parte integrante, de acordo com as especificações do fabricante.	***2)ISO 9001:2008*** ***e ver Manual do Utilizador.***	
4 As características e os níveis de funcionamento referidos nos n.os 1 a 3 do presente anexo não devem ser alterados sempre que as alterações possam comprometer o estado clínico e a segurança dos doentes e, eventualmente, de terceiros, durante a vida útil dos dispositivos prevista pelo fabricante, quando submetidos ao desgaste decorrente das condições normais de utilização.	***1) O software não tem definido tempo de vida útil.*** ***No entanto, quando são feitas alterações ao dispositivo são efetuados registos e feitos novos testes.***	
5 Os dispositivos devem ser concebidos, fabricados e acondicionados de modo que as suas características e níveis de funcionamento, em termos da utilização prevista, não sofram alterações no decurso do armazenamento e do transporte, tendo em conta as instruções e informações fornecidas pelo fabricante.	***2) Ver Manual de Utilizador.***	
6.1 A demonstração da conformidade com os requisitos essenciais deve incluir uma avaliação clínica nos termos do anexo XVI.		
Grupo II - Requisitos relativos à conceção e ao fabrico		
Propriedades químicas, físicas e biológicas		
7.1.1 A seleção dos materiais utilizados, nomeadamente no que respeita à toxicidade e, se for caso disso, à inflamabilidade;		***1) O dispositivo não tem propriedades químicas, físicas e biológicas.***
7.1.2 A compatibilidade recíproca entre os materiais utilizados e os tecidos, as células biológicas e os líquidos corporais, atendendo à finalidade do dispositivo;		***1) O dispositivo não tem propriedades químicas, físicas e biológicas.***
7.1.3 Sempre que aplicável, os resultados das investigações biofísicas ou de modelos cuja validade tenha sido previamente demonstrada.		***1) O dispositivo não tem propriedades químicas, físicas e biológicas.***

7.2 Os dispositivos devem ser concebidos, fabricados e acondicionados por forma a minimizar os riscos relativos a contaminantes e resíduos no que respeita ao pessoal envolvido no transporte, armazenamento e utilização, bem como no que se refere aos doentes, tendo em conta a finalidade do produto, devendo ser prestada especial atenção aos tecidos expostos, bem como à duração e frequência da exposição.		***1) O dispositivo não tem propriedades químicas, físicas e biológicas.***
7.3 Os dispositivos devem ser concebidos e fabricados por forma a poderem ser utilizados em segurança com os materiais, substâncias ou gases com que entrem em contacto no decurso da sua utilização normal ou de processos de rotina e, caso se destinem à administração de medicamentos, devem ser concebidos e fabricados de modo a serem compatíveis com os medicamentos em questão, de acordo com as disposições e restrições que regem esses produtos, de modo que o seu nível de adequação se mantenha conforme à finalidade prevista.		***1) O dispositivo não tem propriedades químicas, físicas e biológicas.***
7.4 Quando um dispositivo inclua, como parte integrante, uma substância que, quando utilizada separadamente, possa ser considerada um medicamento nos termos do Decreto-Lei n.º 176/2006, de 30 de agosto, que procedeu à transposição da Diretiva n.º 2001/83/CE, do Parlamento Europeu e do Conselho, de 6 de novembro, e possa ter efeitos sobre o corpo humano através de uma ação acessória à do dispositivo, deve-se verificar a qualidade, segurança e utilidade da substância, de forma análoga aos métodos previstos no anexo I da Diretiva n.º 2001/83/CE, do Parlamento Europeu e do Conselho, de 6 de novembro (anexo I do Decreto-Lei n.º 176/206, de 30 de agosto).		***1) O dispositivo não tem propriedades químicas, físicas e biológicas.***
7.7 Os dispositivos devem ser concebidos e fabricados por forma a		***1) O dispositivo não tem propriedades***

reduzirem a um mínimo os riscos colocados pela libertação de substâncias do dispositivo, devendo ser concedida especial atenção a substâncias cancerígenas, mutagénicas ou tóxicas para a reprodução, em conformidade com o anexo I da Diretiva 67/548/CEE, do Conselho, de 27 de junho, relativa à aproximação das disposições legislativas, regulamentares e administrativas respeitantes à classificação, embalagem e rotulagem das substâncias perigosas. (Decreto-Lei n.º 280-A/87, de 17 de julho, que estabelece medidas relativas à notificação de substâncias químicas e à classificação, embalagem e rotulagem de substâncias perigosas; Decreto-Lei n.º 82/95, de 22 de abril, que transpõe para a ordem jurídica interna várias diretivas que alteram a Diretiva n.º 67/548/CEE, do Conselho, de 27 de junho, relativa à aproximação das disposições legislativas, regulamentares e administrativas respeitantes à classificação, embalagem e rotulagem de substâncias perigosas; Portaria n.º 732-A/96, de 11 de novembro, que aprova o Regulamento para a Notificação de Substâncias Químicas e para a Classificação, Embalagem e Rotulagem de Substâncias Perigosas.)		***químicas, físicas e biológicas.***
7.7.1 No caso de partes do dispositivo (ou o próprio dispositivo) destinadas a administrar medicamentos, líquidos corporais ou outras substâncias no corpo humano e, ou, a removê-las do corpo humano, ou dispositivos destinados ao transporte e ao armazenamento desses fluidos ou substâncias corporais, contenham ftalatos que sejam classificados como cancerígenos, mutagénicos ou tóxicos para a reprodução, da categoria 1 ou 2, em conformidade com o anexo I da Diretiva n.º 67/548/CEE, do Conselho, de 27 de junho, deve ser aposta na rotulagem do próprio dispositivo e ou na embalagem de cada		***1) O dispositivo não tem propriedades químicas, físicas e biológicas.***

unidade ou, se for caso disso, na embalagem de venda, uma indicação de que se trata de um dispositivo que contém ftalatos.		
7.7.2 Se a utilização pretendida desses dispositivos incluir o tratamento de crianças ou o tratamento de mulheres grávidas ou em aleitamento, o fabricante deve fornecer uma justificação específica para a utilização dessas substâncias no que se refere ao cumprimento dos requisitos essenciais, nomeadamente dos constantes no presente número e nos n.os 7.7 e 7.7.1, na documentação técnica e nas instruções de utilização sobre os riscos residuais para estes grupos de doentes e, se for caso disso, as medidas de precaução adequadas.		***1) O dispositivo não tem propriedades químicas, físicas e biológicas.***
7.8 Os dispositivos devem ser concebidos e fabricados por forma a reduzir ao mínimo os riscos derivados da introdução não intencional de substâncias no dispositivo, tendo em conta o próprio dispositivo e a natureza do meio em que se destina a ser utilizado.		***1) O dispositivo não tem propriedades químicas, físicas e biológicas.***
Infeção e contaminação microbiana		
8.1 Os dispositivos e os respetivos processos de fabrico devem ser concebidos por forma a eliminar ou reduzir, tanto quanto possível, o risco de infeção para o doente, utilizador ou para terceiros, permitir a sua fácil manipulação e, se for caso disso, minimizar a contaminação do dispositivo pelo doente, e vice-versa, no decurso da utilização.		***1) O software não apresenta caraterísticas físicas, não promove infeções nem contaminações.***
8.2 Os tecidos de origem animal devem ser provenientes de animais que tenham sido submetidos a controlos veterinários e a medidas de fiscalização adequadas à utilização prevista para os tecidos, devendo os organismos notificados recolher e manter a informação sobre a origem geográfica dos animais.		***1) O software não apresenta caraterísticas físicas, não promove infeções nem contaminações.***
8.3 Os dispositivos que são fornecidos estéreis devem ser concebidos, fabricados e acondicionados numa embalagem descartável e, ou, em		***1) O software não apresenta caraterísticas físicas, não promove infeções nem contaminações.***

conformidade com procedimentos adequados, por forma a estarem estéreis aquando da sua colocação no mercado e a manterem este estado nas condições previstas de armazenamento e transporte até que seja violada ou aberta a proteção que assegura a esterilidade.		***1) O** software **não apresenta caraterísticas físicas, não promove infeções nem contaminações.***
8.4 Os dispositivos fornecidos estéreis devem ter sido fabricados e esterilizados segundo o método apropriado e validado.		***1) O** software **não apresenta caraterísticas físicas, não promove infeções nem contaminações.***
8.5 Os dispositivos destinados a serem esterilizados devem ser fabricados em condições, nomeadamente de caráter ambiental, adequadas e controladas.		***1) O** software **não apresenta caraterísticas físicas, não promove infeções nem contaminações.***
8.6 Os sistemas de embalagem para dispositivos não estéreis devem conservar o produto sem deterioração do grau de limpeza previsto e, caso se destinem a ser esterilizados antes da utilização, devem minimizar o risco de contaminação microbiana, bem como adequar-se ao método de esterilização indicado pelo fabricante.		***1) O** software **não apresenta caraterísticas físicas, não promove infeções nem contaminações.***
8.7 A embalagem e rotulagem do dispositivo deve permitir distinguir produtos idênticos e análogos vendidos sob a forma esterilizada e não esterilizada.		***1) O** software **não apresenta caraterísticas físicas, não promove infeções nem contaminações.***
Propriedades relativas ao fabrico e condições ambientais		
9.1 Caso um dispositivo se destine a ser utilizado em conjunto com outros dispositivos ou equipamentos, esse conjunto, incluindo o sistema de ligação, deve ser seguro e não prejudicar os níveis de funcionamento previstos, devendo qualquer restrição à utilização ser especificada na rotulagem ou nas instruções.		***1) O** software **não apresenta caraterísticas físicas.***
9.2.1 Os riscos de lesão devidos às suas características físicas, incluindo a relação pressão-volume, e às suas características dimensionais e, eventualmente, ergonómicas;		***1) O** software **não apresenta caraterísticas físicas.***
9.2.2 Os riscos decorrentes de condições ambientais razoavelmente		***1) O** software **não apresenta caraterísticas físicas.***

previsíveis, nomeadamente campos magnéticos, influências elétricas externas, descargas eletrostáticas, pressão, temperatura ou variações de pressão e de aceleração;		
9.2.3 Os riscos de interferência recíproca com outros dispositivos normalmente utilizados nas investigações ou para um determinado tratamento;		*1) O software não apresenta caraterísticas físicas.*
9.2.4 Os riscos resultantes do envelhecimento dos materiais utilizados ou da perda de precisão de qualquer mecanismo de medição ou de controlo, quando não seja possível a manutenção ou calibração (como no caso dos dispositivos implantáveis).		*1) O software não apresenta caraterísticas físicas.*
9.3 Os dispositivos devem ser concebidos e fabricados por forma a minimizar os riscos de incêndio ou explosão em condições normais de utilização ou em situação de primeira avaria, devendo prestar-se especial atenção aos dispositivos cuja utilização implique a exposição a substâncias inflamáveis ou a substâncias suscetíveis de favorecer a combustão.		*1) O software não apresenta caraterísticas físicas.*
Dispositivos com função de medição		
10.1 Os dispositivos com funções de medição devem ser concebidos e fabricados por forma a assegurarem uma suficiente constância e exatidão das medições dentro de limites adequados, atendendo à finalidade dos dispositivos, e indicados pelo fabricante.	***1) Escala milimétrica.*** ***2) Ver Manual do Utilizador.***	
10.2 A escala de medição, de controlo e de leitura deve ser concebida de acordo com princípios ergonómicos e atendendo à finalidade dos dispositivos.	***1) Escala milimétrica.*** ***2) Ver Manual do Utilizador.***	
10.3 As medições feitas por dispositivos com funções de medição devem ser expressas em unidades legais, em conformidade com o disposto na legislação aplicável.	***1) Escala milimétrica.*** ***2) Ver Manual do Utilizador.***	

Proteção contra radiações		
11.1 Os dispositivos são concebidos e fabricados por forma a reduzir ao nível mínimo compatível com o objetivo pretendido a exposição dos doentes, dos utilizadores e de terceiros à emissão de radiações, sem no entanto restringir a aplicação das doses prescritas como apropriadas para efeitos terapêuticos ou de diagnóstico.		***1) O software não apresenta caraterísticas físicas.***
11.2 No caso dos dispositivos concebidos para emitir níveis de radiações com um objetivo médico específico, cujo benefício se considere ser superior aos riscos inerentes à emissão, deve ser possível ao utilizador controlar as radiações, devendo tais dispositivos ser concebidos e fabricados por forma a garantir a reprodutibilidade dos parâmetros variáveis e as respetivas tolerâncias.		***1) O software não apresenta caraterísticas físicas.***
11.3 Os dispositivos que se destinam a emitir radiações visíveis ou invisíveis potencialmente perigosas devem ser equipados, sempre que possível, com indicadores visuais ou sonoros de tais emissões.		***1) O software não apresenta caraterísticas físicas.***
11.4 Os dispositivos devem ser concebidos e fabricados por forma a reduzir o mais possível a exposição de doentes, utilizadores e terceiros à emissão de radiações não intencionais, parasitas ou difusas.		***1) O software não apresenta caraterísticas físicas.***
11.5 As instruções de utilização dos dispositivos que emitem radiações devem conter informações pormenorizadas sobre a natureza das radiações emitidas, os meios de proteção do paciente e do utilizador, a maneira de evitar manipulações erróneas e eliminar os riscos inerentes à instalação.		***1) O software não apresenta caraterísticas físicas.***
11.6 Os dispositivos destinados a emitir radiações ionizantes devem ser concebidos e fabricados por forma a garantir que, sempre que possível, a quantidade, a geometria e a qualidade da radiação emitida possam ser reguladas e controladas em função da finalidade.		***1) O software não apresenta caraterísticas físicas.***

11.6.1 Os dispositivos que emitem radiações ionizantes destinados ao diagnóstico radiológico devem ser concebidos e fabricados por forma a proporcionar uma imagem adequada e, ou, de qualidade para os fins médicos pretendidos, embora com uma exposição às radiações tão baixa quanto possível, tanto do doente como do utilizador.		*1) O software não apresenta caraterísticas físicas.*
11.6.2 Os dispositivos que emitem radiações ionizantes destinados à radioterapia devem ser concebidos e fabricados por forma a permitir a supervisão e um controlo fiáveis da dose administrada, do tipo e energia do feixe e, se for caso disso, da qualidade da radiação.		*1) O software não apresenta caraterísticas físicas.*
Dispositivos médicos ligados a uma fonte de energia ou que dela disponham como equipamento		
12.1 Os dispositivos que integrem sistemas eletrónicos programáveis devem ser concebidos de modo a garantir a recetibilidade, a fiabilidade e o nível de funcionamento desses sistemas, de acordo com a respetiva finalidade, devendo, em caso de avaria, ser adotadas medidas adequadas para eliminar, ou reduzir tanto quanto possível, os riscos que dela possam advir, sendo que no respeitante a dispositivos que incorporem um software ou que sejam eles próprios um software com finalidade médica, este deve ser validado de acordo com o estado da técnica, tendo em consideração os princípios do ciclo de vida, do desenvolvimento, da gestão dos riscos, da validação e da verificação.		*1) Não Integra sistemas eletrónicos programáveis.*
12.2 Os dispositivos que integram uma fonte de energia interna de que dependa a segurança do doente devem dispor de meios que permitam determinar o estado dessa fonte.		*1) O software não tem qualquer ligação com o doente.*
12.3 Os dispositivos ligados a uma fonte de energia externa de que dependa a segurança do doente devem dispor de um sistema de alarme que indique qualquer eventual falta de energia.		*1) O software não tem qualquer ligação com o doente.*

12.4 Os dispositivos destinados à fiscalização de um ou mais parâmetros clínicos de um doente devem dispor de sistemas de alarme adequados que permitam alertar o utilizador para situações suscetíveis de provocar a morte ou uma deterioração grave do estado da saúde do doente.		*1) O software não tem qualquer ligação com o doente.*
12.5 Os dispositivos devem ser concebidos e fabricados por forma a minimizar os riscos decorrentes da criação de campos eletromagnéticos suscetíveis de afetar o funcionamento de outros dispositivos ou equipamentos instalados no meio ambiente.		*1) O software não tem qualquer ligação com o doente.*
• **Proteção contra riscos elétricos**		
12.6 Os dispositivos devem ser concebidos e fabricados por forma a evitar, tanto quanto possível, os riscos de choques elétricos não intencionais em condições normais de utilização e em situações de primeira avaria, desde que os dispositivos estejam corretamente instalados.		*1) O software não apresenta caraterísticas físicas.*
• **Proteção contra riscos mecânicos e térmicos**		
12.7.1 Os dispositivos devem ser concebidos e fabricados por forma a proteger o doente e o utilizador contra riscos mecânicos relacionados, por exemplo, com a resistência, a estabilidade e as peças móveis.		*1) O software não apresenta caraterísticas físicas.*
12.7.2 Os dispositivos devem ser concebidos e fabricados por forma a minimizar, na medida do possível, os riscos decorrentes das vibrações por eles produzidas, atendendo ao progresso técnico e à disponibilidade de redução das vibrações, especialmente na fonte, exceto no caso de as vibrações fazerem parte do funcionamento previsto.		*1) O software não apresenta caraterísticas físicas.*
12.7.3 Os dispositivos devem ser concebidos e fabricados por forma a minimizar, na medida do possível, os riscos decorrentes do ruído		*1) O software não apresenta caraterísticas físicas.*

produzido, atendendo ao progresso técnico e à disponibilidade de meios de redução do ruído produzido, designadamente na fonte, exceto no caso de as emissões sonoras fazerem parte do funcionamento previsto.		
12.7.4 Os terminais e dispositivos de ligação às fontes de energia elétrica, hidráulica, pneumática ou gasosa que devam ser manipulados pelo utilizador devem ser concebidos e construídos por forma a minimizar os riscos eventuais.		***1) O software não apresenta caraterísticas físicas.***
12.7.5 Em condições normais de utilização, as partes acessíveis dos dispositivos, excluindo as partes ou zonas destinadas a fornecer calor ou atingir determinadas temperaturas e o meio circundante, não devem atingir temperaturas suscetíveis de constituir perigo nas condições normais de utilização.		***1) O software não apresenta caraterísticas físicas.***
• **Proteção contra os riscos inerentes ao fornecimento de energia ou administração de substâncias aos doentes**		
12.8.1 A conceção e a construção dos dispositivos destinados a fornecer energia ou administrar substâncias aos doentes devem permitir que o débito seja regulado e mantido com precisão suficiente para garantir a segurança do doente e do utilizador.		***1) O software não apresenta caraterísticas físicas.***
12.8.2 Os dispositivos devem ser dotados de meios que permitam impedir e, ou, assinalar qualquer deficiência no débito que seja suscetível de constituir um perigo, devendo os dispositivos incorporar sistemas adequados que permitam, tanto quanto possível, evitar que os débitos de energia e, ou, substâncias fornecidos pela respetiva fonte de alimentação atinjam, acidentalmente, níveis perigosos.		***1) O software não apresenta caraterísticas físicas.***
12.8.3 A função dos comandos e indicadores deve encontrar-se claramente indicada nos dispositivos e, sempre que um dispositivo		***1) O software não apresenta caraterísticas físicas.***

contenha instruções de funcionamento ou indique parâmetros de funcionamento ou de regulação através de um sistema visual, essas informações devem ser claras para o utilizador e, se for caso disso, para o doente.		
Informações fornecidas pelo fabricante		
13.1 Cada dispositivo deve ser acompanhado das informações necessárias para a sua correta utilização e com segurança e para a identificação do fabricante, tendo em conta a formação e os conhecimentos dos potenciais utilizadores, devendo essas informações ser constituídas pelas indicações constantes da rotulagem e do folheto de instruções.	***2) Ver Manual do Utilizador.***	
13.2 As informações necessárias para a utilização do dispositivo com toda a segurança devem figurar, se exequível e adequado, no próprio dispositivo e, ou, na embalagem individual, ou, eventualmente, na embalagem comercial, mas, se os dispositivos não puderem ser embalados individualmente, as informações devem constar de um folheto de instruções que acompanhe um ou mais dispositivos.	***2) Ver Manual do Utilizador.***	
13.3 Todos os dispositivos devem ser acompanhados de um folheto de instruções, incluído nas respetivas embalagens, sem prejuízo da possibilidade de, a título excecional, o referido folheto de instruções não ser incluído para dispositivos das classes I e IIa, desde que a respetiva segurança de utilização possa ser garantida sem ele.	***2) Ver Manual do Utilizador.***	
13.4 Sempre que adequado, as informações devem ser apresentadas sob a forma de símbolos, os quais, bem como as respetivas cores de identificação, devem estar em conformidade com as normas harmonizadas, ou devem ser descritos na documentação que acompanha o dispositivo, nos domínios em que não existam quaisquer	***1) Ver Manual do Utilizador.***	

normas.		
• **Rotulagem**		
13.5.1 O nome, ou a firma e o endereço do fabricante, sendo que, relativamente aos dispositivos importados para serem distribuídos na União Europeia, o rótulo, a embalagem exterior ou as instruções de utilização devem ainda incluir o nome e o endereço do mandatário do fabricante, sempre que o fabricante não dispuser de sede social na União Europeia;	***1) Informações nas propriedades do software.***	
13.5.2 As informações estritamente necessárias para que o utilizador possa identificar o dispositivo e o conteúdo da embalagem, em especial para os utilizadores;	***1) Informações nas propriedades do software.***	
13.5.3 Se aplicável, a menção «Estéril»;		***1) O dispositivo não apresenta caraterísticas físicas.***
13.5.4 Se aplicável, o código do lote, precedido da menção «Lote», ou o número de série;		
13.5.5 Se aplicável, a data limite de utilização do dispositivo em condições de segurança, expressa pelo ano e mês;		***1) O software não tem data limite de utilização.***
13.5.6 Sempre que aplicável, uma indicação de que o dispositivo é para utilização única, sendo que a indicação do fabricante sobre a utilização única deve ser uniforme em toda a União Europeia;		***1) O dispositivo não é para uso único.***
13.5.7 Para os dispositivos feitos por medida, a menção «Dispositivo feito por medida»;		***1) O dispositivo não foi feito por medida.***
13.5.8 Para os dispositivos destinados à investigação clínica, a menção «Exclusivamente para investigação clínica»;		***1) O software não é para investigação clínica***
13.5.9 Condições especiais de armazenamento e, ou, manuseamento;		***1) O software não apresenta caraterísticas físicas.***
13.5.10 Instruções particulares de utilização;	***2) Ver Manual do Utilizador.***	
13.5.11 Advertências ou precauções a tomar;	***2) Ver Manual do Utilizador.***	
13.5.12 O ano de fabrico para os dispositivos ativos não abrangidos no		

n.º 13.5.5 supra, indicação que pode ser incluída no número do lote ou de série;		
13.5.13 Se aplicável, o método de esterilização;		***1) O dispositivo não apresenta caraterísticas físicas.***
13.5.14 No caso de um dispositivo na aceção do disposto na alínea b) do n.º 2 do artigo 2.º do decreto-lei de que o presente anexo é parte integrante, a menção de que o dispositivo incorpora como parte integrante uma substância derivada do sangue humano.		
13.6 Caso a finalidade prevista de um dispositivo não seja evidente para o utilizador, o fabricante deve especificá-la claramente na rotulagem e nas instruções de utilização.	***1) Informações nas propriedades do software.*** ***2) Ver Manual do Utilizador.***	
13.7 Os dispositivos e os componentes destacáveis devem, se tal se justificar e for exequível, ser identificados em termos de lotes, por forma a possibilitar a realização de ações destinadas a detetar riscos ocasionados pelos dispositivos e pelos componentes destacáveis.		***1) O dispositivo não apresenta componentes destacáveis.***
• **Instruções de utilização**		
13.8.1 As indicações referidas no n.º 13.5, exceto as constantes dos n.os 13.5.4 e 13.5.5;	***1) Informações nas propriedades do software.*** ***2) Ver Manual do Utilizador.***	
13.8.2 Os níveis de adequação referidos no n.º 3, bem como quaisquer efeitos secundários indesejáveis;	***2) Ver Manual do Utilizador.***	
13.8.3 Caso um dispositivo deva ser instalado em ou ligado a outros dispositivos ou equipamentos médicos, para funcionar de acordo com a finalidade prevista, devem ser fornecidos pormenores suficientes das suas características de modo a permitir identificar os dispositivos ou os equipamentos que devem ser utilizados para que se obtenha uma combinação segura;	***2) Ver Manual do Utilizador.***	
13.8.4.1 As instruções de calibração e o manual de manutenção, sempre que aplicável aos produtos em causa;	***1) Ver Manual do Utilizador.***	

13.8.5 Se aplicável, informações úteis para evitar determinados riscos decorrentes da implantação do dispositivo;	***2) Ver Manual do Utilizador.***	
13.8.6 Informações relativas aos riscos de interferência recíproca decorrentes da presença do dispositivo aquando de investigação ou tratamentos específicos;		***1) O dispositivo não apresenta caraterísticas físicas.***
13.8.7 As instruções necessárias em caso de danificação da embalagem que assegura a esterilidade e, se necessário, a indicação dos métodos adequados para se proceder a uma nova esterilização;	***2) Ver Manual do Utilizador.***	
13.8.8 Caso o dispositivo seja reutilizável, informações sobre os processos de reutilização adequados, incluindo a limpeza, desinfeção, acondicionamento e, se for caso disso, método de reesterilização, se o dispositivo tiver de ser novamente esterilizado, bem como quaisquer restrições quanto ao número possível de reutilizações;		***1) O dispositivo não apresenta caraterísticas físicas.***
13.8.9 Caso os dispositivos sejam fornecidos com a condição de serem previamente esterilizados, as instruções relativas à limpeza e esterilização devem ser de molde a garantir que, se forem corretamente respeitadas, o dispositivo satisfaça os requisitos gerais referidos na secção i do presente anexo;		***1) O dispositivo não apresenta caraterísticas físicas.***
13.8.10 Se o dispositivo indicar se destina a utilização única, informações sobre as características conhecidas e os fatores técnicos de que o fabricante tem conhecimento que podem constituir um risco no caso de o dispositivo ser novamente utilizado e, se em conformidade com o n.º 13.3, não sejam necessárias instruções de utilização, as informações devem ser facultadas ao utilizador, a seu pedido;		***1) O dispositivo não é para uso único.***
13.8.11 Caso um dispositivo deva ser submetido a um tratamento ou operação adicional antes de ser utilizado (por exemplo, esterilização,		***1) O dispositivo não apresenta caraterísticas físicas.***

montagem final, etc.), as indicações sobre esse tratamento ou operação;		
13.8.12 Caso um dispositivo emita radiações para fins médicos, as informações relativas à natureza, tipo, intensidade e distribuição das referidas radiações.		***1) O software não apresenta caraterísticas físicas.***
13.9.1 As precauções a tomar em caso de alteração do funcionamento do dispositivo;	***2) Ver Manual do Utilizador.***	
13.9.2 As precauções a tomar no que respeita à exposição, em condições ambientais razoavelmente previsíveis, a campos magnéticos, a influências elétricas externas, a descargas eletrostáticas, à pressão ou às variações de pressão, à aceleração, a fontes térmicas de ignição, etc.;		***1) O software não apresenta caraterísticas físicas.***
13.9.3 Informações adequadas sobre os medicamentos que o dispositivo em questão se destina a administrar, incluindo quaisquer limitações à escolha dessas substâncias;		***1) O software não apresenta caraterísticas físicas.***
13.9.4 As precauções a tomar caso o dispositivo apresente um risco especial ou anormal no que respeita à sua eliminação;	***2) Ver Manual do Utilizador.***	
13.9.5 Os medicamentos ou as substâncias derivadas do sangue humano incorporados no dispositivo como sua parte integrante, em conformidade com os n.os 7.4 e 7.5;		***1) O software não apresenta caraterísticas físicas.***
13.9.6 O grau de precisão exigido para os dispositivos de medição;	***2) Ver Manual do Utilizador.***	
13.9.7 A data da publicação ou da última revisão das instruções de utilização.	***2) Ver Manual do Utilizador.***	

Nota: Os pontos referem-se ao Decreto-Lei nº 145/2009, Anexo I

Os métodos usados para cumprimento podem advir de:

a) conformidade com normas reconhecidas e/ou outras
b) conformidade com métodos de teste industriais comummente aceites
c) conformidade com método desenvolvido pela empresa
d) avaliação dos dados pré-clínicos e clínicos
e) comparação com dipositivo semelhante já disponível no mercado

Instituto Superior de
Engenharia do Porto

Instituto Superior de Engenharia do Porto ⊙ Rua Dr. António Bernardino de Almeida, 431 ⊙ 4200-072 Porto
Tel. +351 228 340 500 ⊙ Fax +351 228 321 159 ⊙ mail@isep.ipp.pt ⊙ www.isep.ipp.pt