UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

Rafael Simon Maia

Cyclops Series Editor: Uma ferramenta de visualização de imagens e auxílio ao laudo e ao diagnóstico

Florianópolis, fevereiro de 2004.

Rafael Simon Maia

Cyclops Series Editor: Uma ferramenta de visualização de imagens e auxílio ao laudo e ao diagnóstico

Trabalho de Conclusão de Curso apresentado
à disciplina do Curso de Ciência da
Computação da Universidade Federal de Santa
Catarina para obtenção do título de bacharel.

Florianópolis, fevereiro de 2004.

Resumo

O surgimento e popularização dos exames radiológicos baseados em tecnologia digital, sua posterior padronização pelo padrão DICOM e o conseqüente barateamento dos produtos relacionados a informática durante a década de oitenta e noventa levou a viabilização e o surgimento de sistemas de arquivamento de informações hospitalares (PACs) e o aumento de sua utilização. A expansão da capacidade computacional também permitiu que um campo já consolidado, a informática médica, crescesse ainda mais e produzissem ferramentas de diagnostico médico de qualidade. Neste contexto, surgiu o projeto Cyclops, que além de tratar da criação e manutenção de PACS também se encarrega do desenvolvimento de aplicações para o auxílio ao diagnóstico.

Com a necessidade de se fornecer uma ferramenta completa, que proporcionasse capacidade de visualização de imagens médicas juntamente com um conjunto de ferramentas de auxílio ao diagnóstico surgiu a idéia de se unir a então ferramenta Cyclops Series Editor com o conjunto de software anteriormente construídos no projeto. Surgiu então uma ferramenta única, com capacidades que nenhuma outra ferramenta similar apresenta no momento, e a custo zero.

Palavras-chave: DICOM, imagens médicas, cliente de imagens médicas, Cyclops, Cyclops Series Editor, Cyclops Brain Atlas, Cyclops PopCorn , Cyclops Stroke, Cyclops Structured Report.

Abstract

With the sprouting and popularization of the digital radiological systems, its posterior standardization in the DICOM standard and the price reduction of computer science related product in the nineties, it became possible the use of the picture archiving

communication system (PACS). The expansion of the computational capacity also allowed that a already consolidated filed, medical informatic, to grow up and produced better quality diagnosiss tools .

At this context, appeared the Cyclops Project, that beyond treat the creation and maintenance of PACS, to put it self of the application development to aid the exams diagnosis.

With the necessity of supplying a complete tool, that provided capacity of visualization of medical images together with a set of aid tools to the diagnosis appeared the idea of joining the tool Cyclops Series Editor with group of software previously build in the project. A new tool appeared then, with capacities that none another similar tool presents at the moment, and cost zero.

Keywords: DICOM, medical images, Cyclops, Cyclops Series Editor, Cyclops Brain Atlas, Cyclops PopCorn , Cyclops Stroke, Cyclops Structured Report.

Sumário

RESUMO **3**

ABSTRACT **4**

1 INTRODUÇÃO **8**

1.1 O PROJETO CYCLOPS **11**
1.2 MOTIVAÇÃO **11**
1.3 OBJETIVOS **12**
1.3.1 OBJETIVOS GERAIS 12
1.3.2 OBJETIVOS ESPECÍFICOS 12
1.4 METODOLOGIA **14**

2 IMAGEM MÉDICA DIGITAL **15**

2.1 ACR/NEMA 1.0 E ACR/NEMA 2.0 ..... **15**
2.2 DICOM 3 ..... **16**
2.3 ESTRUTURA DE INFORMAÇÃO DICOM **19**

3 ESTADO DA ARTE **23**

3.1 FERRAMENTAS EXISTENTES **23**
3.1.1 E-FILM 23
3.1.2 EZDICOM 25
3.1.3 OSIRIS 26
3.1.4 CYCLOPS PERSONAL 26
3.1.5 DICOMSCOPE 27
3.2 ANALISE DO ESTADO DA ARTE **29**

4 O CYCLOPS SERIES EDITOR E SEU SISTEMA DE COMPONENTES **30**

4.1 MODULARIZAÇÃO DO CYCLOPS SERIES EDITOR **31**
4.2 COMO FAZER UM PLUG-IN DE BAIXO ACOPLAMENTO **33**
4.3 COMO FAZER UM PLUG-IN DE ALTO ACOPLAMENTO **33**

5 AS FERRAMENTAS DO CYCLOPS SERIES **35**

5.1 DAS FERRAMENTAS DE PROCESSAMENTO DE IMAGENS **36**
5.1.1 DAS CARACTERÍSTICAS DAS FERRAMENTAS DE PROCESSAMENTO DE IMAGENS 37
5.1.2 AS FERRAMENTAS DE SUPORTE AO LAUDO 43
5.2 OUTRAS FERRAMENTAS **51**
5.2.1 BARRA DE APLICAÇÕES 52
5.2.2 O CYCLOPS MAILER 57
5.2.3 O CYCLOPS IMAGE PRINTER 58
5.2.4 A FERRAMENTA DE MENSURAÇÃO DE ÁREA 58

6 CONSIDERAÇÕES SOBRE PERFORMANCE **59**

6.1 REQUISITOS DE HARDWARE 59
6.2 REQUISITOS DE SOFTWARE 59

7 CONCLUSÕES 60

8 GLOSSÁRIO 61

9 REFERENCIAS BIBLIOGRAFICAS 62

ANEXO 1 - MANUAL DE CONFIGURAÇÃO 64

9.1 CONFIGURANDO O CYCLOPS MEDICAL IMAGE BROWSER 65
9.2 CONFIGURANDO O CYCLOPS MAILER 70
9.3 CONFIGURANDO O CYCLOPS STRUCTURED REPORT 71

ANEXO 2 - MANUAL BÁSICO DE UTILIZAÇÃO DO CYCLOPS SERIES EDITOR E SUAS FERRAMENTAS 73

9.4 UTILIZANDO O CYCLOPS STRUCTURED REPORT 73
9.5 OUTROS BOTÕES 74
9.6 USANDO O CYCLOPS STROKE. 76

ANEXO 3 – ANALISE DE REQUISITOS 77

9.6.1 ORGANOGRAMA 77
9.6.2 DESCRIÇÃO TEXTUAL DOS PROCESSOS. 77
9.7 CASOS DE USO EXTENDIDOS 79
9.8 REQUISITOS 85
9.8.1 REQUISITOS FUNCIONAIS: 85
9.8.2 REQUISITOS NÃO FUNCIONAIS: 86
9.8.3 INTERFACES DO SISTEMA 87

ANEXO 4 – ARTIGO 90

1 RESUMO 90

1. INTRODUÇÃO 90

Lista de Figuras

Figura 1 Um PACS com seus principais componentes. 10
Figura 2 Arquitetura de protocolos do DICOM 3 16
Figura 3 Visualização da estrutura de informação no Cliente Cyclops MIB..... 21
Figura 4 Interface do e-film. Visualizando uma tomografia..... 24
Figura 5 Visualizando a reconstrução tridimensional da mesma 24
Figura 6 Interface do ezDicom..... 25
Figura 7 Interface do Osíris..... 26
Figura 8 Interface do Osíris..... 26
Figura 10 Interface do Cyclops Personal. 27
Figura 11 Interface do Cyclops Personal. 27
Figura 12 Interface de visualização de documentos DICOM SR do DICOMScope 28
Figura 13 Interface de visualização de documentos DICOM SR do DICOMScope 28
Figura 14 Tomografia computadorizada do abdômen original..... 39
Figura 15 Tomografia computadorizada depois de aplicado o algoritmo de Munford-Sha .. 39
Figura 16 Tomografia computadorizada original..... 40
Figura 17 Tomografia computadorizada depois de aplicado o threshold..... 40
Figura 18 Tomografia computadorizada do abdômen original..... 42
Figura 19 Tomografia computadorizada depois de aplicado o Canny 42
Figura 20 Tomografia computadorizada original..... 43
Figura 21 Tomografia computadorizada depois de aplicado o VDRF..... 43
Figura 22 – Organograma básico de funcionamento do Cyclops Series Editor 77
Figura 23 - Interface do Cyclops Medical Image Browser..... 87
Figura 24 – Interface inicial do Cyclops Series Editor..... 87
Figura 25 Exibição das recontruções multiplanares de uma serie de tomografias..... 88
Figura 26 Ajuste de window radiológica..... 88
Figura 27 – Interface de visualização de uma única imagem (como algoritmo canny aplicado) 89
Figura 28 – Ferramenta de seleção de formato de exportação de imagem..... 89
Figura 29 – Interface do Cyclops Series Editor com o Cyclops Brain Atlas funcionando em conjuto..... 44
Figura 30 – Interface do Cyclops Series Editor funcionando em conjunto com o Cyclops Structured Report Editor : fácil criação de laudos estruturados. 46
Figura 31 Inteface do Cyclops Series Editor em conjunto com a ferramenta Cyclops Stroke Quantifier..... 50
Figura 32 Interface de ajuste e preview da window radiologica..... 54
Figura 33 – Interface de visualização das recontruções multiplanares..... 55
Figura 34 – Interface da ferramenta de e-mail do Cyclops Series Editor..... 57

1 Introdução

Em meados da década de 70, surgiu uma nova modalidade de aparelhos radiológicos: a Tomografia Computadorizada. A partir dessa data, surgiu todo um aparato tecnológico que consistido de técnicas de armazenamento, recuperação e transmissão de imagens (arquivos digitais de imagens ao invés de apenas filme). O aprimoramento dessas tecnologias com a crescente utilização da informática em ambientes clínicos e hospitalares levaram ao surgimento de padrões, que, na maioria das vezes, empregavam tecnologias de arquivamento e transmissão completamente incompatíveis, tornando o compartilhamento de recursos dos dispositivos impossível.

Foi então que, em 1983, surgiram as primeiras idéias de estudar os padrões existentes e tentar criar um que fosse amplamente aceito por todos os fabricantes. Com essa meta em mãos, os grupos ACR (American College of Radiology) e NEMA (National Electrical Manufacturers Association) formaram um comitê associado com a missão desenvolver essa interface entre equipamentos de aquisição de imagens e facilitar o desenvolvimento expansão do PACS/HIS (do inglês Hospital Information Systems - Sistemas de Informações Hospitalares).

Além de especificações à conexão de hardware, o padrão deveria ser desenvolvido para incluir um dicionário dos elementos de dados necessários para exibir imagem própria e sua respectiva interpretação.

Assim, em 1985 a ACR-NEMA Standards Publication No. 300-1985, publicada em 1985 foi chamada de versão 1.0. Posteriormente o padrão foi revisado ainda duas vezes: em outubro de 1986 e em janeiro de 1988. A ACR-NEMA Standards Publication No. 300- 1988 foi chamada de versão 2.0. Essas publicações padrão forneciam especificações para uma interface de hardware, um conjunto mínimo de comandos de software e um conjunto consistente de formatos de dados. Esse padrão sofria de várias deficiências, principalmente de

não ter sido desenvolvido para suportar os ambientes de rede de computadores já disponíveis nas clínicas e hospitais na época de sua publicação.

Em 1992, uma nova versão, com mais capacidade de expressão e que mantinha a compatibilidade com as anteriores foi publicada, sob o novo nome de DICOM 3 (do inglês Digital Imaging and Communications in Medicine) em ACR/NEMA Standards Publication PS3.

Esse foi o passo inicial, que juntamente com a expansão acelerada da computação pessoal na década de oitenta e noventa ajudaram a expandir os Sistemas de Comunicação e Arquivamento de Imagens, mais conhecidos como PACS (do inglês, Picture Archive and Communication System).

Atualmente, o fato de um aparelho ter capacidade de exportar suas imagens digitais em DICOM 3 fornece a garantia de que ele poderá ser integrado facilmente em um PACS já existente, devido a utilização de tecnologias de redes amplamente difundidas (e portanto baratas, como TCP/IP) , sendo portanto um padrão de fato para os sistemas de automação hospitalares.

A abrangência do DICOM 3 em sistemas de arquivamento e comunicação de Imagens vai desde a codificação dos dados das imagens e informações associadas, passando pela definição de diversas classes de serviços, como armazenamento, recuperação, pesquisa e impressão de imagens, formatos utilizados no armazenamento das imagens em meios removíveis, processos de negociação de associações para a transmissão dos dados das imagens através de redes, etc. Softwares aplicativos para a área médica também podem obter vantagens ao oferecer suporte ao padrão, pelas mesmas razões acima citadas. Na figura 2 podemos ver um exemplo de um PACS com seus componentes principais.

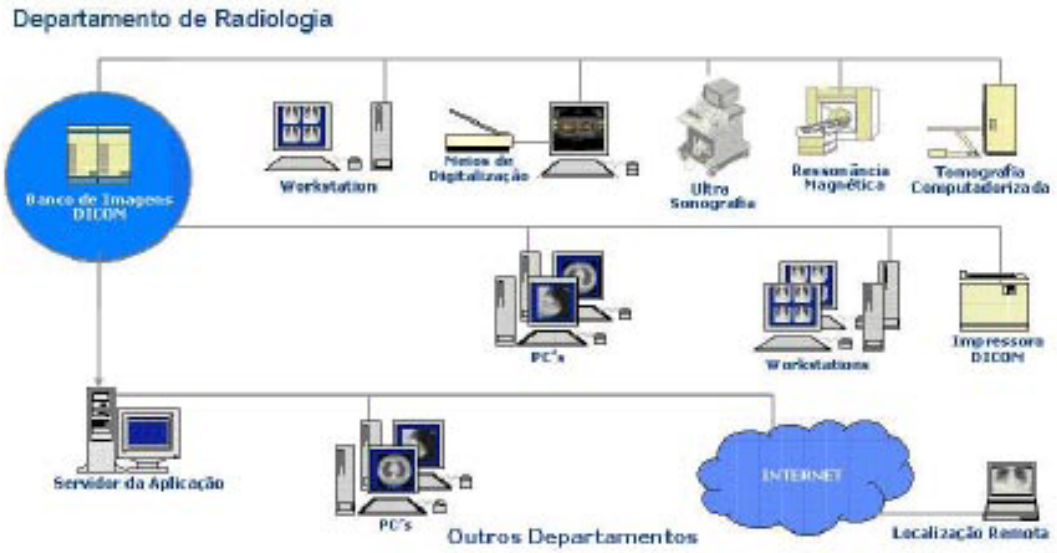


Figura 1 Um PACS com seus principais componentes.

Para que se obtenha a grande vantagem da utilização de sistemas de Comunicação e Arquivamento de Imagens em ambientes clínicos e hospitalares é necessária a utilização de boas ferramentas de visualização e recuperação dos dados armazenados no banco de dados. Dentro do Projeto Cyclops, eram se utilizadas duas ferramentas que exerciam essas funções: o *Cyclops DICOM Editor* e o *Cyclops DICOM Series Editor*. Ambas as ferramentas foram incialmente construídas por Paulo Roberto Dellani. A segunda ferramenta, então foi escolhida para não só ser remodelada como ter uma grande ampliação de funções que permitiria que ela fosse usada como uma ferramenta de visualização de imagens radiológicas, mas também como uma ampla ferramenta de auxilio ao laudo e ao diagnostico, assimilando todo um grupo de ferramentas desenvolvidas anteriormente durante o projeto cyclops.

1.1 O Projeto Cyclops

O Projeto Cyclops foi criado em 1992 como um projeto de pesquisa binacional e de longo prazo pelos professores Dr. Aldo von Wangenheim, da Universidade Federal de Santa Catarina e pelo professor Dr. Michael M. Richter, da Universidade de Kaiserslautem.

O projeto tem como objetivo desenvolver novos métodos e ferramentas de análise de imagens médicas utilizando-se da inteligência artificial e visão computacional. Também é meta do projeto desenvolver uma estrutura de serviço PACS de baixo custo para ser utilizada em hospitais e clínicas do Brasil.

No Brasil existem também parceiros, tais como a clínica DMI em São José-SC e o Hospital Universitário da UFSC.

O consorcio internacional de pesquisa e desenvolvimento é constituído por Universidades de ambos os países, parceiros industriais da área de software, parceiros médicos e empresas fabricantes de equipamentos radiológicos de ambos os países.

1.2 Motivação

Todo sistema PACS é basicamente constituído de quatro partes:

- Um banco de dados onde são armazenados os arquivos DICOM;
- Um servidor DICOM conectado ao banco de dados e que fornece a sintaxe de transferência DICOM de acordo com as especificações;
- Um cliente para se conectar ao Servidor DICOM;
- E uma ferramenta para a visualização das imagens DICOM.

O software apresentado neste trabalho de conclusão de curso se enquadra, portanto, na ultima categoria. Entretanto ele não se restringe a isso, pois era necessário que o software apresentasse uma serie de características que facilitassem o trabalho de médico na hora do laudo e do diagnóstico e que também auxiliasse na futura expansão do software.

Com isso surgiu a idéia de integrar as mais importantes ferramentas de auxilio ao diagnóstico criadas no projeto Cyclops, juntamente com uma ferramenta de leitura e criação de DICOM *Structured Reports,* fornecendo-se assim, uma ferramenta completa e fácil de usar que pudesse facilitar a vida do médico ao analisar imagens radiológicas e a partir desta analise realizar o laudo da patologia.

1.3 Objetivos

1.3.1 Objetivos Gerais

Desenvolver um software capaz de visualizar arquivos no formato DICOM, de fácil utilização e que seja facilmente expansível com o acréscimo de novas funcionalidades e que forneças as mesmas ferramentas disponíveis nos software similares, além de fornecer um conjunto de ferramentas que auxiliem no laudo e na descrição do mesmo.

1.3.2 Objetivos Específicos

Para realizar a implementação deste software foram definidos os seguintes objetivos específicos:

- Modelar e implementar um software capaz de exibir imagens DICOM;

- Fornecer nesse software funcionalidades que auxiliem os médicos no exame das imagens, utilizando-se pra isso de algoritmos de visão computacional, edição gráfica e impressão;
- Criar uma estrutura expansível, de modo que novas funcionalidades possam ser facilmente adicionadas ao software;
- Integrar alguns softwares anteriormente construídos no âmbito do Projeto Cyclops: Stroke, Brain Atlas, WaveFormViewer, PopCorn, Structured Report Editor;
- Criar um sistema de impressão e criação de páginas em html das imagens e laudos para utilização na web como portal de acesso aos laudos para os médicos e pacientes;
- Estudar o framework DICOM já implementado e estudar possíveis mudanças para melhora de performance;

Requisto:

- A implementação deve ocorrer na linguagem Smalltalk, devido a grande maioria das ferramentas desenvolvidas no projeto terem sido realizados nessa linguagem.

1.4 Metodologia

Para atingir os objetivos específicos destes trabalhos foi utilizada a seguinte metodologia:

a) Conhecer o padrão DICOM, estudando suas definições no “DraftDICOM Standard“e principalmente analisando a implementação já existente do projeto, o framework DICOM que vem sendo construído desde 1993;

b) Entender o como funcionada a exibição das imagens DICOM na tela do computador;

c) Entender outros tipos de arquivos DICOM, como ECG e SR;

d) Entender o funcionamento das outras ferramentas que serão assimiladas por esse software, já que elas posteriormente serão parte do mesmo;

e) Entender o funcionamento dos algoritmos de visão computacional utilizadas pelas ferramentas de processamento de imagens que serão utilizadas.

2 Imagem médica digital

O ramo de imagens médicas digitais tem uma série de termos técnicos e padrões, cujo esclarecimento prévio faz-se necessário antes da continuidade da leitura dos demais capítulos.

2.1 ACR/NEMA 1.0 e ACR/NEMA 2.0

A iniciativa de padronização para PACS foi tomada pelo ACR (do inglês American College of Radiology) e pelo NEMA (do inglês National Electrical Manufacturers Association) no início dos anos 80 (1983), e em 1985 a versão 1.0 do padrão foi publicada, sendo então referenciada como ACR/ NEMA Standards Publication No. 300-1985. O padrão foi revisado várias vezes e em 1988 foi publicada a versão 2.0, descrita no ACR/NEMA Standards Publication No. 300-1988. Assim, em 1988, a codificação e intercâmbio de imagens médicas digitais começaram a ser verdadeiramente padronizada. Apesar de começar a ser largamente utilizada, esta segunda versão do padrão ainda não fornecia uma comunicação confiável, em virtude das diversas modificações incorporadas a esta versão (CLUNIE, 2000).

Na verdade, o problema foi que em 1988 muitos usuários queriam uma interface entre seus dispositivos de imagem e suas redes. Apesar disto já ser permitido pela versão corrente do padrão, o mesmo ainda necessitava de melhoramentos a fim de permitir uma comunicação em rede robusta. Por exemplo, um dispositivo poderia enviar uma mensagem contendo a imagem e seu cabeçalho para um outro dispositivo qualquer, mas este último não tinha como saber o que o primeiro queria fazer com as informações contidas na mensagem. Isso acontecia pois o padrão de 1988 não foi desenvolvido exatamente para suportar comunicação em rede. Este ponto ainda era secundário para o padrão, já que naquela época a existência de redes de computadores em ambientes hospitalares eram muito pequena.

Este padrão deve sua importância principalmente ao fato de que muitos formatos proprietários e outros padrões de facto adotaram o formato de mensagens do ACR/NEMA e seu correspondente dicionário de dados e mecanismos de expansão.(CLUNIE, 2000).

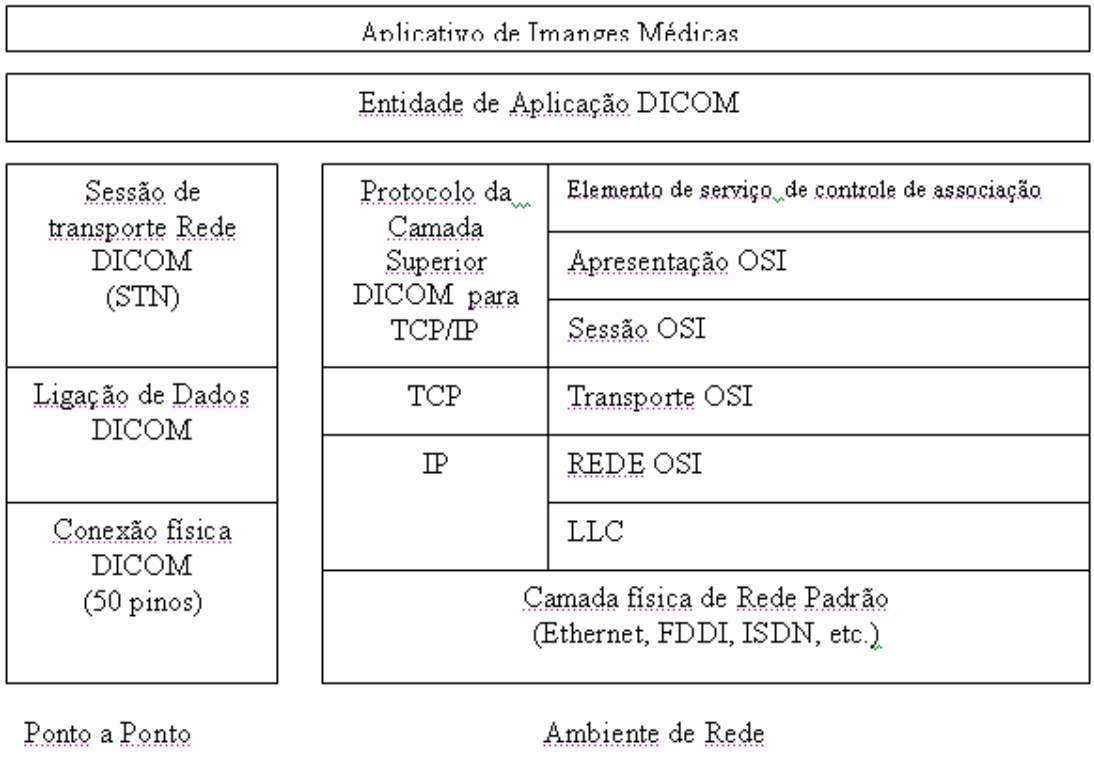


Figura 2 Arquitetura de protocolos do DICOM 3

2.2 DICOM 3

Entre 1992-1993 foi publicada uma nova versão radicalmente revisada, mas completamente compatível com o padrão anterior, sob o nome de ACR/NEMA Standards Publication S3, também referenciada como DICOM 3 (acrônimo do inglês Digital Imaging and Communications in Medicina).

Adicionalmente, um rápido exame dos tipos de serviços necessários à comunicação em rede, mostrou que a definição de uma classe de serviços básicos permitiria que um processo de alto nível (na camada de aplicação) fosse capaz de se comunicar com um grande número de diferentes protocolos de rede. Assim, deu se o desenvolvimento de uma nova versão, quase um novo projeto. Estes protocolos são modelados como uma série de camadas superpostas, também conhecidas como “pilha”. (CLUNIE, 2000).

Na versão 2.0 do DICOM já existia uma “pilha” que definia uma comunicação ponto-a-ponto. Mais tarde foram inseridas, baseadas em sua popularidade e possibilidade de expansão, as pilhas de protocolos de rede TCP/IP (do inglês Transmission Control Protocol / Internet Protocol) e a ISO/OSI (do inglês International Standards Organization / Open Systems Interconnection).

A figura 2 (página anterior) apresenta um diagrama do modelo de comunicação desenvolvido. A filosofia básica é que uma dada aplicação de imagens médicas possa se comunicar sobre qualquer das pilhas disponíveis com qualquer outro dispositivo que utilize a mesma pilha. Com esta nova filosofia, tornou-se possível a troca de pilhas, sempre que necessário, sem ter de reescrever todo o código dos programas aplicativos que utilizassem o padrão DICOM. Após três anos de trabalho, com muitas sugestões da indústria e da área acadêmica, foi dado por completo o DICOM 3.0. Esta versão é muito mais abrangente e robusta que as versões anteriores.

A sua extensão pode ser vista na maneira como foram divididos os capítulos que definem o padrão, na sua última versão publicada (2000):

- PS 3.1-2000 *Introduction and Overview*: uma breve introdução ao padrão DICOM;
- PS 3.2-2000 *Conformance*: define os princípios para todas as implementações que busquem estar em conformidade com o padrão DICOM.

- PS 3.3-2000 *Information Object Definitions*: descreve como os objetos de informação são construídos. Lista todos os grupos de Entidades de Informação e Definições de Objetos de Informações. Estes grupos são coleções e seqüências de elementos de dados.
- PS 3.4-2000 *Service Class Specifications*: contém a especificação das Classes de Serviço. Explica como várias classes de serviço são implementadas pelas entidades de aplicação. As regras de SCU (usuário do serviço) e SCP (fornecedor do serviço) também são definidas nesta parte. É uma boa referência/tutorial para programadores de aplicações DICOM.
- PS 3.5-2000 *Data Structures and Encoding*: especifica o método de codificação de dados para a transmissão e decodificação após a recepção dos dados.
- PS 3.6-2000 *Data Dictionary*: provê uma lista com todas as Tags correspondentes aos atributos dos objetos de informação que fazem parte do padrão.
- PS 3.7-2000 *Message Exchange*: descreve os Serviços DICOM comumente nreferenciados como DIMSE-C e DIMSE-N (do inglês DICOM Message Service Element - Composite / Normalized, respectivamente). Estes serviços são encapsulados pela comunicação DICOM.
- PS 3.8-2000 *Network Communication Support for Message Exchange*:especifica o processo de negociação de associação, protocolos da camada superior, transporte dos dados e estado da máquina. Este é o componente mais complexo para implementadores DICOM.
- PS 3.9-2000 *Point to Point Communication Support for Message Exchange*: especifica a utilização do padrão DICOM sobre conexões ponto-a-ponto.

- PS 3.10-2000 Media Storage and File Format for Media Exchange;
- PS 3.11-2000 Media Storage Application Profiles;
- PS 3.12-2000 *Media Formats and Physical Media for Media Interchange*: explicam os formatos utilizados para armazenar dados em meios removíveis 11 como discos WORM, visando facilitar a transferência de informações entre sistemas computadorizados para imagens digitais em ambientes médicos não interligados em rede.
- PS 3.13-2000 *Print Management Point-to-Point Communication Support*: especifica os serviços e protocolo necessários para suportar a comunicação de Entidades de Aplicação de Gerenciamento de Impressão DICOM.
- PS 3.14-2000 *Grayscale Standard Display Function*: especifica uma função de exibição padronizada para a exibição de imagens em escala de cinza. Provê exemplos de métodos para mensuração da curva característica de um sistema de exibição em particular, com propósitos de calibração do mesmo, ou para a mensuração da conformidade de um sistema de exibição à função de exibição em escala de cinza padrão.
- PS 3.15-2000 *Security Profiles*: aborda a conformidade de implementações utilizando o padrão DICOM quanto às questões de segurança.

2.3 Estrutura de Informação DICOM

O padrão DICOM não define apenas métodos para comunicação de imagens. ele também define toda uma modelagem de objetos de informação que vão além a própria imagem. Assim, a forma como as informações sobre pacientes, estudos e séries de imagens também tem uma definição de como devem ser transmitidas e armazenadas.

Assim sendo, cada imagem tem associadas a si mesma um grande conjunto de informações. Como existem várias informações que existem de forma hierárquica é possível organizar as imagens da mesma maneira.

De acordo com o padrão DICOM, é possível organizar as informações no seguinte nível hierárquico:

a) **Paciente:** é uma pessoa que recebe ou está apto a receber tratamento médico.

(PS 3.3, 2000).

b) **Estudo:** é uma coleção de séries de imagens médicas e outras informações logicamente relacionadas com o objetivo de diagnosticar um paciente. Um estudo está associado sempre a somente um paciente. (PS 3.3, 2000).

c) **Série:** são um conjunto de atributos utilizados para agrupar várias instancias de informação (comumente imagens) em diferentes conjuntos. Para agrupar várias instancias em uma série, todas as instâncias devem compartilhar o mesmo quadro de referência (frame of reference) e equipamento, caso tenham sido especificados, e ser da mesma modalidade (PS 3.3, 2000).

d) **Imagem:** além da informação visual contém também outras informações relacionadas à mesma, tais como parâmetros da máquina de captura de imagens.

Assim, cada uma dessas entidades (ou objetos) do modelo são descritas formalmente como definicoes de objetos de informação ou IODs (sigla em inglês para Information Objects Definition). Existem no padrão DICOM dois tipos de IODs: IODs normalizados e IODs compostos. Os IODs normalizados representam objetos atômicos do mundo real e modelam apenas uma entidade de informação, ou IE (sigla em inglês para Information Entity). Logo IEs são todos os objetos do mundo real do modelo que podem ser descritos através de um IOD normalizado.

Objetos do mundo real que são excessivamente complexos, entretanto , são definidos por IODs compostos. IODs compostos modelam objetos que possuem mais de uma IE, como por exemplo, uma imagem de ressonância magnética ou tomografia computadorizada. Dentro do Cyclops Series Editor, os IODs compostos foram modelados como Classes.

Conjuntos de características de objetos compostos do mundo ral que são interrrelacionadas são descritas pelo padrão DICOM como modeulos de objeto de informação ou IOM (do inglês dInformation Object Modules). IOMs são tipos especias de IODs normalizados que sempre representam conjuntos dados do tipo numérico ou texto e que estão relacionadaos a um IOD complexo. IOMs sempre represetam conjuntos de dados do tipo numérico ou texto e que estão relacionados a um IOD complexo.

Figura 3 Visualização da estrutura de informação no Cliente Cyclops MIB

Acima podemos ver como a estrutura de informação do DICOM 3 foi implementada dentro do Cyclops Medical Image Browser . Note que o contexto da informação que é mais textual, como paciente, serie e estudos pertencem ao campo lista esquerda, pois essa geralmente apresenta um número menor de itens comparado com o número de itens que pode haver no campo de informação imagem. É comum haver tomografias ou ressonâncias magnéticas com mais de 200 cortes por serie por isso ela melhor exibida na lista a direita.

3 Estado da Arte

3.1 Ferramentas Existentes

3.1.1 e-Film

O e-Film é o mais conceituado e conhecido cliente de imagens médicas existente na atualidade. Podemos dizer que ele é um software completo, possuindo ferramentas para comunicação em redes, ferramentas de auxilio a diagnóstico, filtros para alteração de contraste e ferramenta de impressão. Até o meio do ano 2001, este software era freeware, mas após esta data passou a ser cobrado. Outro ponto interessante é que o e-film possui sérios problemas com seu driver de impressão, inviabilizando algumas tarefas de impressão, quando o volume a ser impresso excede um certo limite.

Este software foi um dos melhores que surgiram até o momento para a plataforma

de computadores PC, no que diz respeito ao padrão DICOM. Trata-se de um software que faz o papel de uma “estação de trabalho” para imagens médicas digitais no padrão DICOM.

Algumas de suas funcionalidades:

- Estação de Visualização;
- Armazenamento e Transmissão;
- Impressão DICOM;
- Armazenamento em Mídias.

Como estação de visualização de imagens médicas digitais, o e-Film possui uma série de características notáveis, tais como a possibilidade de visualização simultânea de estudos de pacientes e suas respectivas imagens, fácil controle da “janela” utilizada na visualização da

imagem com o mouse, conversão das imagens para outros formatos digitais de imagens (JPEG, TIFF, etc.), impressão das imagens em impressoras comuns, manipulação das informações sobre pacientes, estudos e séries de cada imagem no formato DICOM, transferência destas imagens e informações para outras entidades de aplicação DICOM via rede de computadores, etc. O e-Film vem com um servidor DICOM com algumas funções de gerenciamento, tal com shutdown, reinicialização, definição de bandas de utilização do espaço em disco no sistema, monitoramento de filas de transferência de imagens de/para outras entidades de aplicação, etc. Conta também com algumas facilidades para a criação de CD-R e outras mídias de alta capacidade de armazenamento, com imagens selecionadas do seu arquivo. Podem-se incluir as imagens em formato DICOM, HTML+JPEG ou ambos em uma estrutura de diretórios para, posteriormente, serem transferidas para mídia, juntamente com o e-Film Lite, uma versão mais simples do e-Film.

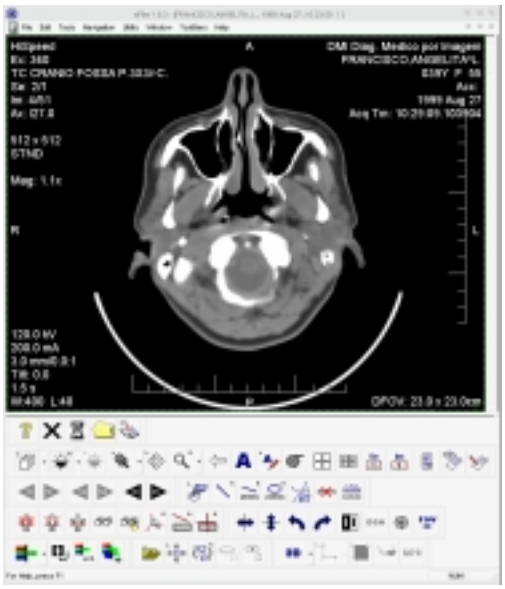


Figura 4 Interface do e-film. Visualizando uma tomografia

Figura 5 Visualizando a reconstrução tridimensional da mesma

3.1.2 ezDicom

O EzDicom é um software freeware que decodifica arquivos DICOM, inclusive imagens “multi-frame”, ou seja, de filmes obtidos a partir de raios x ou tomografos como os utilizados numa biopsia . Ele apresenta uma ótima performance. Porém, ele só trabalha com imagens DICOM separadas, e não dá pouco suporte para o trabalho com séries de imagens. O máximo que ele permite é a leitura da informação contida em cada um dos “Tags” que ela apresenta. Ele também apresenta uma boa ferramenta para ajuste de Window (contraste e brilho da imagem), que é muito apreciada pelos médicos para auxiliar no diagnóstico de determinados problemas. O “zoom” na imagem também é permitido.

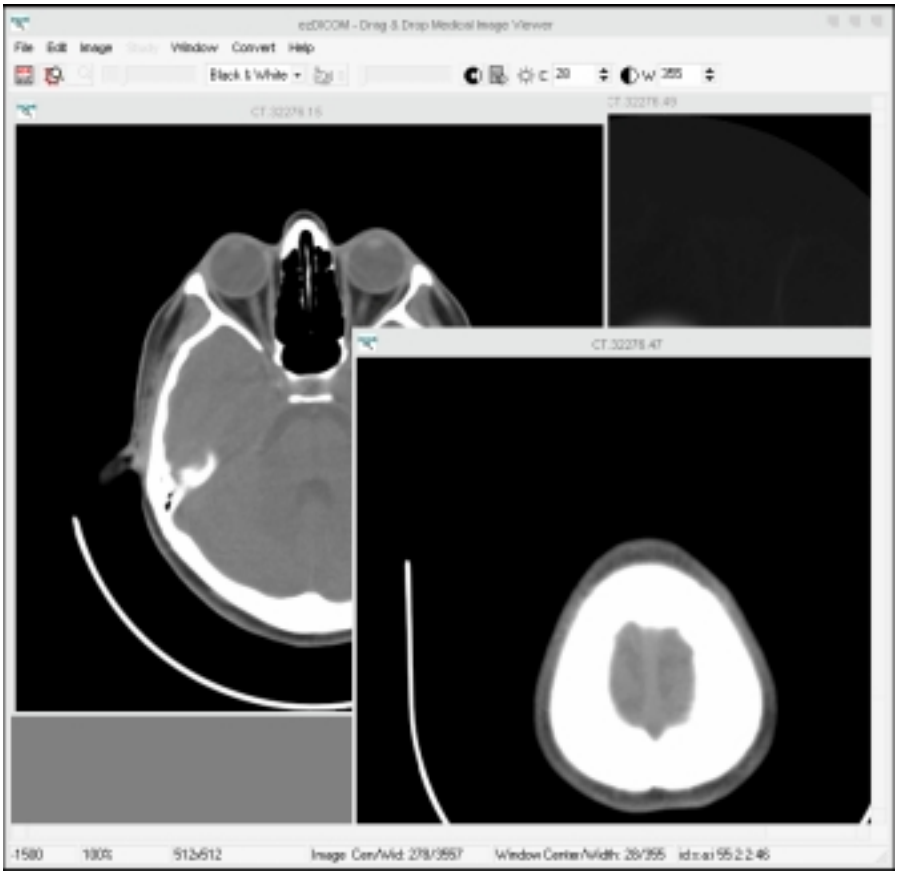


Figura 6 Interface do ezDicom

3.1.3 Osiris

Este programa foi por muito tempo considerado o melhor cliente para imagens médicas no padrão DICOM. Como podemos observar na figura 6, ele possui um conjunto de ferramentas de analise de imagens simples que pode proporcionar algum auxilio ao médico na hora de estudar a imagem. Entretanto ele não fornece nenhuma ferramenta que auxilie a escrita do laudo, inclusive apresentando uma interface com pouco recursos.

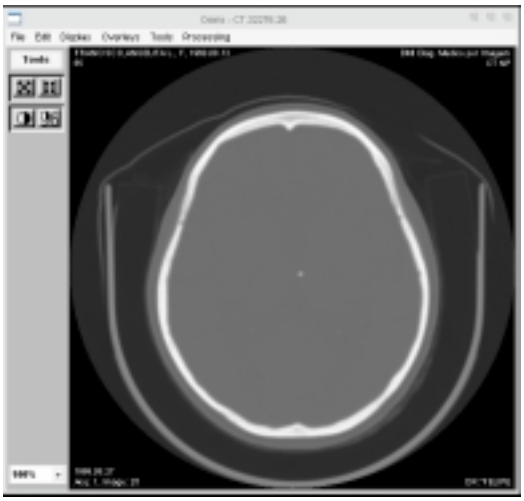


Figura 7 Interface do Osíris.

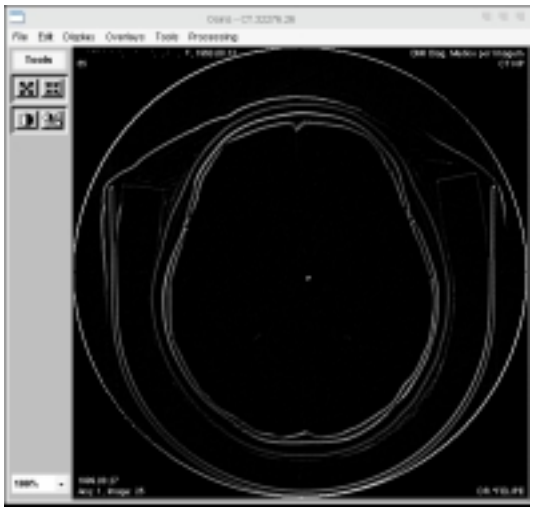


Figura 8 Interface do Osíris.

3.1.4 Cyclops Personal

O CyclopsPersonal foi desenvolvido por Daniel Duarte Abdala em seu trabalho de conclusão de curso, com o objetivo de sanar uma das grandes dificuldades dos softwares desenvolvidos no projeto Cyclops, o fato de serem desenvolvidos em uma linguagem interpretada, o Smalltalk que não é propicia ao uso comercial. Ele apresenta basicamente as mesmas funcionalidades práticas que os outros softwares, como ajuste de window, impressão das imagens, zoom. Entretanto vale lembrar que ele foi feito para ser uma ferramenta gratuita

e de fácil expansibilidade, sendo que é provável que as ferramentas aqui testadas sejam posteriormente adicionadas a ele.

Figura 9 Interface do Cyclops Personal.

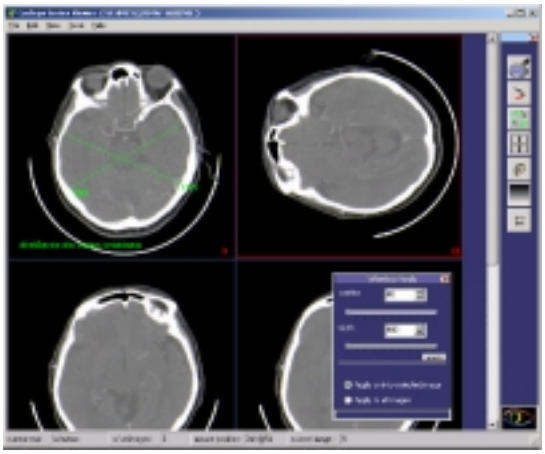


Figura 10 Interface do Cyclops Personal.

3.1.5 Dicomscope

O DicomScope é um visualizador de arquivos DICOM para Windows, desenvolvido pela OFFIS e é atualmente uma das melhores ferramentas de visualização de imagens DICOM disponível. Ë gratuito e foi desenvolvido em Java, e mesmo sendo um software interpretado como o apresentado neste trabalho, possui um desempenho extraordinário, sendo mais rápido que todas a outras ferramentas citadas. Ele apresenta as mesma ferramentas básicas que as outras, tendo como diferencial a capacidade de criar e ler laudos SR através de uma rede DICOM. Compreende ainda um browser para estudos, um componente para processar e visualizar imagens DICOM, ferramentas para impressão de imagens e documentos DICOM SR, e também um gerenciador de impressão. O visualizador é gratuito e o código fonte em Java está disponível para download na página do OFFIS (DICOMSCOPE, 2002).

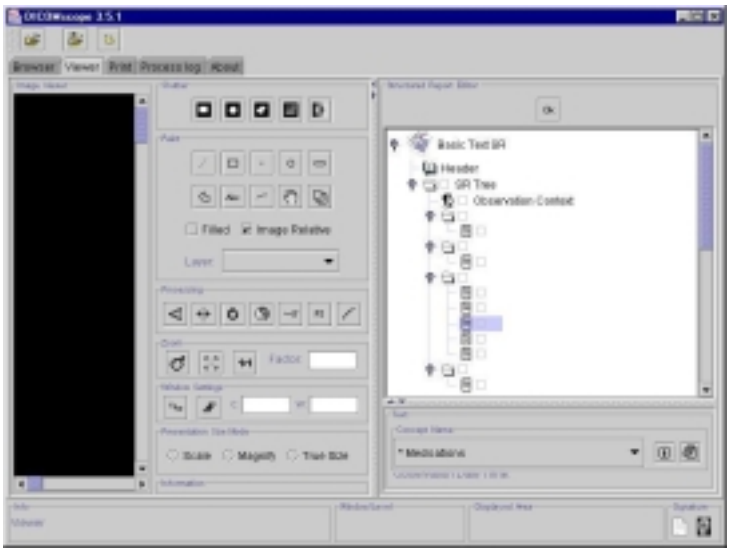


Figura 11 Interface de visualização de documentos DICOM SR do DICOMScope

Figura 12 Interface de visualização de documentos DICOM SR do DICOMScope

3.2 Analise do estado da arte

Poderíamos pensar que devido a grande quantidade de ferramentas de visualização de imagens em formato DICOM seria inútil a construção de mais uma ferramenta similar, mas esta conclusão é completamente errônea, ainda mais levando-se em conta a realidade dos hospitais públicos brasileiros, que sofrem com falta de verbas até mesmo para a manutenção do seus antiquados sistemas de radiologia. Geralmente sistemas de comunicação e arquivamento, os PACS, são vendidos em kits fechados que podem custar algumas centenas de milhares de reais. Se apenas as ferramentas de visualização forem compradas esse valor ficara próximo de 1500 reais (preço do e-filme no dia 16 de janeiro de 2004). Vale lembrar que a ferramenta aqui desenvolvida é gratuita e possui um grande número de funções que não são presentes nas ferramentas similares, onde muitas vezes elas terão que ser comprada separadamente a preços semelhantes.

A combinação aqui apresentada de ferramentas de auxilio ao diagnostico permite ao médico tomar uma decisão mais rápida e acertada referente ao diagnostico e por seqüente laudo do paciente. Com o auxilio da ferramenta CyclopsPopcorn, o médico poderá automaticamente detectar neurocisticercose e calcificações no cérebro do paciente apartir de tomografias. Com o auxilio do Atlas Cerebral de Talairach poderá detectar quais áreas funcionais do cérebro sofreram danos com este neurocisticercose ou com um acidente vascular cerebral descoberto pelo Cyclops Stroke Quantifier e prescrever o melhor tratamento com o CyclopsStructuredReport Editor sem ter que abrir uma nova aplicação ou aprender uma nova ferramenta complexa.

4 O Cyclops Series Editor e seu sistema de componentes

Durante os últimos dez anos, foram desenvolvidos no Projeto Cyclops muitas ferramentas que fornecem ao médico auxilio ao diagnostico. Mas um dos grandes problemas desses aplicativos, do ponto de vista de um cientista da computação, foi a maneira como foram construídos, na maioria das vezes, por pessoas com pouco ou nenhum conhecimento de programação orientada a objetos, em especial da linguagem smalltalk, e a não utilização de metodologias da engenharia de software. Ou seja, além de mal programados, os sistemas não tinham nenhum planejamento anterior, e acabaram por se tornar, difíceis de usar e de alterar. Esses softwares foram construídos durante projetos de mestrado e após a conclusão dos mesmos, as pessoas responsáveis por eles deixavam o projeto, e deixavam sem documentação as ferramentas que elas construíram. Como fazer então para integrar um conjunto de ferramentas cujo código não contem documentação e é muito difícil de compreender em um curto período de tempo, num única ferramenta, com mínimas alterações na ferramenta original e que venha a permitir novas expansões no futuro, sem que ela própria seja modificada radicalmente?

Deste problema, surgiu-se a idéia da utilização de plug-ins, que poderiam ser definidos como trechos de código altamente modularizados (componentes), que poderiam ser encaixados para forma uma novo aplicativo e que localizariam e minimizariam as regiões onde poderiam ocorrer modificações.

Analisando as ferramentas desenvolvidas no projeto, percebeu-se claramente uma divisão: um tipo de ferramenta possuía alto acoplamento com a imagem, ou porque trabalhava diretamente com ela, executando algum tipo de processamento, ou porque necessitava exibir informações diretamente sobre ela. O outro tipo era claramente independente da imagem, ou

seja, possuía em relação a imagem um acoplamento extremamente baixo, fazendo, no máximo, apenas uma referencia a ela.

Você verá claramente a diferença entre esses dois tipos de aplicações quando em um capítulo à frente ver o funcionamento do Cyclops Brain Atlas e do Cyclops Structured Report.

4.1 Modularização do Cyclops Series Editor

Uma das coisas que você vai perceber quando estudar o Cyclops Series Editor é que ele é todo formado por mini aplicações que são organizadas em camadas:

Camada de Apresentação Series Editor
Camada de Tabulação de Plug-ins
Camada de Exibição de série de imagem
Camada de Exibição de Imagem individual

Na camada mais baixa se encontra a classe Cyclops Image Viewer que é responsável por exibir uma única imagem e prover uma fachada para seu acesso. Na camada superior se encontra a classe Cyclops Series Viewer, que anexa várias classes Cyclops Image Viewer para poder realizar a exibição da série de imagens. Na terceira camada, de tabulação e de plug-ins, é onde funciona o acoplamento das novas funções. Ela "vê" a classe CyclopsImageViewer como uma fachada para as classes CyclopsDicomImage e MRCachedImage, além do Model/View/Controler pertencente a cada classe.

Foram detectados que existem dois tipos de plug-ins:

- Tipo altamente acoplado a imagem: Este tipo de plug-in é geralmente derivado da antiga Classe DICOMSeriesEditor , ou seja, embora o objetivos de vários desses plug-ins fosse completamente diferente, todos eles contem um núcleo em comum, onde a grande diferença se localiza em algum código acrescentado a subclasse e a um novo MVC.este tipo de plugin provavelmente implicará pouca alteração na aplicação original, apenas as mudança de algum métodos de acesso ao MVC da classe CyclopsImageViewer. As Classes CyclopsSEview, CyclopsSEmodel E CyclopsSEContoller provavelmente precisarão ser modificadas, adicionando-se a ela o código que será necessário a exibição da informação requerida pelo plug-in. Uma das coisa que foi-se pensada durante o desenvolvimento deste trabalho, foi a utilização de diversos MVC, um para cada plugin altamente acoplado. O problema dessa abordagem é que ela minimizaria a utilização em conjunto dos plugins, e é exatamente a idéia contrária a essa que motivou este trabalho. Assim, você terá que acrescentar algum código e algumas variáveis nessas classes, de acordo com o tipo de aplicação que você desejar.

- Tipo de baixo acoplamento Este tipo de plugin, ao contrário do anterior, não é uma derivação da classes DicomSeriesEditor, sendo provavelmente feito do zero como uma nova aplicação.Este tipo de plug-in requer pouquíssima alteração de código , porque o seu funcionamento independe de um sistema de visualização de imagens, embora posso exibir outros tipos de sinais médicos, como uma waveform. Geralmente, seu funcionamento difere de tal forma do primeiro tipo, que até mesmo o local onde sua interface será exibida será

diferente; ele será exibo em uma nova tabua ao invés da região escondida denomina barra de ferramentas.

4.2 Como fazer um plug-in de baixo acoplamento

Basicamente você pode construir a ferramenta, completamente despreocupado de como funcionará a integração posterior bastando se preocupar com os seguintes fatores:

- Na hora de desenvolver a interface gráfica, certificar-se de que ela não necessitará de chamadas a superclasse nos método open.
- Adicionar uma variável para armazenar a instancia do cyclops Series Editor que vai englobar a aplicação posteriormente e apenas mais 3 métodos:
    - O método refresh deve conter procedimentos para recontruir a interface, entretanto ele pode ser deixado em branco
    - O método deverá deactivate e isAccepted retornaram sempre retornar true se naquele determinado momento, a ferramenta tem permisão para se desativada, em um procedimento de troca de tabua por exemplo

4.3 Como fazer um plug-in de alto acoplamento

Este modelo é um pouco mais complicado, para se fazer à integração. Você pode começar desenvolvendo a aplicação do zero ou derivar uma subclasse de DicomSeriesEditor e depois adaptá-la .

Se você optar por criar uma subclasse de DicomSeriesEditor você terá muito mais trabalho, pois terá que criar também um novo Model/View/Controler, que depois será

inutilizado com a passagem do código para o MVC do CyclopsSeriesEditor. Então é melhor começar diretamente com uma nova classe. Crie uma interface, que tenha no máximo 260 pixels de altura e 600 pixels de largura. Se esse espaço for insuficiente, utilize um widget de tabulação ou um notebook que permite que você utilize um esquema de tabuas bidimensionais, o que multiplicará a área útil em várias vezes. Crie o programa que você deseja centrando todo o código de processamento diretamente na classe da interface.

Modifique as classes CyclopsSEView, CyclopsSEModel e CyclopsSEController, de modo que você possa adicionar o código necessário a exibção de informações, e menus.

Caso você tenha feito pelo primeiro modo, siga esses passos:

- Mude a interface excluído a parte de exibição de imagens para somente a menor parte possível onde serão mostradas as informações de configurações e onde deverá ser executado todo o processamento.
- Acrescente uma variável para armazena o CyclopsSeriesEditor. A seguir você deve localizar todas os métodos que possuem referências ao canvas de exibição de imagens:
    - Entao isso: (((self builder namedComponents at: #lichtbrettSubcanvas) widget components at: 1) component widget components at: 3) component model removeChildrenLines.
    - Irá virar algo assim: self seriesEditor pageHolder selection imageVieweList first removeChildrenLines.

Em seguida, você deve analisar sua modelagem do MVC e copiar a parte necessária para o Modelo MVC do Cyclops Series Editor. Essa é a tarefa mais complicada, já que possivelmente você terá que modificar parte do código.

5 As Ferramentas Do Cyclops Series

Ficou definido que o software desenvolvido durante este trabalho de conclusão de curso deveria integrar os seguintes aplicativos anteriormente desenvolvidos no Projeto Cyclops:

- Cyclops Brain Atlas, desenvolvido durante o mestrado do professor Harley Wagner.
- Cyclops Waveform Viewer, desenvolvido durante o mestrado do professor Gilson Araújo.
- Cyclops PopCorn, desenvolvido durante o mestrado do Eros Comunello.
- Cyclops Stroke, ainda em desenvolvimento.
- Cyclops Structured Report Editor e Viewer, desenvolvido durante o mestrado de Mariana K. Bortoluzzi.

Além dessas ferramentas, o Cyclops Series ainda deveriam ser integrados ou funcionar em conjunto com esse outro conjunto de ferramentas:

- Cyclops MIB, desenvolvido durante o mestrado do Paulo P. Dellani;
- Um plug-in de processamento de imagens que utilizaria diversos algoritmos existentes no pacote Khoros 2.2 e Vista.
- Cyclops Mailer, um programa capaz de enviar as imagens por e-mail, juntamente com o laudo estruturado associado.

5.1 Das ferramentas de Processamento de Imagens

A principio, o software deverá conter as seguintes ferramentas classificadas conforme o tipo de analise que ela faz sobre a imagem.

Segmentação:

- Munford & Sha;
- WaterShed Adapt WaterGrow;
- Threshold
- Crescimento de região
- Snake
- Difussão anisotrópica
- Get Segments
- Merge Segments

Analise de bordas

- Canny
- VDRF

Ferramentas de medição de distância, e área:

- Área
- Perímetro
- Leveled Complexity
- Mean Value
- Center
- Geração de polígono.

5.1.1 Das Características das Ferramentas de Processamento de Imagens

Para a construção da ferramenta Cyclops Series Editor, esta sendo utilizado a linguagem Smalltalk a partir do ambiente de desenvolvimento Cincom VisualWorks (Cincom, 2003). Entretanto, por ser uma linguagem interpretada, a utilização da mesma para processamento de imagens é inadequada e ilógica, por isso foram escolhidos dois pacotes de processamento de imagens gratuitos:

- O pacote de processamento de imagens Khoros 2.2 (Khoros, 2003), hoje um produto comercial em sua versão 3.0, foi desenvolvido originalmente pela University of New Mexico em colaboração com a UNICAMP e é o mais utilizado sistema para teste e experimentação de soluções de Visão Computacional. Ele provê um conjunto bastante extenso de implementações de algoritmos constituindo mais que 300 ferramentas de manipulação de imagens.
- O outro pacote de processamento de imagens, o Vista, foi desenvolvido por Art Pop durante seu projeto de doutorado e possui um conjunto considerado de ferramentas de visão computacional.

Além disso, é necessário o programa de conexão ao banco de dados, que implementa as rotinas da sintaxe de transferência DICOM, o Cyclops MIB para a obtenção dos arquivos DICOM de um servidor.

5.1.1.1 Dos algoritmos de Segmentação.

Os algoritmos de segmentação têm por objetivo reduzir as informações da imagem em regiões mais ou menos homogêneas, onde somente a parte importante da imagem está presente e os detalhes e ruídos são eliminados.

5.1.1.1.1 Segmentação por Munford & Sha

O algoritmo de Munford-Sha (Munford-Sha, ????) faz parte dos algoritmos classificados como de segmentação sendo destes um dos mais preciso. Nele cada região é formada por um grupo de pixels com uma "borracha ao redor". A região cresce de acordo com a elasticidade da borracha, onde a elasticidade é definida pela variação de intensidade dos pixels de uma região, portanto quanto maior a variação menor é a elasticidade da borracha e quanto menor a variação maior a elasticidade. Além disso, há ainda o critério de união de regiões que é determinada pela variação dos tons de cinza.

Seu funcionamento é baseado na *Equação Funcional* de Munford & Sha:

$$E(u,K) = \iint_{\Omega} \|u - g\|^2 \, dxdy + \iint_{\Omega / K} \|\nabla u\|^2 \, dxdy + \lambda * l(K)$$

Como o objetivo desse algoritmo é exatamente extrair da imagem apenas as partes importantes (que realmente possuem alguma informação) ele pode ser muito útil ao médico na identificação de tecidos e destacá-los frente a outros. Este algoritmo, por exemplo, é utilizado pelo software Cyclops Popcorn para auxiliar na detecção das neurocisticercose.

Na figura abaixo, podemos claramente observar que ele foi capaz de identificar com clareza a região composta de tecido ósseo.

Figura 13 Tomografia computadorizada do abdômen original

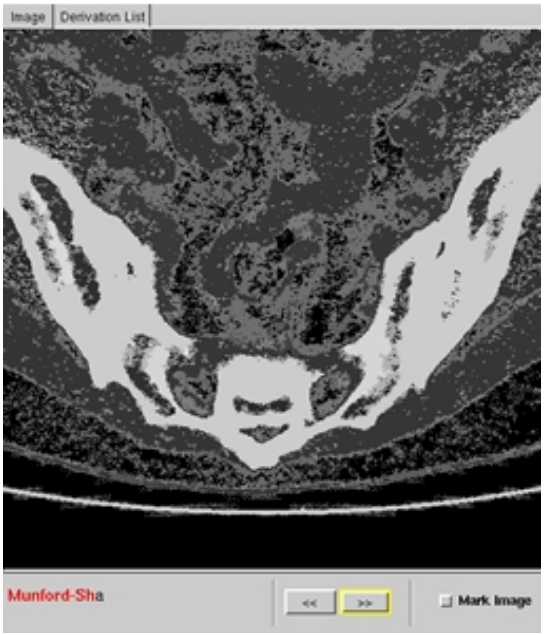


Figura 14 Tomografia computadorizada depois de aplicado o algoritmo de Munford-Sha

5.1.1.1.2 Threshold

O algoritmo do limiar de cor, como também é conhecido o threshold, funciona analisando todos os pixels da imagem e comparando com um determinado valor dado (o limiar). Caso o valor do pixel seja maior que o valor do limiar, então esse pixel passara a ser um pixel preto, se não, caso o valor do pixel seja menor, este assumira o valor de um pixel branco.

Este algoritmo permite a segmentação de estruturas através do seu valor de densidade radiológica, por exemplo, separando a área óssea da área de massa cinzenta.

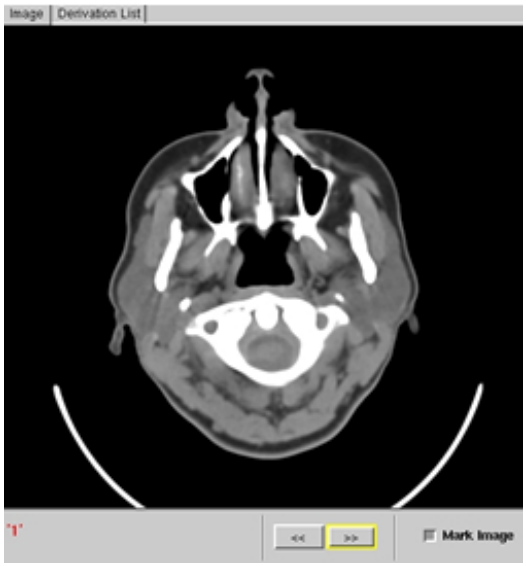


Figura 15 Tomografia computadorizada original

Figura 16 Tomografia computadorizada depois de aplicado o threshold

5.1.1.1.3 WaterShed Adapt WaterGrow

Imagine uma imagem como sendo um modelo topográfico tridimensional, onde as variações de cor nos pixels seriam equivalentes as suas altitudes (valores mais altos são picos, mais baixos são depressões). Agora imagine que essas regiões estão sendo inundadas na mesma velocidade para cada mínimo local da imagem, onde a água de diferentes mínimos se encontrar, são construídas barreiras para separá-las. Essas barreiras constituem-se as linhas divisórias do watershed.

5.1.1.2 Dos algoritmos de detecção de bordas.

Bordas são o resultado de uma mudança de propriedade física ou espacial, sendo que a maioria dos detectores de bordas utiliza-se de operadores diferenciais de primeira e segunda ordem para a sua detecção.

5.1.1.2.1 Detecção de bordas utilizando o algoritmo Canny

O canny utiliza-se de um operador gaussiano de primeira derivada que suaviza os ruídos e localiza as bordas. Ele se ampara em quatro estágios:

- Uniformização da imagem: a imagem é uniformizada utilizando-se duas funções gaussianas unidimensionais, uma pra cada eixo, que aproxima uma função gaussiana 2-D.
- Diferenciação: A imagem uniformizada é separada em cada eixo para que seja possível calcular o gradiente. Isso se dá aplicando a função unidimensional X nos valores uniformizados na direção Y e vice-versa.
- Omissão de pontos de mínima intensidade: Depois de encontrado o valor da intensidade para cada ponto, precisa-se localizar as bordas, localizando então os pontos de máxima intensidade ou localizando os pontos de mínima intensidade que precisam ser omitidos. Um valor de máxima intensidade ocorre no local mais alto da função gradiente ou onde a derivada da função gradiente possui valor zero.
- Limiarização da borda: A limiarização usado no algoritmo Canny usa o método chamado "histerese", ou seja, pixel com um valor acima do limite são incluídos são aceitos e quando abaixo do limite são rejeitados.

Figura 17 Tomografia computadorizada do abdômen original

Figura 18 Tomografia computadorizada depois de aplicado o Canny

5.1.1.2.2 Detecção de bordas utilizando o algoritmo VDRF

O algoritmo de detecção de bordas VDRF utiliza-se dos mesmos princípios do Canny, mas com a utilização da derivada primeira ao invés de uma função gaussiana. Ambos os algoritmos de detecção de bordas podem ser úteis para a identificação com clareza dos limites de regiões de interesse ao médico, e podem também ser útil em reconstrução tridimensional da parte óssea de uma series de imagens radiológicas.

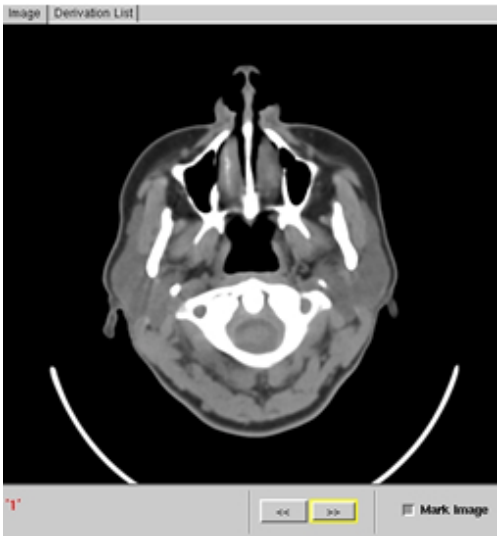


Figura 19 Tomografia computadorizada original

Figura 20 Tomografia computadorizada depois de aplicado o VDRF

Modelos de domínio do problema

5.1.2 As ferramentas de suporte ao laudo

5.1.2.1 O CyclopsBrainAtlas

O CyclopsBrainAtlas é um Atlas deformável digital do cérebro humano baseado no modelo do atlas de Talairach, que permitea identificação dos quadrantes de Talairach e automaticamente mostra as áreas de Brodmann relacionadas a cada quadrante. Isto permite a identificação e a localização semi-automática das áreas funcionais e anatômicas do cérebro afetadas por alguma patologia visível como, por exemplo, um cisticerco. O ajuste é executado com a adaptação, posicionamento e rotação interativas do volume inteiro do Atlas, que é mostrado na relação de usuário através das interseções planas com as fatias tomográficas.

A respeito do modelo do Atlas de talairach

O Atlas de talairach foi desenvolvido a partir de um único cérebro humano (de uma senhora de 60 anos), onde a maioria dos estudos de áreas funcionais foram feitos em pessoas vivas!

Figura 21 – Interface do Cyclops Series Editor com o Cyclops Brain Atlas funcionando em conjuto

5.1.2.2 Structured Report

Embora registros médicos em formato convencional (papel ou filme) seja rotina na maioria absoluta dos hospitais e clinicas do país, cada vez mais as instituições de saúde

implementam sistemas de registro clínico eletrônico buscando maior agilidade no acesso aos dados de paciente, e melhorando assim o atendimento.

Existem inúmeros termos para se referir a sistemas computacionais para registro de dados clínicos do paciente, no Brasil o termo geralmente usado é o prontuário eletrônico.

A parte mais importante de qualquer sistema de prontuário eletrônico é um sistema que permita a edição confortável de laudos e observações clínicas, por parte dos profissionais encarregados dos cuidados aos pacientes.

Como explicando anteriormente, um arquivo DICOM pode conter referências a imagens, eletrocardiogramas e arquivos de áudio, além disso, ele também pode conter objetos de informação que codificam dados a respeito do paciente e seus dados relativos à sua saúde em forma de texto estruturado formal. Este ultimo leva o nome DICOM Structured Report – DICOM SR .

O padrão DICOM SR "utiliza terminologia controlada, e desta forma evita a ambigüidade das linguagens naturais, facilita a busca por informações específicas e a internacionalização do conteúdo"(Bortoluzzi,2003), assim estabelece como devem ser formados armazenados e transferidos documentos estruturados que podem representar laudos, ou qualquer tipo de observação clínica. Estes documentos contêm informações de contexto, tais como procedimentos que devem ser executados para o sucesso de um tratamento, e dados sobre profissionais de saúde envolvidos. Um objeto no padrão pode conter referências embutidas a imagens, eletrocardiogramas, e arquivos de áudio bem como a outros documentos no mesmo padrão.

Cada objeto codifica apenas informações semânticas, e não contém informações sobre como o documento representado pelo objeto deve ser apresentado, ou impresso. Portanto, cada implementação de prontuário eletrônico pode ter um formato para apresentação que lhe for mais adequado. Além disso, objetos no padrão fazem uso de terminologia controlada, o

que evita as ambigüidades da linguagem natural, facilita o entendimento automatizado do conteúdo, a busca por informações específicas, e a tradução do conteúdo.

O CyclopsSRReport Editor e um editor de documentos de interface amigável, capaz de produzir documentos de acordo com o padrão DICOM SR. Devido a necessidade de fácil acesso a essa ferramenta, ela foi integrada diretamente dentro do Cyclops Series Editor. Uma das grandes facilidades que ela apresenta é a capacidade de ler modelos (Templates) de laudos, com parte do texto já preenchida. Isto é importante porque grande parte dos laudos é extremamente repetitiva, constituindo-se basicamente de reproduções de laudos anteriores, e mesmo o laudo inicial muitas vezes necessita de apenas pequenas modificações estruturais e de textuais.

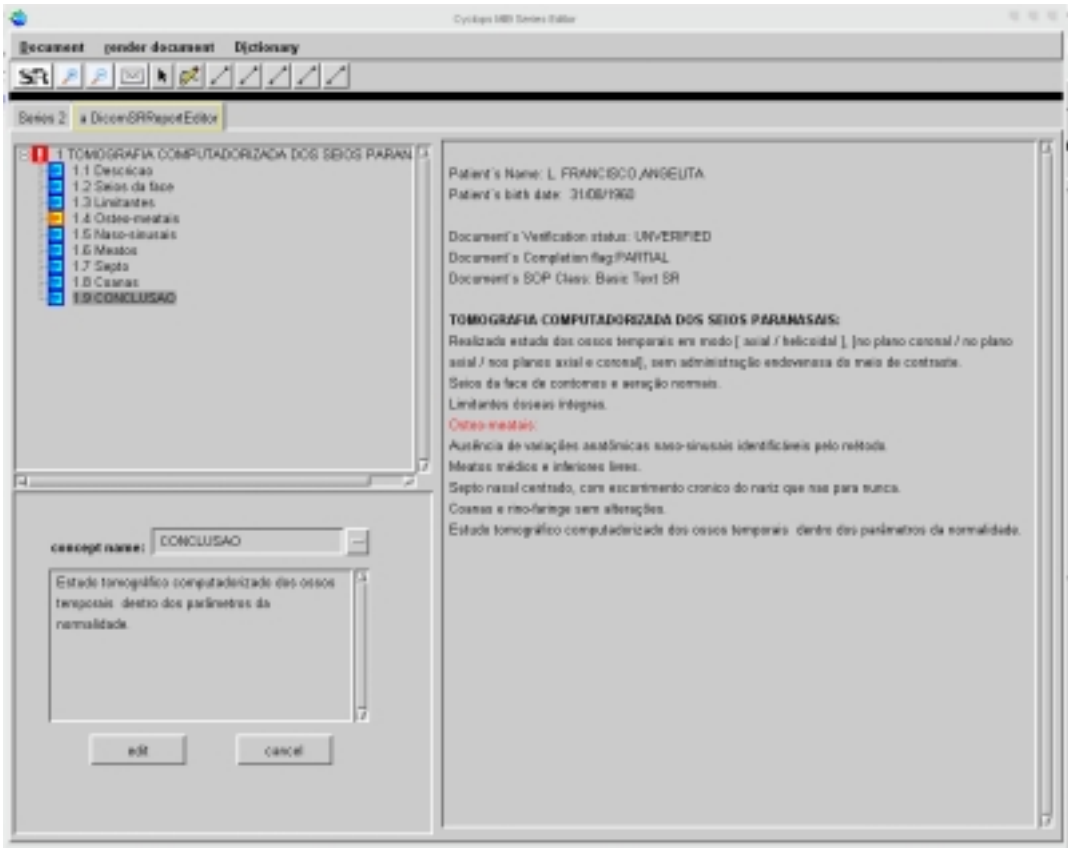


Figura 22 – Interface do Cyclops Series Editor funcionando em conjunto com o Cyclops Structured Report Editor : fácil criação de laudos estruturados.

5.1.2.3 Cyclops Stroke Quantifier

Acidentes vasculares cerebrais, conhecidos popularmente no Brasil com derrame ou infarto cerebral, correspondem a qualquer lesão encefálica surgida a partir da alteração da área vascular, ou seja, do sistema de irrigação sanguínea do cérebro.

> Quando o fluxo sanguíneo, em uma determinada região cerebral, cai abaixo de 20ml/100g/min, falha a atividade elétrica cerebral e surgem sintomas. Com maiores reduções, em torno de 10ml/100g/min, ocorre uma área central de isquemia com conseqüente morte dos neurônios. Ao redor desta área há uma região onde existe redução do fluxo, porém a perfusão é mantida pela circulação colateral. Esses neurônios que não funcionam, mas estão vivos, formam a zona de penumbra, que é uma região metabolicamente inativa, a qual pode ser recuperada com a intervenção médica, se a terapia for iniciada rapidamente. (Wardlaw et al. 1999);

Assim, logo após ser diagnosticado o AVC isquêmico, a terapia dever "ser feita com a utilização de agentes neuroprotetores ou com a utilização de substâncias trombolíticas na tentativa de restabelecer o fluxo na região obstruída. Este tipo de terapia tem a finalidade de restaurar o fluxo sangüíneo o mais breve possível para limitar as perdas neuronais na área de penumbra" (Wardlaw et al. 1999);

Alguns dos fatores limitantes para adoção da terapia com trombolíticos são:

- A grande dificuldade que os radiologistas encontram em identificar AVC isquêmico com 6 horas de evolução usando apenas a Tomografia Computadorizada como método diagnóstico (Wardlaw et al. 1999);

- A possibilidade de visualização de sinais de isquemia pelo radiologista dentro desta janela de tempo já é por si só um fator independente de mau prognóstico (Wardlaw et al. 1998);
- Dificuldade de identificação e mensuração da zona de penumbra, que seria a região potencialmente recuperável com a terapia.

Embora não existam dados precisos no Brasil, sabemos que os acidentes vasculares cerebrais são a principal causa de morte relacionada com patologias clínicas e a segunda causa mais freqüente de morbidade neurológica nos paises desenvolvidos. Oitenta por cento de todos os AVCs são causados por problemas no fluxo sanguíneo (AVC isquêmico), sendo que destes, 75 % são devido a obstruções (Fieschi et al. 1989).

O Cyclops Stroke Quantifier foi criado para fornecer suporte a decisão na terapia do AVC isquêmico calculando e representando graficamente o fluxo sangüíneo no cérebro com base em Tomográfica computadorizada dinâmica (cálculo da variação local relativa de densidade radiológica). Para isto foi realizado o desenvolvimento de um software para auxiliar ao radiologista e aos médicos generalistas a identificarem precocemente o AVC isquêmico para proceder com o tratamento adequado além de permitir que façamos a correlação entre a área atingida com o déficit das funções atribuídas a esta área utilizando–se do Atlas cerebral. E calcular o tamanho da área resultante do AVC utilizando – se de uma ferramenta para tal.

Para o cálculo da variação de sinal e elaboração dos mapas de perfusão foi calculada inicialmente, para cada região da imagem correspondente a parênquima cerebral, a curva de variação percentual de sinal em relação àquele ponto na imagem nativa (corte 1). As áreas de parênquima são automaticamente selecionadas pelo software em função de sua densidade radiológica e de suas vizinhanças. Essas curvas de absorção de agente de contraste são armazenadas para a geração dos mapas de perfusão corte a corte e podem ser visualizadas

pelo usuário do sistema simplesmente através da movimentação do mouse sobre qualquer um dos cortes.

Os mapas de perfusão são calculados através da comparação das variações relativas de absorção de agente de contraste previamente calculadas com a variação média de absorção de contraste de um volume de referência (VOR) determinado pelo usuário. O sistema calcula uma curva de variação relativa média de absorção de agente de contraste passando pelo VOR em todos os cortes. Esta curva, denominada curva de referência, será utilizada como parâmetro para o cálculo dos mapas de perfusão em cada corte. Para isto, calcula-se o desvio percentual (positivo ou negativo) entre a curva de referência e as curvas de variação percentual de sinal de cada ponto. Isto permite que se construa um mapa de comparação entre uma dinâmica de agente de contraste tomada como referência para aquele paciente e todos os pontos do parênquima. Este método permite também que se tome por referência uma área sadia do próprio paciente, e que se calcule diversos mapas de perfusão baseados em diferentes áreas de referência, podendo-se comparar, por exemplo, a absorção de agente de contraste com a absorção típica de uma área de substância branca ou com uma área do córtex ou mesmo com uma área apresentando patologia.

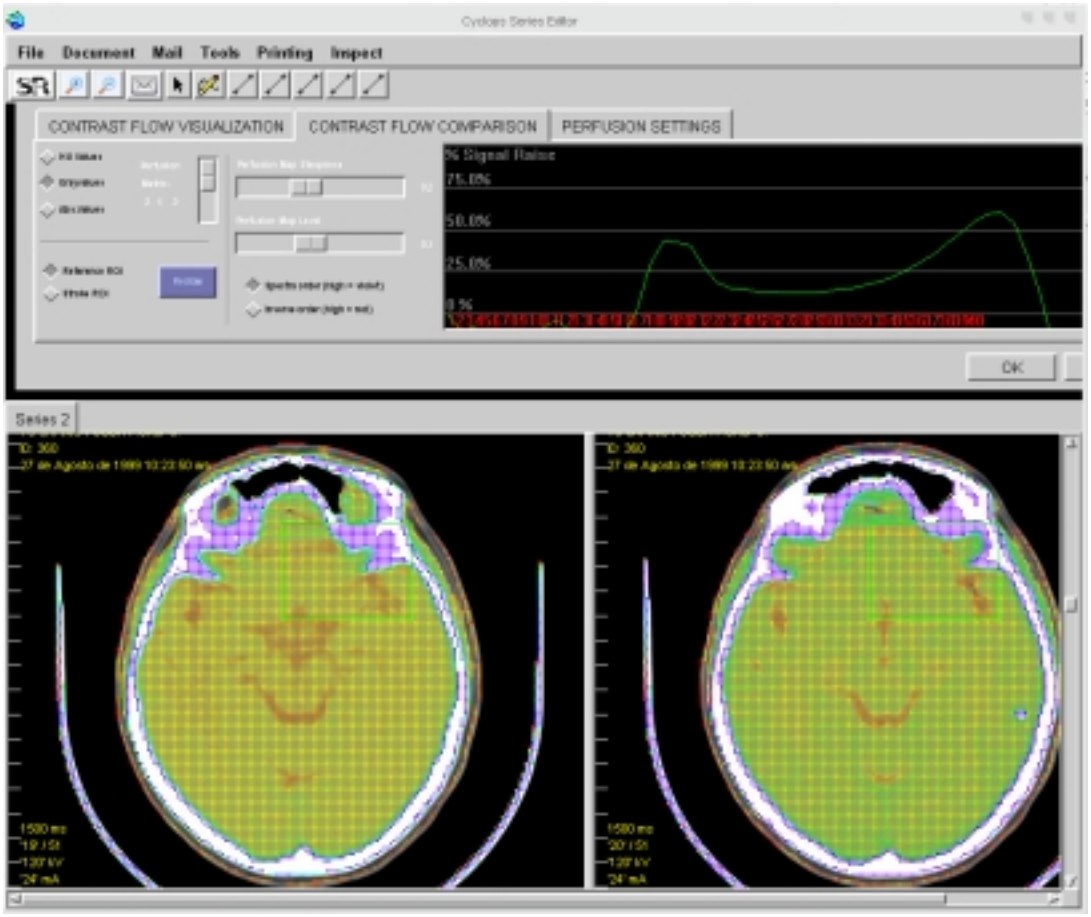


Figura 23 Inteface do Cyclops Series Editor em conjunto com a ferramenta Cyclops Stroke Quantifier

5.1.2.4 Cyclops Popcorn

A neurocisticercose é uma das principais causas da epilepsia, segundo afirmam especialistas. Todavia, não existem estatísticas confiáveis sobre esta relação.

O Cyclops Popcorn é uma ferramenta de diagnósticos que utiliza métodos de visão computacional para a identificação, contagem, mensuração e localização de calcificações relacionadas a esse tipo de infecção em imagens de tomografia computadorizada do crânio.

O procedimento é resultado de um trabalho conjunto realizado entre o Grupo de Sistemas baseados em Conhecimento, da Universidade de Kaiserslautern, Alemanha; o Departamento de Informática da Universidade Federal de Santa Catarina (UFSC); o

Ambulatório de Epilepsia de Florianópolis e a Clínica Radiológica Budedenbrock, Blasinger e Benz, Alemanha.

O sistema segmenta imagens de tons cinza obtidas pelo tomógrafo computadorizado, realizando uma classificação por meio de técnicas de inteligência artificial, utilizando para isso de uma rede neural feedfoward. Também arca as calcificações no cérebro consideradas como relacionadas à neurocisticercose, assinalando as áreas consideradas afetadas com cores especiais. Os resultados são comparados com o panorama obtido em testes cognitivos, que detectam os sintomas do paciente.

5.1.2.5 Cyclops Waveform Viewer

O cyclops waveform é um visualizador de ondas obtidas através de aparelhos de eletroencefalograma ou eletrocardiograma que permite uma analise semi-automática da onda auxiliando o médico na detecção de problemas cardíacos ou cerebrais.

5.2 Outras ferramentas

O *Cyclops Series Editor* é constituído de alguns recursos interessantes como o uso de tabuas para a divisão da área útil, o que permite ao usuário médico visualizar as várias séries de um mesmo estudo simultaneamente. Além disso, esse recurso permite ao médico também a criação de novas tabuas para a utilização de o software auxiliar Cyclops Structured Report Editor. Caso existam, além de imagens, uma waveform de qualquer tipo (EEG ou ECG) ela também será exibida em uma nova tabua, assim como um laudo SR preexistente será visualizado pelo Cyclops SR Viewer.

5.2.1 Barra de aplicações

Escondida abaixo da barra de menus existe um widget que permite acesso a um painel de configuração das ferramentas que complementam a funcionalidade do programa. A ferramentas disponíveis são as seguintes:

Ajuste de contraste de window

Essa é uma das ferramentas mais importantes e mais usadas no programa. Com as tomografias computadorizadas se obtém além da imagem propriamente dita alguns dados extras de características de tecidos como a densidade relativa de uma estrutura, medida em HU, ou unidades de Hounsfield (em homenagem a Sir Geoffrey Hounsfield, inventor da Tomografia Computadorizada). (CHENG, 2001)

Esta medida estipula que a densidade da água têm valor 0, enquanto a densidade do ar é tipicamente -1000.

A seguir está uma tabela com exemplo de valores HU para algumas estruturas:

Estrutura	Valor em HU
Ar	-1000
Gordura	-100 a 400
Fluido	0 a 20
Tecidos leves	20 a 100
Osso	1000
Hemorragia intracranial aguda	55 a 75
Massa cinzenta	30 a 40

Relação de estruturas X HU values

Para a visualização de imagens codificadas nesta medida, é necessário realizar uma conversão destes valores para a escala RGB, comumente utilizada em sistemas gráficos. Para realizar esta operação, é necessário definir limites dentro dos valores em HU. Estes limites são normalmente chamados de “Window”.

Uma “Window” é definida dois valores em HU: Centro da “Window” (*Window Center*) e Comprimento da Window (*Window Width*). Estes dois valores definem um intervalo dentro da escala de HU da seguinte maneira: o número central deste intervalo é o Centro da Window, e o tamanho total deste intervalo é o Comprimento da Window. Assim:

$$\textbf{Window} = X \mid WC - \frac{WW}{2} \leq X \leq WC - \frac{WW}{2}$$

Sendo:

- WC = Centro da “Window” (*Window Center*)
- WW = Comprimento da “Window” (*Window Width*)

A conversão dos valores em HU para RGB pode ser feita fazendo com que valores inferiores ao intervalo da Window sejam definidos como preto (valor 0 em RGB), e valores maiores que o intervalo são mostrados como branco (valor 255). Os valores dentro do intervalo são definidos proporcionalmente entre 0 e 255.Para realizar a converão de um valor em HU para um intervalo da escala

RGB, pode-se usar a seguinte fórmula:

$$RGB = \left(\frac{(HU - WC + \frac{WW}{2})}{WW} \right) * 256$$

Abaixo vemos a janela de configuração onde podemos alterar as configurações de Window Center e Window Width e ver em tempo real uma aproximação da imagem que será gerada a partir desses valores.

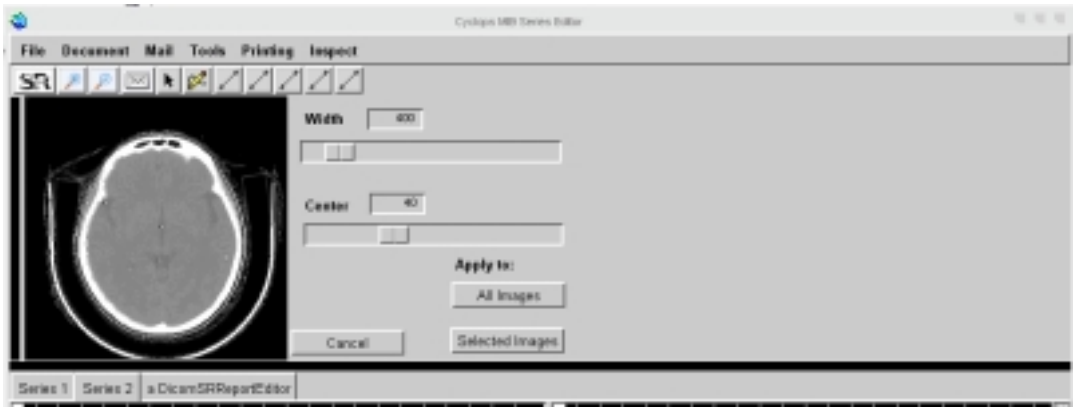


Figura 24 Interface de ajuste e preview da window radiologica

Reconstruções Multiplanares

No canto esquerdo da área escondida pelo widget, existe uma área semelhante, que quando clicada exibirá duas imagens, conhecidas com reconstruções multiplanares da imagens.

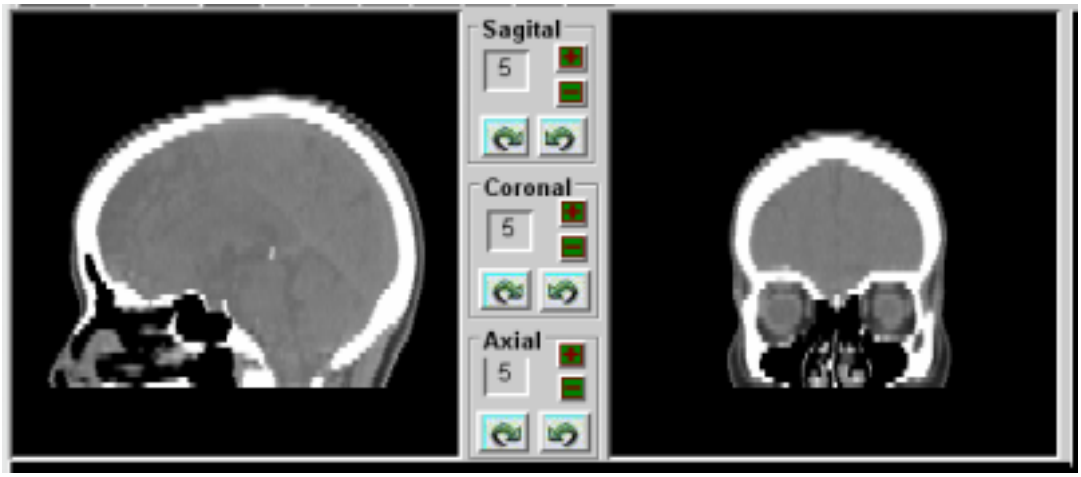


Figura 25 – Interface de visualização das recontruções multiplanares

Em uma série de tomografias computadorizadas usuais, os cortes (as imagens obtidas) são usualmente feitas no sentido xy(sentido axial), tomando o valor de z fixo. Para ter-se condições de se vizualizar o volume em 3d, é necessária a existência de cortes no sentido sagital e coronoal , quando a série não apresenta uma outra tomografia ou raio x nesses sentidos(isso é conhecido como piloto) é nessecario gera-lo artificialmente, através de um proceso que apartir das imagens originais gera uma reconstrução multiplanar.

A MPR, que corta todo o volume no sentido yz ou sentido sagital, toma o valor de x fixo onde o corte é realizado (pode ser observado na figura 32 a esquerda.) .A mpr que corta todo o volume no sentido xz que tomando dessa vez o valor de y como fixo,

Funcionamento do algoritmo

Uma das funcionalidades extras que uma imagem DICOM possui é saber exatamente o tamanho que cada pixel tem na realidade. Então para se obter uma reconstrução planar bastaria utilizar o tamanho do pixel, a posição do corte desejado e um certo numero de

imagens que você gostaria, obter os pixel de cada uma dessas imagens e concatená-los para formar uma nova imagem.

Abaixo o algoritmo que faz isso no Cyclops Series Editor:

```
aPalette := originalImages first mrCachedImage image palette.
"Copy the first image palette"
newBits := ByteArray new.
originalImages first sliceThickness asNumber
/ (originalImages first pixelSpacing at: 1) asNumber.
originalImages do:
[:aDicomImage |
myImage := aDicomImage mrCachedImage image.
aOrientationMode = #horizontal
ifTrue: [line := myImage packedRowAt: aPoint y]
ifFalse: [line := aDicomImage mrCachedImage colAt: aPoint x].
newBits := line , newBits].
newImage := MRCachedImage new.
newImage setImage: (Depth16Image
extent: myImage width @ originalImages size
depth: 16
palette: aPalette
bits: newBits).
pilotImage := newImage shrunkenBy: 2 @ 0.3.
```

5.2.2 O Cyclops Mailer

É muito comum no contexto de uma clinica de exames médicos que após ser efetuado um exame de captura de imagens como por exemplo uma tomografia, o radiologista responsável de o laudo e envie as imagens mais interessantes juntamente com o laudo para o médico que requisitou o exame. Isso acelera o processo de diagnóstico do médico requisitante, mas sem uma ferramenta especifica e integrada ao cliente de imagens DICOM, esta tarefa pode ser bastante cansativa.

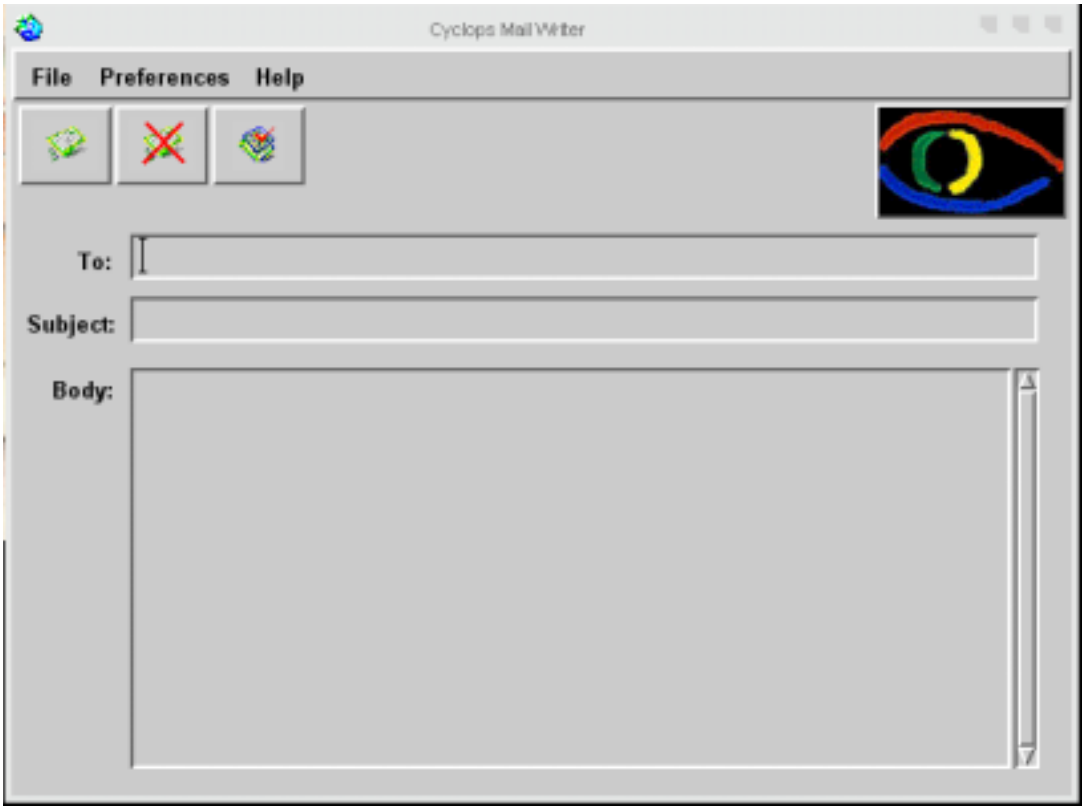


Figura 26 – Interface da ferramenta de e-mail do Cyclops Series Editor

5.2.3 O Cyclops Image Printer

Devido a inexistência de um sistema de impressão postscript funcional no ambiente de desenvolvimento, foi-se necessário apelar a modos alternativos para conseguir-se imprimir uma exame e seu respectivo laudo. Para isso foi criado um template em HTML capaz de automaticamente se adaptar conforme o tipo de exame(que pode ser tomagrafia, Ressonância magnética, ultrasonografia.). Esse arquivo html então é exportado para arquivo e pode então ser aberto automaticamente por um browser ou então transformado em arquivo postscript por uma ferramenta externa como o html2ps existente no ambiente linux.

5.2.4 A ferramenta de mensuração de área

Uma das ferramentas mais úteis num ferramenta de auxilio ao laudo é simplesmente uma ferramenta de mensuração de área, muito importante por exemplo para calcular por exemplo, o tamanho de uma determinada região de interesse, como um neurocisticerco ou a área em que ocorreu um AVC. Como dito anteriormente, o padrão DICOM armazena não somente dados de cor da imagem, mas também de tamanho e densidade de cada pixel. Com essa informação em mãos, podemos utilizar uma ferramenta simples, baseada em splines para gerar uma forma qualquer a vontade do usuário e após isso, calcular a quantidade de pixels dentros dela. Como conhecemos o valor da área de cada pixel, logo descobrimos o valor de área da região selecionada.

6 Considerações sobre Performance

6.1 Requisitos de hardware

Devido ao fato deste projeto tratar de informações que são apresentadas em grandes volumes(uma imagem dicom geralmente possui 500 kb de tamanho, levando-se em conta que uma series pode ter algumas dezenas delas...) algumas considerações a respeito dos requisitos de software e hardware devem ser feitas visando o bom funcionamento do mesmo.

Como o smalltalk, diferentemente do Java, armazena todo o seu código fonte em um único arquivo denominado imagem, a quantidade de memória RAM disponível em hardware tem que ser relativamente grande, pois a quantia de memória ram usada pelo programa pode facilmente chegar a 100 mb. Um processador adequado e uma boa placa de vídeo também são fundamentais para a usabilidade do programa.

6.2 Requisitos de software

Quanto aos requisitos de software o núcleo básico do programa pode funcionar em qualquer sistema operacional, desde os comuns Windows e linux até estações sunSparcs, graças a sua maquina virtual. Entretanto o funcionamento dos algoritmos de processamento de imagens estão até o momento dependentes do sistema operacional UNIX ou derivados pois foi desenvolvido tendo em mente uma arquitetura GPL. Entretanto existe a possibilidade de porte dos algoritmos ou da utilização de um sistema de emulação de linux dentro do Windows como por exemplo o GNU cygwin.

7 Conclusões

A implementação de uma ferramenta de visualização de imagens médicas no formato DICOM é uma tarefa complicada e muitas vezes entediante. Graças a quantidade de código que foi construído desde a origem do projeto cyclops em 1993, tem-se hoje um framework médico de grande valor, sendo muito mais fácil construir uma aplicação médica do que partir do zero.

O real objetivo por trás deste protótipo é a união da maioria das ferramentas elaboradas no projeto cyclops, visando uma maior utilidade de uma ferramenta médica e uma maior aceitação por parte dos usuários médicos de ferramentas computacionais para auxilio ao diagnostico.

Acredita-se que o objetivo deste trabalho tenha sido alcançado, pois a ferramenta apresentada aqui apresenta grande facilidade na utilização eu um desempenho aceitável para uma software desenvolvido em uma linguagem interpretada. Embora o software apresentado aqui apresente todas as características de um versão final, ele terá seu trabalho continuado após a minha graduação e provavelmente apresentará novas características que auxiliaram os médicos ainda mais na sua tarefa de melhorarem a qualidade de vida de seus pacientes.

8 Glossário

ACR	American College of Radiology
AE	Application Entity
CT	Computed Tomography
CTN	Central Test Node
DICOM	Digital Image Communications on Medicine
HIS	Hospital Information Systems
IOD	Information Object Definition
IOM	Information Object Module
IP	Internet Protocol
LLC	Logical Link Layer
MR	Magnetic Ressonance
NEMA	National Electrical Manufacturers Association
NM	Nuclear Medicine
OSI	Open Systems Interface
PACS	Picture Archiving and Communications Systems
TCP/IP	Transfer Control Protocol/Internet Protocol
UID	Unique Identifier
US	Ultra-Sound

9 Referencias Bibliograficas

Abdala,Daniel Duarte. Cyclops Personal – **Uma Ferramenta para Gerenciamento e Visualização de Imagens Médicas no Padrão DICOM 3.0**. Florianópolis, 5 de julho de 2002. 217 pág. Trabalho de Conclusão de Curso em Ciência da Computação - Universidade Federal de Santa Catarina - INE, 2002.

Dellani, Paulo Roberto . **Desenvolvimento De Um Servidor De Imagens Médicas Digitais No Padrão Dicom**. Florianópolis, fevereiro de 2001. 98 pág. Dissertação (Mestrado em Ciência da Computação) - Programa de Pós-Graduação em Ciência da Computação - PPGCC, Universidade Federal de Santa Catarina - INE, 2001.

Bortoluzzi, Mariana Kessler. **Desenvolvimento e Implementação De Um Editor De Documentos Estruturados No Padrâo Dicom Structured Report**. Florianópolis, Março de 2003. 98 pag. Dissertação (Mestrado em Ciencias da Computação) - Programa de Pós-Graduação em Ciencias da Computação - PPGCC,Universidade Federal de Santa Catarina – INE , 2003.

WANGENHEIM, Aldo von, **Página da disciplina de Visão Computacional**. Disponível em: <http://www.inf.ufsc.br/~visao/>. Acesso em: 15 de jul. 2003.

Berti, L. A. **Implementação de um protótipo de software para análise, mensuração e Reconstrução de aneurismas de artéria aorta abdominal**. 2003. Trabalho de Conclusão de Curso de Engenharia da Computação - Universidade do Vale do Itajaí - Unisul, Santa Catarina

PRATT,WILLIAM K.. **Digital Image Processing**. Addisson Wesley .2001

BECK, Kent. Smalltalk: Best Practice Patterns. New Jersey: Prentice

Hall, 1997. 224 p.

Abdala, Daniel Duarte;Wangenheim, Aldo von. **Conhecendo Smalltalk. Florianópolis**: Editora Visual Books, 2002. V. 1. 300 pág.

CLUNIE, David A. Medical Image Format FAQ. **Questões freqüentemente perguntadas sobre Formatos de Imagens Médicas, com enfoque para o padrão DICOM**. Disponível em: <http://www.dclunie.com/medical-image-faq/html/>. Acesso em: 10 de novembro de 2003.

EFILM MEDICAL INC. **eFilm Workstation 1.5.3 Software** . Disponível em: <http://www.efilm.ca/ >. Acesso em: 05 de fevereiro de 2001.

NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION. **Digital Imaging and Communications in Medicine (DICOM) – Todas as partes.** Virginia, 2000. Disponível em: <ftp://medical.nema.org/medical/dicom/2000/draft/ >. Acesso em: 10 de novembro de 2003.

UNIVERSITY HOSPITAL OF GENEVA. **The Osiris Imaging Software**. Disponível em: <http://www.expasy.ch/www/UIN/html1/projects/osiris/osiris.html>. Acesso em: 01 de fevereiro de 2001.

Wagner,Harley Miguel . **ATLAS CEREBRAL DIGITAL: DESENVOLVIMENTO DE UMA FERRAMENTA COMPUTACIONAL PARA MAPEAMENTO FUNCIONAL E ANATÔMICO DE ÁREAS CEREBRAIS, BASEADO NO ATLAS DE TALAIRACH.** Florianópolis, Março de 2001, 88 pag. Dissertação (Mestrado em Ciencias da Computação) - Programa de Pós-Graduação em Ciencias da Computação - PPGCC,Universidade Federal de Santa Catarina – INE , 2001.

Anexo 1 - Manual de Configuração

Para instalar o Cyclops Series Editor você deve inicialmente instalar a máquina virtual do Cincom VisualWorks, que pode ser encontrado em http://www.cincom.com/smalltalk . Após baixar o arquivo, deve-se executá-lo e seguir as instruções fornecidas pelo programa. O arquivo de instalação rodará tanto no Linux quanto no Windows, devido ao fato dele próprio ser uma imagem smalltalk e, portanto independente de sistema operacional.

Quando a instalação for completada, deve-se copiar o arquivo de imagem do Cyclops Series Editor para o diretório desejado. Esse arquivo terá extensão .im e seu nome provelmente sera algo como cyclopsMIB ou cyclopsSeries. Caso você não disponha de um arquivo de imagem da aplicação, e sim dos parcels (fontes) que a constituem, você deve proceder da seguinte maneira:

- Copie os arquivo visualnc.im e visualnc.cha do Cincom Smalltalk, geralmente encontrada (...)\vw7nc\image\ para o diretório desejado.
- Execute o arquivo imagem, clicando duas vezes nele ou então construa um atalho para a maquina virtual, geralmente localizada em: (...) vw7nc\bin\win no caso do Windows e depois modifique o alvo (Target) do arquivo para algo semelhante a figura 34. Ou então no diretório da imagem digite : *"Local da máquina virtual" –h 100m "local da imagem"*
- No menu file do launcher do Visual Works, escolha a opção file browser e navega até a pasta onde se encontram os parcels. Carregue-os de acordo com a seqüência:

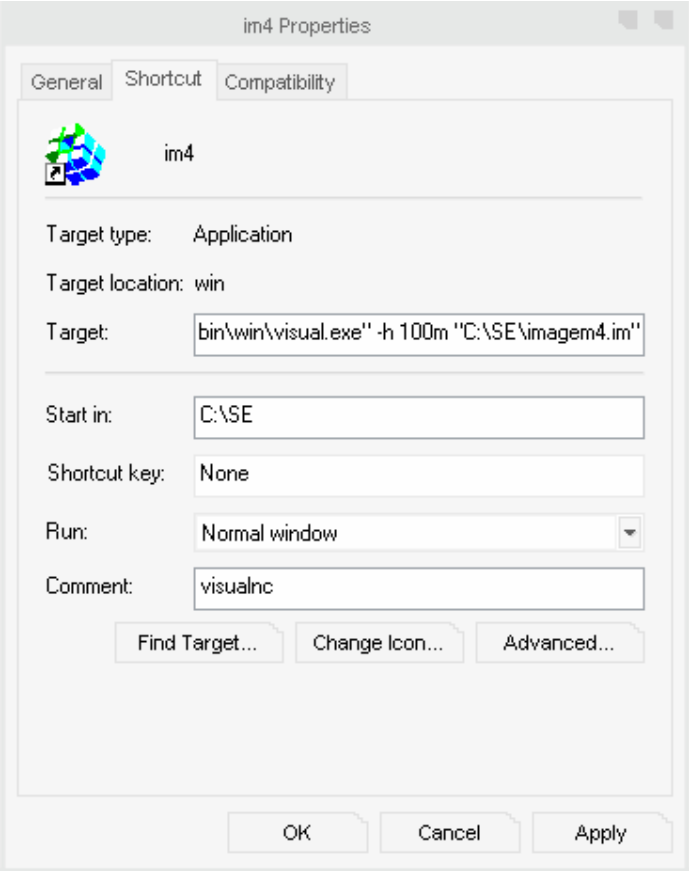


Figura 27 – Exemplo de atalho para roda uma imagem do Cyclops Series Editor

9.1 Configurando o Cyclops Medical Image Browser

Na mesma pasta onde você colocou o arquivo imagem, deve existir um arquivo XML chamado CyclopsMIB.xml. Este arquivo contém as informações de configuração para o aplicativo Cyclops Medical Image Browser. Abaixo um exemplo do arquivo:

```
<ApplicationEntities name="HU/UFSC">
  <head>
    <site>Laboratorio de Telemedicina / CES / HU / UFSC</site>
    <address>Campus Universitario UFSC</address>
    <city>Florianopolis</city>
    <state>SC</state>
    <country>Brasil</country>
    <www>http://www.telemedicina.ufsc.br</www>
    <eMail>cyclops@telemedicina.ufsc.br</eMail>
    <browser>/usr/local/cyclops/netscape/netscape</browser>
  </head>

 <dicomUpperLayer>
    <port>4007</port>
    <maximumPDULenght>65536</maximumPDULenght>
```

```
    <maximumSimultaneousConnections>50</maximumSimultaneousConnections>
    <ARTIM>
      <requestTime>50</requestTime>
      <rejectTime>50</rejectTime>
      <releaseTime>25</releaseTime>
    </ARTIM>
  </dicomUpperLayer>

  <remoteAE>
    <default>CYCLOPS</default>
        <dicom>
      <AETitle>ABDOMEN-PELVE</AETitle>
      <label>ABDOMEN-PELVE</label>
      <host>cranio.telemedicina.ufsc.br</host>
      <address>#[150 162 67 1]</address>
      <port>4001</port>
    </dicom>
  </remoteAE>

  <localAE>
    <dicom>
      <AETitle>Cranio</AETitle>
      <label>Cranio</label>
      <host>cranio.telemedicina.ufsc.br</host>
      <address>#[150 162 67 1]</address>
      <port>4007</port>
    </dicom>
    </localAE>
</ApplicationEntities>
```

Dentro da tag <head> são contidas apenas informações de localização reais da máquina e informações da instituição onde ela existe. A partir daí teremos a tag <dicomUpperLayer> e sua subtags:

- <dicomUpperLayer> - inicial o trecho de configuração de comunicação com o servidor.
- <port>4007</port> - defini a porta padrão de conexão ao servidor.
- <maximumPDULenght>65536</maximumPDULenght> - define o tamanho máximo do pacote de dados a ser recebido e enviado pelo servidor DICOM
- <maximumSimultaneousConnections>50</maximumSimultaneousConnections> - Define o número máximo de conexões simultâneas a serem aceitas pelo servidor.

- As configurações abaixo pode ser deixadas como estão por padrão.

```
<ARTIM>
<requestTime>50</requestTime>
<rejectTime>50</rejectTime>
<releaseTime>25</releaseTime>
</ARTIM>
```

- </dicomUpperLayer> - fecha o trecho de configuração de comunicação com o servidor

O próximo trecho de configuração será o pertencente a tag <remoteAE>. Nela serão listados os bancos de dados onde se fará a conexão remota, ou seja, os endereços ips e hostnames das máquinas onde estão hospedadas as imagens DICOM.

- <remoteAE>. – inicial configuração de conexão remota;
- <default>CYCLOPS</default> - define o servidor padrão;
- <filesystem> - lista um sistema de arquivos remoto que poderá ser usado para receber imagens;
    - <AETitle> Home Dir</AETitle> - o nome do banco de dados a que pertence essa configuração;
    - <host>cerebro.com.br. </host> O hostname ou ip da maquina a ser contatada;
    - <label>Home Dir</label> - O nome que será exibido na janela do programa
    - <address>#[150 162 67 1]</address> - o endereço ip, substituindo-se os pontos por espaços, do servidor;
    - <port>4001</port> A porta de conexão ao servidor

    - <baseDirectory>/home/simon</baseDirectory> O diretório que será acessado pelo servidor para fornecer os arquivos DICOM.
- </filesystem> fecha a tag de sistema de aquivo.

Aqui inicia a configuração de um banco de dados

- <dicom> - lista uma servidor remoto que poderá ser usado para lista e receber imagens de um banco de dados;
- <AETitle>ABDOMEN-PELVE</AETitle> - nome do banco de dados a que pertence essa configuração;
- <label>ABDOMEN-PELVE</label>- O nome que será exibido na janela do programa
- <host>cerebro.com.br</host> o hostname ou o ip da maquina a ser contatada
- <address>#[150 162 67 1]</address> - o endereço ip, substituindo-se os pontos por espaços, do servidor
- <port>4001</port> A porta de conexão ao servidor
- </dicom> termina a tag
- </remoteAE> termina a tag

A configuração das subtags pertencentes a <LocalAE> seguem os mesmos critérios dos da tag <RemoteAe>. Após o termino da configuração do arquivo XML, você deve executar a máquina virtual e abrir o Cyclops MIB para conferir se tudo está correto.. Para isso você pode proceder de duas maneiras, dependendo do sistema operacional que você estiver utilizando:

- Dentro de um sistema UNIX como o linux, você deve estar no diretório onde se encontra a imagem e o arquivo XML, e digitar o comando: visual7 –h 100m nomedaimagem.im & . Este comando automaticamente abrirá o programa,

caso a imagem seja um executável. Se a imagem for uma imagem de desenvolvimento, você deverá abrir um workspace e digitar CyclopsMIB open.

- Caso o sistema for Windows, você deve seguir o exemplo dado na página anterior e construir um atalho à maquina virtual e a imagem executével.

Isso abrirá a janela exibida na figura abaixo:

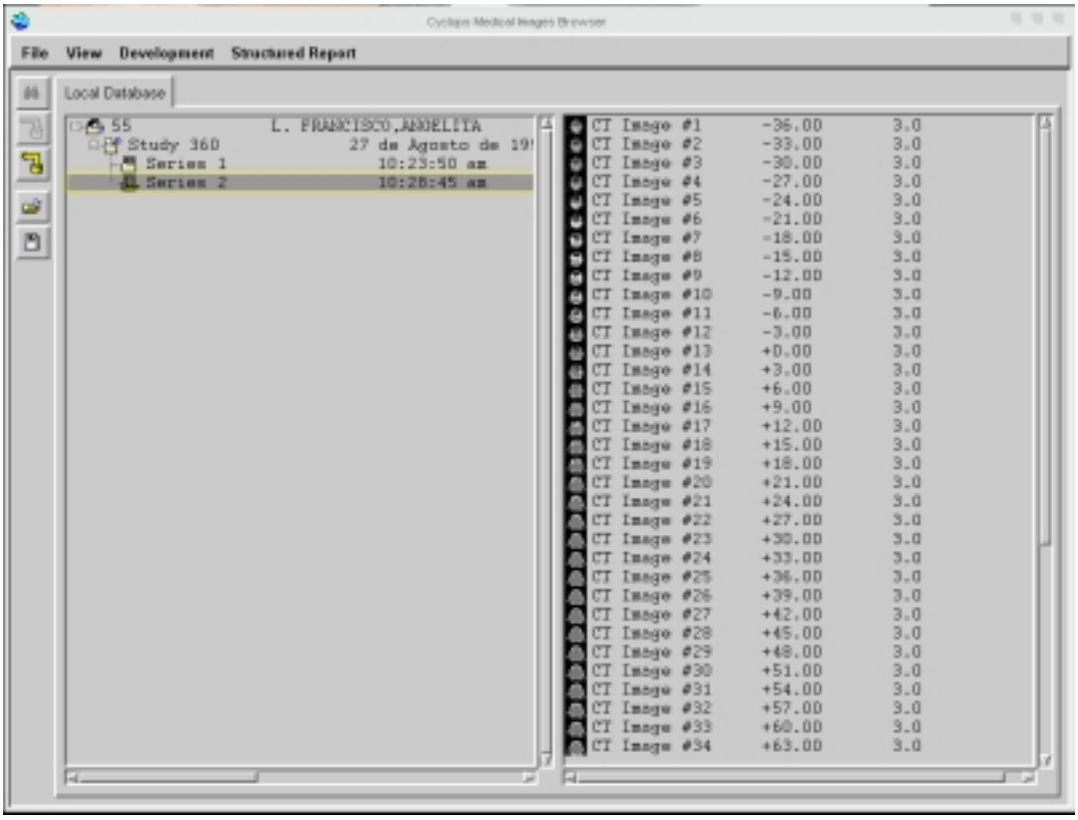


Figura 28 – Cyclops MIB

Para abrir o series editor deves se clicar na lista esquerda, nos campos pertencentes ao nível de estudo ou série com o botão direito, ir a opção lançar aplicação e então clicar sobre Cyclops Series Editor. Veja a figura 36.

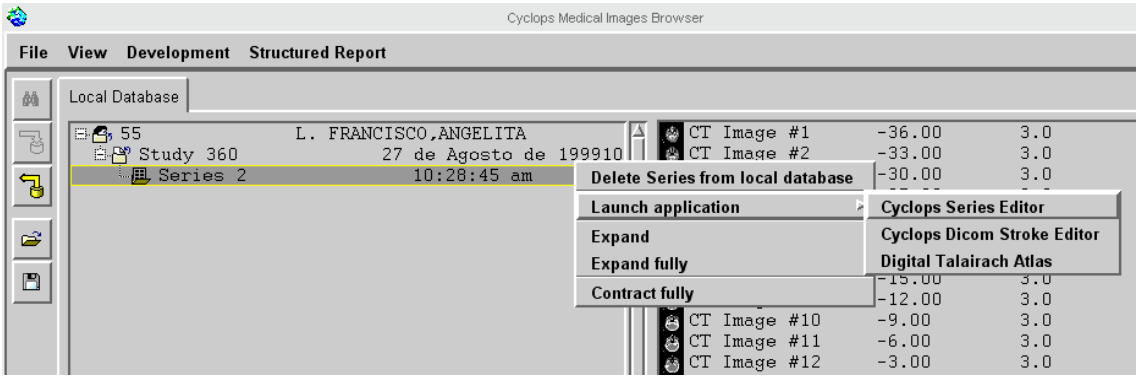
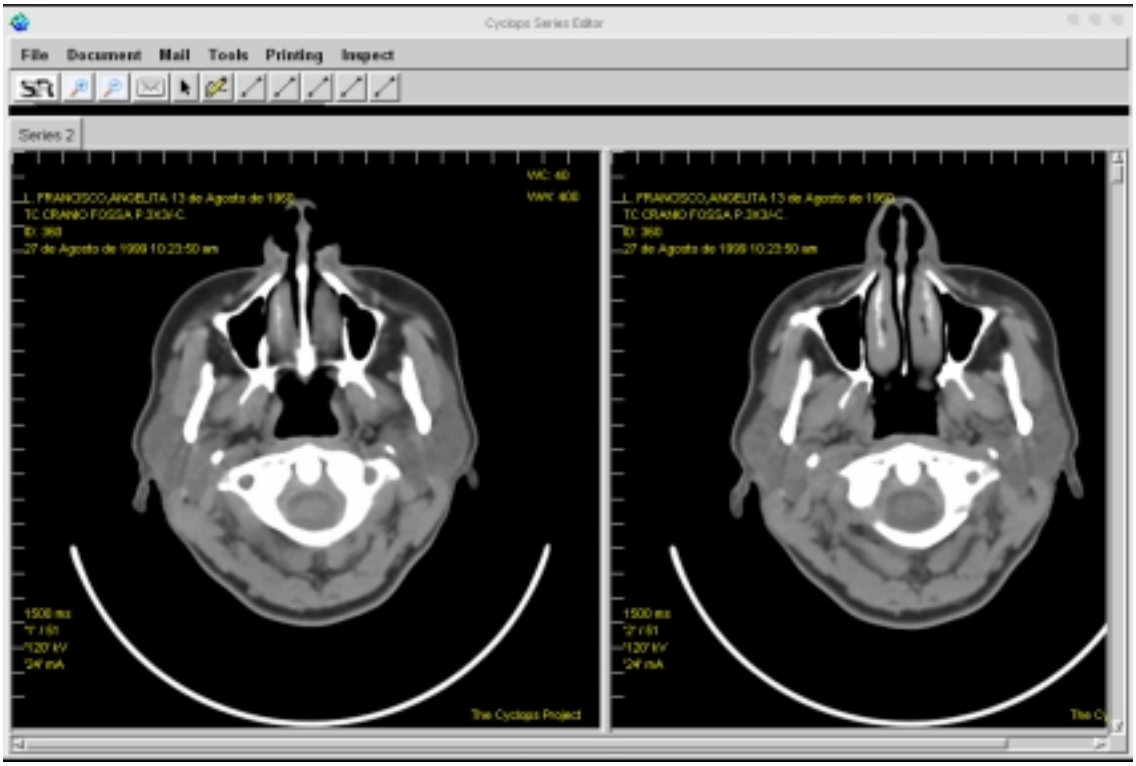


Figura 29 - Abrindo o Cyclops Series Editor

Isso abrirá a janela abaixo que dará acesso ao Cyclops Series Editor e as suas diversas ferramentas :



9.2 Configurando o cyclops Mailer

Clicando com o botão direito em cima de uma imagem qualquer você pode marca-lá para enviar por e-mail. Mas para isso você precisará antes configurar o Cyclops Mailer fornecendo os servidores de STMP:

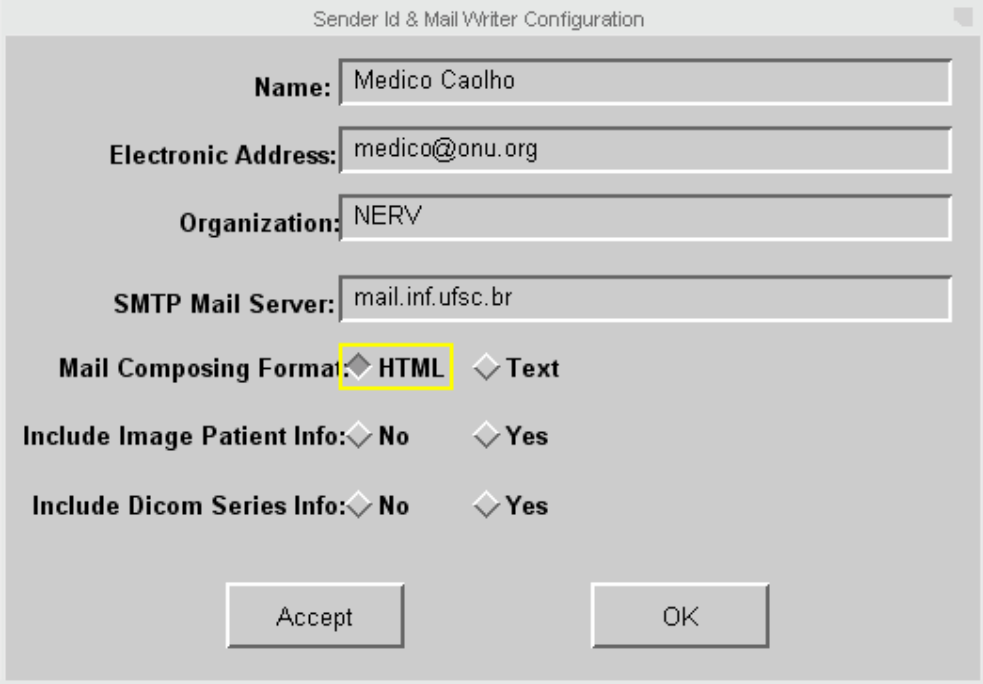


Figura 30 Configuração do Cyclops Mailer.

9.3 Configurando o Cyclops Structured Report

Para configurar o Cyclops Structured Report Editor você precisará de uma série de arquivos de dicionários de termos médicos que estão contidos nas seguintes pastas:

- Dicionários
- XMLTemplates

A pasta dicionário é composta de arquivos de dicionário de informações, que podem ser termos médicos, unidades ou qualquer outra coisa.

Veja um exemplo:

```
0101dmi1|TOMOGRAFIA COMPUTADORIZADA DO ABDOME E PELVE|
340102dmi1|Descricao|Realizado estudo do abdome e pelve em modo [ axial /
helicoidal ], após administração de contraste por via oral, e antes,
durante e após administração endovenosa do meio de contraste iodado.|
340103dmi1|Figado|Fígado de dimensões e contornos normais.|
340104dmi1|Parenquima|Parênquima  hepático com coeficientes de atenuação
normais aos raios X e impregnação homogênea.|
340105dmi1|Vias biliares|Vias biliares intra e extra-hepáticas de calibre
normal.|
340106dmi1|Vesicula|Vesícula biliar normodistendida, de paredes finas, com
conteúdo líquido [homogêneo].|
```

A pasta XMLTemplates contem uma séries de arquivos que contém modelos pré prontos de laudo que podem ser editados e assim poupar tempo ao médico, já que a grande parte do trabalho de laudo diário é a repetição do laudo anterior.

```
<CyclopsDicomSR_IOD>
<CyclopsDicomSRDocumentContentIOM valueType="CONTAINER"
continuityOfContent="SEPARATE" >
      <CyclopsDicomConceptNameCodeSequenceItem codeValue="340113"
codingSchemeDesignator="dmi" codingMeaning="Adenomegalias"
codingSchemeVersion="1" />
      <CyclopsDicomContentSequenceItem valueType="TEXT"
relationshipType="CONTAINS">
            <CyclopsDicomConceptNameCodeSequenceItem codeValue="340199"
codingSchemeDesignator="dmi" codingMeaning="CONCLUSAO"
codingSchemeVersion="1" />
      </CyclopsDicomContentSequenceItem>
      <CyclopsDicomContentSequenceItem valueType="TEXT"
relationshipType="CONTAINS">
            <CyclopsDicomConceptNameCodeSequenceItem codeValue="340114"
codingSchemeDesignator="dmi" codingMeaning="Bexiga" codingSchemeVersion="1"
/>
      </CyclopsDicomContentSequenceItem>
</CyclopsDicomSRDocumentContentIOM>
 </CyclopsDicomSR_IOD>
```

Anexo 2 - Manual básico de utilização do Cyclops Series Editor e suas ferramentas

Após abrir o Cyclops Series editor você pode através da barra de ferramentas acessar diversos botões, abaixo estão listadas suas funcionalidades:

9.4 Utilizando o Cyclops Structured Report

Botão SR - Para criar um SR você pode proceder de duas maneiras, clicando diretamente no botão SR na barra de ferramentas ou indo ao menu Document e clicando em "new SR", isto abrirá uma janela que perguntará qual modelo (template) de laudo você gostaria de criar.

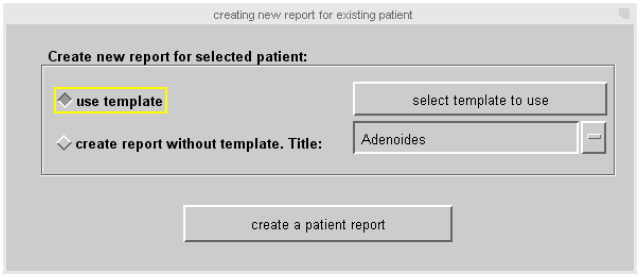


Figura 31 - Seleção de template do Cyclops Structured Report

Após selecionar o modelo você deve clicar em “create a patient report”. Isso irá adicionar uma nova tabua na janela do Cyclops Series Editor e exibirá a janela abaixo:

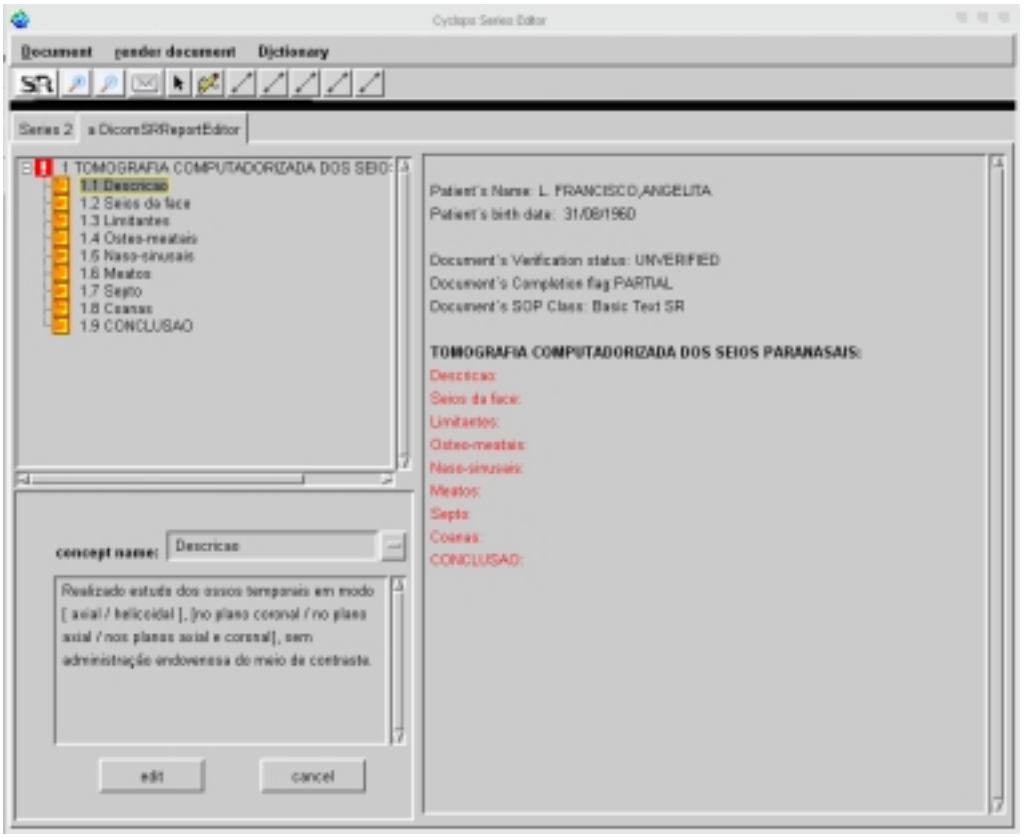


Figura 32 - Nova tabua contendo uma versão do Cyclops SR

Para editar um item basta clicar sobre ele e mudar a descrição na campo de edição abaixo da lista. Quando você acabar de escrever, clique novamente em Edit para confirmar, o campo selecionado na lista mudara de cor para azul.

Clicando no menu Documento você poderá salvar o documento Sr criado em arquivo ou no banco de dados local, lá você poderá mandá-lo para o banco de dados após revisá-lo.

9.5 Outros botões

Clicando nos botões   você pode incrementar ou decrementar o tamanho da imagem em 10 %. O único problema disso é que é relativamente demorado.

Clicando no botão  você automaticamente lançara o Cyclops Mailer, entretanto para conseguir mandar uma imagem por e-mail, você deverá anteriormente marcá-las, clicando sobre elas com o botão direito.

Clicando no botão  dará acesso a ferramenta desenho livre.

Clicando na ferramenta  dará acesso a ferramenta de mensuração de distância.

Clicando sobre a imagem com o botão direito você pode marcar e desmarcar uma imagem, para utilizá-la posteriormente no mailer ou com ferramentas de processamento de imagens. Perceba que após você marcar uma imagem, ela passará a ser exibida com uma borda vermelha.

Clicando sobre os botões  de reconstrução multiplanar você pode mudar a posição do corte que aparecera.

Para visualizar as reconstruções multiplanares você precisa clicar na barra abaixo da barra de ferramentas e depois novamente na barra que aparecerá a esquerda da tela. Essa organização foi necessária para poder se trabalhar com resoluções aceitáveis para o padrão brasileiro, que geralmente possui monitores de 14 polegadas com no máximo 1024 x 800 pontos.

Através do menu Tools você poderá acessar o grupo de ferramentas de auxilio ao diagnóstico do Cyclops Series Editor. Abaixo uma breve explicação de sua utilização:

O Cyclops PopCorn é muito simples de utilizar, basta selecionar um grupo de imagens e clicar em Detect, após a execução, as neurocisticercoses que o programa identificar serão marcadas por uma bolinha vermelha.

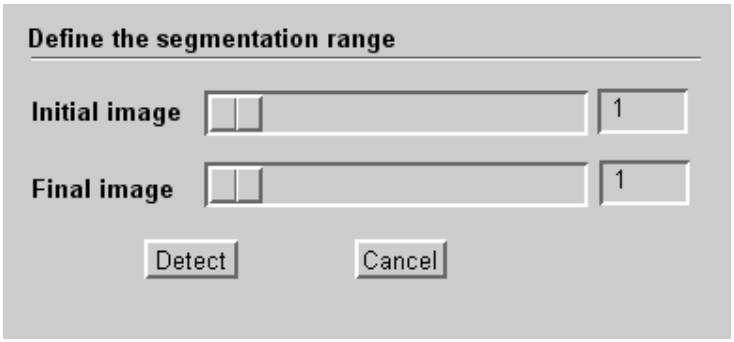


Figura 33 - Cyclops PopCorn

Selecionando o Brain Atlas no menu Tools, dois novos botões apareceram na barra de ferramentas:  .O primeiro botão dá acesso ao cubo inicial onde você poderá fazer os ajustes manuais de largura e altura do Atlas. Bastando para isso clica no botão de ajuste na mesma barra de ferramentas. O Segundo exibirá o Atlas propriamente dito.Entre as duas reconstruções multiplanares existe uma área de ajuste, onde você poderá rotacionar os Atlas conforme a necessidade. Um quarto botão, o Locate, permite que você selecione uma área funcional. Informações sobre essa área então serão exibidas na janela do Series.

9.6 Usando o Cyclops Stroke.

Após selecionar o Stroke você deve selecionar a tabua “Contrast Flow Visualization” e passar o mouse em uma região de interesse.

Após isso mude para a tabua para “Contrast Flow Comparision”.

Clique numa imagem e selecione uma área de interesse e desenhe um retângulo sobre ela. Clique com o botão direito e depois em Calculate.

Anexo 3 – Analise de Requisitos

Neste capítulo serão apresentados: o organograma do sistema CyclopsMIB Series Editor, e alguns de seus casos de usos extendidos

9.6.1 Organograma

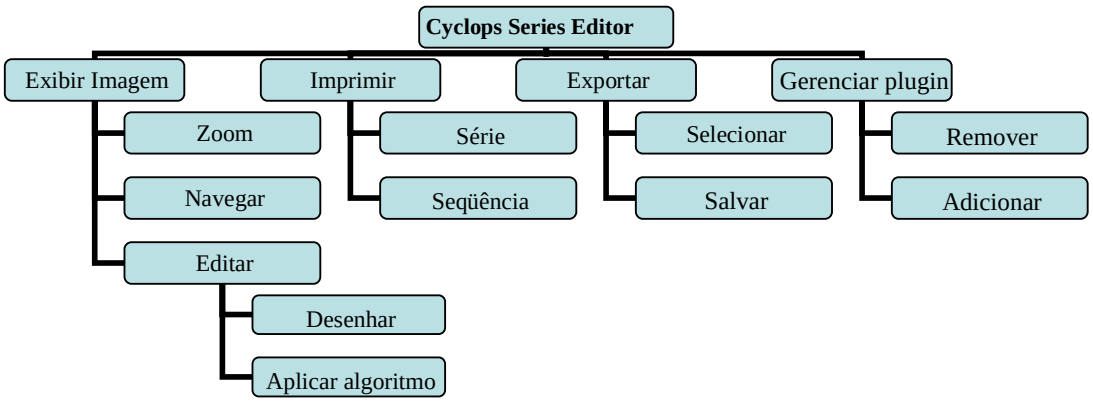


Figura 34 – Organograma básico de funcionamento do Cyclops Series Editor

9.6.2 Descrição Textual dos Processos.

Processo 1: Exibir Imagens

Cada imagem é exibida individualmente de acordo com seu tamanho original, o que pode variar a forma com que ela será disposta na tela. Caso uma imagem possua outras imagens derivadas, será possível navegar por ela como se folheássemos um álbum de fotografias.

Processo 2 : Imprimir

As imagens poderão ser impressas de várias formas:

- Uma imagem individualmente.
- Toda a série de imagens carregadas
- Todas as imagens derivadas de uma imagem original da série.

Processo 3: Exportar Imagens

Será possível, após selecionar o formato, salvar uma imagem ou uma série de imagens. Essas imagens poderão estar nos seguintes formatos: raw, vista, viff, sun Raster, pnm e pgm.

Processo 4: Gerenciar Plugins

Um plug-in nada mais será que um programa constituindo de uma expansão das capacidades atuais do Series Editor, isso poderá ser desde um novo algoritmo de analise de imagens até algo mais complexo. Eles poderão facilmente ser incluídos e removidos a partir do menu ferramentas (tools). É denominado aqui plug-in o software adicionado ao sistema que não cria modificações no código original do programa, como por exemplo, o “kit de processamento de imagens” e o CyclopsPopCorn, Cyclops WaveForm e Cyclops Structured Report.

Processo 5: Gerenciar Add-ons

Um add-on neste caso, será um programa constituindo de uma expansão das capacidades

atuais do Series Editor que necessita de pequenas alterações no código fonte original para prover os novos recursos. Entretanto ele não alterará o funcionamento dos recursos anteriores. Esse é o caso das ferramentas Cyclops Brain Atlas e do Cyclops Stroke Quantifier.

9.7 Casos de Uso Extendidos

Caso de uso: Impressão da série original

Atores: usuário médico

Finalidade: Imprimir a série de imagens e os dados que ela possui e permitir ao ator a inserção de comentários.

Tipo: primário e essencial

Pré-condições: as imagens estão corretamente carregadas

Pós-condições: O sistema não sofre alterações.

Visão geral: O usuário médico, desejando imprimir todas as séries de imagens, aponta o mouse ao menu File, seleciona a opção imprimir série aguarda o término da impressão.

Seqüência Típica de Eventos

Ação do ator:	**Resposta do sistema:**
1. Esse caso de uso começa quando o usuário deseja imprimir a serie de imagens que esta vendo. Para isso ele escolhe a opção adequada no menu.	**2.** Abrir uma janela com opções de impressão: Imprimir todas as imagens; Imprimir todas as imagens derivadas da imagem selecionada; Inserir comentários;

Recuperar dados dos pacientes e confirmar;

3. O usuário seleciona a opção desejada e confirma.

4. O sistema requisita ao usuário que o mesmo entre com o comentário sobre a serie.

5. O usuário faz o comentário e confirma.

6. O sistema gera as imagens e o HTML e a seguir,manda abrir o browser com o arquivo para impressão, ou gerá um arquivo postscript através do programa html2ps e envia-o diretamente a impressora.

Caso de uso: Aplicar algoritmo de processamento de imagem

Ator: usuário médico

Finalidade: Ver o resultado do processamento de uma imagem radiológica

Visão geral: O médico escolhe o algoritmo que deseja e seleciona a configuração desejada e aplica-o sobre as imagens selecionadas. No final o usuário visualiza a imagem resultante.

Seqüência Típica de Eventos

Ação do ator:	**Resposta do sistema:**
1. Este caso de uso começa quando	**2.** O sistema aplica o algoritmo para

o medico decide que precisa de uma informação ou imagem resultante de um algoritmo de analise. Ele seleciona as imagens onde deseja aplicar os algoritmos e configura de acordo com o a imagem que deseja obter. Aplica o algoritmo.

cada imagem e adiciona como imagem derivada a cada respectiva imagem. Quando o algoritmo termina é exibida uma mensagem avisando o usuário.

3. o usuário visualiza a imagem.

Seqüência Alternativa:

3a. O sistema retornou mais de uma imagem (porque o algoritmo gerou assim) então o usuário deve navegar pelas imagens, usando os botões de navegação.

2a. O algoritmo não pode ser aplicado com a configuração desejada: Informa ao usuário que a configuração é inadequada e sugere uma nova configuração ideal.

2b. O algoritmo falha não geração da imagem devido a falta de algum componente. Avisa o usuário de que não foi possível executar corretamente o algoritmo e desabilita a opção.

Caso de uso: Desenhar ou Medir

Atores: Usuário médico

Finalidade: Destacar ou medir uma determinada imagem.

Visão geral: O medico seleciona a ferramenta de desenho desejada e aplica diretamente sobre a imagem.

Tipo: primário

Seqüência Típica de Eventos

Ação do ator:	**Resposta do sistema:**
1. Este caso de uso começa o usuário tem a necessidade de medir ou destacar alguma área. Então ele seleciona a ferramenta desejada e clica sobre a imagem, fazendo um desenho qualquer.	**2.** O sistema guarda os pontos onde o usuário clicou e de acordo com a ferramenta escolhida realiza um desenho ou calculo e exibe isso nos pontos anteriormente definidos.

Seqüência Alternativa:

1. O usuário poderá arrastar o desenho e movê-lo para uma outra área.

2. O usuário poderá desejar que em caso de medição de uma área que essa informação aparecerá numa outra cor ou que ele possa fazer algum comentário sobre a região.

Caso de uso: Navegar Sobre Derivações

Atores: usuário médico

Finalidade: Visualizar uma seqüência de derivações de uma imagem original..

Visão geral: O medico escolhe uma imagem especifica e através dos botões de navegação ou da lista de imagens exibe as imagens numa seqüência qualquer.

Tipo: primário e fundamental.

Seqüência Típica de Eventos

Ação do ator:	**Resposta do sistema:**
1. Este caso de uso começa o usuário tem a necessidade de navegar sobre as imagens derivadas (a partir de um algoritmo de analise) de uma imagem original. Ele pode navegar pelas imagens clicando nos botões de navegação de cada imagem	**2.** O sistema verifica a imagem atual e prossegue exibindo a imagem derivada armazenada de acordo com os comandos do usuário.

Seqüência Alternativa:

1a. O usuário poderá exibir uma imagem fora de ordem, bastando pra isso exibir a lista de imagens derivadas e clicar em cima dela.

Caso de uso: Executar ferramenta auxiliar

Atores: usuário médico

Finalidade: Adicionar novas funcionalidades ao CyclopsMIB Series Editor

Tipo: secundária

Seqüência Típica de Eventos

Ação do ator:	**Resposta do sistema:**

1. Este caso de uso começa o usuário tem a necessidade expandir a capacidade do Cyclops Series Editor original. Ele seleciona a nova aplicação no menu ferramentas.

2.O sistema substitui a ferramenta que estiver em operação atualmente (saso exista) e executa os procedimentos necessário para ativá-la.

9.8 Requisitos

Aqui apresentamos os requisitos básicos do sistema que foram elaborados inicialmente.

9.8.1 Requisitos Funcionais:

R1 - O Sistema deve ser capaz de exibir as imagens em qualquer tamanho e redimensioná-las em qualquer momento, acordo com a vontade do usuário.

R2 - O sistema tem que permitir a impressão uma imagem.

R3 - O sistema tem que permitir a impressão uma serie de imagens.

R4 - O sistema tem que permitir a impressão uma sequencia de imagens,obtidas a partir de uma original.

R5 - O sistema deve fornecer informações sobre a série, como nome do paciente, data do exame, numero do slice, slice thikness, posição do slice, etc.

R6 - Deve ser capaz de aplicar algoritmos de analise de imagens sobre qualquer imagem marcada ou sobre todas as imagens de acordo com a vontade do usuário.

R7 -Deve ser possível navegar pela serie de imagens derivadas de uma imagem sem que se tenha que mudar ou gerar derivações semelhantes nas outras imagens.

R8 - Deve ser possível exportar as imagens em outros formatos, seja essa imagem original ou derivada.

R9 - Deve ser possível gerar reconstruções das imagens que permitiram a visualização do plano sagital e coronal da serie de imagens.

R10 -Deve ser possível gerar um desenho sobre a imagem e salvar esses desenhos.

9.8.2 Requisitos Não Funcionais:

RN 1 - Interface simples: o programa tem que conter uma interface bastantes simplificada, para que os usuários médicos possam utilizar com máxima eficiência, mas mesmo assim ser capaz de exibir o máximo de informações possíveis.

RN 2 - Tempo de resposta: É conveniente que o sistema tenha um tempo de resposta razoável de acordo com o tipo de operação que estiver realizando. Entretanto alguns algoritmos que o Series Editor possui podem demorar um tempo muito longo, então isso so será garantido em alguns casos as serem discutidos.

RN 3 - Confirmação de operação concluída: O sistema deve informar quando acabou de realizar uma operação e sobre a corretude da mesma.

RN 4 - Independência de Sistema Operacional: O Sistema deve ser independente de sistema operacional e ser capaz de executar todos os algoritmos seja ele qual for.

9.8.3 Interfaces do Sistema

Para executar o Cyclops Series Edito é necessário antes ter carragado as imagens DICOM através do programa Cyclops Medical Image Browser e clicar no nível de serie ou estudo pertencente a lista esquerda.

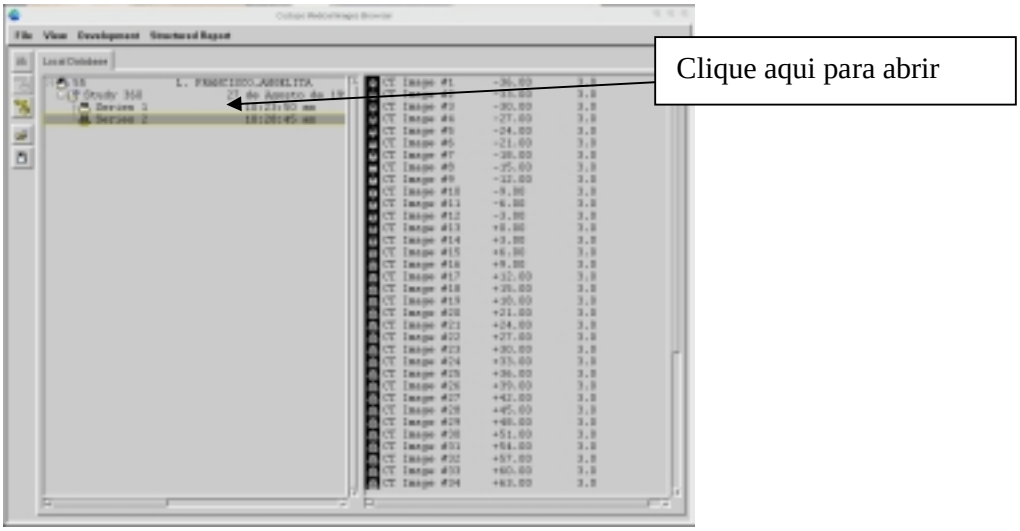


Figura 35 - Interface do Cyclops Medical Image Browser

Após abrir o Cyclops Series Editor, o usuário se defrontará com a interface abaixo:

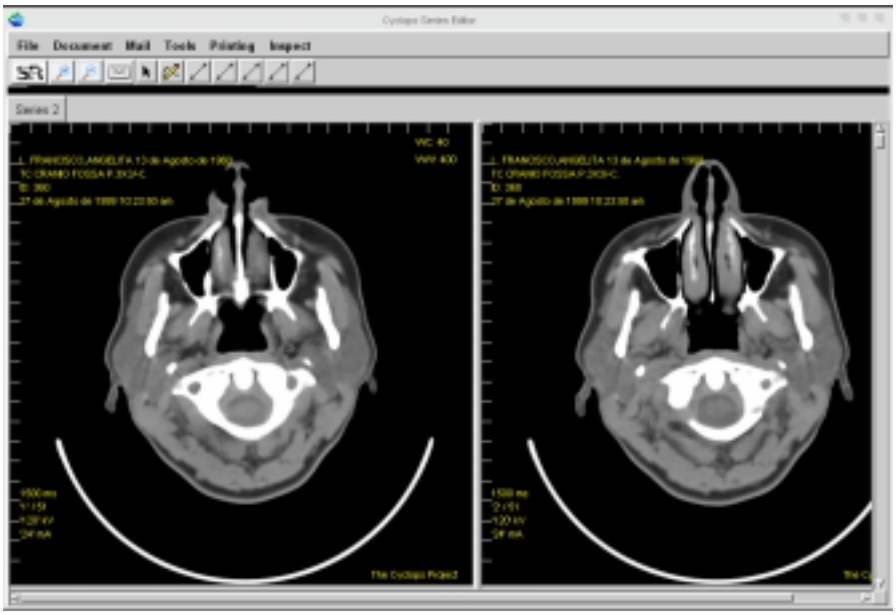


Figura 36 – Interface inicial do Cyclops Series Editor

Clicando na tarja preta abaixo da barra de ferramentas, aparecerá uma área escondida, desenvolvida para exibir o painel de configuração das aplicações auxiliares e para exibir a reconstrução multiplanar das serie de imagens exibida.

Área de exibição das reconstruções multiplanares:

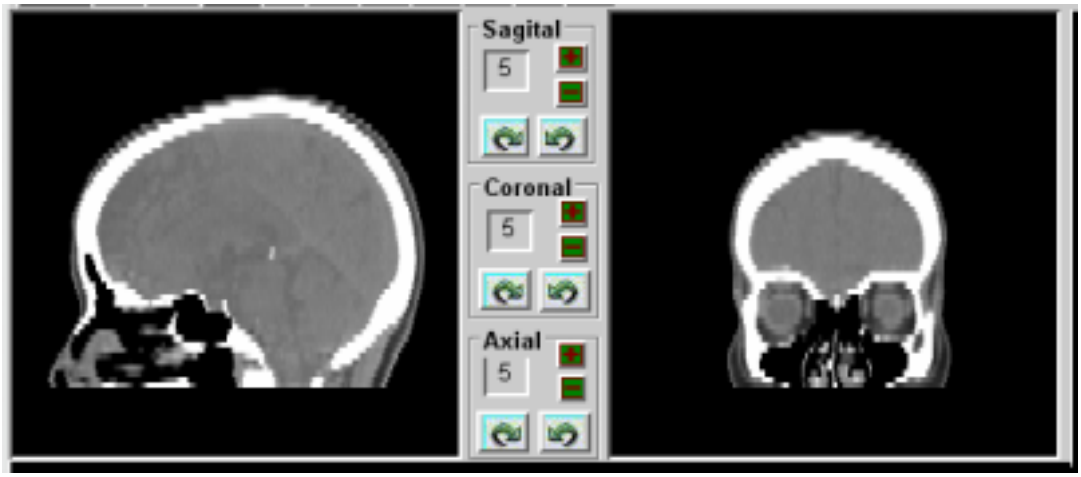


Figura 37 Exibição das recontruções multiplanares de uma serie de tomografias

Área de exibição de ferramentas:

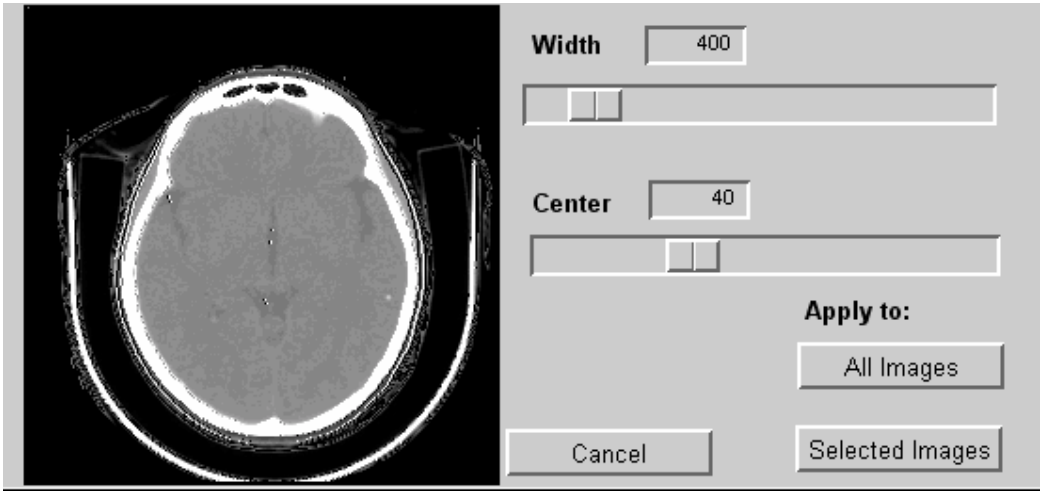


Figura 38 Ajuste de window radiológica

No campo de visualização de imagens temos a aplicação responsável por exibir as imagens e permitir a navegação entre as imagens derivadas a partir da original.

Cada imagem será exibida numa tela que fará parte de uma tab, que além de exibir a imagem exibirá uma lista de imagens derivadas. Através dos botões de navegação abaixo da imagem, será possível visualizar as imagens derivadas, quando elas existirem. Selecionando-se o campo Mark image selecionara a imagem principal, para que possa aplicar um novo algoritmo de processamento de imagens. Se você quiser selecionar um grupo de imagens derivadas bastará ir na lista de imagens derivadas e marcá-las.. Haverá também um identificador para a imagem no canto esquerdo inferior que mostra o nome da imagem.

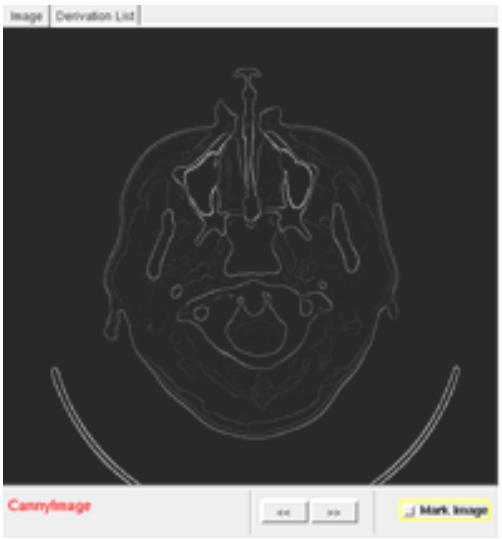


Figura 39 – Interface de visualização de uma única imagem (como algoritmo canny aplicado)

A janela para se selecionar o formato de exportação de imagem será acessível através do menu File, no canto esquerdo superior

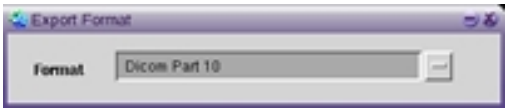


Figura 40 – Ferramenta de seleção de formato de exportação de imagem

Anexo 4 – Artigo

Cyclops Series Editor: Uma ferramenta de visualização de imagens e auxílio ao laudo e ao diagnóstico

Rafael Simon Maia

Ciências da Computação, Departamento de Informática e Estatística - INE
Universidade Federal de Santa Catarina (UFSC), Brasil, 88040-900
Fone (0XX48)333-9999, Fax (0XX48)333-9999
simon@ctc.ufsc.br

Resumo

Este artigo apresenta uma breve explanação do desenvolvimento de uma ferramenta de auxilio ao diagnóstico, ao laudo e a visualização de imagens.

Palavras-chave: DICOM, imagens médicas, cliente de imagens médicas, Cyclops, Cyclops Series Editor, Cyclops Brain Atlas, Cyclops PopCorn , Cyclops Stroke, Cyclops Structured Report.

Introdução

Com o surgimento da tomografia computadorizada na década de setenta e a aplicação dessa modalidade de exame na prática, começou-se a utilizar computadores em ambientes clínicos e hospitalares para manipular e armazenar as imagens radiológicas, e com isso surgiram padrões proprietários de armazenamento para cada fabricante.
Esses padrões eram incompatíveis entre si, e logo foi preciso criar um padrão único, para facilitar a vida de todos. Surgiu então o ACR-NEMA 1.0 que mais tarde, em sua terceira versão viria a ser chamado DICOM 3.0. Então a utilização de workstations para visualização das imagens radiológicas começou a se tornar rotina .

Motivação

Poderíamos pensar que devido a grande quantidade de ferramentas de visualização de imagens em formato DICOM seria inútil a construção de mais uma ferramenta similar, mas esta conclusão é completamente errônea, ainda mais levando-se em conta a realidade dos hospitais públicos brasileiros, que sofrem com falta de verbas até mesmo para a manutenção do seus antiquados sistemas de radiologia. Geralmente sistemas de comunicação e arquivamento, os PACS, são vendidos em kits fechados que podem custar algumas centenas de milhares de reais. Se apenas as ferramentas de visualização forem compradas esse valor ficara próximo de 1500 reais (preço do e-filme no dia 16 de janeiro de 2004). Vale lembrar que a ferramenta aqui desenvolvida é gratuita e possui um grande número de funções que não são presentes nas ferramentas similares, onde muitas vezes elas terão que ser comprada separadamente a preços semelhantes.
A combinação aqui apresentada de ferramentas de auxilio ao diagnostico permite ao médico tomar uma decisão mais rápida e acertada referente ao diagnostico e por seqüente laudo do paciente.

Projeto Cyclops

Nos últimos dez anos, foram desenvolvidas dentro do Projeto Cyclops muitas ferramentas que fornecem ao médico auxilio ao diagnostico. Mas um dos grandes problemas desses aplicativos, do ponto de vista de um cientista da computação, foi a maneira como foram construídos, na maioria das vezes, por pessoas com pouco ou nenhum conhecimento de programação orientada a objetos, em especial da linguagem smalltalk, e a não utilização de metodologias da engenharia de software.
Esses softwares foram construídos durante projetos de mestrado e após a conclusão dos mesmos, as pessoas responsáveis por eles deixavam o projeto, e deixavam sem documentação as ferramentas que elas construíram. Como fazer então para integrar um conjunto de ferramentas cujo código não contem documentação e é muito difícil de compreender em um curto período de tempo, numa única ferramenta, com mínimas alterações na ferramenta original e que venha a permitir novas expansões no futuro, sem que ela própria seja modificada radicalmente?

Deste problema, surgiu-se a idéia da utilização de plug-ins, que poderiam ser definidos como trechos de código altamente modularizados (componentes), que poderiam ser encaixados para forma uma novo aplicativo e que localizariam e minimizariam as regiões onde poderiam ocorrer modificações.

Componentes?

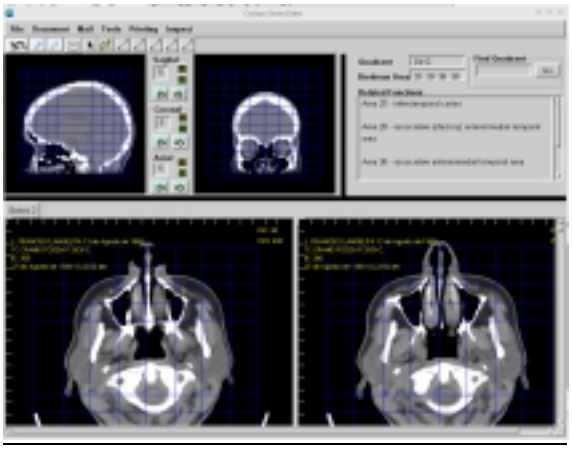
Analisando as ferramentas desenvolvidas no projeto, foram detectados que existem dois tipos de plug-ins:
•Tipo altamente acoplado a imagem: Este tipo de plug-in é geralmente derivado da antiga Classe DICOMSeriesEditor , ou seja, embora o objetivos de vários desses plug-ins fosse completamente diferente, todos eles contem um núcleo em comum, onde a grande diferença se localiza em algum código acrescentado a subclasse e a um novo MVC.este tipo de plugin provavelmente implicará pouca alteração na aplicação original, apenas as mudança de algum métodos de acesso ao MVC da classe CyclopsImageViewer. As Classes CyclopsSEview, CyclopsSEmodel E CyclopsSEContoller provavelmente precisarão ser modificadas, adicionando-se a ela o código que será necessário a exibição da informação requerida pelo plug-in. Uma das coisas que foram pensadas durante o desenvolvimento deste trabalho, foi a utilização de diversos MVC, um para cada plug-in altamente acoplado. O problema dessa abordagem é que ela minimizaria a utilização em conjunto dos plug-ins, e é exatamente a idéia contrária a essa que motivou este trabalho. Assim, você terá que acrescentar algum código e algumas variáveis nessas classes, de acordo com o tipo de aplicação que você desejar.

•Tipo de baixo acoplamento Este tipo de plug-in, ao contrário do anterior, não é uma derivação da classe DicomSeriesEditor, sendo provavelmente feito do zero como uma nova aplicação. Este tipo de plug-in requer pouquíssima alteração de código, porque o seu funcionamento independe de um sistema de visualização de imagens, embora posso exibir outros tipos de sinais médicos, como uma waveform. Geralmente, seu funcionamento difere de tal forma do primeiro tipo, que até mesmo o local onde sua interface será exibida será diferente; ele será exibo em uma nova tabua ao invés da região escondida denomina barra de ferramentas.

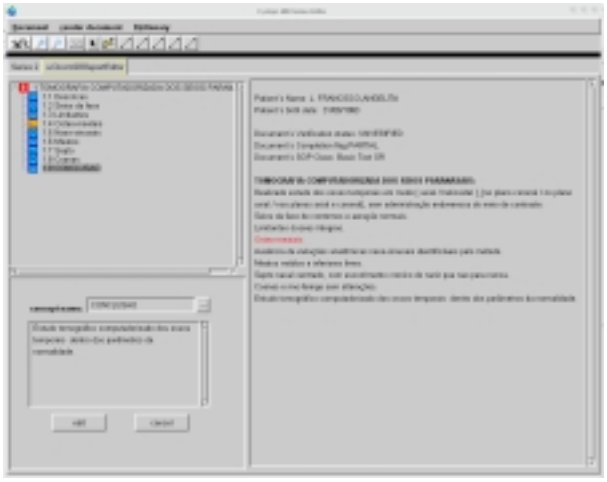
As Ferramentas Do Cyclops Series

Ficou definido que o software desenvolvido durante este trabalho de conclusão de curso deveria integrar os seguintes aplicativos anteriormente desenvolvidos no Projeto Cyclops:
•Cyclops Brain Atlas, desenvolvido durante o mestrado do professor Harley Wagner.
•Cyclops Waveform Viewer, desenvolvido durante o mestrado do professor Gilson Araújo.
•Cyclops PopCorn, desenvolvido durante o mestrado do Eros Comunello.
•Cyclops Stroke, ainda em desenvolvimento.
•Cyclops Structured Report Editor e Viewer, desenvolvido durante o mestrado de Mariana K. Bortoluzzi.

O CyclopsBrainAtlas é um Atlas deformável digital do cérebro humano baseado no modelo do atlas de Talairach, que permite a identificação dos quadrantes de Talairach e automaticamente mostra as áreas de Brodmann relacionadas a cada quadrante. Isto permite a identificação e a localização semi-automática das áreas funcionais e anatômicas do cérebro afetadas por alguma patologia visível como, por exemplo, um cisticerco. O ajuste é executado com a adaptação, posicionamento e rotação interativas do volume inteiro do Atlas, que é mostrado na relação de usuário através das interseções planas com as fatias tomográficas.



O CyclopsSRReport Editor e um editor de documentos de interface amigável, capaz de produzir documentos de acordo com o padrão DICOM SR. Devido a necessidade de fácil acesso a essa ferramenta, ela foi integrada diretamente dentro do Cyclops Series Editor. Uma das grandes facilidades que ela apresenta é a capacidade de ler modelos (Templates) de laudos, com parte do texto já preenchida. Isto é importante porque grande parte dos laudos é extremamente repetitiva, constituindo-se basicamente de reproduções de laudos anteriores, e mesmo o laudo inicial muitas vezes necessita de apenas pequenas modificações estruturais e de textuais.



O Cyclops Stroke Quantifier foi criado para fornecer suporte a decisão na terapia do AVC isquêmico calculando e representando graficamente o fluxo sangüíneo no cérebro com base em tomográfica computadorizada dinâmica (cálculo da variação local relativa de densidade radiológica). Para isto foi realizado o desenvolvimento de um software para auxiliar ao radiologista e aos médicos generalistas a identificarem precocemente o AVC isquêmico para proceder com o tratamento adequado além de permitir que façamos a correlação entre a área atingida com o déficit das funções atribuídas a esta área utilizando–se do Atlas cerebral. E calcular o tamanho da área resultante do AVC utilizando – se de uma ferramenta para tal.

O Cyclops Popcorn é uma ferramenta de diagnósticos que utiliza métodos de visão computacional para a identificação, contagem, mensuração e localização de calcificações relacionadas a esse tipo de infecção em imagens de tomografia computadorizada do crânio.

Além dessas ferramentas, o Cyclops Series Editor também possui o Cyclops Waveform, que é um visualizador de ondas obtidas através de aparelhos de eletroencefalograma ou eletrocardiograma que permite uma analise semi-automática da onda auxiliando o médico na detecção de problemas cardíacos ou cerebrais, e uma ferramenta de e-mail que permite o envio das imagens através do e-mail.

Conclusão

O real objetivo por trás deste protótipo é a união da maioria das ferramentas elaboradas no projeto cyclops, visando uma maior utilidade de uma ferramenta médica e uma maior aceitação por parte dos usuários médicos de ferramentas computacionais para auxilio ao diagnostico.
Acredita-se que o objetivo deste trabalho tenha sido alcançado, pois a ferramenta apresentada aqui apresenta grande facilidade na utilização eu um desempenho aceitável para uma software desenvolvido em uma linguagem interpretada.

Referências

Abdala,Daniel Duarte. Cyclops Personal – **Uma Ferramenta para Gerenciamento e Visualização de Imagens Médicas no Padrão DICOM 3.0**. Florianópolis, 5 de julho de 2002. 217 pág. Trabalho de Conclusão de Curso em Ciência da Computação - Universidade Federal de Santa Catarina - INE, 2002.

Dellani, Paulo Roberto . **Desenvolvimento De Um Servidor De Imagens Médicas Digitais No Padrão Dicom**. Florianópolis, fevereiro de 2001. 98 pág. Dissertação (Mestrado em Ciência da Computação) - Programa de Pós-Graduação em Ciência da Computação - PPGCC, Universidade Federal de Santa Catarina - INE, 2001.

Bortoluzzi, Mariana Kessler. **Desenvolvimento e Implementação De Um Editor De Documentos Estruturados No Padrâo Dicom Structured Report**. Florianópolis, Março de 2003. 98 pag. Dissertação (Mestrado em Ciencias da Computação) - Programa de Pós-Graduação em Ciencias da Computação - PPGCC,Universidade Federal de Santa Catarina – INE , 2003.

WANGENHEIM, Aldo von, **Página da disciplina de Visão Computacional**. Disponível em: <http://www.inf.ufsc.br/~visao/>. Acesso em: 15 de jul. 2003.

Berti, L. A. **Implementação de um protótipo de software para análise, mensuração e Reconstrução de aneurismas de artéria aorta abdominal**. 2003. Trabalho de Conclusão de Curso de Engenharia da Computação - Universidade do Vale do Itajaí - Unisul, Santa Catarina

PRATT,WILLIAM K.. **Digital Image Processing**. Addisson Wesley .2001

BECK, Kent. Smalltalk: Best Practice Patterns. New Jersey: Prentice Hall, 1997. 224 p.

Abdala, Daniel Duarte;Wangenheim, Aldo von. **Conhecendo Smalltalk. Florianópolis**: Editora Visual Books, 2002. V. 1. 300 pág.

CLUNIE, David A. Medical Image Format FAQ. **Questões freqüentemente perguntadas sobre Formatos de Imagens Médicas, com enfoque para o padrão DICOM**. Disponível em: <http://www.dclunie.com/medical-image-faq/html/>. Acesso em: 10 de novembro de 2003.

EFILM MEDICAL INC. **eFilm Workstation 1.5.3 Software** . Disponível em: <http://www.efilm.ca/ >. Acesso em: 05 de fevereiro de 2001.

NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION. **Digital Imaging and Communications in Medicine (DICOM) – Todas as partes.** Virginia, 2000. Disponível em: <ftp://medical.nema.org/medical/dicom/2000/draft/ >. Acesso em: 10 de novembro de 2003.

Wagner,Harley Miguel . **ATLAS CEREBRAL DIGITAL: DESENVOLVIMENTO DE UMA FERRAMENTA COMPUTACIONAL PARA MAPEAMENTO FUNCIONAL E ANATÔMICO DE ÁREAS CEREBRAIS, BASEADO NO ATLAS DE TALAIRACH**. Florianópolis, Março de 2001, 88 pag. Dissertação (Mestrado em Ciencias da Computação) - Programa de Pós-Graduação em Ciencias da Computação - PPGCC,Universidade Federal de Santa Catarina – INE , 2001.

Anexo 5 – Código Fonte

'From VisualWorks® NonCommercial, Release 7 of 14 de Junho de 2002 on 9 de Fevereiro de 2004 at 7:14:23 pm'!

```
!CyclopsSeriesEditor methodsFor: 'initialize'!

initialize
"***This is decompiled code.***


self initializeEditorWindowLevel.
self pageHolder list: (List with: Blank new).
self shrunkFactor: OrderedCollection new.
self coPilotView: ((CyclopsWindowToolView new model: ((CyclopsWindowTool
newOn: self)
name: 'a' printString asSymbol))
owner: self).
self pilotView: ((CyclopsWindowToolView new model: ((CyclopsWindowTool
newOn: self)
name: 'a' printString asSymbol))
owner: self).
self coPilotView bigOne: #coronal.
self pilotView bigOne: #sagital.
(format := 'Dicom Part 10' asValue) onChangeSend: #changedFormat to: self.
self filenameForExporting: 'image.dcm'.
self currentPosition: ValueHolder new.
self pilot: OrderedCollection new.
self coPilot: OrderedCollection new.
^self!

initializeEditorWindowLevel


self windowW value: nil.
self windowC value: nil.!

mount
"this class initialize the paging system, creating a page for each Image
Series or SR"

| srViewer seriesViewer n imagesToRender renderPercentDone ecgViewer |
n := 0.
self seriesViewersList: OrderedCollection new."This code open a
imageLoader"
imagesToRender := 0.
self cyclopsSeries do:
[:aSeries |
aSeries modality = 'SR'
ifFalse: [imagesToRender := imagesToRender + aSeries images size]].
imagesToRender > 0
ifTrue:
[renderPercentDone := CyclopsMIBImageRendering new.
renderPercentDone filesToLoad: imagesToRender.
renderPercentDone open].
self cyclopsSeries do:
[:aSeries |
aSeries modality = 'SR'
```

```
ifTrue:
[aSeries images asOrderedCollection do:
[:aSR |
"Create a SRViewer and add a page for it"

srViewer := DicomSRReportViewer newOn: aSR.
self addAndSelectPage: srViewer."Add the page"
srViewer open2]].
((aSeries modality = 'ECG') | (aSeries modality = 'HD'))ifTrue:
[
aSeries images asOrderedCollection do:
[:aSR |
self halt.
ecgViewer := CyclopsWaveformViewer newOn: aSR.
self addAndSelectPage: ecgViewer.
ecgViewer open2]]
ifFalse:
["Create a SeriesViewer and add a page for it"

self shrunkFactor add: 1.0 asValue."Add a zoom factor for each
SeriesViewer"
seriesViewer := SeriesViewer newOn: aSeries.
"seriesViewer := BrainAtlasInterface newOn: aSeries."
seriesViewer seriesEditor: self.
aSeries images do:
[:aCyclopsImage |
seriesViewer initializeReferencedPalettesForOneImage: aCyclopsImage.
aCyclopsImage mrCachedImage asRetainedMedium.
renderPercentDone renderNumberIncrement].
"initialize the zoom factor and set the toolFlag. The tool Flag must be the
same to
all of them"
seriesViewer
shrunkFactor: self shrunkFactor last;
toolFlag: self toolFlag.
self seriesViewersList add: seriesViewer.
self addAndSelectPage: seriesViewer.
n = 0
ifTrue:
["Add the page"

"Don't know what this do"

n := 1.
self lastSeries: self pageHolder listHolder value size - 1]]]."Remove the
first page, who is a blank page"
self removePage: (self pages at: 1).
self pageHolder selectionIndex: 1.
self seriesViewersList notEmpty
ifTrue:
["Set the page one to be the first"

self createPilots]
ifFalse: [].
imagesToRender > 0 ifTrue: [renderPercentDone closeRequest]!

postBuildWith: aBuilder
| atlas |
super postBuildWith: aBuilder.
self pageHolder selectionIndexHolder onChangeSend: #pageChanged to: self.
self taskBarSelector onChangeSend: #changedTaskBar to: self.
```

```
self taskBarSelector2 onChangeSend: #changedTaskBar2 to: self.
(self widgetAt: #ResizingSplitter2) moveBy: -600 @ 0.
(self widgetAt: #ResizingSplitter1) moveBy: 0 @ -256.

"xunxo do atlas"

"atlas:=self pageHolder selection.

atlas sortedImages: atlas cyclopsSeries images asSortedCollection.
atlas postBuildWith: aBuilder ."!

postOpenWith: aBuilder

| temp |
super postOpenWith: aBuilder.
self pageHolder selection: self pageHolder list first.
temp:=self pageHolder selection class printString.
temp ~= 'SeriesViewer' ifTrue:[
self unistallCurrentMenu.
temp = 'DicomSRReportViewer' ifTrue:[
self updateMenuToSRviewer.
].

].

"self pageHolder selection  placeImagesOnLightscreen: self pageHolder
selection builder."! !

!CyclopsSeriesEditor methodsFor: 'accessing'!

aCyclopsStudy
^aCyclopsStudy!

aCyclopsStudy: anObject
aCyclopsStudy := anObject!

coPilot
^coPilot!

coPilot: anObject
coPilot := anObject!

coPilotView
^coPilotView!

coPilotView: anObject
coPilotView := anObject!

currentImage
^currentImage!

currentImage: anObject
currentImage := anObject!

currentPosition
^currentPosition!

currentPosition: anObject
currentPosition := anObject!

cyclopsMIB
```

```
^cyclopsMIB!

cyclopsMIB: anObject
cyclopsMIB := anObject!

cyclopsSeries
^cyclopsSeries!

cyclopsSeries: anObject
cyclopsSeries := anObject!

dicomEditor
^cyclopsMIB!

dicomEditor: anObject
cyclopsMIB := anObject!

directoryForExporting
^directoryForExporting!

filenameForExporting
^filenameForExporting!

filenameForExporting: anObject
filenameForExporting := anObject!

formatTypes: anObject
formatTypes := anObject!

lasrSRReportEditor
^lasrSRReportEditor!

lasrSRReportEditor: anObject
lasrSRReportEditor := anObject!

lastInstalledPage
^lastInstalledPage!

lastInstalledPage: anObject
lastInstalledPage := anObject!

lastSeries
^lastSeries!

lastSeries: anObject
lastSeries := anObject!

pilot
^pilot!

pilot: anObject
pilot := anObject!

pilotView
^pilotView!

pilotView: anObject
pilotView := anObject!

seriesViewersList
^seriesViewersList!
```

```
seriesViewersList: anObject
seriesViewersList := anObject!

shrunkFactor
^shrunkFactor!

shrunkFactor: anObject
shrunkFactor := anObject!

taskBar
^taskBar!

taskBar: anObject
taskBar := anObject!

text
^text!

text: anObject
text := anObject!

windowC
"This method was generated by UIDefiner.  Any edits made here
may be lost whenever methods are automatically defined.  The
initialization provided below may have been preempted by an
initialize method."

^windowC isNil
ifTrue:
[windowC := 0 asValue]
ifFalse:
[windowC]!

windowC: anObject
windowC := anObject!

windowR
^windowR!

windowR: anObject
windowR := anObject!

windowW
"This method was generated by UIDefiner.  Any edits made here
may be lost whenever methods are automatically defined.  The
initialization provided below may have been preempted by an
initialize method."

^windowW isNil
ifTrue:
[windowW := 0 asValue]
ifFalse:
[windowW]!

windowW: anObject
windowW := anObject! !

!CyclopsSeriesEditor methodsFor: 'actions'!

changePilots
```

```
"This stub method was generated by UIDefiner"

^self!

exportImageToFile

| result dicomSeriesEditorChooser markedImages |
self halt.
(self markedImages size = 0)
ifTrue:
[DicomDialog warn: 'No image is marked.']
ifFalse:
[(markedImages size > 1)
ifTrue:
[dicomSeriesEditorChooser := DicomSeriesEditorChooser
newOn: markedImages
forPurpose:'exportImage'.
dicomSeriesEditorChooser dicomSeriesEditor: self.
dicomSeriesEditorChooser open.]
ifFalse:
["fileBrowser := DicomFileBrowserSaving new.
fileBrowser model: self dicomEditor.
result := fileBrowser openInterface: #windowSpec."
result := FileSelectionDialog allowDirectoryResult:  true allowFileResult:
false.
(result isNil)
ifFalse:
[(result asFilename definitelyExists)
ifTrue:
[self directoryForExporting: result.
self exportImage]
ifFalse:
[(DicomDialog confirm: result, ' does not exist or is not accessible. Try
again?')
ifTrue:
[self exportImageToFile]]]]].!

exportMarkedImagesToFiles

| result markedImages |
markedImages := self markedImages.

(markedImages size = 0)
ifTrue:
[DicomDialog warn: 'No image is marked.']
ifFalse:
[
result := FileSelectionDialog allowDirectoryResult:  true allowFileResult:
false.
(result isNil)

ifFalse:
[(result asFilename definitelyExists)
ifTrue:
[self directoryForExporting: result.
self exportMarkedImagesAsSeries]
ifFalse:
[(DicomDialog confirm: result, ' does not exist or is not accessible. Try
again?')
ifTrue:
```

```
[self exportMarkedImagesToFiles]]]].!

exportSeriesToFiles

| result |

result := FileSelectionDialog allowDirectoryResult:  true allowFileResult:
false.
(result isNil) ifFalse: [
(result asFilename definitelyExists) ifTrue: [
self directoryForExporting: result.
self exportSeries
]
ifFalse: [
(DicomDialog confirm: result, ' does not exist or is not accessible. Try
again?') ifTrue: [
self exportSeriesToFiles
]
]
].!

rotateLeft

"self child removeAllSuchThat: [:elem | elem = elem]."
"(((self builder namedComponents at: #lichtbrettSubcanvas) widget
components at: 1) component widget components at: 3) component model
removeAllLines."
"(((self builder namedComponents at: #lichtbrettSubcanvas) widget
components at: 1) component widget components at: 3) component model
removeChildrenLines."
self taskBar cubo children: OrderedCollection new.
self taskBar cubo genericRotate: self degree value in: #y.
self taskBar  drawCubo.
self taskBar  calcPropagPointsAxial!

rotateLeftCoronal

| temp |
"self child removeAllSuchThat: [:elem | elem = elem]."
self taskBar cubo children: OrderedCollection new.
temp := self  degree  value negated.
self taskBar cubo genericRotate: temp in: #z.
self taskBar  drawCubo.
self taskBar  calcPropagPointsAxial!

rotateLeftSagital

"self child removeAllSuchThat: [:elem | elem = elem]."
self taskBar cubo children: OrderedCollection new.
self taskBar cubo genericRotate: self  degree value  in: #x.
self taskBar  drawCubo.
self taskBar calcPropagPointsAxial!

rotateRight

| temp |
"self child removeAllSuchThat: [:elem | elem = elem]."
self taskBar cubo children: OrderedCollection new.
temp := self degree value negated.
self taskBar cubo genericRotate: temp in: #y.
self taskBar drawCubo.
```

```
self taskBar  calcPropagPointsAxial!

rotateRightCoronal

"self child removeAllSuchThat: [:elem | elem = elem]."
self taskBar cubo children: OrderedCollection new.
self taskBar cubo genericRotate: self degree value in: #z.
self taskBar drawCubo.
self taskBar  calcPropagPointsAxial!

rotateRightSagital

| temp |
"self child removeAllSuchThat: [:elem | elem = elem]."
self taskBar cubo children: OrderedCollection new.
temp := self degree value negated.
self taskBar cubo genericRotate: temp in: #x.
self taskBar  drawCubo.
self taskBar  calcPropagPointsAxial! !

!CyclopsSeriesEditor methodsFor: 'aspects'!

bigOne
"This method was generated by UIDefiner.  Any edits made here
may be lost whenever methods are automatically defined.  The
initialization provided below may have been preempted by an
initialize method."

^bigOne isNil
ifTrue:
[bigOne:=#coronal asValue]
ifFalse:
[bigOne].!

degree
"This method was generated by UIDefiner.  Any edits made here
may be lost whenever methods are automatically defined.  The
initialization provided below may have been preempted by an
initialize method."

^degree isNil
ifTrue:
[degree := 5 asValue]
ifFalse:
[degree]!

embeddedApplication

^nil asValue!

format
"This method was generated by UIDefiner.  Any edits made here
may be lost whenever methods are automatically defined.  The
initialization provided below may have been preempted by an
initialize method."

^format isNil
ifTrue:
[format := String new asValue]
ifFalse:
[format]!
```

```
formatTypes
|list|

list := List new.
list add: 'Dicom Part 10';
add: 'Colored Image Sun Raster File';
add: 'Image PGM';
add: 'Image PNM Binary';
add: 'Image PNM Ascii';
add: 'Image RAW';
add: 'Image Sun Raster File';
add: 'Image VIFF';
add: 'Image Vista'.

^list asValue.!

menuSelector
"This method was generated by UIDefiner.  Any edits made here
may be lost whenever methods are automatically defined.  The
initialization provided below may have been preempted by an
initialize method."

^menuSelector isNil
ifTrue:
[menuSelector := nil asValue]
ifFalse:
[menuSelector]!

pageHolder
"This method was generated by UIDefiner.  Any edits made here
may be lost whenever methods are automatically defined.  The
initialization provided below may have been preempted by an
initialize method."

^pageHolder isNil
ifTrue:
[pageHolder := SelectionInList new]
ifFalse:
[pageHolder]!

taskBarSelector
"This method was generated by UIDefiner.  Any edits made here
may be lost whenever methods are automatically defined.  The
initialization provided below may have been preempted by an
initialize method."

^taskBarSelector isNil
ifTrue:
[taskBarSelector := nil asValue]
ifFalse:
[taskBarSelector]!

taskBarSelector2
"This method was generated by UIDefiner.  Any edits made here
may be lost whenever methods are automatically defined.  The
initialization provided below may have been preempted by an
initialize method."

^taskBarSelector2 isNil
ifTrue:
```

```
[taskBarSelector2 := nil asValue]
ifFalse:
[taskBarSelector2]!

toolFlag
"This method was generated by UIDefiner.  Any edits made here
may be lost whenever methods are automatically defined.  The
initialization provided below may have been preempted by an
initialize method."

^toolFlag isNil
ifTrue:
[toolFlag := #marker asValue]
ifFalse:
[toolFlag]! !

!CyclopsSeriesEditor methodsFor: 'events'!

changeRequest
"On change, do someting very important , but i don't know what"

| unsaved choice |
unsaved := self pages reject: [:some | some isAccepted].
unsaved isEmpty ifTrue: [^true].
choice := self suggestSaving: unsaved.
choice = #no ifTrue: [^true].
choice = #cancel ifTrue: [^false].
"The choice is #yes, i.e. save all"
unsaved do: [:each | each saveIntoFile ifFalse: [^false]].
^true!

pageChanged
"When anothe page is selected, we need to change the current application to
the application on
the new page, always checking if the current menus is the correct one "

| selection currentPageClass numb |
selection := self pageHolder selection.
lastInstalledPage notNil
ifTrue:
[lastInstalledPage deactivate.
lastInstalledPage class printString = 'SeriesViewer'
ifTrue:
["self halt."

numb := self pageHolder listHolder value indexOf: lastInstalledPage
ifAbsent: [].
numb notNil ifTrue: [self lastSeries: numb]]
ifFalse:
[lastInstalledPage class printString = 'DicomSRReportEditor'
ifTrue:
["self halt."

numb := self pageHolder listHolder value indexOf: lastInstalledPage
ifAbsent: [].
numb notNil ifTrue: [self lasrSRReportEditor:
(self pageHolder listHolder value at: numb)]].
lastInstalledPage := nil]].
selection isNil ifTrue: [^self].
self installPage: selection.
self windowEvent: #open from: self builder window.
```

```
currentPageClass := self pageHolder selection class printString.
currentPageClass = 'SeriesViewer'
ifTrue:
[self unistallCurrentMenu.
self updateSeriesEditorMenu.
numb := self pageHolder listHolder value indexOf: selection ifAbsent: [].
numb notNil ifTrue: [self lastSeries: numb].
"self mainWindow display."
self pilotView redisplay.
self coPilotView redisplay.
self showWindowTaskBar]
ifFalse:
[currentPageClass = 'DicomSRReportViewer'
ifTrue: [self unistallCurrentMenu.
    self updateMenuToSRviewer.

]
ifFalse:
[currentPageClass = 'DicomSRReportEditor'
ifTrue:
[self unistallCurrentMenu.
self updateMenuToSReditor.
self pageHolder selection treeModel refreshRoot.
"self mainWindow display"]]]!

pageClose

self canClosePage ifFalse: [self error: 'cannot close this page'].
self currentTextPage changeRequest ifFalse: [^self].
self removePage: self currentTextPage!

updatePage
"Page-related things such as labels may have changed for the current page."

"super updatePage."
self pages changed: #at: with: self pageHolder selectionIndex!

windowEvent: anEvent from: aWindow

self myDependents
update: #windowState
with: anEvent
from: aWindow.
^self! !

!CyclopsSeriesEditor methodsFor: 'export'!

addImageNumberAtFilename: imageNumber

| filename test index lastIndex newFilename |
filename := filenameForExporting.

"Search for the last '.' in filename"
test := 1.
[test = 0]
whileFalse:
[index := test+1.
lastIndex := test.
test := filename findString: '.' startingAt: index.].

newFilename := String new.
```

```
newFilename := filename copyFrom: 1 to: (lastIndex-1).
newFilename := newFilename, imageNumber asNumber printString.
newFilename := newFilename, (filename copyFrom: (lastIndex) to: (filename
size)).

^newFilename.!

changedFormat
| currentFormat |
currentFormat := format value.

(currentFormat = 'Dicom Part 10') ifTrue: [self changeExtIn: 'dcm'].
(currentFormat = 'Colored Image Sun Raster File') ifTrue: [self
changeExtIn: 'ras'].
(currentFormat = 'Image PGM') ifTrue: [self changeExtIn: 'pgm'].
(currentFormat = 'Image PNM Binary') ifTrue: [self changeExtIn: 'pnm'].
(currentFormat = 'Image PNM Ascii') ifTrue: [self changeExtIn: 'pnm'].
(currentFormat = 'Image RAW') ifTrue: [self changeExtIn: 'raw'].
(currentFormat = 'Image Sun Raster File') ifTrue: [self changeExtIn:
'ras'].
(currentFormat = 'Image VIFF') ifTrue: [self changeExtIn: 'vif'].
(currentFormat = 'Image Vista') ifTrue: [self changeExtIn: 'vis'].!

changeExtIn: anExtension

| filename test index lastIndex newFilename |
filename := filenameForExporting.

"Search for the last '.' in filename"
test := 1.
[test = 0]
whileFalse:
[index := test+1.
lastIndex := test.
test := filename findString: '.' startingAt: index.].

newFilename := String new.
newFilename := filename copyFrom: 1 to: lastIndex.
newFilename := newFilename, anExtension.

self filenameForExporting: newFilename.!

directoryForExporting: aString

| sep |
sep := ByteString with: Filename separator.
(aString last = sep)
ifTrue:
[directoryForExporting := aString]
ifFalse:
[directoryForExporting := aString,sep]!

exportImage
|currentFormat|
self halt.
Cursor bull showWhile:
[currentFormat := format value.

(currentFormat = 'Dicom Part 10')ifTrue: [self exportDicomImage].
(currentFormat = 'Colored Image Sun Raster File')ifTrue: [self
exportColoredImageSunRasterFile].
```

```
(currentFormat = 'Image PGM') ifTrue: [self exportImagePGM].
(currentFormat = 'Image PNM Binary') ifTrue: [self exportImagePNM].
(currentFormat = 'Image PNM Ascii')ifTrue: [self exportImagePNMAscii].
(currentFormat = 'Image RAW') ifTrue: [self exportImageRaw].
(currentFormat = 'Image Sun Raster File') ifTrue: [self
exportImageSunRasterFile].
(currentFormat = 'Image VIFF') ifTrue: [self exportImageVIFF].
(currentFormat = 'Image Vista') ifTrue: [self exportImageVista].].!

exportMarkedImagesAsSeries
|currentFormat filename min max extremum sep markedImages |

markedImages := self markedImages.


Cursor bull showWhile:
[currentFormat := format value.

(currentFormat = 'Dicom Part 10')
ifTrue:
[sep := ByteString with: Filename separator.
(self directoryForExporting, sep, (cyclopsSeries study
studyInstanceUID))asFilename makeDirectory.
self directoryForExporting: (self directoryForExporting, sep,
(cyclopsSeries study studyInstanceUID), sep, (cyclopsSeries
seriesInstanceUID)).
self directoryForExporting asFilename makeDirectory.
markedImages do: [:image|
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportDicomImage: filename]].

(currentFormat = 'Colored Image Sun Raster File')
ifTrue:
[markedImages do: [:image|
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportColoredImageSunRasterFile: filename]].

(currentFormat = 'Image PGM')
ifTrue:
[markedImages do: [:image|
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportImagePGM: filename]].

(currentFormat = 'Image PNM Binary')
ifTrue:
[min := 255. max :=0.
markedImages do: [:image|
extremum := image mrCachedImage calcExtrema.
min > (extremum at:#min) ifTrue:[min := (extremum at:#min).].
max <(extremum at:#max) ifTrue:[max := (extremum at:#max).].].
markedImages do: [:image|
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportImagePNM: filename min: min max: max]].

(currentFormat = 'Image PNM Ascii')
ifTrue:
[markedImages do: [:image|
```

```
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportImagePNMAscii: filename]].

(currentFormat = 'Image RAW')
ifTrue:
[markedImages do: [:image|
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportImageRaw: filename]].

(currentFormat = 'Image Sun Raster File')
ifTrue:
[markedImages do: [:image|
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportImageSunRasterFile: filename]].

(currentFormat = 'Image VIFF')
ifTrue:
[markedImages do: [:image|
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportImageVIFF: filename]].

(currentFormat = 'Image Vista')
ifTrue:
[markedImages do: [:image|
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportImageVista: filename]].

].!

exportSeries
|currentFormat filename min max extremum sep study |

cyclopsSeries:=self pageHolder selection cyclopsSeries.
Cursor bull showWhile:
[currentFormat := format value.

(currentFormat = 'Dicom Part 10')
ifTrue:
[sep := ByteString with: Filename separator.
(self directoryForExporting, sep, (cyclopsSeries study
studyInstanceUID))asFilename makeDirectory.
self directoryForExporting: (self directoryForExporting, sep,
(cyclopsSeries study studyInstanceUID), sep, (cyclopsSeries
seriesInstanceUID)).
self directoryForExporting asFilename makeDirectory.
cyclopsSeries images do: [:image|
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportDicomImage: filename]].

(currentFormat = 'Colored Image Sun Raster File')
ifTrue:
[cyclopsSeries images do: [:image|
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportColoredImageSunRasterFile: filename]].
```

```
(currentFormat = 'Image PGM')
ifTrue:
[cyclopsSeries images do: [:image|
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportImagePGM: filename]].

(currentFormat = 'Image PNM Binary')
ifTrue:
[min := 255. max :=0.
cyclopsSeries images do: [:image|
extremum := image mrCachedImage calcExtrema.
min > (extremum at:#min) ifTrue:[min := (extremum at:#min).].
max <(extremum at:#max) ifTrue:[max := (extremum at:#max).].].
cyclopsSeries images do: [:image|
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportImagePNM: filename min: min max: max]].

(currentFormat = 'Image PNM Ascii')
ifTrue:
[cyclopsSeries images do: [:image|
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportImagePNMAscii: filename]].

(currentFormat = 'Image RAW')
ifTrue:
[cyclopsSeries images do: [:image|
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportImageRaw: filename]].

(currentFormat = 'Image Sun Raster File')
ifTrue:
[cyclopsSeries images do: [:image|
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportImageSunRasterFile: filename]].

(currentFormat = 'Image VIFF')
ifTrue:
[cyclopsSeries images do: [:image|
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportImageVIFF: filename]].

(currentFormat = 'Image Vista')
ifTrue:
[cyclopsSeries images do: [:image|
filename := self addImageNumberAtFilename: image imageNumber.
self currentImage: image.
self exportImageVista: filename]].

].!

launchExportFormatWindow

|bldr|
```

```
bldr := UIBuilder new.
bldr source: self.
bldr  add: (self class interfaceSpecFor: #exportFormatSpec).
bldr openAt: (self
originFor: bldr window
nextTo: #pilot).!

markedImages

|temp|
temp := OrderedCollection new.
self  pageHolder selection imageVieweList do: [: aImageViewer |
(aImageViewer markedImage value) ifTrue: [
temp add: aImageViewer cyclopsImage .
]

].
 ^temp.!

originFor: aWindow nextTo: aWidgetID
"Answer the point at which a subwindow such as the
history window (aWindow) should be opened so that it is
positioned beside the invoking button (aWidgetID)."
| gutter windowBorder |
gutter := 10 @ 0.
windowBorder := 0 @ 21."Nonportable."
^self builder window getDisplayBox origin
- gutter
+ windowBorder
- (aWindow width @ 0)
+ (0 @ (self builder componentAt: aWidgetID) bounds origin y).! !

!CyclopsSeriesEditor methodsFor: 'export formats'!

exportColoredImageSunRasterFile
currentImage mrCachedImage exportColoredImageSunRasterFile:
(directoryForExporting, filenameForExporting)asString.!

exportColoredImageSunRasterFile: filename
currentImage mrCachedImage exportColoredImageSunRasterFile:
(directoryForExporting, filename)asString.!

exportDicomImage

| aDCMFile |
aDCMFile := CyclopsDicomFile dicomIOD: self currentImage dicomIOD.
aDCMFile writeTo: (directoryForExporting, filenameForExporting) asString
asFilename.!

exportDicomImage: filename

| aDCMFile |
aDCMFile := CyclopsDicomFile dicomIOD: self currentImage dicomIOD.
aDCMFile writeTo: (directoryForExporting, filename) asString asFilename!

exportImagePGM
  | image |
image:= currentImage mrCachedImage.
(image image depth = 8)
ifTrue:[image exportImagePGM: (directoryForExporting, filenameForExporting)
asString.]
```

```
ifFalse:[(image convert16To8Bit) exportImagePGM: (directoryForExporting,
filenameForExporting) asString.].!

exportImagePGM: filename
currentImage mrCachedImage exportImagePGM: (directoryForExporting,
filename)asString.!

exportImagePNM
| image |
image:= currentImage mrCachedImage.
(image image depth = 8)
ifTrue:[image exportImagePNM: (directoryForExporting, filenameForExporting)
asString.]
ifFalse:[(image convert16To8Bit) exportImagePNM: (directoryForExporting,
filenameForExporting) asString.].!

exportImagePNM: filename
currentImage mrCachedImage exportImagePNM: (directoryForExporting,
filename)asString.!

exportImagePNM: filename min: min max: max
[currentImage mrCachedImage exportImagePNM: (directoryForExporting,
filename)asString min:min max:max.] on: Exception do:[: msg| Dialog warn:
'Houve um erro ao exportar!!' ].!

exportImagePNMAscii
currentImage mrCachedImage exportImagePNMAscii: (directoryForExporting,
filenameForExporting)asString.!

exportImagePNMAscii: filename
currentImage mrCachedImage exportImagePNMAscii: (directoryForExporting,
filename)asString.!

exportImageRaw
currentImage mrCachedImage exportImageRaw: (directoryForExporting,
filenameForExporting)asString.!

exportImageRaw: filename
currentImage mrCachedImage exportImageRaw: (directoryForExporting,
filename)asString.!

exportImageSunRasterFile
currentImage mrCachedImage exportImageSunRasterFile:
(directoryForExporting, filenameForExporting)asString.!

exportImageSunRasterFile: filename
currentImage mrCachedImage exportImageSunRasterFile:
(directoryForExporting, filename)asString.!

exportImageVIFF
currentImage mrCachedImage exportImageVIFF: (directoryForExporting,
filenameForExporting)asString.!

exportImageVIFF: filename
currentImage mrCachedImage exportImageVIFF: (directoryForExporting,
filename)asString.!

exportImageVista
currentImage mrCachedImage exportImageVista: (directoryForExporting,
filenameForExporting)asString.!
```

```
exportImageVista: filename
currentImage mrCachedImage exportImageVista: (directoryForExporting,
filename)asString.! !

!CyclopsSeriesEditor methodsFor: 'html'!

generateHTMLPrintFile
"Export the Images with the selected radiological window to the working
directory..."

"Export the images to the directory"

| g htmlText |
htmlText := self openSRselector value asString.
g := CyclopsHtmlForPrint new.
g viewsList: self pageHolder selection imageVieweList.
g generateHTMLPrintFile: htmlText.!

generateHTMLPrintFileOnMarkedAllSeries

"Create a html file with jpeg images of all CyclopsImages on the current
page"
| g novaLista |
novaLista := OrderedCollection new.
self pageHolder list do:
[:aPage |
aPage class printString = 'SeriesViewer'
ifTrue: [novaLista := novaLista , aPage markedImages]].
g := CyclopsHtmlForPrint new.
g viewsList: novaLista.
g generateHTMLPrintFile!

generateHTMLPrintFileOnMarkeds

"Create a html file with jpeg images of the marked CyclopsImages on the
current page"

| g htmlText |
htmlText := self openSRselector value asString.
g := CyclopsHtmlForPrint new.
g viewsList: self pageHolder selection markedImages.
g generateHTMLPrintFile: htmlText!

openSRselector
"Open the a dialog to the user choose which of the current SRs he want to
print"
| response temp |

temp := OrderedCollection new.
self pageHolder list do:
[:aPage |
aPage class printString = 'DicomSRReportViewer'
| (aPage class printString = 'DicomSRReportEditor')
ifTrue: [temp add: aPage]].
response := Dialog
choose: 'Choose the Structured Report.'
fromList: temp
values: temp
lines: 8
cancel: [^''].
^response textEditor! !
```

```
!CyclopsSeriesEditor methodsFor: 'inspect'!

inspectX

"Inspect the current app"
self inspect.! !

!CyclopsSeriesEditor methodsFor: 'mail'!

mailImages
"
This method creates a new instance of
CyclopsMailWriterInterface passing the
marked Images for Delivering and open the MailWriter Interface

PRD - 05-08.2000 "

self mailImages2.
"| aCyclopsMailWriterInterface markedImages selected |

selected := self pageHolder selection.
selected class printString = 'SeriesViewer'
ifTrue: [markedImages := selected markedCyclopsImages.
markedImages isEmpty
ifTrue:
[DicomDialog warn: 'No image(s) selected!!'.
^self].
aCyclopsMailWriterInterface := CyclopsMailWriterInterface
newOn: markedImages.
aCyclopsMailWriterInterface open.

]ifFalse:[

]"!

mailImages2
"
This method creates a new instance of
CyclopsMailWriterInterface passing the
marked Images for Delivering and open the MailWriter Interface

PRD - 05-08.2000
"

| aCyclopsMailWriterInterface markedImages selected |

selected := self pageHolder selection.
selected class printString = 'SeriesViewer'
ifTrue: [markedImages := selected markedImages.
markedImages isEmpty
ifTrue:
[DicomDialog warn: 'No image(s) selected!!'.
^self].
aCyclopsMailWriterInterface := CyclopsMailWriterInterface2
newOn: markedImages.
aCyclopsMailWriterInterface open.

]ifFalse:[

]! !
```

```
!CyclopsSeriesEditor methodsFor: 'menusteste'!

colorSelectedMenu

| mb |
mb := MenuBuilder new.
mb add: 'Add Above' -> #unimplemented;
line;
add: 'Delete' -> #unimplemented;
add: 'Delete All' -> #unimplemented.
^mb menu!

menuHolder

self colorSelectedMenu.
"self nothingSelectedMenu."!

nothingSelectedMenu

| mb |
mb := MenuBuilder new.
mb add: 'Add At Bottom' -> #unimplemented;
line;
add: 'Delete All' -> #unimplemented.
^mb menu!

unimplemented

self halt.!

unistallCurrentMenu

"Unistall the current menu bar"
self windowMenuBar components clean.
self windowMenuBar menuButtons clean.
self mainWindow display.!

updateMenuToSREditor

| mb menu submenu mmm newButton |

"Update the current menu bar to a SREditor menu bar"

menu:=Menu new.
menu addItemLabel: 'Create a new SR' value: 'createSR' asSymbol.
menu addItemLabel: 'Print Document' value: 'printText' asSymbol.
menu addItemLabel: 'Save to DICOM 3.0 file' value: 'saveIODToDcmFile'
asSymbol.
menu addItemLabel: 'Store on Cyclops MIB' value: 'saveIODToDicomEditor'
asSymbol.


newButton:=(self windowMenuBar newButtonLabeled: 'Document'
accessCharacter: 1 menu: menu).
self windowMenuBar add: newButton at: 2@2.


menu:=Menu new.
menu addItemLabel: 'show documents details' value:
'renderDocumentShowIndexRelationshipsAndIdentation' asSymbol.
```

```
menu addItemLabel: 'hide all relationships identation and indexes' value:
'renderDocumentSimpleOptions' asSymbol.


newButton:=(self windowMenuBar newButtonLabeled: 'render document'
accessCharacter: 1 menu: menu).
self windowMenuBar add: newButton at: (self windowMenuBar components last
bounds corner x+5)@2.


menu:=Menu new.
menu addItemLabel:'load a Concept Name dictionar' value:
'loadConceptNameCodeDictionary' asSymbol.
menu addItemLabel: 'load a Measurement Units dictionary' value:
'loadMeasurementUnitsCodeDictionary' asSymbol.



newButton:=(self windowMenuBar newButtonLabeled: 'Dictionary'
accessCharacter: 2 menu:  menu).
self windowMenuBar add: newButton at: (self windowMenuBar components last
bounds corner x+5)@2.!

updateMenuToSRviewer

| mb menu submenu mmm newButton |

"Update the current menu bar to a SREditor menu bar"

menu:=Menu new.
menu addItemLabel: 'Create a new SR' value: 'createSR' asSymbol.

newButton:=(self windowMenuBar newButtonLabeled: 'Document'
accessCharacter: 1 menu: menu).
self windowMenuBar add: newButton at: 2@2.!

updateSeriesEditorMenu

"Update the current menu bar to a standard SeriesEditor menu bar"
self windowMenuBar updateMenu.! !

!CyclopsSeriesEditor methodsFor: 'mpr creation
'!

createPilot: aPoint orientation: aOrientationMode

| myImage newBits newImage line scaleRatio  timeNow pilotPath resizeCommand
pilotImage aPalette convertCommand shrinkCommand insertCommand yPosition
shrinkImage pilotName |
aPalette := (originalImages   first) mrCachedImage image  palette.
"calcula o numero de repeticoes de uma linha com base na largura do corte e
no tamanho do pixel"
newBits := ByteArray new.
scaleRatio := (((originalImages first) sliceThickness asNumber) /
((originalImages   first)  pixelSpacing at: 1 )asNumber) .
originalImages do: [ :aDicomImage |

myImage := aDicomImage mrCachedImage image.


(aOrientationMode = #horizontal)
```

```
ifTrue:[
line:= myImage packedRowAt: (aPoint y).
]
ifFalse:[
line:= aDicomImage mrCachedImage colAt:(aPoint x).
].
newBits := line, newBits.
].

newImage := MRCachedImage new.
newImage setImage: (
Depth16Image
extent:( myImage width )@(originalImages size)
depth: 16
palette: (aPalette)
bits:  newBits
 ).

timeNow := Time now asSeconds printString.
pilotPath := '/tmp/'. "lugar onde o pilot sera criado"
pilotName := pilotPath,'orig',timeNow,'.viff'.
newImage exportImageVIFF: pilotName.


Cursor currentCursor: Cursor wait.
Transcript show: 'Iniciando tratamento da imagem...';cr.
resizeCommand := 'kresample -i ',pilotPath,'orig' ,timeNow,'.viff -o
',pilotPath,'resize',timeNow,'.viff -wmag 1.0 -hmag ', (scaleRatio
printString).
Transcript show: (UnixProcess cshOne: resizeCommand); show: resizeCommand.

shrinkCommand := 'kshrink -i ',pilotPath,'resize',timeNow,'.viff -o
',pilotPath,'shrink',timeNow,'.viff -wmag 0.50 -hmag 0.5'.
Transcript show: (UnixProcess cshOne: shrinkCommand); show: shrinkCommand.

convertCommand := 'kformats -i ',pilotPath,'shrink',timeNow,'.viff -o
',pilotPath,'final',timeNow,'.viff -viff'.
Transcript show: (UnixProcess cshOne: convertCommand); show:
convertCommand.

shrinkImage := (MRCachedImage new) importImageVIFF:
pilotPath,'final',timeNow,'.viff'.
yPosition := ((((originalImages first mrCachedImage height) - (shrinkImage
image height)) / 4)/2) rounded.

insertCommand :='kinset -i1 data/kernel/image512.xv -i2
',pilotPath,'shrink',timeNow,'.viff -o ',pilotPath,'final',timeNow,'.viff -
hoff '  ,yPosition printString.
Transcript show: (UnixProcess cshOne: insertCommand); show: insertCommand.

convertCommand :=  'kformats -i ',pilotPath,'final',timeNow,'.viff -o
',pilotPath,'khoros1',timeNow,'.viff -viff'.
Transcript show: (UnixProcess cshOne: convertCommand); show:
convertCommand.

      pilotImage := (MRCachedImage new) importImageVIFF16:
pilotPath,'khoros1',timeNow,'.viff'.
pilotImage retainedMedium: nil.

pilotImage image palette: (aPalette).
Cursor currentCursor: Cursor normal.
```

```
^pilotImage.!

createPilot: aPoint orientation: aOrientationMode onSeries: originalImages

| myImage newBits newImage line pilotImage aPalette |
"This function get a line of pixel on each image and create a multiplanar
reconstruction of the image, from a point gived "

aPalette := originalImages first mrCachedImage image palette.
"Copy the first image palette"
newBits := ByteArray new.
originalImages first sliceThickness asNumber
/ (originalImages first pixelSpacing at: 1) asNumber.
originalImages do:
[:aDicomImage |
myImage := aDicomImage mrCachedImage image.
aOrientationMode = #horizontal
ifTrue: [line := myImage packedRowAt: aPoint y]
ifFalse: [line := aDicomImage mrCachedImage colAt: aPoint x].
newBits := line , newBits].
newImage := MRCachedImage new.
newImage setImage: (Depth16Image
extent: myImage width @ originalImages size
depth: 16
palette: aPalette
bits: newBits).
pilotImage := newImage shrunkenBy: 2 @ 0.3.


^pilotImage!

createPilots


| aPonto serie image |
"Call the creator of multiplanar images in accordance with  the type of the
image"

self cyclopsSeries do:
[:aSeries |
aSeries modality ~= 'SR'
ifTrue:
[ serie := aSeries images asSortedCollection.
image := (serie at:1) mrCachedImage .
aPonto := (image height/2)@(image width/2).
"serie first sopClassUID = '1.2.840.10008.5.1.4.1.1.2' & "(serie size > 5)
ifTrue:
[self pilot add: (self
createPilot: aPonto
orientation: #vertical
onSeries: serie).
self coPilot add: (self
createPilot: aPonto
orientation: #horizontal
onSeries: serie)]
ifFalse:
[self pilot add: ((serie at: 1) mrCachedImage shrunkenBy: 2 @ 2).
self coPilot add: self pilot last]]]!

currentCoPilot
```

```
"Select the current or last MPR image to show on the viewHolder, if you
change the tab to a SR program, the last MPR will be show"

| txt emph label |
self seriesViewersList notEmpty ifTrue:[
^self coPilot at: self lastSeries.]ifFalse:[

txt := 'MPR not available' asText.
emph := Array
with: #bold
with: #color->ColorValue red.
txt emphasizeFrom: 1 to: 17 with: emph.
label := Label
with: txt
attributes: (TextAttributes styleNamed: #large).
^label
]!

currentPilot

"Select the current or last MPR image to show on the viewHolder, if you
change the tab to a SR program, the last MPR will be show"
| txt emph label |
self seriesViewersList notEmpty ifTrue:[
^self pilot at: self lastSeries.]ifFalse:[

txt := 'MPR not available' asText.
emph := Array
with: #bold
with: #color->ColorValue red.
txt emphasizeFrom: 1 to: 17 with: emph.
label := Label
with: txt
attributes: (TextAttributes styleNamed: #large).
^label
]! !

!CyclopsSeriesEditor methodsFor: 'pages'!

pages

"Return the list of the pages"
^self pageHolder list!

tabControl

"Return the tabs"
^(self widgetAt: #pages)! !

!CyclopsSeriesEditor methodsFor: 'pages add - remove'!

addAndSelectPage: aWorkspacePage


"Add a page to the tabControl"
self addDependent: aWorkspacePage.
self addPage: aWorkspacePage.
self pageHolder selection: aWorkspacePage!

addPage: aWorkspacePage
```

```
"The Variables page always remains the last, so any new page
always ends up being one before the last."
| inspectorPage |
inspectorPage := self pages last.
self pages add: aWorkspacePage after: inspectorPage!

removePage: aWorkspacePage


"Remove the page"
self pages remove: aWorkspacePage! !

!CyclopsSeriesEditor methodsFor: 'paging functions'!

deactivate

^true.!

isAccepted

^true.!

label

^'aaaaaaaaaa'.!

refresh!

verboseLabel

^'aaaaaaaaaa'.!

verboseLevel

^'aaaaaaaaaa'.! !

!CyclopsSeriesEditor methodsFor: 'SR'!

addImageToSR
"Mariana Kessler Bortoluzzi January 16th, 2003"

| anImageUID aSOPClassUID |
(self markedImages notNil and: [self markedImages size = 1])
ifTrue:
[anImageUID := (self markedImages at: 1) dicomIOD cyclopsDicomSOPCommonIOM
sopInstanceUID.
aSOPClassUID := (self markedImages at: 1) dicomIOD cyclopsDicomSOPCommonIOM
sopClassUID]
ifFalse: [nil].
self dicomSRReportEditor
applyUIDToSelectedImageItemWith: anImageUID
classUID: aSOPClassUID
andClose: self!

createSR
| g menu dialogModel dialogBuilder CreatingNewSRInNewSeriesforMIBSeries |

g := CreatingSRInNewSeriesForSeriesEditor newOn: self aCyclopsStudy.

g dicomEditor: self cyclopsMIB.
```

```
dialogModel := SimpleDialog new.
g dialog: dialogModel.
dialogBuilder := dialogModel builder.
dialogModel openFor: g interface: #windowSpec.
"g openDialogInterface: #windowSpec."
g srReportEditor notNil
ifTrue:
[self addAndSelectPage: g srReportEditor.
g srReportEditor mIBSeriesEditor:  self.
g srReportEditor open2.
menu := self menuFor: #menu.
self lasrSRReportEditor: g srReportEditor.

]! !

!CyclopsSeriesEditor methodsFor: 'SREditor actions'!

aboutSREditor



self pageHolder selection aboutSREditor.!

loadConceptNameCodeDictionary



self pageHolder selection loadConceptNameCodeDictionary.!

loadMeasurementUnitsCodeDictionary



self pageHolder selection loadMeasurementUnitsCodeDictionary.!

renderDocumentShowIndexRelationshipsAndIdentation



self pageHolder selection
renderDocumentShowIndexRelationshipsAndIdentation.!

renderDocumentSimpleOptions



self pageHolder selection renderDocumentSimpleOptions.!

saveIODToDcmFile



self pageHolder selection saveIODToDcmFile.!

saveIODToDicomEditor



self pageHolder selection saveIODToDicomEditor.! !
```

```
!CyclopsSeriesEditor methodsFor: 'tabs creation'!

installPage: aWorkspacePage

lastInstalledPage := aWorkspacePage.
aWorkspacePage builder: nil.
self tabControl
client: aWorkspacePage
spec: #windowSpec
builder: builder newSubBuilder.
self updatePage.
aWorkspacePage refresh!

reInstallInterface
"It is suspicious that the selection is not preserved while
rebuilding, but for now let's just work around it."

| pageIndex |
pageIndex := self pageHolder selectionIndex.
super reInstallInterface.
self pageHolder selectionIndex: pageIndex! !

!CyclopsSeriesEditor methodsFor: 'taskbar'!

changedTaskBar
| newTaskBar |

newTaskBar := self taskBarSelector value.


     newTaskBar == #analise ifTrue: [
self showAnaliseTaskBar.
     ].

newTaskBar == #window ifTrue: [
self showWindowTaskBar.
     ].!

changedTaskBar2
| newTaskBar |

newTaskBar := self taskBarSelector2 value.


     newTaskBar = #saveIODToDicomEditor ifTrue: [
self halt.
self saveIODToDicomEditor
     ].!

hide


self inspect.
self moveBy: 100@-100.!

hidetaskbar

"Da um resize na taskbar por si so."


"
```

```
(self widgetAt: #ResizingSplitter1 )moveBy: 100@100."

(self widgetAt: #ResizingSplitter2 )moveBy: 100@200.!

popcorn


"| subcanvas spec application |

""Create the subapplication and initialize it.""
application := PCRangeDefiner newMin2: 1 max: (self lastSeries
imageVieweList size) editor: self .

""Get the spec object for the embedded canvas.""
spec := PCRangeDefiner interfaceSpecFor: #windowSpec.

""Get the subcanvas and install the editing application.""
subcanvas := (self builder componentAt: #taskBar) widget.
subcanvas client: application spec: spec.
application displayNotes."!

showAnaliseTaskBar


| subcanvas spec application |

"Create the subapplication and initialize it."
application := CyclopsSeriesEditorTaskBar new.
 application owner: self.
"Get the spec object for the embedded canvas."
spec := CyclopsSeriesEditorTaskBar interfaceSpecFor: #windowSpec.

"Get the subcanvas and install the editing application."
subcanvas := (self builder componentAt: #taskBar) widget.
subcanvas client: application spec: spec.
application displayNotes.!

showAtlasTaskBar


| subcanvas spec application |

"Create the subapplication and initialize it."
application :=BrainAtlasTaskBar new.
 application owner: self.
"Get the spec object for the embedded canvas."
spec := BrainAtlasTaskBar interfaceSpecFor: #windowSpec.

"Get the subcanvas and install the editing application."
subcanvas := (self builder componentAt: #taskBar) widget.
subcanvas client: application spec: spec.
self taskBar: application.!

showPopCornTaskBar


| subcanvas spec application tmp |

"Create the subapplication and initialize it."
tmp := (self pageHolder listHolder value at: self lastSeries).
```

```
application := PCRangeDefiner newMin2: 1 max: (tmp imageVieweList size)
editor: tmp .

"Get the spec object for the embedded canvas."
spec := PCRangeDefiner interfaceSpecFor: #windowSpec.

"Get the subcanvas and install the editing application."
subcanvas := (self builder componentAt: #taskBar) widget.
subcanvas client: application spec: spec.!

showStrokeTaskBar


| subcanvas spec application |

"Create the subapplication and initialize it."
application := StrokeTaskBar newOn: self pageHolder selection.
"Get the spec object for the embedded canvas."
spec:= StrokeTaskBar interfaceSpecFor: #windowSpec.

"Get the subcanvas and install the editing application."
subcanvas := (self builder componentAt: #taskBar) widget.
subcanvas client: application spec: spec.
application showFirstTab.
self taskBar: application.!

showWindowTaskBar

| subcanvas spec |

seriesViewersList notEmpty ifTrue:[
"Create the subapplication and initialize it."
self openPreviewWindowOn: (self pageHolder selection cyclopsSeries images
asOrderedCollection at: 1 ).

"Get the spec object for the embedded canvas."
spec := CyclopsSeriesEditorWindowBar interfaceSpecFor: #windowSpec.

"Get the subcanvas and install the editing application."
subcanvas := (self builder componentAt: #taskBar) widget.
subcanvas client: windowR spec: spec.].! !

!CyclopsSeriesEditor methodsFor: 'toolbar actions'!

coronalMPR

self toolFlag value: #coronal.!

distance

"Set the current tool to distance measurer"
self toolFlag value: #distance.!

drawCubo


self taskBar drawCubo.!

drawCuboChildren
```

```
self taskBar drawChildren.!

normal

"Set the current tool to normal selector"
self toolFlag value: #marker.!

pen
"***This is decompiled code.***
The source was unavailable because the source pointer appears to point to
an incorrect position in
the file. The file may have been modified after this method was updated."

self toolFlag value: #pen.
^self!

sagitalMPR

self toolFlag value: #sagital.!

setToolFlagAtlas
"***This is decompiled code.***
The source was unavailable because the source pointer appears to point to
an incorrect position in
the file. The file may have been modified after this method was updated."

self toolFlag value: #atlas.
^self!

setToolFlagLocate
"***This is decompiled code.***
The source was unavailable because the source pointer appears to point to
an incorrect position in
the file. The file may have been modified after this method was updated."

self toolFlag value: #locate.
^self!

zoomMinus
"Change the size of all image to a minor size"

| index shrunk |
index := self pageHolder selectionIndex.

self pageHolder selection class printString = 'SeriesViewer' ifTrue:[
shrunk := self shrunkFactor at: index.
shrunk value < 15 ifTrue: [shrunk value: shrunk value + 0.1].
self changedShrunkFactor: index].!

ZoomPlus
"Change the size of all image to a major size"

| index shrunk |
index := self pageHolder selectionIndex.

self pageHolder selection class printString = 'SeriesViewer' ifTrue:[
shrunk := self shrunkFactor at: index.
shrunk value > 0.1 ifTrue: [shrunk value: shrunk value - 0.1].
self changedShrunkFactor: index.]! !

!CyclopsSeriesEditor methodsFor: 'trecos'!
```

```
exit! !

!CyclopsSeriesEditor methodsFor: 'views'!

copilotImageView
"***This is decompiled code.***
The source was unavailable because the source pointer appears to point to
an incorrect position in
the file. The file may have been modified after this method was updated."

(coPilotView model: ((CyclopsWindowTool newOn: self)
name: 'a' printString asSymbol))
owner: self.
coPilotView model name: #coronal.
^coPilotView!

pilotImageView
"***This is decompiled code.***
The source was unavailable because the source pointer appears to point to
an incorrect position in
the file. The file may have been modified after this method was updated."

(pilotView model: ((CyclopsWindowTool newOn: self)
name: 'a' printString asSymbol))
owner: self.
pilotView model name: #sagital.
^pilotView! !

!CyclopsSeriesEditor methodsFor: 'window'!

createPreviewImageOn: aCyclopsDicomImage
| newIm tempIm pixDat pal factorX factorY |
factorX := aCyclopsDicomImage columns / 256.
factorY := aCyclopsDicomImage rows / 256.
tempIm := aCyclopsDicomImage mrCachedImage image
shrunkenBy: factorX @ factorY.
aCyclopsDicomImage bitsAllocated = 8
ifTrue:
[pixDat := tempIm bits.
pal := tempIm palette copy]
ifFalse:
[aCyclopsDicomImage bitsStored = 8
ifTrue:
[pixDat := self bitsAllocated16BitsStored8HighBit7With: tempIm bits.
pal := self
create8BitPalette: aCyclopsDicomImage privatePresentationState wC
@ aCyclopsDicomImage privatePresentationState wW
for: aCyclopsDicomImage]
ifFalse:
[aCyclopsDicomImage bitsStored = 12
ifTrue:
[pixDat := self bitsAllocated16BitsStored12HighBit11With: tempIm bits.
pal := self
create8BitPalette: (aCyclopsDicomImage privatePresentationState wC / 16)
truncated @ (aCyclopsDicomImage privatePresentationState wW / 16) truncated
for: aCyclopsDicomImage]
ifFalse:
[aCyclopsDicomImage pixelRepresentation = 1
ifTrue:
[pixDat := self
```

```
bitsAllocated16BitsStored16HighBit15PixelRepresentation1With: tempIm bits.
pal := self
create8BitPalette: (aCyclopsDicomImage privatePresentationState wC / 16)
truncated @ (aCyclopsDicomImage privatePresentationState wW / 16) truncated
for: aCyclopsDicomImage]
ifFalse:
[pixDat := self bitsAllocated16BitsStored16HighBit15With: tempIm bits.
pal := self
create8BitPalette: (aCyclopsDicomImage privatePresentationState wC / 256)
truncated
@ (aCyclopsDicomImage privatePresentationState wW / 256) truncated
for: aCyclopsDicomImage]]]].
newIm := Image
extent: 256 @ 256
depth: 8
bitsPerPixel: 8
palette: pal
usingBits: pixDat.
^newIm.!

openPreviewWindowOn: aCyclopsDicomImage
| presState newIm photo convFactor pt preview result |
presState := aCyclopsDicomImage privatePresentationState.
newIm := self createPreviewImageOn: aCyclopsDicomImage.
(aCyclopsDicomImage photometricInterpretation includes: 50 asCharacter)
ifTrue: [photo := 2]
ifFalse: [photo := 1]."looking for MONOCHROME2 - (50 asCharacter) = 2"
"looking for MONOCHROME1 - (49 asCharacter) = 1"
aCyclopsDicomImage bitsAllocated = 8 | (aCyclopsDicomImage bitsStored = 8)
ifTrue: [convFactor := 1]
ifFalse:
[aCyclopsDicomImage bitsStored = 12
ifTrue: [convFactor := 16]
ifFalse:
[convFactor := 16.
aCyclopsDicomImage pixelRepresentation = 0 ifTrue: [convFactor := 256]]].
pt := presState wW @ presState wC.
self windowR: (CyclopsSeriesEditorWindowBar
newOn: newIm
wAndC: pt
photometric: photo
conversion: convFactor
pixRepresentation: aCyclopsDicomImage pixelRepresentation).
self windowR owner: self!

updateAllImages

"self validateWindowC.
self validateWindowW."
| wc ww img newImage factor |
Cursor wait show.
factor := (shrunkFactor at: self pageHolder selectionIndex) value.
self pageHolder selection imageVieweList do: [ :im | im cyclopsImage
privatePresentationState wW: (self windowW value).
im cyclopsImage privatePresentationState wC: (self windowC value).
" im original:  im cyclopsImage mrCachedImage."
wc :=  (im cyclopsImage privatePresentationState wC).
ww :=  (im cyclopsImage privatePresentationState wW).
im cyclopsImage setPalette: (self pageHolder selection referencedPalettes
at: wc@ww ifAbsentPut: [im cyclopsImage
createPaletteFromPresentationState]).
```

```
img :=im cyclopsImage mrCachedImage image .
img convertToPalette: (img palette).
im cyclopsImage mrCachedImage setImage: (img).
newImage:= im cyclopsImage mrCachedImage shrunkenBy: (factor@factor).
newImage asRetainedMedium.
im imageList at:1 put: newImage .
im currentImage: newImage.



].
"self redisplayAllImages."
Cursor normal show.
ObjectMemory globalGarbageCollect.!

updateMarkedImages

"self validateWindowC.
self validateWindowW."
"]"

| wc ww img newImage factor |
Cursor wait show.
factor := (shrunkFactor at: self pageHolder selectionIndex) value.
self pageHolder selection imageVieweList do: [ :im |(im markedImage value)
ifTrue: [ im cyclopsImage privatePresentationState wW: (self windowW
value).
im cyclopsImage privatePresentationState wC: (self windowC value).
" im original:  im cyclopsImage mrCachedImage."
wc :=  (im cyclopsImage privatePresentationState wC).
ww :=  (im cyclopsImage privatePresentationState wW).
im cyclopsImage setPalette: (self pageHolder selection referencedPalettes
at: wc@ww ifAbsentPut: [im cyclopsImage
createPaletteFromPresentationState]).
img :=im cyclopsImage mrCachedImage image .
img convertToPalette: (img palette).
im cyclopsImage mrCachedImage setImage: (img).
newImage:= im cyclopsImage mrCachedImage shrunkenBy: (factor@factor).
newImage asRetainedMedium.
im imageList at:1 put: newImage .
im currentImage: newImage.
].


].
"self redisplayAllImages."
Cursor normal show.
ObjectMemory globalGarbageCollect.! !

!CyclopsSeriesEditor methodsFor: 'zoom functions'!

changedShrunkFactor: tabIndex



|seriesViewer|
"Set the zoom factor for the selected page"
seriesViewer:=(self seriesViewersList at:tabIndex).
seriesViewer changedShrunkFactor.! !

!CyclopsSeriesEditor methodsFor: 'private'!
```

```
bitsAllocated16BitsStored12HighBit11With: temp
|pixel bitsLow bitsUp index|

"Idea of this algorithm: see comments"

pixel := ByteArray new: (temp size / 2).
"pixel := ByteArray new: (newImage rows * newImage columns)."
index := 1.
((temp size / 2) - 1)
timesRepeat:
[bitsLow := temp at: (index * 2).
bitsLow := bitsLow bitAnd: 240.
bitsLow := bitsLow bitShift: -4.
bitsUp := temp at: (index * 2) + 1.
bitsUp := bitsUp bitAnd: 15.
bitsUp := bitsUp bitShift: 4.
pixel at: index + 1 put: (bitsUp bitOr: bitsLow).
index := index + 1].
^pixel.!

bitsAllocated16BitsStored16HighBit15PixelRepresentation1With: temp
|pixel bitsUp index|
" Assumes that image is in 'CT-like style' with effictive pixel values from
-4096 to 4095."


pixel := ByteArray new: (temp size).
index := 1.
(temp size / 2)
timesRepeat:
[bitsUp := temp at: index.
bitsUp > 15
ifTrue: [bitsUp := bitsUp bitShift: -4].
pixel at: index put: bitsUp.
index := index + 2].
index := 2.
(temp size / 2)
timesRepeat:
[pixel at: index put: (temp at: index).
index := index + 2].
pixel := self bitsAllocated16BitsStored12HighBit11With: pixel.
^pixel.!

bitsAllocated16BitsStored16HighBit15With: temp
|pixel bitsUp index|

"comments"

pixel := ByteArray new: (temp size) / 2.
index := 0.
((temp size / 2) - 1)
timesRepeat:
[bitsUp := temp at: (index * 2) + 1.
pixel at: index + 1 put: bitsUp.
index := index + 1].
^pixel.!

bitsAllocated16BitsStored8HighBit7With: temp
|pixel index|
```

```
pixel := ByteArray new: (temp size) / 2.
index := 1.
temp do:[ :byte | (index even) ifTrue: [pixel at: (index / 2) put: byte].
index := index + 1.].

^pixel.!

create8BitPalette: aPoint for: anImage

| wc ww upper lower type palette |

type := 8.
wc := aPoint x.
ww := aPoint y.
(ww = 0) ifTrue: [ ww := 1].
lower := wc - (((ww - 1) / 2) rounded).
upper := wc + (((ww - 1) / 2) rounded).

anImage pixelRepresentation = 1 ifTrue: [
(anImage photometricInterpretation includes: 50 asCharacter) ifTrue: [
"looking for MONOCHROME2 - (50 asCharacter) = 2"
palette := MRMappedPalette initializeForM2SignedPixelDataWithCenter: wc
width: ww depth: type].
(anImage photometricInterpretation includes: 49 asCharacter) ifTrue: [
"looking for MONOCHROME1 - (49 asCharacter) = 1"
palette := MRMappedPalette initializeForM1SignedPixelDataWithCenter: wc
width: ww depth: type].
].
anImage pixelRepresentation = 0 ifTrue: [
(anImage photometricInterpretation includes: 50 asCharacter) ifTrue: [
"looking for MONOCHROME2 - (50 asCharacter) = 2"
palette := MRMappedPalette initializeWithLower: lower asInteger withUpper:
upper asInteger depth: type].
(anImage photometricInterpretation includes: 49 asCharacter) ifTrue: [
"looking for MONOCHROME1 - (49 asCharacter) = 1"
palette := MRMappedPalette initializeWithLower: upper withUpper: lower
depth: type].
].

^palette!

evaluateResultOfPaletteManipulation: anAssociation
anAssociation notNil
ifTrue:
[anAssociation key = #selected
ifTrue:
[anAssociation value x > 0
ifTrue: [self windowW value: anAssociation value x]
ifFalse: [self windowW value: 1].
self windowC value: anAssociation value y.
self updateMarkedImages]
ifFalse:
[anAssociation value x > 0
ifTrue: [self windowW value: anAssociation value x]
ifFalse: [self windowW value: 1].
self windowC value: anAssociation value y.
self updateAllImages]]! !
```