Faculdade de Engenharia da Universidade do Porto

Broker de comunicação em protocolos de comunicação para a saúde

João Miguel Monteiro Pinto

Relatório de Projecto realizado(a) no Âmbito do
Mestrado Integrado em Engenharia Informática e Computação

Orientador: António Miguel Pontes Pimenta Monteiro (Doutor)

Julho de 2008

Broker de comunicação em protocolos de comunicação para a saúde

João Miguel Monteiro Pinto

Relatório de Projecto realizado(a) no Âmbito do
Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Jaime dos Santos Cardoso (Doutor)

Arguente: Carlos Manuel Azevedo Costa (Doutor)

Vogal: António Miguel Pontes Pimenta Monteiro (Doutor)

17 de Julho de 2008

Resumo

Este trabalho foi realizado no âmbito da disciplina de Projecto a decorrer no segundo semestre do quinto ano do MIEIC - Mestrado Integrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto. O projecto foi realizado na Siemens Medical Solutions (MED), durante um período aproximado de vinte semanas.

Este projecto, denominado de DCM4HT, pode ser definido como um módulo adicional ao produto em desenvolvimento Healthy Talkie que visa permitir a comunicação entre sistemas não compatíveis, dentro de uma instituição de Saúde.

O objectivo deste projecto é de aumentar as capacidades de integração do Healthy Talkie com um novo protocolo, o protocolo DICOM habilitando-o na recepção, envio e processamento de mensagens deste tipo, mantendo a actual arquitectura do produto de modo a ser possível a tradução de uma mensagem DICOM para um outro protocolo, consoante as necessidades do sistema destinatário.

Este protocolo, criado pela ACR (*American College of Radiology*) e NEMA (*National Electrical Manufacturers Association*) em 1983, surge da necessidade de criar um método padrão para transmissão de imagens médicas e informação associada, entre diferentes dispositivos.

Espera-se que esta ferramenta permita, por exemplo, a recepção de um pedido de *Modality Worklist*, que construa uma mensagem de resposta com base em informações existentes numa base de dados ou num outro equipamento e a envie com sucesso para o destinatário.

Para a realização deste trabalho, foi necessário o envolvimento com a actual equipa do Healthy Talkie, utilizando o processo ágil de desenvolvimento *Scrum*.

Dos resultados obtidos, podemos salientar a implementação completa dos mecanismos de comunicação de rede do protocolo DICOM, sem recurso a APIs externas e o suporte ao serviço DICOM *Modality Worklist*.

Quanto ao trabalho a desenvolver no futuro, é necessário garantir que a conversão de uma mensagem DICOM para um outro protocolo como HL7 ou CSV é correctamente realizada sendo vital garantir a integridade da informação no ambiente a que este produto se destina.

Abstract

This project has been develop under the class "Project", on the second semester of the fifth year of MIEIC (Informatics and Computers Integrated Masters of FEUP). The estimated project duration was of twenty weeks and was undertaken on Siemens Medical Solutions(MED).

Project DCM4HT, as it was entitled, is an additional module developed for the product Healthy Talkie. The module itself was developed envisaging communications between incompatible systems on Health Care Institutions to be possible and that is in fact the module's function.

The enhancement of the integration capabilities of Healthy Talkie with the DICOM protocol was the project's ultimate goal. With this new enhancement, Healthy Talkie would not only be able to receive, transmit and process messages of this type but it would also allow it to maintain the product's own architecture so that a translation to other protocols would still be possible on demand of the receiving system.

The DICOM protocol, created by ACR (*American College of Radiology*) and NEMA (*National Electrical Manufacturers Association*) in 1983, came as an answer to the need for medical images and related information to be transmitted by a standard method between different peripherals.

This tool is thought to be able to receive a Modality Worklist request and, from a data base's information or from information stored on other peripherals, to work out and successfully send a message to the sender.

It is imperative for this project's realization the involvement with the actual Healthy Talkie's team using the Scrum's agile process development.

From the results obtained we can stand out the complete implementation of the DICOM protocol on the network's communication mechanisms without resorting to the use of external APIs as well as the DICOM's Modality Worklist service support.

As for future development plans, it is necessary to ensure that the conversion of a DICOM message to other protocols, mainly HL7 or CSV, is successfully carried out while ensuring the product's information integrity on the intended destination.

Agradecimentos

Ao Professor Raul Moreira Vidal e ao actual Director do MIEIC, o Professor Augusto Sousa por me terem proporcionado uma formação exemplar e por assegurarem da melhor forma o prestígio merecido que este curso possui.

Ao meu Orientador de projecto, o Professor António Miguel Pimenta Monteiro, pela ajuda preciosa que ofereceu durante estas vinte semanas.

Ao meu supervisor na Siemens S.A., o Eng. António Martins, por ter partilhado a sua vasta experiência e conhecimentos, e pelo rigor sempre presente nas suas observações.

A todos o colegas que tive o prazer de conhecer nesta instituição, mas em especial ao Diogo Bastos e Rui Fontinha que sempre me ajudaram a ultrapassar os obstáculos que surgiram neste percurso e que sem eles não o teria concluído com sucesso.

Á minha família, a quem devo tudo aquilo que sou, e por estarem sempre presentes, quer nos bons momentos, quer nos momentos menos fáceis deste período académico.

Á minha querida Paula, por me fazer sonhar e por dedicar um sabor único a cada dia que passa.

A todos vós, muito obrigado!

O Autor

Abreviaturas e Símbolos

API Application Programming Interface

AS Age String

CS Code String

DA Date

CVS Concurrent Version System

DICOM Digital Imaging Communications in Medicine

HL7 Health Level 7

IDE Integrated development environment

JRE Java Runtime Environment

MIEIC Mestrado Integrado em Engenharia Informática e Computação

PDU Protocol Data Unit

RIS Radiology Information System

SCP Service Class Provider

SCU Service Class User

SQ Sequence of Items

SQL Structured Query Language

XML Extensible Markup Language

Conteúdo

1 Introdução **1**
- 1.1 Contexto/Enquadramento . . . 1
- 1.2 Projecto . . . 1
- 1.3 Motivação e Objectivos . . . 2
- 1.4 Metodologias de desenvolvimento . . . 2
- 1.5 Estrutura do relatório . . . 2

2 Estado da Arte **5**
- 2.1 Healthy Talkie . . . 5
- 2.2 O protocolo DICOM . . . 7
  - 2.2.1 Serviços DICOM . . . 7
  - 2.2.2 Modelo de Informação . . . 8
  - 2.2.3 Objectos DICOM . . . 9
  - 2.2.4 Transporte . . . 11
    - 2.2.4.1 Negociação . . . 11
- 2.3 A Solução DCM4HT . . . 12
  - 2.3.1 Funcionalidades Principais . . . 12
- 2.4 Outras soluções para integração de sistemas . . . 12
  - 2.4.1 Mirth Project . . . 12
  - 2.4.2 MEDxConnect . . . 13
  - 2.4.3 Emageon . . . 14
  - 2.4.4 IC2E Collaboration Suite . . . 15
  - 2.4.5 Hipax Workstation 4 . . . 15
- 2.5 Comparação dos produtos mencionados . . . 16
- 2.6 Tecnologias de desenvolvimento . . . 17
- 2.7 Conclusões . . . 18

3 Projecto e implementação do sistema DCM4HT **19**
- 3.1 Especificação de Requisitos . . . 19
- 3.2 Arquitectura . . . 24
- 3.3 Implementação . . . 30
  - 3.3.1 Estabelecimento da Comunicação . . . 34
  - 3.3.2 Negociação entre DICOM SCU e SCP . . . 36
  - 3.3.3 Recepção da mensagem DICOM (P-DATA-TF PDUs) . . . 38
  - 3.3.4 Processo *Untranslate* . . . 40
  - 3.3.5 Processamento da mensagem DICOM . . . 43
  - 3.3.6 Processo *Translate* . . . 45

3.3.7 Envio da mensagem DICOM (P-DATA-TF PDUs) 46
3.3.8 Fecho da associação entre DICOM SCP e SCU 47
3.4 Testes e funcionamento 48

4 Conclusões e Trabalho Futuro **55**
4.1 Satisfação dos Objectivos 55
4.2 Trabalho Futuro 56

Referências **57**

A Detalhes do protocolo DICOM **59**
A.1 Estrutura das principais mensagens 59

Lista de Figuras

2.1 Exemplo de uma rede DICOM 8
2.2 DICOM Tags 10
2.3 IOD - Estrutura Organizacional 10
2.4 Um *scanner* inicia a rotina de transferência de imagens pela fase de associação. São negociados vários detalhes durante o estabelecimento da associação, para que a *workstation* esteja preparada para processar a imagem que está para receber. 11
2.5 Pedido de associação 12

3.1 Cenário de uso: *Modality Worklist* 22
3.2 Cenário de uso: Exame sem marcação 22
3.3 Cenário de uso: *Modality Performed Procedures Steps* 23
3.4 Cenário de uso: anonimização de dados 23
3.5 Cenário de uso: reconhecimento e autorização de DICOM SCU 24
3.6 Arquitectura geral do DCM4HT 25
3.7 Sequência de recepção 26
3.8 Sequência de envio 26
3.9 Modelo do processo *receive* 27
3.10 Modelo do processo *untranslate* 28
3.11 Fluxo do módulo *in-process* 28
3.12 O módulo *Route* é o responsável por direccionar uma mensagem para um ou mais destinos. 29
3.13 Modelo do processo *out-process* 29
3.14 Modelo do processo *translate* 30
3.15 Fluxo do módulo *send* 31
3.16 Exemplo de tabela de integração 31
3.17 Fluxo de informação DICOM *Modality Worklist* 32
3.18 Simulador TIANI DICOM *Modality Worklist* SCU 33
3.19 Fluxo do mecanismo C-FIND entre *Service Class User* e *Service Class Provider* 34
3.20 Paralelismo entre um *select* de SQL e um C-FIND-RQ 34
3.21 Exemplo de um pedido C-FIND e respectiva resposta 35
3.22 Estrutura detalhada do P-DATA-TF PDU 39
3.23 Modelo de uma MetadataTree 41
3.24 Estrutura de um elemento DICOM do tipo implícito 42
3.25 Fluxo de informação no envio de um C-FIND-RSP 47
3.26 Mensagens trocadas durante fecho de associação 48

3.27 Fluxo de informação do cenário de teste 49
3.28 Aplicação TIANI Modality SCU 49
3.29 Tabela Base de dados MySQL 50
3.30 Execução do Healthy Talkie 52
3.31 Aplicação TIANI Modality SCU com valores do pedido 53
3.32 Aplicação TIANI Modality SCU com o resultado da pesquisa 54

Lista de Tabelas

2.1 Serviços DICOM V3.0 suportados como SCU (Service Class User) . . 17
2.2 Serviços DICOM V3.0 suportados como SCP (Service Class Provider) 17
2.3 *Transfer Syntaxes* suportadas 17

3.1 Requisitos especificados inicialmente para o DCM4HT (*Release Beta*) 21
3.2 Requisitos especificados inicialmente para o DCM4HT (*Release Final*) 21

Capítulo 1

Introdução

Este capítulo pretende apresentar de um modo geral, o enquadramento deste projecto, o ambiente no qual este foi desenvolvido e a importância deste para a empresa. Por fim, será fornecida uma descrição da estrutura deste relatório.

1.1 Contexto/Enquadramento

Este trabalho surge no contexto da disciplina de Projecto a decorrer no segundo semestre do quinto ano do MIEIC - Mestrado Integrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto. O projecto foi realizado na Siemens Medical Solutions (MED), após acordo mútuo entre as duas partes, durante um periodo aproximado de quatro meses e meio. Todo o processo de desenvolvimento esteve sob orientação do Prof. Doutor António Miguel Pimenta Monteiro e do responsável pelo projecto na instituição externa, Eng. António Cardoso Martins.

1.2 Projecto

Este projecto pode ser definido como um módulo adicional ao produto em desenvolvimento na Siemens MED chamado Healthy Talkie.

O Healthy Talkie é uma interface de comunicação. Visa permitir a comunicação (transferência de informação) entre sistemas não compatíveis, dentro de uma instituição de Saúde. Ou seja, recorrendo ao Healthy Talkie podemos interligar duas aplicações que usam diferentes protocolos de comunicação, podendo estes variar entre HL7, CSV, XML, etc...

O objectivo do projecto consistem em aumentar as capacidades de integração do Healthy Talkie com um novo protocolo, o protocolo DICOM, habilitando-o na recepção, envio e processamento de mensagens DICOM, mantendo a actual arquitectura do produto de modo a ser possível a tradução de uma mensagem DICOM para um outro protocolo, consoante as necessidades do sistema destinatário.

1.3 Motivação e Objectivos

Particularizando um pouco mais, o objectivo deste projecto consiste no desenvolvimento da capacidade de recepção, processamento e envio de dados em protocolo DICOM v3.0, como uma nova funcionalidade do software de interface de comunicação Healthy Talkie (excepto a manipulação de imagem), o qual neste momento, apenas suporta HL7 e CSV.

Como resultado final, espera-se que a ferramenta permita, por exemplo, a recepção de um pedido de *Modality Worklist*, que construa uma mensagem de resposta com base em informações existentes numa base de dados ou num outro equipamento, e a envie com sucesso para o destinatário.

Esta aplicação deve também suportar os principais serviços DICOM como *Modality Worklist* ou *Storage*, fundamentais para a comunicação de informação com sistemas *PACS*, *RIS*, etc...

1.4 Metodologias de desenvolvimento

Para a realização do trabalho, foi necessário o envolvimento com a actual equipa do Healthy Talkie, utilizando o processo ágil de desenvolvimento *Scrum*.

Com *Scrum* o desenvolvimento do produto é dividido em múltiplas iterações, onde cada iteração tem uma duração definida (geralmente de 30 dias) e onde são definidas por ordem de prioridade as tarefas a realizar nesse período. Este método visa promover a capacidade de resposta da equipa, melhorando a eficiência na gestão dos processos.

1.5 Estrutura do relatório

Este documento encontra-se estruturado da seguinte forma. O actual capítulo, "Introdução", que situa o leitor no contexto do projecto, descrevendo o meio no qual este foi desenvolvido e os objectivos a cumprir.

O segundo capítulo, "Estado da Arte", pretende documentar o que foi criado até ao momento no tema em desenvolvimento neste relatório. Será apresentado o produto Healthy Talkie, as suas principais funcionalidades e onde está a ser utilizado

actualmente. Será descrita na generalidade a norma DICOM, que sendo um protocolo tão complexo e extenso se torna importante referir a sua influência na área da Saúde. Outro aspecto a ser apresentado neste capítulo é a análise de soluções semelhantes ao DCM4HT o que permitiu definir a vantagem competitiva deste produto face aos já existentes no mercado. Por fim, serão justificadas as opções tomadas quanto às tecnologias utilizadas para a implementação do sistema DCM4HT.

No terceiro capítulo, "Projecto e implementação do sistema DCM4HT", encontramos uma análise detalhada à aplicação desenvolvida, fazendo referência à sua arquitectura assim como a detalhes de implementação.

Por fim, as conclusões que podemos realçar deste projecto e possíveis melhoramentos a desenvolver no futuro.

Capítulo 2

Estado da Arte

Para a análise e desenho deste projecto é necessário avaliar as soluções já existentes no mercado e conhecer as funcionalidades que estas nos podem oferecer. Actualmente é possível encontrar algumas aplicações úteis como um motor de integração do protocolo DICOM num ambiente potencialmente heterogéneo. Desta forma serão descritos alguns produtos de integração, muito úteis para a definição da vantagem competitiva da nossa solução, DCM4HT - DICOM for Healthy Talkie.

2.1 Healthy Talkie

Muitas vezes, a informação nas instituições de Saúde flui por diversos sistemas, diferentes entre si no que toca ao método de comunicação ou linguagens suportadas. De modo a aproveitar e rentabilizar a informação que cada um possui, houve necessidade de interligar estes sistemas que à partida seriam incompatíveis, surgiram assim os sistemas de interface.

No entanto, pelo que podemos constatar, estas interfaces são desenvolvidas com um carácter específico e não genérico para um problema em causa. Podemos afirmar que estas interfaces são mal desenvolvidas e crescem exponencialmente, visto que temos uma interface específica para cada par de sistemas a interligar.

Se não estabelecermos um processo normalizado de comunicação, são formadas ilhas de informação onde encontramos a mesma informação repetida inúmeras vezes, causando problemas de gestão ou até uma elevada probabilidade de imprecisão na informação do doente, como também temos uma visão errónea e distorcida da realidade podendo dar origem a potenciais erros clínicos.

Os mais recentes sistemas de informação na Saúde suportam um ou mais protocolos *Standard* de comunicação de forma a responder às necessidades de partilha de

informação, como para reduzir o número de interfaces proprietárias para aplicações de diferentes sistemas ou cenários. No entanto, continuarão a existir discrepâncias em detalhes como o formato dos dados ou como o protocolo fora correcta ou incorrectamente interpretado.

A necessidade de um motor de integração genérico capaz de traduzir a informação entre os diferentes protocolos é reconhecida e é precisamente esta lacuna que o Healthy Talkie pretende eliminar.

O Healthy Talkie permite a partilha de informação entre todas as aplicações, como relatórios médicos, imagens ou prescrições, dentro de um sistema de informação hospitalar, facilitando as migrações entre sistemas rentabilizando o projecto e o investimento efectuado.

O sistema Healthy Talkie pode ser visto como um facilitador, validador / tradutor, assim como um roteador e um motor de integração.

Facilitador - Muitos sistemas suportam o protocolo DICOM mas não HL7, ou suportam HL7 mas não permitem a comunicação de objectos DICOM, ou suportam XML mas não HL7. Outras situações existem em que as versões suportadas dos mesmos protocolos diferem entre si fazendo com que os sistemas possam suportar transferências de dados apenas em campos privados do protocolo. Ainda podemos encontrar sistemas que não dispõem de nenhum mecanismo de comunicação da informação que possuem. A necessidade de um complemento a estes tipos de sistemas, que seja simples e pouco invasivo, mas que permita a troca de informação, é claramente uma mais valia.

Validador ou tradutor - Muitos produtos caracterizam-se pela sua conformidade a um determinado protocolo (ex: HL7), no entanto podem haver diferenças nas mensagem produzidas por estes, evidenciando a necessidade de um broker de comunicação entre estes.

Roteador - Um sistema pode necessitar de enviar informação para mais do que destinatário. Por vezes esta necessidade é condicional, consoante os agentes envolvidos ou a situação em causa. É portanto útil a existência de uma interface com a responsabilidade de decisão do encaminhamento da informação, segundo regras definidas.

Motor de Integração - A configuração é simplificada se uma aplicação enviar informação clínica estruturada segundo um determinado protocolo, para uma interface de comunicação, que por sua vez pode reencaminhar as mensagens para outros sistemas usando diferentes protocolos. Desta forma, todo o processo de manipulação e encaminhamento da informação é completamente transparente para a entidade emissora.

2.2 O protocolo DICOM

Digital Imaging and Communications in Medicine (DICOM) é um protocolo orientado à manipulação, armazenamento, impressão e comunicação da informação de Imagiologia.

Até meados dos anos 80 cada fabricante criava o seu próprio protocolo para troca de dados entre os seus produtos. Esta situação é altamente desvantajosa visto que requer um custo e esforço superiores para integração de equipamentos de outros fabricantes.

Este protocolo, criado pela ACR (*American College of Radiology*) e NEMA (*National Electrical Manufacturers Association*) em 1983, surge da necessidade de criar um método padronizado para transmissão de imagens médicas e informação associada, entre diferentes dispositivos.

Este *Standard* define o formato dos objectos que contêm os dados, define um protocolo de comunicação de rede baseado em TCP/IP (Figura 2.1), define os serviços que podem ser aplicados aos objectos, sendo ainda independente do fabricante, dispositivo, sistema operativo ou *software*.

O DICOM torna-se assim semelhante a outros formatos comuns como JPEG, que são criados e visualizados por diversas aplicações, sem apresentarem problemas de interoperabilidade. Outro exemplo são os servidores de *email*, capazes de enviar mensagens para qualquer outro servidor de *email* independentemente do *software* ou *hardware* em causa.

DICOM é um *standard* aberto, caracterizado pela sua complexidade, sendo já composto por mais de duas mil páginas [1].

Alguns dispositivos que utilizam esta norma, são:

- Modalidades (CT, MR, CR)
- Sistemas PACS (*Picture Archiving and Communitation System*)
- Estações de *reporting* e *post-processing*
- *Printers* e *print servers*

2.2.1 Serviços DICOM

Os serviços DICOM estão organizados em dois grupos: serviços compostos ou normalizados. Os serviços compostos estão estruturados de modo a manterem a compatibilidade com as versões anteriores do *Standard* ACR-NEMA. Estes foram orientados desde início para os serviços *storage* (C-STORE), *query* (C-FIND), *retrieval* (C-GET) e *transfer* (C-MOVE) de imagens. No entanto os serviços compostos também são úteis para outros tipos de informação, como os relatórios [2].

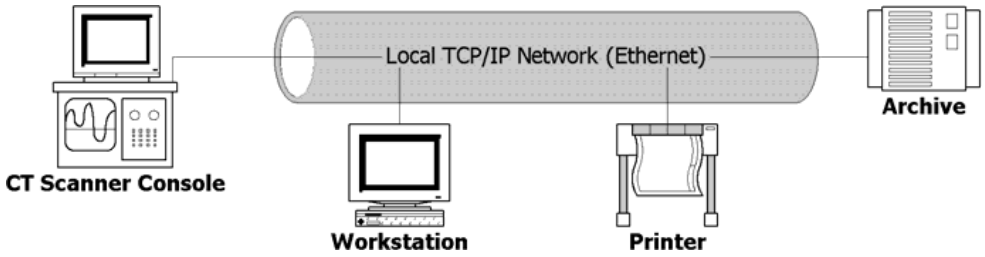


Figura 2.1: Exemplo de uma rede DICOM

É de salientar o facto do grupo dos serviços compostos não disponibilizar um serviço de *update*. Esta omissão é intencional. Os arquitectos do *standard* original ACR-NEMA, concluíram que ao omitir o serviço de *update* conseguiriam reduzir a possibilidade de alteração indevida de uma imagem armazenada. Assim, os serviços compostos são optimizados para a transferência de imagens.

Os serviços normalizados foram desenhados para oferecerem funcionalidades de gestão de informação. São então orientados para a informação representante de uma única entidade do mundo-real, pelo que, os serviços compostos são orientados a documentos (imagens) que contenham informação com origem em um ou mais entidades do mundo-real (como *pixel data*, número de identificação do equipamento ou do doente). Os serviços normalizados disponibilizam operações básicas de gestão da informação como: *create* (N-CREATE), *delete* (N-DELETE), *update* (N-SET) e *retrieve* (N-GET).

Entidades do mundo-real (como imagens, procedimentos, relatórios) são representados dentro da semântica DICOM através de *templates*. Estes *templates* estão documentados nas descrições dos objectos de informação DICOM (IODs) [3].

2.2.2 Modelo de Informação

DICOM tem um *modelo de informação* que o diferencia de outros protocolos, em particular da norma HL7 v2.x muito conhecida na área da saúde para a comunicação de informação do doente.

Os objectos de informação DICOM são definições para a informação a ser transferida. Ou seja, cada tipo de imagem clínica tem características específicas, logo também será constituído por um objecto de informação diferente. Por exemplo uma imagem CT requer um conjunto de informações existente no cabeçalho da imagem diferente das existentes numa imagem de ultrasom ou de uma imagem de oftalmologia. Estes objectos de informação podem ser vistos como um template com uma identificação única e que estão registados pela NEMA [4].

Objectos de informação são também uma parte importante das Classes Par Serviço-Objecto (SOP). Um exemplo de uma Classe SOP é, *CT Storage SOP Class*, que permite transferir as imagens CT.

Apesar de existir uma ideia generalizada de associação de objectos DICOM a imagens, é importante salientar que uma lista de marcações de doentes a ser enviada para uma impressora é também um objecto DICOM definido por um template específico [3].

2.2.3 Objectos DICOM

Uma rede DICOM contém informação sobre um número de diferentes doentes. Cada doente pode ter sido examinado através de múltiplas imagens digitais.

Depois de um exame estar concluído o *software* do *scanner* cria um conjunto de imagens, chamado **estudo**.

Por vezes um estudo é composto por imagens que correspondem a diversos ângulos de uma zona do corpo humano (vista) ou por diversos tipos de aquisição. Cada vista ou aquisição é denominado **série**.

Cada série é constituída por um número de imagens ou camadas onde cada camada é um **Objecto de Informação DICOM**.

Quando um exame é impresso, existe uma porção de informação sobre o doente e respectivo estudo que é apresentada nos cantos de cada imagem. Assim, basta analisar a impressão para conhecer o respectivo doente e qual o estudo no qual a imagem está inserida. Os Objectos de Informação DICOM seguem uma ideia semelhante. Cada imagem ou relatório DICOM contém informação suficiente para especificar o doente e o estudo ao qual pertence.

Para cada tipo de Objecto de Informação, existe uma Definição do Objecto de Informação (IOD - *Information Object Definition*) que especifica qual o conjunto de dados que este objecto deverá conter e como é que a informação deverá ser armazenada [5].

Cada elemento da informação de um objecto DICOM é uma **Tag DICOM**. Como é apresentado na Figura 2.2, uma Tag DICOM consiste num nome do campo e um valor. Os nomes dos campos são numéricos e ordenados de modo crescente. Uma tag contém essencialmente, informação como: "Nome = Mourinho, José", "Data de Nascimento = 23 Out 1954", etc. Mesmo as imagens estão inseridas numa *tag*.

Um IOD para imagens CT (Figura 2.3), especifica o conjunto de *tags* que devem estar presentes em cada imagem CT, assim como o valor válido para algumas *tags*. Por exemplo, uma imagem CT deve conter sempre o campo *Modality* com o valor "CT".

```
(0008,0020) DA    [20000720]                                #     8,  1  StudyDate
(0008,0021) DA    [20000720]                                #     8,  1  SeriesDate
(0008,0022) DA    [20000720]                                #     8,  1  AcquisitionDate
(0008,0023) DA    [20000720]                                #     8,  1  ContentDate
(0008,0030) TM    [231035.000000]                           #    14,  1  StudyTime
(0008,0031) TM    [231258.000000]                           #    14,  1  SeriesTime
(0008,0032) TM    [231441.000000]                           #    14,  1  AcquisitionTime
(0008,0033) TM    [231528.000000]                           #    14,  1  ContentTime
(0008,0050) SH    [N188322]                                 #     8,  1  AccessionNumber
(0008,0060) CS    [CT]                                      #     2,  1  Modality
```

Figura 2.2: DICOM Tags

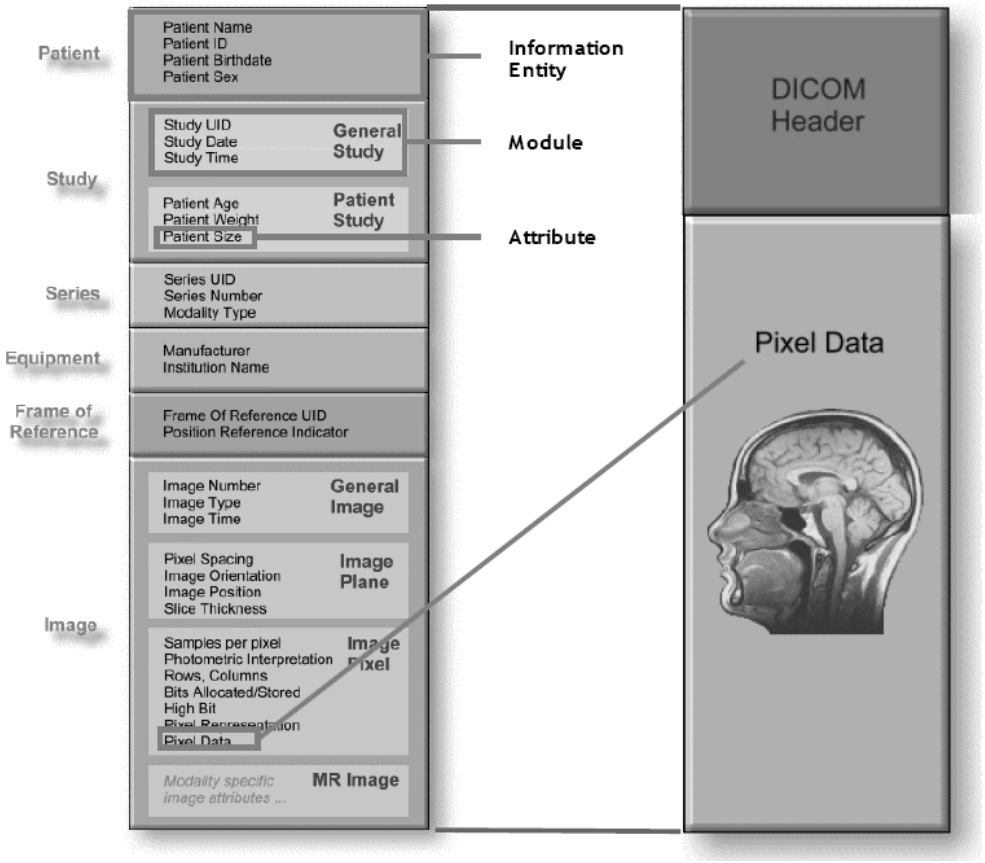


Figura 2.3: IOD - Estrutura Organizacional

2.2.4 Transporte

O protocolo DICOM é robusto no que toca ao transporte de dados (Figura 2.4). Um dispositivo em conformidade com a norma DICOM não envia apenas um objecto. Terá de estabelecer uma negociação com a entidade receptora para assegurar que o tipo de objecto transmitido, assim como o tipo de serviço a ser efectuado sobre este, são suportados pelo destinatário. No caso do destinatário não suportar, por exemplo, o tipo de codificação da informação, pode requerer ao emissor um outro tipo de codificação.

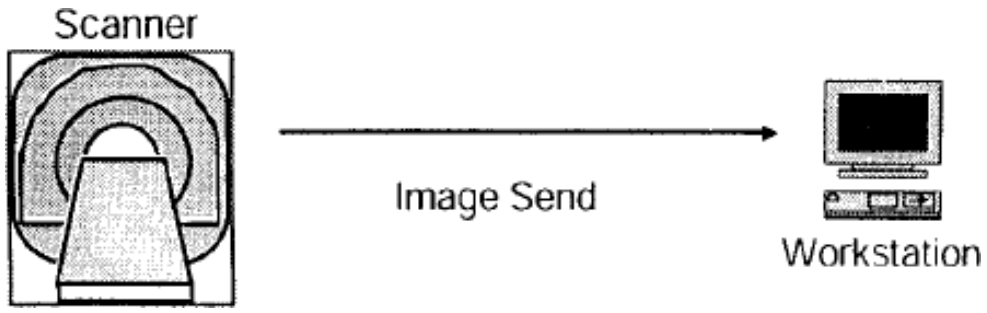


Figura 2.4: Um *scanner* inicia a rotina de transferência de imagens pela fase de associação. São negociados vários detalhes durante o estabelecimento da associação, para que a *workstation* esteja preparada para processar a imagem que está para receber.

2.2.4.1 Negociação

Esta negociação é denominada de *association establishment* (figura 2.5), que para além de negociar o tipo de serviço também o faz para a *transfer syntax*. A *transfer syntax* não é mais do que um género de codificação da mensagem trocada (ordem dos bytes e método de compressão). Um determinado dispositivo pode suportar a *transfer syntax* padrão (*Implicit Little Endian*) ou uma codificação específica com compressão JPEG que permita transmitir a informação num menor período de tempo [6].

As funcionalidades suportadas por cada dispositivo deverão ser explicitamente detalhadas num documento *DICOM Conformance Statement*, incluindo as classes SOP e as *transfer syntaxes* suportadas. Este documento permite determinar se duas máquinas são compatíveis entre si, sendo utilizados não só pelos utilizadores directos ou compradores, mas também pelas pessoas que trabalham com estes sistemas, como integradores de sistemas ou *testers*.

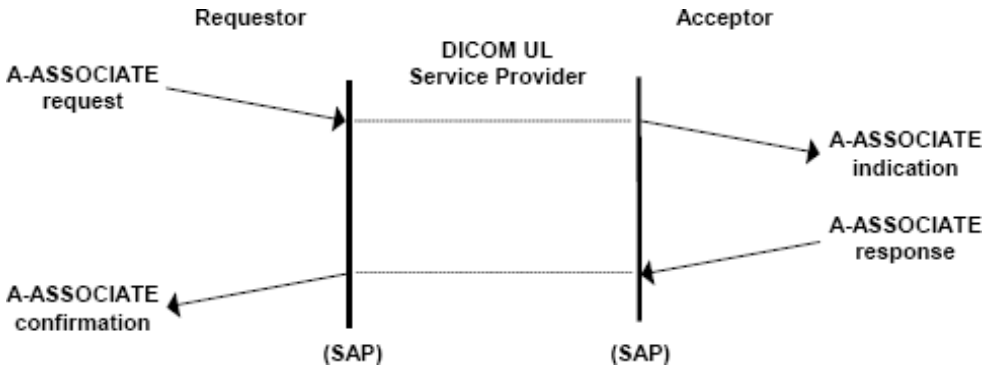


Figura 2.5: Pedido de associação

2.3 A Solução DCM4HT

O módulo DICOM do Healthy Talkie é o responsável pelo suporte à comunicação de imagens digitais clínicas e informação relacionada, disponibilizando um conjunto de serviços de manipulação, transporte e armazenamento de objectos DICOM v3.0.

2.3.1 Funcionalidades Principais

Este módulo foi desenvolvido segundo os requisitos levantados, que propõem uma solução a um conjunto de cenários de uso habituais (Figuras 3.1, 3.2, 3.3, 3.4 e 3.5) num panorama clínico. Estas funcionalidades estão ordenadas pela sua prioridade e serão apresentadas numa perspectiva de alto nível de implementação.

- DICOM Verification
- Storage SCU/SCP
- Separação / União da imagem clínica num objecto DICOM
- Storage Commitment SCU/SCP
- Query Retrieve SCU/SCP
- Modality Worklist

2.4 Outras soluções para integração de sistemas

2.4.1 Mirth Project

O Mirth Project constitui um projecto Open Source que permite a troca de informação HL7 e DICOM, entre sistemas e aplicações, como também operações de modificação e encaminhamento destas mensagens segundo regras pré-determinadas.

A criação de uma interface para sistemas já existentes é conseguida através de uma aplicação baseada na web com um *wizard* que permite associar diversas aplicações aos componentes do motor Mirth.

O suporte aos protocolos HL7 e DICOM torna-se essencial para que o Mirth se identifique como um intermediário capaz de transformar as mensagens trocadas por sistemas díspares.

- Suporte para mensagens HL7, X12, NCPDP, DICOM, EDI, XML;
- Protocolos de comunicação MLLP, HTTP, HTTPS, FTP, SFTP, JDBC, PDF, RTF, SMTP, POP3, JMS, Files, Web services;
- Base de dados MySQL, PostgreSQL, Oracle, Microsoft SQL Server, Microsoft Access, Apache Derby.

O Mirth Project é baseado na biblioteca open source Dcm4che e suporta os serviços:

- Leitura e escrita de mensagens DICOM;
- Adicionar, modificar e remover elementos do header DICOM;
- Parser DICOM para XML;
- Parser XML para DICOM;
- Encaminhamento de mensagens DICOM.

Os serviços a desenvolver no futuro são:

- Dcmsend
- Dcmrcv
- Dcmmwl
- Dcmqpwl
- Dcmqr

2.4.2 MEDxConnect

Medical Message Mediator (M3) é a principal aplicação da solução *MEDxConnect* da Compressus Inc. Esta suporta HL7 e DICOM entre outros protocolos, usando o módulo M3 o protocolo de comunicação e a semântica de cada um dos sistemas a si associados para controlar o fluxo de imagens, relatórios, mensagens, informações demográficas do paciente e outras informações relevantes ao diagnostico e planeamento entre sistemas heterogéneos.

O M3 realiza o mapeamento dos elementos da mensagem e facilita a interoperabilidade entre diferentes aplicações criando um sistema virtual integrado. Com esta solução é possível gerir, regras de encaminhamento de estudos sobre o paciente, distribuição dos exames e notificação da criação e entrega de relatórios.

A aplicação M3 suporta DICOM, HL7, XML, SSL, TLS, assim como outros *standards* para assegurar a conformidade segundo o HIPAA.

O *MEDxConnect Virtual Worklist* é um produto que corre sobre o servidor MEDxConnect M3 aumentando as suas funcionalidades. O principal objectivo da Virtual Worklist em conjunto com o M3 é de disponibilizar e armazenar imagens de diagnóstico nos arquivos DICOM, permitir o acesso à *work list* de um paciente em qualquer ponto da rede, assim como a capacidade de encaminhar as imagens consoante regras definidas pelo utilizador.

Os serviços suportados por esta solução incluem:

- *Retrieve* e *store* de imagens em arquivos DICOM;
- Criação e acesso da *work list* do paciente;
- Encaminhamento (*route*) de imagens DICOM baseado em regras criadas pelo utilizador.

2.4.3 Emageon

Emageon é constituído por múltiplas ferramentas que visam disponibilizar aos especialistas de cuidados médicos, meios para gerir, aceder e visualizar as informações das diversas especialidades em qualquer ponto da instituição. O gestor de arquivos do Emageon suporta um vasta gama de operações DICOM que podem ser necessários para um arquivo, assim como operações não suportadas por outros fabricantes. Como este gestor permite ao administrador do sistema a definição das classes suportadas pela AE local (Application Entity), este pode facilmente suportar todas as modalidades DICOM para serviços query/retrieve e storage como CT, CR, MR, US, NM, DR, SC, ou outras.

Este arquivo suporta os serviços:

- Dicom *query/retrieve* (C-MOVE SCP, C-GET SCP,C-FIND SCP);
- *Storage* (C-STORE SCU, SCP);
- *Verification* (C-ECHO as SCU and SCP);

Suporta também as seguintes sintaxes

- *Little endian implicit value representation;*

- *Little endian explicit value representation;*
- *Big endian explicit value representation;*
- JPEG *lossless* e JPEG *lossy*

2.4.4 IC2E Collaboration Suite

IC2E Collaboration Suite é uma solução que visa a integração, comunicação, colaboração e troca de informação e imagens clínicas do paciente para diversas instituições.

Esta aplicação suporta a troca de informação em tempo real, informações como imagens, formulários, relatórios, com origem em sistemas HIS, RIS, CIS e PACS díspares. Uma das ferramentas desta aplicação é o IC2E Image ou DICOM Router, que permite a interligação de múltiplas modalidades a um sistema PACS através do protocolo DICOM. Mais concretamente, este suporta comunicações DICOM Muitos-para-Muitos, 1-para-Muitos, Muitos-para-1, dependendo apenas das regras de encaminhamento.

Ao suportar múltiplos standards, as soluções IC2E's estão em conformidade com as normas XDS e HIPAA

2.4.5 Hipax Workstation 4

Devido à sua arquitectura modular, o sistema Hipax Workstation 4 pode ser facilmente adaptado a diferentes ambientes dependendo da sua função. Alguns dos módulos que constituem esta aplicação listam-se de seguida:

- DICOM Worklist: Recepção automática de worklists de sistemas HIS e RIS
- DICOM Print: Serviço de impressão da imagem DICOM
- DICOM Communication: DICOM Storage, Query/Retrieve
- DICOM Email: Envio e recepção de imagens via Email independentemente do fabricante
- Base Module Standard: módulo base que inclui funções de processamento de imagem;
- Base Module "light": módulo base que inclui funções restritas para processamento de imagem;
- CR/DR Connection: recolha directa de imagens dos sistemas CR/DR de diferentes fabricantes;

- X-ray Journal: armazenamento de informação de Raios-X
- Vidar X-ray Digitizing: ligação a digitalizadores de Raios-X Vidar
- TWAIN X-ray Digitizing: ligação a qualquer digitalizador com controladores TWAIN
- Video Interface: recolha de imagem únicas ou sequências de imagens de sistema de vídeo

Com o módulo de comunicação DICOM, o sistema Hipax pode receber e enviar imagens através de um canal de transporte TCP/IP, assim como receber imagens de modalidades que respeitem o protocolo DICOM assim como modalidades de outros fabricantes. O envio das imagens pode ser realizado com diferentes modos de compressão:

- JPEG 2000 lossless
- JPEG, JPEG 2000

Os comandos DICOM suportandos são:

- Selecção do paciente (Find/Query SCU)
- Mover imagens (Move SCU)
- Envio de imagens (Store SCU)
- Recepção de imagens (Store SCP)
- Query/retrieve
- Armazenamento automático das imagens recebidas, numa base de dados
- Comunicação com a rede, assim como outras estações externas à rede
- Encriptação das imagens e documentos para transmissão externa
- Diferente modos de compressão, com factor de 3 a 20
- Suporte de múltiplas ligações em simultâneo

2.5 Comparação dos produtos mencionados

Este comparativo apresenta as principais funcionalidades de cada um dos produtos supra mencionados e destaca as diferenças entre estes. O levantamento das funcionalidades foi baseado nos *Conformance Statements*, de cada solução e desta

forma, dado que os sistemas Mirth Project e IC2E Collaboration Suite não disponibilizam este documento, não serão contemplados no comparativo. O comparativo está apresentado nas tabelas 2.1, 2.2, 2.3.

Tabela 2.1: Serviços DICOM V3.0 suportados como SCU (Service Class User)

	MEDxConnect	Emageon	Hipax Workstation 4
Storage	X	X	X
Structured Reporting	X	X	
Query/Retrive	X		X
MPPS	X	X	
MPPS Notification	X		
Modality Worklist		X	X
Grayscale Print			X

Tabela 2.2: Serviços DICOM V3.0 suportados como SCP (Service Class Provider)

	MEDxConnect	Emageon	Hipax Workstation 4
Verification	X	X	X
Storage	X	X	X
Structured Reporting	X	X	
Query/Retrive	X	X	
Storage Commitment	X	X	
MPPS	X	X	

Tabela 2.3: *Transfer Syntaxes* suportadas

	MEDxConnect	Emageon	Hipax Workstation 4
Implicit VR Little Endian	X	X	X
Explicit VR Little Endian	X	X	
Explicit VR Big Endian		X	
Deflated Explicit VR Little Endian		X	
JPEG Baseline Process 1	X	X	
JPEG Extended Process 2	X		
JPEG Extended Process 4		X	
JPEG Lossless Process 14	X	X	

2.6 Tecnologias de desenvolvimento

As tecnologias usadas para o desenvolvimento foram necessariamente as utilizadas pela equipa Healthy Talkie. Ou seja, Java 6 com o ambiente de desenvolvimento Eclipse e controlo de versões através de repositório CVS.

A razão para a escolha da linguagem Java na implementação do Healthy Talkie, deve-se, em primeiro lugar, ao facto desta oferecer uma agilidade de desenvolvimento consideravelmente satisfatória para as necessidades que a equipa apresenta diariamente, e, em segundo lugar, devido à tecnologia ser pouco invasiva para o equipamento que comporta o produto, bastando apenas que este tenha instalado o JRE 6.

2.7 Conclusões

Após uma análise detalhada das funcionalidades dos sistemas actualmente disponíveis, que têm como objectivo a integração de dispositivos de diferentes fabricantes, com diferentes especificações, para suporte de serviços DICOM, é possível retirar algumas ilações sobre o posicionamento e a orientação que o Healthy Talkie deverá tomar.

Todos os produtos acima descritos, possuem algumas características semelhantes ao módulo que se pretende desenvolver, mas é possível destacar alguns para uma análise mais aprofundada.

O Mirth Project, um sistema Open Source, é talvez a solução mais interessante das cinco analisadas. É muito semelhante ao sistema Healthy Talkie, dado os protocolos de comunicação por ele suportados, como o HL7, XML e DICOM. Foi desenvolvido com base na biblioteca Dcm4che, que pode também vir a ser útil na implementação do módulo DICOM para o Healthy Talkie. O grande defeito apresentado por este projecto consiste na falta de documentação disponibilizada.

Das soluções comerciais analisadas podemos salientar a MEDxConnect e a Emageon, que para além de oferecerem uma gama bastante completa de funcionalidades, disponibilizam toda a documentação de especificação da norma DICOM V3.0, necessária para a sua integração com qualquer outro sistema.

Capítulo 3

Projecto e implementação do sistema DCM4HT

Este capítulo é dedicado aos detalhes de mais baixo nível da solução DCM4HT. Serão descritos os requisitos que foram acordados desenvolver com a instituição Siemens MED, apresentando a arquitectura escolhida e expondo alguns detalhes de implementação.

3.1 Especificação de Requisitos

A Siemens MED foi a entidade que definiu os requisitos para esta aplicação.

Inicialmente a perspectiva global desta aplicação foi descrita da seguinte forma pela Siemens: *"O objectivo deste projecto é o de desenvolver a capacidade de recepção, processamento e envio de dados em protocolo DICOM v3.0, como uma nova funcionalidade do software de interface de comunicação Healthy Talkie (excepto manipulação de imagem), o qual apenas suporta HL7. Como resultados finais, espera-se que a ferramenta permita por exemplo a validação dos dados constantes numa imagem DICOM, recorrendo a uma mensagem HL7 do tipo QRY, à qual um sistema de identificação de pacientes responderá com os dados completos de um paciente, devendo-se posteriormente preencher ou corrigir os dados da imagem DICOM original."*

Verificou-se que os requisitos que esta aplicação deveria cumprir sofreram pequenas alterações ao longo do tempo. O protocolo DICOM tem diferentes funcionalidades e particularidades que geralmente são desconhecidas, mesmo por especialistas

na área da Imagiologia. Assim, quando se analisa o protocolo com um maior detalhe é que se ganha conhecimento suficiente para definir realmente o que é viável implementar e de que forma o conseguimos atingir, no tempo que temos disponl.

A metodologia de desenvolvimento adoptada pela equipa do Healthy Talkie é o *Scrum*. Os requisitos que são levantados para todo o *software* estão definidos num documento denominado *product backlog*, que são apresentados em forma de tarefas durante o processo de desenvolvimento. Esta lista de tarefas é ordenada por prioridade, onde a tarefa mais importante se encontra no topo da lista. As tarefas têm também uma estimativa do esforço despendido em dias, o que permite ter uma ideia do período de tempo necessário para concluir o projecto.

Inicialmente, a lista de tarefas decidida para o projecto revelou-se demasiado extensa porque o somatório dos dias estimados para cada tarefa era superior ao tempo que ainda restava para a implementação do projecto. Foi necessário dividir esta lista em duas *Releases* da aplicação:

- *Release Beta* (Tabela 3.1)
- *Release Final* (Tabela 3.2)

Estes são os requisitos pretendidos para a solução DCM4HT, ou seja, as funcionalidades que deverão ser adicionadas ao Healthy Talkie, a médio ou longo prazo, não estão definidas exclusivamente no âmbito deste projecto.

Como é referido, o DCM4HT deve ser totalmente flexível e configurável por meio de ficheiros XML, desde a definição do próprio protocolo DICOM quanto à estrutura das mensagens que podemos receber ou enviar, até à definição dos serviços DICOM que podemos suportar. A vantagem deste sistema de configuração, é que no caso do protocolo DICOM sofrer alterações ao longo do tempo, o DCM4HT pode ser adaptado facilmente às novas exigências, não sendo necessário efectuar estas alterações programaticamente.

Sos os cenários de uso que pretendemos que o DCM4HT venha a suportar com sucesso a médio ou longo prazo, apresentados nas Figuras 3.1 a 3.5 .

DICOM *Worklist*, apresentado na Figura 3.1, é uma funcionalidade que permite receber um pedido de uma modalidade (equipamento externo). Este pedido pode ser por exemplo: Todos os exames marcados para o dia de hoje, entre as 14h00 e as 16h00. Ao recebermos um pedido deste tipo, devemos recorrer a uma fonte de informação como uma base de dados SONHO, muito comum nos sistemas de informação hospitalares, e devolver os registos médicos encontrados. Esta funcionalidade permite evitar a re-introdução de dados dos pacientes evitando erros de escrita.

O DCM4HT permite que os exames sem marcação (sem informação previamente disponível sobre o doente), Figura 3.2, sejam verificados no que toca à integridade da

Tabela 3.1: Requisitos especificados inicialmente para o DCM4HT (*Release Beta*)

Descrição	Estimativa (dias)
Serviço DICOM C-ECHO, para verificar se outro sistema está disponível (idêntico ao ping)	5
Adaptar as classes que suportam peers.xml e peers.dtd, por forma a suportar peers do tipo DICOM	7
DICOM C-STORE como SCP/SCU (capacidade de enviar e receber objectos DICOM)	10
Alterar as classes Receive e Send para que separe/junte a *tag* que contém a imagem dos restantes dados de uma mensagem DICOM, a coloque no sistema de ficheiros e introduza uma *tag* com uma referência para a imagem	4
Definição do ficheiro XML (protocols.xml) que define as regras de construção de mensagens DICOM com a codificação *Implit Little Endian*	15
Derivar a classe translate e untranslate que converte um objecto DICOM em informação neutra, intepretável pelo Healthy Talkie com a ajuda do ficheiro de configuração protocols.xml	12
Implementar DICOM *storage commitment* (N-ACTION) como SCP/SCU alterando as classes send e receive	4
Implementar *Query/Retrieve* (SCU/SCP), capacidade de inquirir sobre a existência de um estudo DICOM, e solicitar o seu envio	5
Implementar *Modality Worklist* – capacidade de criar uma lista de tarefas para uma determinada modalidade a partir dos dados existentes no RIS (radiology information system)	6
RELEASE Beta	**68**

Tabela 3.2: Requisitos especificados inicialmente para o DCM4HT (*Release Final*)

Descrição	Estimativa (dias)
Suporte de ficheiro DicomDIR, que permite a utilização de suportes amovíveis	7
Garantir que são aceites imagens DICOM de tamanhos muito elevados, sem comprometer o desempenho do Healthy Talkie	2
Suportar *Transfer syntax Explicit Big Endian*	6
Suportar *Transfer syntax Explicit Little Endian*	7
Testar o cenário de utilização "Exame na hora"	3
Testar o cenário de utilização "Exame marcado"	3
Testar o cenário de utilização "Anonimização"	3
Testar cenário de utilização autorização	3
Implementar Dicom MPPS - *Modality Performed Procedure Step*	6
Print management, capacidade de imprimir uma imagem Dicom e gerir uma fila de impressão	18
DICOM *conformance statement* para o Healthy Talkie	6
GUI para o configurador das mensagens DICOM no Healthy Talkie	13
RELEASE Final	**59**

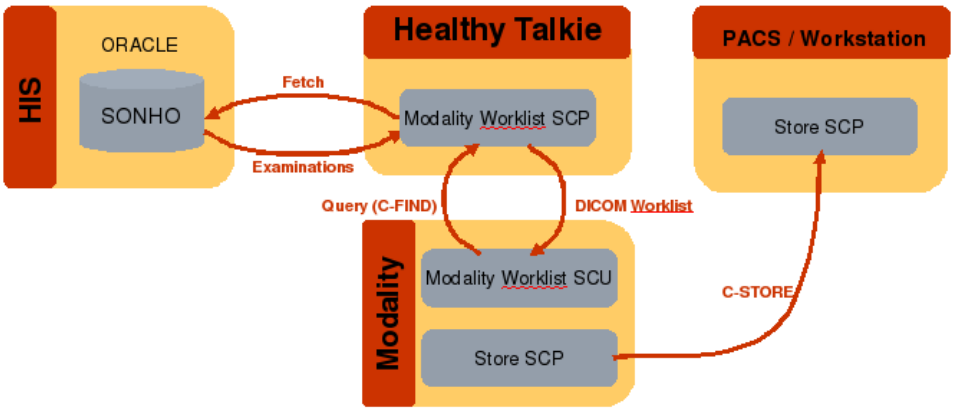


Figura 3.1: Cenário de uso: *Modality Worklist*

informação que contêm. Sempre que a consistência da informação não é verificada, é enviada uma notificação para um endereço de *email* previamente definido.

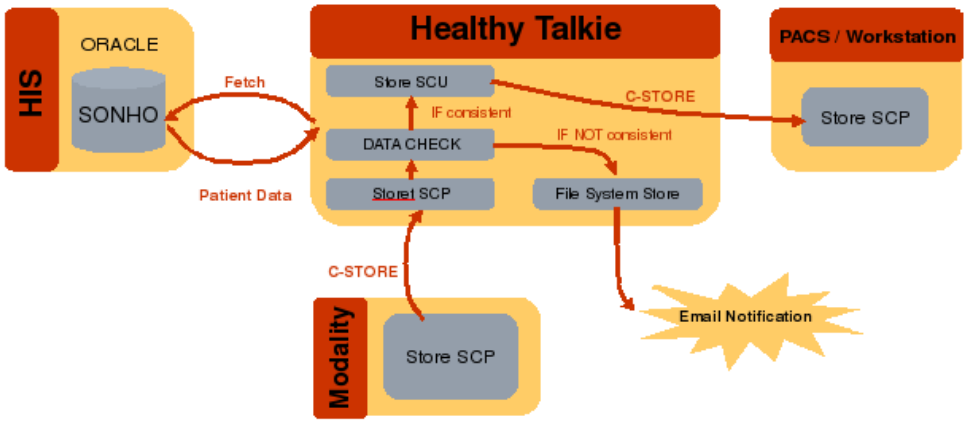


Figura 3.2: Cenário de uso: Exame sem marcação

O serviço MPPS - *Modality Performed Procedure Step* (Figura 3.3)- permite que uma modalidade comunique ao Healthy Talkie cada etapa concluída num exame a um doente e que posteriormente estes dados sejam armazenados numa base de dados do RIS.

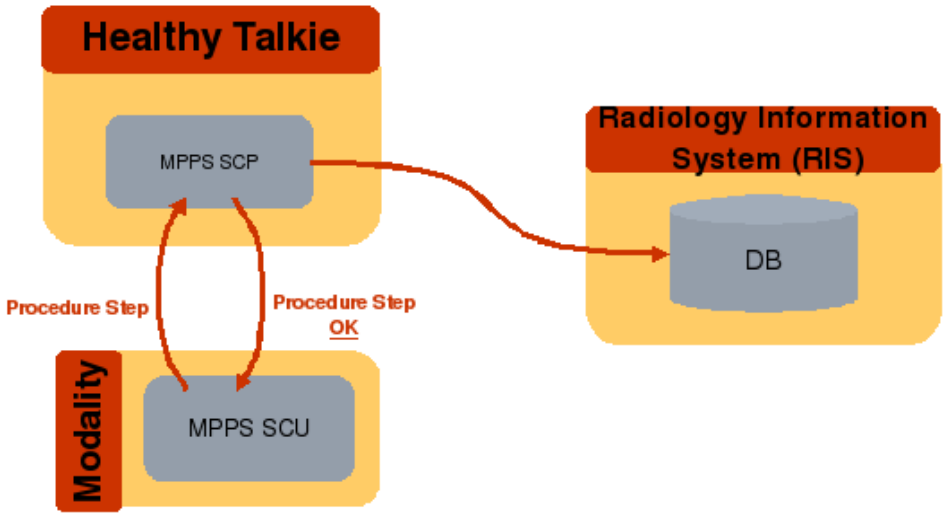


Figura 3.3: Cenário de uso: *Modality Performed Procedures Steps*

Com o serviço de anonimização do DCM4HT (Figura 3.4) conseguimos extrair toda a informação sensível de uma mensagem DICOM, isto é, os dados pessoais do doente.

Figura 3.4: Cenário de uso: anonimização de dados

O serviço de autorização (Figura 3.5) é um sistema de segurança da solução DCM4HT com o objectivo de garantir que as comunicações só são efectuadas com equipamentos registados no Healthy Talkie. Todavia todos os equipamentos não registados que tentarem comunicar com o Healthy Talkie serão imediatamente rejeitados. Este serviço de autorização está implicitamente desenvolvido na fase de associação que antecede uma transferência em DICOM.

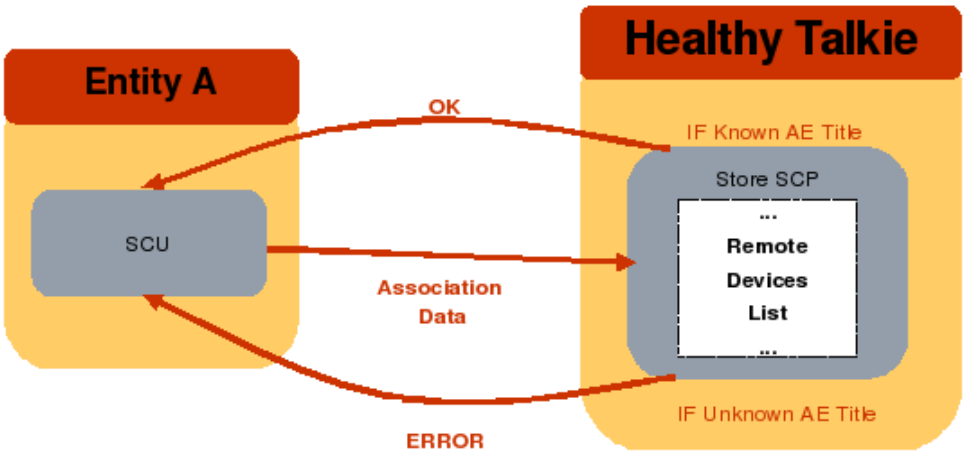


Figura 3.5: Cenário de uso: reconhecimento e autorização de DICOM SCU

3.2 Arquitectura

Um dos requisitos para este projecto é a utilização da arquitectura actual do Healthy Talkie. Pretende-se que todas as funcionalidades adicionadas sejam verdadeiramente integradas no Healthy Talkie, de forma a mantermos uma implementação mais limpa e inteligente, reutilizando o código existente sempre que possível. Assim, a arquitectura da solução DCM4HT que será exposta nesta secção é fundamentada na totalidade no produto Healthy Talkie.

Tal como apresentado na Figura 3.6, o DCM4HT é composto por três módulos principais:

- *RWorker*
- *Route*
- *SWorker*

O mo *RWorker* é o responsável pela recepção de mensagens provenientes de diferentes protocolos de comunicação e pelo tratamento da mensagem de modo a ser interpretável pelo Healthy Talkie.

O mo*Route* baseia-se em regras definidas pelo utilizador para reencaminhar a mensagem recebida para um determinado destino segundo um determinado protocolo.

O mo *SWorker* manipula a mensagem segundo as necessidades do destinatário e envia-a através de um protocolo de comunicação.

Podemos verificar nas Figuras 3.7 e 3.8 que estes módulos são muito semelhantes entre si e que apresentam funcionamentos inversos. O primeiro recebe uma

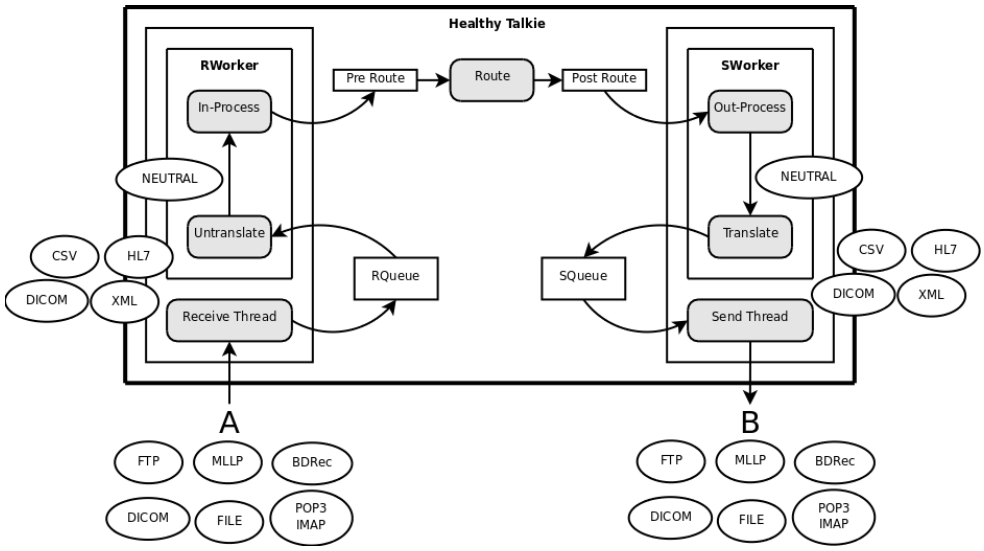


Figura 3.6: Arquitectura geral do DCM4HT

mensagem em DICOM/HL7/CSV/XML e transforma os seus dados em informação neutra, isto é, sem formatação e independente de qualquer protocolo. Só quando possuímos esta mensagem no estado neutro é que a podemos processar, efectuar correcções, adicionar ou remover informação, o que significa ser perceptível pelo Healthy Talkie.

O segundo módulo faz precisamente a operação inversa, isto é, dispõe de um conjunto de dados em formato neutro e de um destino que é atribuído pelo *Route*, traduz a informação para o protocolo de destino e comunica a mensagem construída para uma entidade externa.

Analisando o *RWorker* encontramos três sub-módulos: o *Receive* (Figura 3.9), o *Untranslate* (Figura 3.10) e o *Process* (Figura 3.11). É importante referir que existe um sub-módulo denominado de *Process* tanto no *RWorker* como no *SWorker* que são distinguidos pelo nome *In-Process* e *Out-Process* respectivamente.

O *Receive* contém todas as funções para comunicação de rede baseadas em TCP/IP (excepto no caso *file*). As comunicações são configuráveis, ou seja, apenas as ligações que foram adicionadas ao ficheiro de configuração que contém os dados dos equipamentos externos (IP e Porta) é que serão iniciadas pelo *Receive*. O resultado desta comunicação é uma mensagem que transporta a informação a ser posteriormente transformada para formato neutro.

O processo *Untranslate* recebe uma mensagem codificada de acordo com o protocolo da origem (DICOM, HL7, CSV, XML) e extrai a informação útil através de regras de *parse* definidas num ficheiro de configuração pelo utizador, para uma tabela de integração em formato neutro (ver Figura 3.16). Só depois desta mensagem

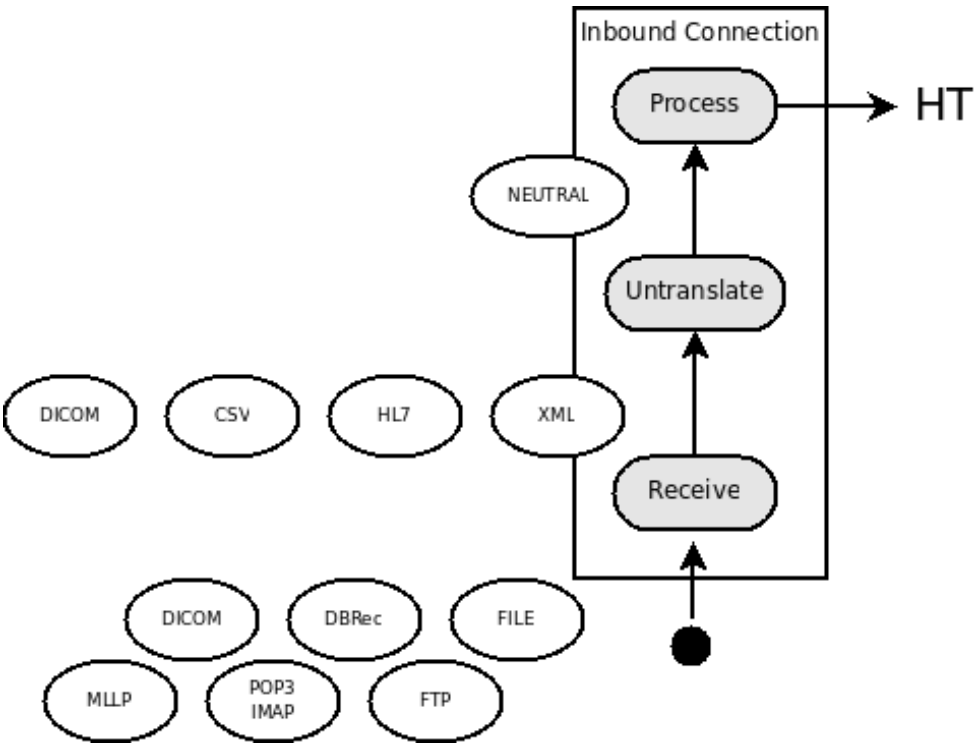


Figura 3.7: Sequência de recepção

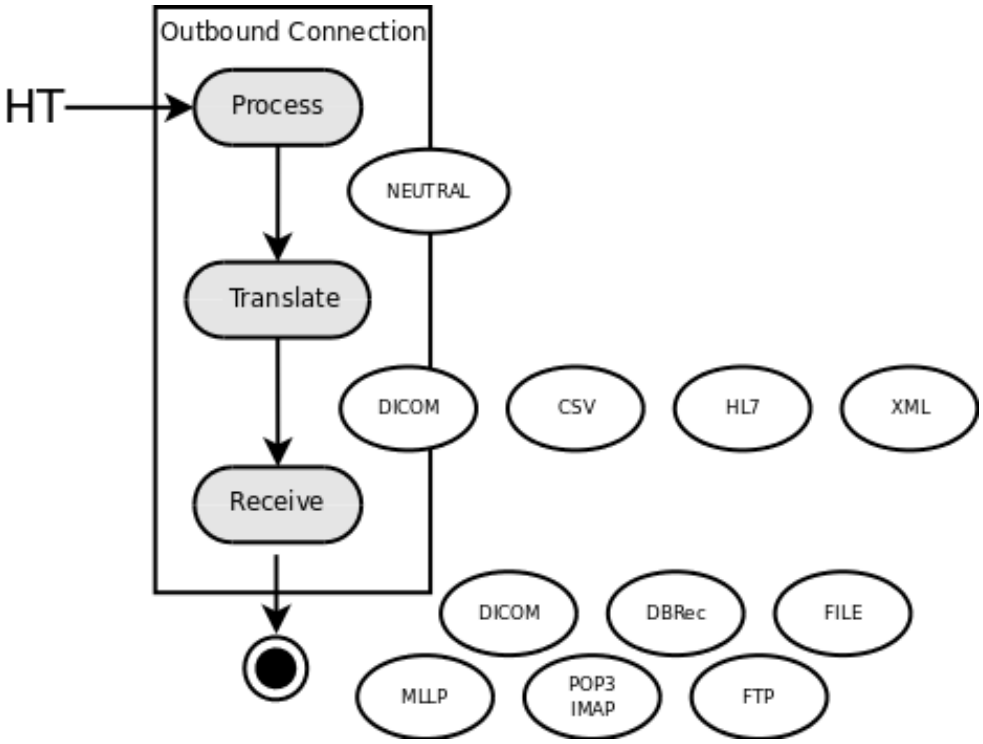


Figura 3.8: Sequência de envio

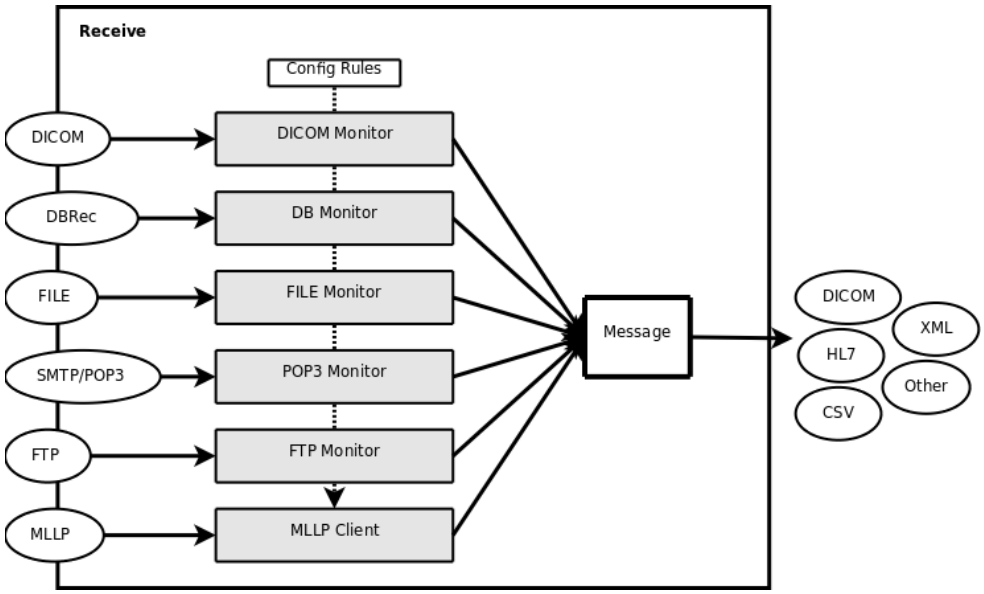


Figura 3.9: Modelo do processo *receive*

se encontrar devidamente dividida e normalizada dentro da tabela de integração é que podemos avançar para a primeira fase de processamento, *In-Process*.

O *In-Process* permite complementar a informação da tabela de integração com dados oriundos de outra ligação. Através de regras ou funções é possível fazer um *bind* da informação de origem com a de destino. Por fim, é possível guardar a informação gerada após este processo.

Com o módulo *Route*, apresentado na Figura 3.12, tornou-se possível o encaminhamento de uma mensagem para um ou mais destinos. As regras de encaminhamento são definidas num ficheiro XML, sendo de fácil acesso para o utilizador. Um dado importante neste processo é que para cada destino gerado pelo *Route* vamos ter uma *thread SWorker* diferente.

O módulo *Out-Process* (Figura 3.13) é muito semelhante ao módulo *In-Process* visto que é composto pelas mesmas funções de *Fetch*, *Bind* e *Store*. A diferença é que nesta fase, as alterações efectuadas aos dados existentes na tabela de integração pretendem satisfazer necessidades específicas do destinatário da mensagem. Isto significa que a mensagem é sujeita a diferentes alterações consoante o seu destino.

A informação contida na tabela de integração encontra-se totalmente preenchida, manipulada e pronta para ser traduzida para o protocolo do equipamento de destino. O módulo *Translate* (Figura 3.14) lê a informação da tabela de integração e consoante as regras do protocolo definidas num ficheiro XML constrói uma mensagem DICOM, HL7, CSV ou XML.

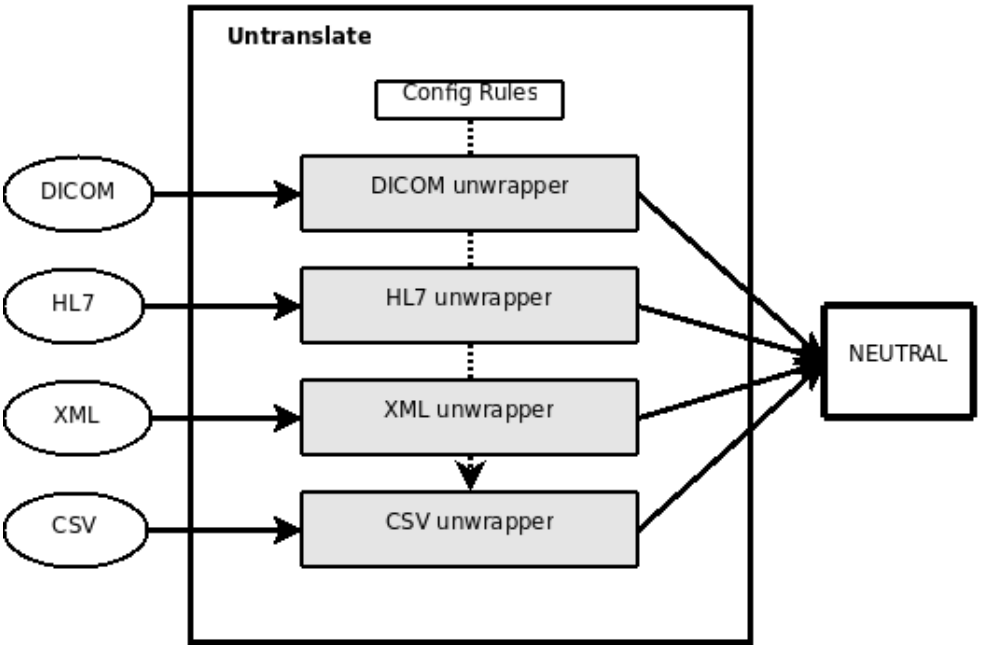


Figura 3.10: Modelo do processo *untranslate*

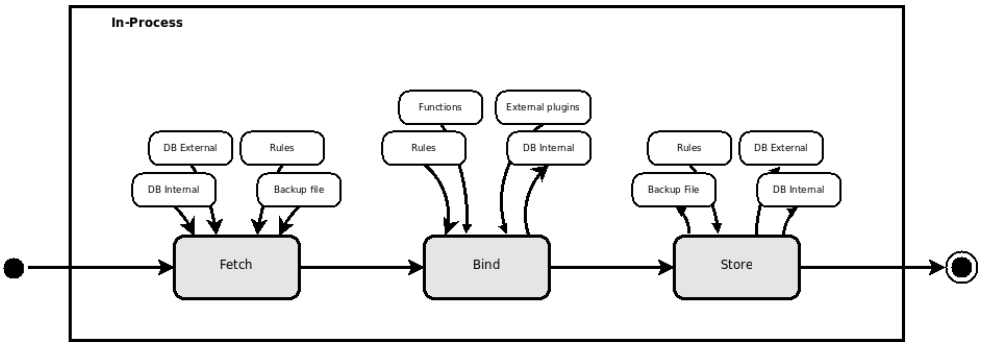


Figura 3.11: Fluxo do módulo *in-process*

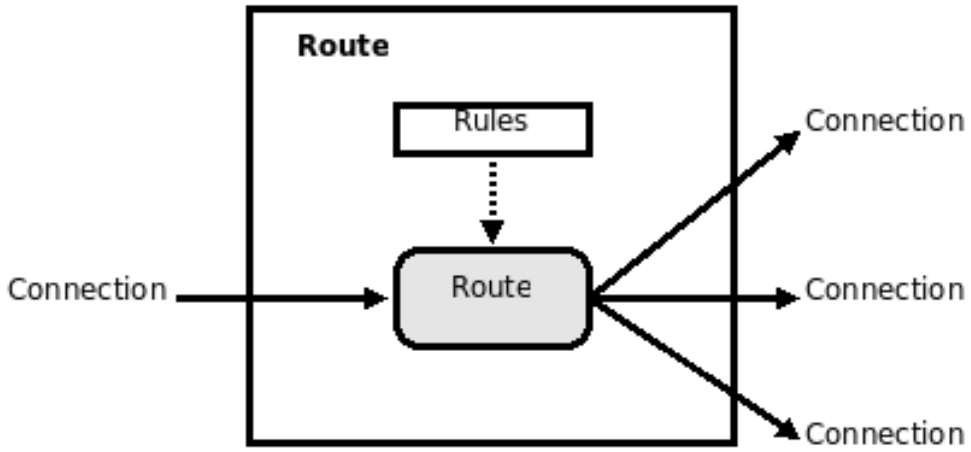


Figura 3.12: O módulo *Route* é o responsável por direccionar uma mensagem para um ou mais destinos.

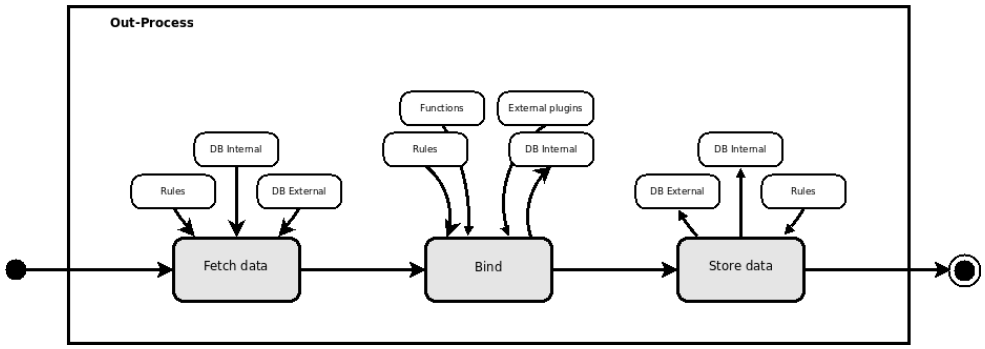


Figura 3.13: Modelo do processo *out-process*

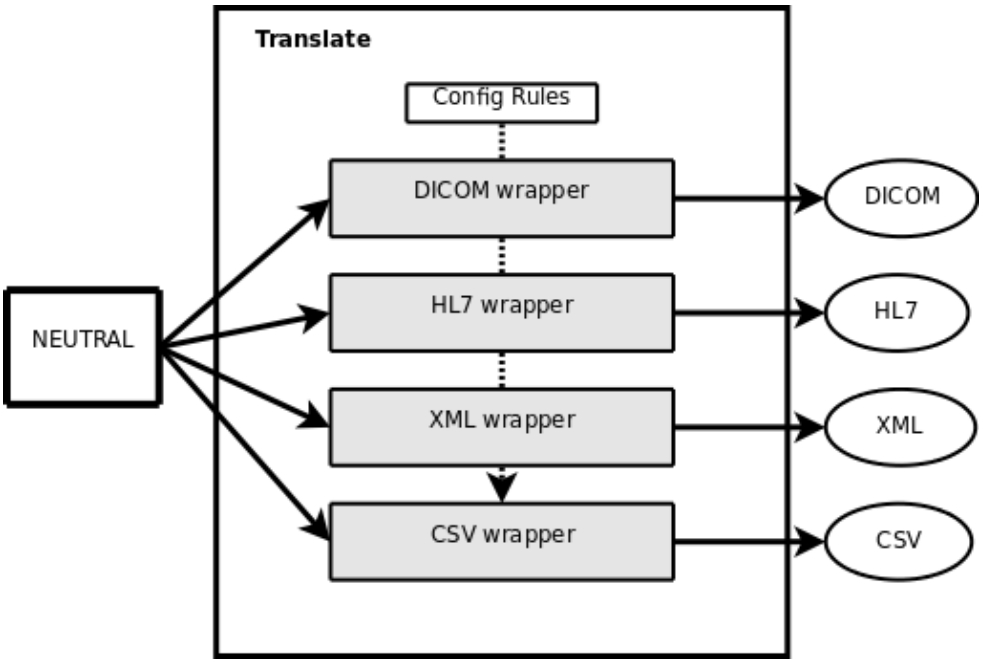


Figura 3.14: Modelo do processo *translate*

O processo *Send* é o responsável pelos detalhes de comunicação baseada em TCP/IP, excepto no caso *file*, e envia a mensagem para o destinatário.

Um elemento fundamental no funcionamento do Healthy Talkie e no sistema DCM4HT são as tabelas de integração. Estas são utilizadas para guardar variáveis temporárias associadas às mensagens que serão capazes de suportar um número ilimitado de entradas: cada uma com diferentes tamanhos e conteúdos. Os campos repetidos têm uma cardinalidade identificada no seu nome, exemplo "nomepaciente = josé" e "nomepaciente.2 = rui".

3.3 Implementação

Na implementação deste projecto foi necessário tomar decisões estratégicas, quanto ao desenvolvimento de todas as funcionalidades relacionadas com o protocolo DICOM de raiz ou utilizando uma API de código aberto.

As APIs encontradas e que nos poderiam ser úteis são:

- DCM4CHE (Java)
- DCMTK da Offis (C++)
- dicom4j (Java)

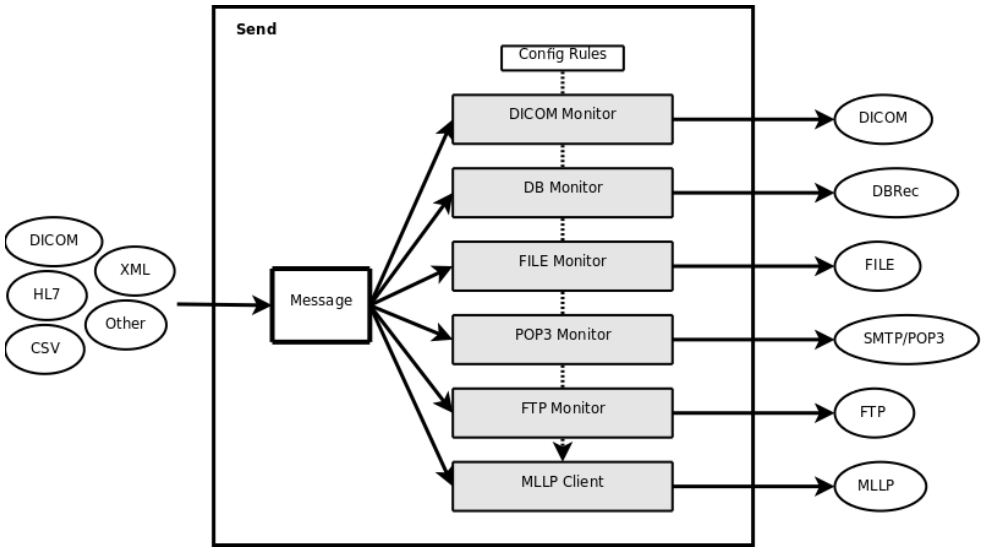


Figura 3.15: Fluxo do módulo *send*

Message Reference	Data Name	Data Type	Data Value
1992883727	PATIENT.NAME	String	John Doe
	PATIENT.AGE	Integer	34
19938839	PATIENT.SSN	String	9918282273
	MESSAGE.TYPE	String	ADT^A01
	PATIENT.BIRTH	Date	1985-08-07

Figura 3.16: Exemplo de tabela de integração

A utilização de APIs DICOM proporcionar-nos-ia uma vantagem significativa, visto que a parte de comunicação das mensagens via *sockets* TCP/IP estaria completa e correctamente implementada. Assim, poderiamos investir todo o nosso esforço em dois pontos:

1. Garantir que as funções de trasporte (envio ou recepção) de mensagens DICOM eram correctamente invocadas e que toda a parte de comunicação via rede se encontrava funcional.

2. Implementar as funções de manipulação das mensagens de acordo com a arquitectura do Healthy Talkie.

Apesar das vantagens mencionadas, a utilização de uma API não se enquadrava na orientação do Healthy Talkie. Como produto pretende-se que este tenha o mínimo de dependências e não seja invasivo, ou seja, necessite, no sistema onde stalado, de instalas subsidias ou bibliotecas adicionais.

Opts por implementar de raiz todos os mecanismos de rede DICOM, o que aumentou drasticamente a complexidade do projecto. Todas as nossas espectativas tiveram de ser revistas, visto que as estimativas de tempo iniciais seriam totalmente ultrapassadas.

Não podemos esquecer que o protocolo DICOM se diferencia de outros protocolos pela sua robustez de rede. Para além da fase de transferência das mensagens (leitura e escrita de PDUs (*Protocol Data Units*)) é indispensável desenvolver também a fase de associação do Healthy Talkie com o equipamento que pretendemos comunicar.

O nosso objectivo primordial seria a implementação do serviço DICOM *Worklist*, com suporte à *transfer syntax Implicit Little Endian*.

O fluxo de informação deste cenário de uso é apresentado na Figura 3.17.

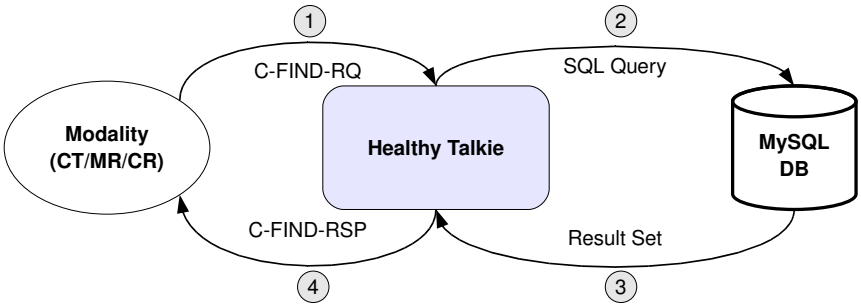


Figura 3.17: Fluxo de informação DICOM *Modality Worklist*

Neste cenário de uso, é apresentada uma modalidade DICOM que desencadeia todo o processo de comunicação através de um pedido de informação ao Healthy Talkie. Este é efectuado por meio de um comando C-FIND-RQ.

Como no local de desenvolvimento deste projecto não temos nenhum equipamento de imagiologia como CT, MR ou CR, tivemos de recorrer a um simulador disponibilizado pela TIANI (Figura 3.18).

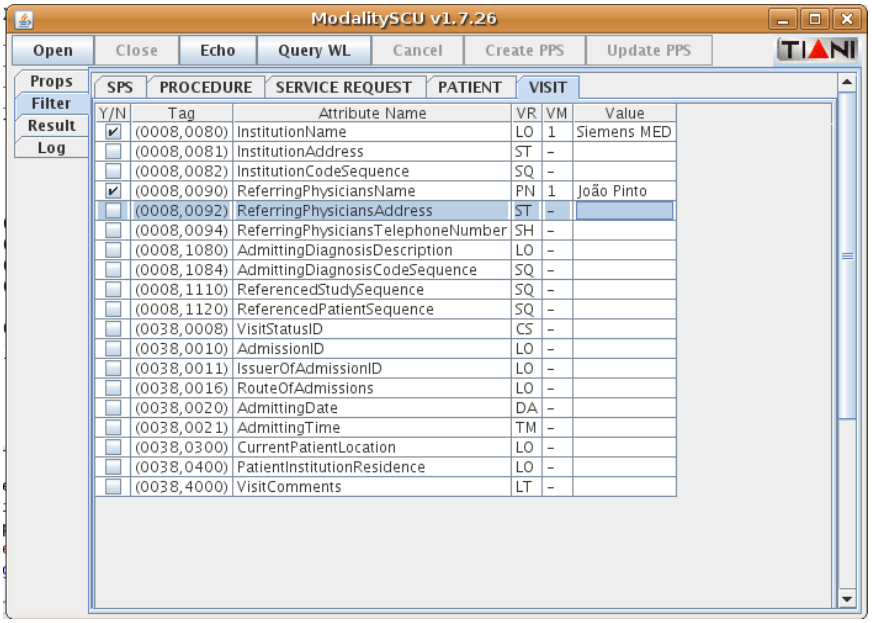


Figura 3.18: Simulador TIANI DICOM *Modality Worklist* SCU

Esta aplicação *Modality Worlist SCU (Service Class User)* permite que o utilizador escolha a informação que vai enviar para a aplicação servidora, *Modality Worklist SCP*, papel desempenhado pelo Healthy Talkie.

Antes de avançarmos para os detalhes de implementação é essencial percebermos como funciona o mecanismo C-FIND (ver Figura 3.19).

Este é constituído por um pedido (C-FIND-RQ) e por uma resposta (C-FIND-RSP). O comando C-FIND-RQ tem um conjunto de dados que são atributos. Estes atributos podem ter dois significados diferentes consoante o valor que comportam.

Os atributos de um C-FIND-RQ, que tenham conteúdo igual a vazio (*string* "") são *returnig keys*, ou seja, definem os campos que deverão ser retornados com informação que a modalidade pretente conhecer.

Por sua vez, os campos de um C-FIND-RQ com conteúdo diferente de vazio são *matching keys*, de todo semelhante ao *select* da sintaxe SQL, onde são filtrados os resultados que apenas contenham o valor destes campos. Podemos constatar a semelhança entre um C-FIND-RQ e uma *query* SQL na Figura 3.20.

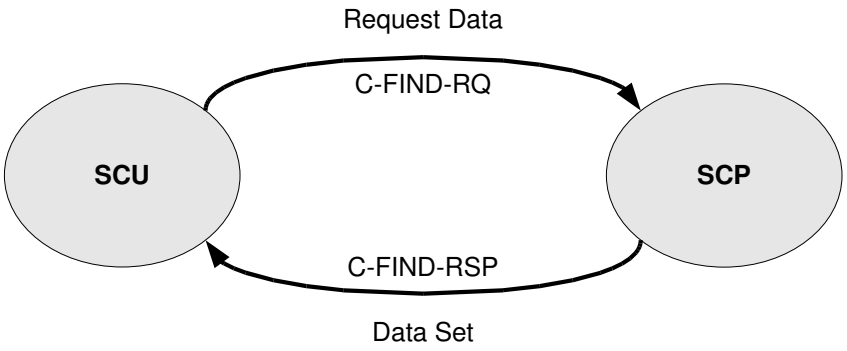


Figura 3.19: Fluxo do mecanismo C-FIND entre *Service Class User* e *Service Class Provider*

A resposta a um pedido é conseguida através de um comando C-FIND-RSP. O conteúdo deste comando é de todo semelhante ao C-FIND-RQ com a diferença de que os campos vazios estão agora preenchidos com os resultados encontrados pelo servidor (Figura 3.21).

Agora que conhecemos os detalhes para esta implementação e os resultados esperados, podemos avançar para a descrição das classes e funções responsáveis pelo funcionamento da solução DCM4HT.

Este sub-capítulo está organizado pelas funcionalidades oferecidas pelo DCM4HT.

3.3.1 Estabelecimento da Comunicação

Para iniciarmos a comunicação com uma modalidade é necessário adicionar uma ligação ao Healthy Talkie do tipo DICOM, de forma a criar um servidor (DICOM SCP). As ligações são definidas no ficheiro connections.xml do *package* config.

```
1 <connection peer="Dicom MWL In" name="Dicom MWL In" ack="none" type="
    inbound" priority="3" interactive="true">
2     <protocol name="DICOM" version="3.0 Implicit Little" header="no"/>
3     <support>
4         <appcontext name="1.2.840.10008.3.1.1.1" />
```

Figura 3.20: Paralelismo entre um *select* de SQL e um C-FIND-RQ

C-FIND-RQ

Tag	Attribute Name	Value
0010,0010	Patient Name	João Pinto
0010,1010	Patient Age	

C-FIND-RSP

Tag	Attribute Name	Value
0010,0010	Patient Name	João Pinto
0010,1010	Patient Age	23

Figura 3.21: Exemplo de um pedido C-FIND e respectiva resposta

```
        <callingae title="ModalitySCU" />
        <calledae title="RisServer" />
    </support>
...
```

As definições supra mencionadas dão origem a uma ligação do tipo *inbound* com o nome "Dicom MWL In", protocolo de comunicação DICOM v3.0 e codificação *Implicit Little*. É indispensável definir *peer* desta ligação, ou seja, os atributos de rede para o dispositivo "Dicom MWL In". Estes atributos podem ser encontrados no ficheiro peers.xml do *package* config.

```
...
<dicom id="Dicom MWL In" hostname="127.0.0.1" port="6104"/>
...
```

Como podemos observar, este *peer* é do tipo DICOM. Sempre que o Healthy Talkie for executado será detectado o tipo deste *peer* e criada uma nova *thread* DICOM para esta ligação. Estes métodos estão implementados na classe Receive do *package* connection.

```
...
if(peer.getType().equals(Receive.METHOD_DICOM)) {
    DICOMRun();
}
...

/**
 * Dicom Message Exchange. Here we establish an Association, exchange
     data, and close Association
 *
 * @return void
 */
private void DICOMRun() {
    DICOMExchangeThread dexThread = new DICOMExchangeThread(this.alarm,
         this.connection, this.peer, this, this.internalDB);
    dexThread.run();
}
```

Esta *thread* do tipo DICOMExchangeThread é responsável por:

1. Iniciar o DICOM SCP, para criação do *socket* TCP utilizado posteriormente pelo DICOM SCU (simulador TIANI);
2. Estabelecimento da associação das duas entidades (fase de negociação);
3. Recepção da mensagem DICOM transmitida pelo DICOM SCU;
4. Encaminhamento da mensagem recebida para o núcleo do Healthy Talkie onde a mensagem será processada, através da fila RQueue (ver Figura 3.6).

A criação do servidor TCP que irá escutar a porta definida para este *peer* (porta 6104), é a primeira função que podemos encontrar dentro desta *thread*.

```
1 public void run() {
2   try {
3   /*
4    * Start DICOM SCP
5    */
6   Socket socket = DicomMessage.startProvider(this.peer.getPort());
7
8 ...
9 }
```

3.3.2 Negociação entre DICOM SCU e SCP

Logo apbertura do canal de comunicação entre as duas aplicações prossegue-se com o estabelecimento da associação.

```
1 /*
2  * Negociate Association
3  */
4
5 boolean accepted = DicomMessage.associationEstablishment(socket, this.connection.
      getAppContextNames(),this.connection.getCallingAETitles(), this.connection.getCalledAETitles
      ());
```

Em primeiro lugar, o Healthy Talkie como DICOM SCP irá aguardar a única mensagem possível no ínicio deste processo, um pedido A-ASSOCIATE (A-ASSOCIATE-RQ) como é mostrado na Figura 2.5. Caso seja recebida um outro tipo de mensagem a comunicação é imediatamente interrompida.

Esta mensagem contém informações relevantes como:

1. *Called AE Title* - Nome da aplicação DICOM destinatária;
2. *Calling AE Title* - Nome da aplicação DICOM remetente;
3. *Abstract Syntax Names* - Nomes dos serviços suportados pelo SCU;

4. *Transfer Syntax Names* - Nomes das codificações suportadas para cada serviço;

5. *Max. Length Received* - Tamanho máximo dos PDUs recebidos.

Com base nas informações desta ligação, inseridas no ficheiro connections.xml, podemos validar se o remetente desta mensagem se encontra na lista dos dispositivos aceites. Esta validação é um requisito deste projecto e apresentado no cenário de uso "Autorização" da Figura 3.5. Caso o nome do remetende não se encontre autorizado este pedido de associação será rejeitado, enviando para tal uma mensagem A-ASSOCIATE-RJ (ver apêndice A).

Esta validação é efectuada recorrendo ao método validateAAssociateRQ da classe DicomMessage.

```
1  /**
2   * Validate A-ASSOCIATE-RQ
3   */
4  int validationResult = validateAAssociateRQ(associateRQ,
       supportedAppContextNames, supportedCallingAETitles,
       supportedCalledAETitles);
5
6  ...
7
8  if(validationResult==0) {
9    /**
10    * Send A-ASSOCIATE-AC PDU
11    */
12   sendAAssociateAC(socket,associateRQ);
13
14   ...
15 }
```

Caso o valor retornado pelo método validateAAssociateRQ seja igual a 0, então o pedido de associação é válido e o Healthy Talkie comunica a sua decisão por meio de um A-ASSOCIATE-AC (ver apêndice A).

Um detalhe importante na construção na mensagem A-ASSOCIATE-AC é a criação da lista de serviços DICOM aceites e respectivas *transfer syntaxes* suportadas. Ou seja, para cada serviço e respectivas *transfer syntaxes* da lista recebida pela mensagem A-ASSOCIATE-RQ vamos comunicar quais destas serão aceites ou recusadas. Esta funcionalidade encontra-se implementada no método buildPrContextArray da classe DicomMessage.

```
1  private static ArrayList<PresentationContext> buildPrContextArray(AAssociateRQ_AC associateRQ)
       throws Exception {
2      ArrayList<String> supportedAbstractSyntax = new ArrayList<String>();
3      ArrayList<String> supportedTransferSyntax = new ArrayList<String>();
4
5      // Modality Worklist Information Model - FIND SOP Class
```

```
6       supportedAbstractSyntax.add("1.2.840.10008.5.1.4.31");
7
8       // Implicit LE
9       supportedTransferSyntax.add("1.2.840.10008.1.2");
10      ArrayList<PresentationContext> prContextArray = new ArrayList<PresentationContext>();
11
12      /*
13       * For each Presentation Context,
14       * check if it Abstraxt Syntax Name is one of the supported
15       */
16      for (PresentationContext pc : associateRQ.prContextArray) {
17          String absSyntaxName = new String(pc.absSyntax.absSyntaxName,"ISO-8859-1");
18          absSyntaxName = absSyntaxName.trim();
19          PresentationContext prContext = null;
20
21          // If absSyntaxName is supported
22          if(supportedAbstractSyntax.contains(absSyntaxName)) {
23              /*
24               * Check which Transfer Syntaxes are supported
25               */
26              boolean atLeastOneTr = false;
27              for (TransferSyntax ts : pc.trSyntaxArray) {
28                ...
29              }
30              // If no transfers syntax in this presentation context are suppoted,
31              // it doesn't accept this Abstract Syntax Name
32              if(!atLeastOneTr) {
33                  ...
34              }
35          }
36          else { // if abstract syntax not suported
37            ...
38          }
39          prContextArray.add(prContext);
40      }
41      return prContextArray;
42  }
```

Outro valor que é negociado entre as duas aplicações é o tamanho máximo que poderr cada PDU a receber, isto é, a capacidade máxima de recepção por PDU.

Concluída a fase de negociação, a entidade DICOM SCU irá enviar mensagens que apenas solicitem os serviços aceites pelo SCP e codificadas com *transfer syntaxes* suportadas.

3.3.3 Recepção da mensagem DICOM (P-DATA-TF PDUs)

Uma mensagem DICOM deve ser comunicada em múltiplos PDUs, no qual o tamanho de cada PDU não deve exceder o valor negociado durante a fase de associação. A estrutura de um P-DATA-TF PDU é apresentada no apêndice A.

A leitura destes PDUs a partir do *stream* do *socket* TCP é realizada dentro da classe DICOMExchangeThread com o método readPDataPDUs.

```
1
2   /*
3    * If Association Accepted
4    */
5   if(accepted) {
6       /*
7        * Read P-DATA-TF PDUs
8        */
```

```
9     byte[] pDataPDUs = DicomMessage.readPDataPDUs(socket);
10    String msg = new String(pDataPDUs,"ISO-8859-1");
11 ...
```

O mecanismo de leitura dos PDUs deverá reconhecer o início de cada PDU através do *byte* que é lido em primeiro lugar. Este *byte* contém o valor 4 representado na base 16, como apresentado na Figura 3.22.

PDU bytes	Field name	Description of field
1	PDU-type	04H
2	Reserved	This reserved field shall be sent with a value 00H but not tested to this value when received.
3-6	PDU-length	This PDU-length shall be the number of bytes from the first byte of the following field to the last byte of the variable field. It shall be encoded as an unsigned binary number.
7-xxx	Presentation-data-value Item(s),	This variable data field shall contain one or more Presentation-data-value Items(s). For a complete description of the use of this field see Section 9.3.5.1

Figura 3.22: Estrutura detalhada do P-DATA-TF PDU

Para cada PDU é necessário extrair a informação nos *bytes* 3 a 6 que indicam o número de *bytes* que falta processar.

Depois de concluída a recepção de todos os P-DATA-TF PDUs que contêm a mensagem é necessário converter o *array* resultante para uma *string*. Esta operação é fundamental dado que todo o núcleo do Healthy Talkie foi preparado para a recepção das mensagens apenas neste formato.

Este pormenor revelou-se uma fonte de problemas. Uma vez que a mensagem é um binário e não um conjunto de caracteres ASCII, verifica-se a perda de informação do *array* de *bytes* após ter sido convertido para uma *string*.

Este obstáculo foi ultrapassado depois de se descobrir que seria necessário definir o *charset* com o valor de "ISO 8859-1" (Latin-1) nos processos de conversão.

```
1 ...
2 // Convert data byte[] to String
3 byte[] pDataPDUs = DicomMessage.readPDataPDUs(socket);
4 String msg = new String(pDataPDUs,"ISO-8859-1");
5 ...
```

```
1 ...
2 // Convert data String to data byte[]
3 byte[] dataBytes = data.getBytes("ISO-8859-1");
4 ...
```

Depois de construída a *string* que contém a mensagem que a aplicação DICOM SCU pretende transmitir, podemos avançar para o processo de conversão da mesma para formato neutro (*Untranslate*).

Para tal é necessário colocar a mensagem na fila RQueue (ver Figura 3.6) que será lida posteriormente pelo módulo *Untranslate*.

```
1  ...
2  /*
3   * Send message to RQUEUE and Start Untranslate
4   */
5
6  if (msg != null) {
7      internalDB.RQueueInsert(connection.getID(), new Date(), msg, false,
           false, null, null);
8      this.receive.startUntranslate();
9  }
10 else {
11     alarm.error(connection.getID(), "No P-DATA-TF PDU's received");
12 }
13 ...
```

3.3.4 Processo *Untranslate*

O processo de *untranslate* é dividido em três fases:

1. Conversão da mensagem DICOM para uma árvore n-ária;
2. Validação dos dados da árvore;
3. Conversão dos nós-folha da árvore para uma tabela de integração.

A primeira fase mencionada é importante para facilitar a subdivisão da mensagem em múltiplos atributos. Cada abributo DICOM é identificado por uma *tag* e possui um determinado conteúdo. Criou-se um tipo de árvore que satisfizesse por inteiro as nossas necessidades, sendo esta implementada na classe MetadataTree. É relevante referir que cada objecto do tipo MetadataTree é constituído por:

- Variável - preenchido com a *tag* do atributo;
- Tipo de dados - todos os atributos DICOM têm um tipo de dados como "AS", "CS" ou "DA";
- Valor - o conteúdo do atributo;
- Nós-árvore - lista de MetadataTrees que correspondem aos filhos deste nó.

A raíz desta MetadataTree deverá ser sempre um nó com a variável igual a "ROOT" e que irá conter todos os atributos DICOM da mensagem recebida. O modelo representativo desta árvore é apresentado na Figura 3.23.

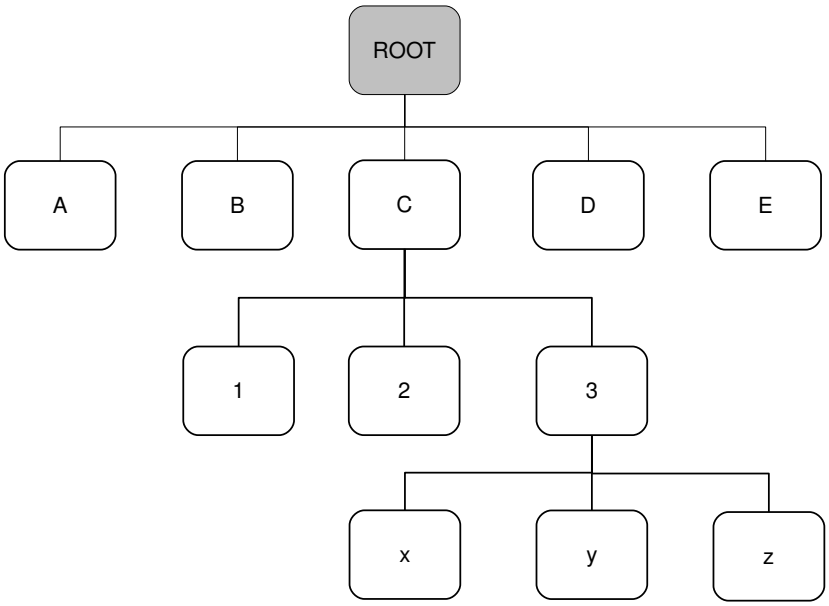


Figura 3.23: Modelo de uma MetadataTree

O processo de leitura e conversão da mensagem é baseado em regras definidas no ficheiro protocol.xml, onde especificamos a estrutura dos elementos DICOM. Como opts, numa primeira fase, por suportar apenas a *transfer syntax Implicit Little Endian*, então a estrutura que devemos respeitar será:

```
1 <level name="Element Implicit" dictionaryreference="Tag">
2     <level name="Tag" size="4"/>
3     <level name="Length" size="4"/>
4     <level name="Value" sizelocation="Length"/>
5 </level>
```

Este excerto define os elementos com codificação do tipo implícita, e é baseado no protocolo DICOM, como apresentado na Figura 3.24.

Uma mensagem DICOM é um conjunto de elementos de igual formato. E estando definida a estrutura deste elemento então é possível ler correctamente toda a mensagem. O método buildTreeNode da classe DicomUntranslate permite efectuar aconstrução da árvore. Este método lê sequencialmente todos os elementos da mensagem e sempre que encontra um elemento do tipo "SQ" (elemento DICOM que comporta mais elementos dentro de si) efectua uma chamada recursiva de modo a construir uma árvore na sua totalidade.

Tag		Value Length	Value
Group Number (16-bit unsigned integer)	Element Number (16-bit unsigned integer)	32-bit unsigned integer	Even number of bytes containing the Data Elements Value encoded according to the VR specified in PS 3.6 and the negotiated Transfer Syntax. Delimited with Sequence Delimitation Item if of Undefined Length.
2 bytes	2 bytes	4 bytes	'Value Length' bytes or Undefined Length

Figura 3.24: Estrutura de um elemento DICOM do tipo implícito

A fase de validação dos dados da árvore, verifica se todos os elementos são válidos para este tipo de mensagem. Como estamos a simular um pedido de DICOM *Modality Worklist*, recorrendo ao comando C-FIND-RQ, então através do protocolo [7, chap. PS 3.8] podemos conhecer os elementos previstos para esta mensagem.

Uma representação dos elementos do comando C-FIND-RQ no ficheiro protocols.xml foi absolutamente essencial para a realização da validação.

```
<message name="C-FIND-RQ" typelocationtag="00000100" typeforcevalue="0020">
    <includes>
        <group name="Command Set" range="(1,1)">
            <include name="Element Implicit" tag="00000000" range="(1,1)"/>
            <include name="Element Implicit" tag="00000002" range="(1,1)"/>
            <include name="Element Implicit" tag="00000100" range="(1,1)"/>
            <include name="Element Implicit" tag="00000110" range="(1,1)"/>
            <include name="Element Implicit" tag="00000700" range="(1,1)"/>
            <include name="Element Implicit" tag="00000800" range="(1,1)"/>
        </group>
        <group name="Data Set" range="(0,1)">
            <include name="Element Implicit" tag="00400000" range="(0,1)"/>
            <include name="Element Implicit" tag="00400100" range="(0,1)"/>
            <include name="Element Implicit" tag="00400100.fffee000" range="(0,N)"/>
            <include name="Element Implicit" tag="00400100.fffee000.00080060" range="(0,1)"/>
            <include name="Element Implicit" tag="00400100.fffee000.00400001" range="(0,1)"/>
            <include name="Element Implicit" tag="00400100.fffee000.00400002" range="(0,1)"/>
            <include name="Element Implicit" tag="00400100.fffee000.00400003" range="(0,1)"/>
            <include name="Element Implicit" tag="00400100.fffee000.00400006" range="(0,1)"/>
        </group>
    </includes>
</message>
```

Neste excerto, uma mensagem do tipo C-FIND-RQ é composta por um *Command Set* e um *Data Set*. Por sua vez, cada um destes contém um conjunto de *includes* que correspondem aos elementos da mensagem. Esta definição permite validar o nome das variáveis dos nós da árvore comparando-as com o valor do atributo "tag", assim como o número de ocorrências deste elemento na árvore através do atributo "range".

É importante notar neste excerto XML que existem elementos em que o seu atributo "tag" contém o caracter "." como o elemento que se segue:

```
1 <include name="Element Implicit" tag="00400100.fffee000" range="(0,N)"/
    >
```

O caracter "." estabelece uma hierarquia entre os elementos. Neste caso em particular informa que o elemento "fffee000" é um nó filho do elemento "00400100".

Conclui-se que o processo de transferência dos nós-folha desta árvore n-ária para uma tabela de integração se revela bastante simplificado, bastando adicionar uma entrada à tabela por cada nó-folha encontrado.

3.3.5 Processamento da mensagem DICOM

Pretendemos nesta fase da implementação completar os campos vazios da tabela de integração com informação existente numa base de dados MySQL, e construir com estes dados uma mensagem C-FIND-RSP.

Como podemos ver na Figura 3.6, após a conversão da mensagem para formato neutro, encontramos:

- *In-Process*
- *Route*
- *Out-Process*

Foi decidido implementar o processamento da mensagem DICOM dentro do módulo *In-Process* e na classe Process, tendo o método *fetch* sido adaptado às necessidades do DCM4HT.

Este método separa os campos preenchidos dos campos sem conteúdo da mensagem e constrói uma *query* SQL, como apresentado na Figura 3.20. Para a construção da *query* é necessário associar as *tags* dos atributos DICOM com os campos da base de dados. Basta adicionar as regras no ficheiro connections.xml como se mostra no seguinte exemplo:

```
1 <fetch>
2     <!--These rules are adapted to Dicom. The goal is just to associate
          a dicom tag to an database field-->
3     <fetchbinding type="string" destination="00400006">
4     <origin type="external" peer="RIS" database="dicom" table="
          ScheduledProceduresSteps" field="
          scheduledPerformingPhysicianName" variabletype="string"/>
5     </fetchbinding>
6     <fetchbinding type="string" destination="00080060">
7      <origin type="external" peer="RIS" database="dicom" table="
          ScheduledProceduresSteps" field="modality" variabletype="
          string"/>
8     </fetchbinding>
```

```
</fetch>
```

Neste exemplo estamos a associar o atributo DICOM com *tag* "00400006" à coluna "scheduledPerformingPhysicianName" e o atributo com *tag* "00080060" à coluna "modality" da tabela "ScheduledProceduresSteps".

Para que o Healthy Talkie seja capaz de comunicar com esta base de dados é necessário adicionar um *peer* ao ficheiro de configuração peers.xml como se v seguinte exemplo.

```
<!-- ris database -->
<database id="RIS" hostname="127.0.0.1" port="3306" databasename="dicom
    " vendor="MySQL" version="5.0.51a-3ubuntu5.1" username="root"
    password="password"/>
```

O método *fetch* é responsável pelo pedido à base de dados que é o preenchimento dos campos vazios da mensagem DICOM com os resultados provenientes da base de dados.

O próximo passo é o encaminhamento da mensagem para um destino, através do módulo *Route* implementado na classe *Route* do Healthy Talkie. Todos os mecanismos de encaminhamento do Route são baseados em regras definidas no ficheiro routing.xml, onde se indica o *peer* destinatário para as mensagens com origem num *peer* remetente.

```
<!-- This is a dicom worklist routing -->
<origin connection="Dicom MWL In">
  <route>
    <destination connection="Dicom MWL Out" action="C-FIND-RSP"/>
  </route>
</origin>
```

No exemplo mencionado, todas as mensagens com origem no *peer* "Dicom MWL In" serão encaminhadas para o *peer* "Dicom MWL Out". O tipo de mensagem que deverá ser construída para o *peer* de destino é especificada no atributo XML "action". Neste caso pretendemos construír uma mensagem C-FIND-RSP com a resposta ao pedido DICOM *Modality Worklist*.

Para concluir o processamento da mensagem, falta adicionar ou remover alguns dados da tabela de integração de forma a que a mensagem inclua todos os atributos de um C-FIND-RSP definidos em protocols.xml. Este processo é realizado no módulo Out-Process, ou seja, na classe OutProcess do HealthyTalkie.

```
<message name="C-FIND-RSP" typelocationtag="00000100" typeforcevalue="8020">
    <includes>
        <group name="Command Set" range="(1,1)">
            <include name="Element Implicit" tag="00000000" range="(1,1)"/>
            <include name="Element Implicit" tag="00000002" range="(1,1)"/>
            <include name="Element Implicit" tag="00000100" range="(1,1)"/>
            <include name="Element Implicit" tag="00000120" range="(1,1)"/>
```

```
8             <include name="Element Implicit" tag="00000800" range="(1,1)"/>
9             <include name="Element Implicit" tag="00000900" range="(1,1)"/>
10        </group>
11        <group name="Data Set" range="(0,1)">
12            <include name="Element Implicit" tag="00400000" range="(0,1)"/>
13            <include name="Element Implicit" tag="00400100" range="(0,1)"/>
14            <include name="Element Implicit" tag="00400100.fffee000" range="(0,N)"/>
15            <include name="Element Implicit" tag="00400100.fffee000.00080060" range="(0,1)"/>
16            <include name="Element Implicit" tag="00400100.fffee000.00400001" range="(0,1)"/>
17            <include name="Element Implicit" tag="00400100.fffee000.00400002" range="(0,1)"/>
18            <include name="Element Implicit" tag="00400100.fffee000.00400003" range="(0,1)"/>
19            <include name="Element Implicit" tag="00400100.fffee000.00400006" range="(0,1)"/>
20        </group>
21    </includes>
22 </message>
```

Os dados da tabela de integração deverão cumprir com as regras definidas no ficheiro protocols.xml para este tipo de mensagem (C-FIND-RSP), para avançarmos para o mecanismo de conversão da tabela de intregração para a mensagem DICOM C-FIND-RSP.

3.3.6 Processo *Translate*

O processo Translate encontra-se implementado na classe Translate do Healthy Talkie. Dentro do método *performTranslate* desta classe é detectado o tipo da mensagem que pretendemos traduzir. Sendo a mensagem C-FIND-RSP do tipo DICOM (um binário), então será invocado o método *build* da classe DicomTranslate com o protocolo a utilizar (protocol), o formato da mensagem final (message) e a tabela de integração com a informação a ser convertida (ie).

```
1 if (protocol.getStructure().getType().equals(ConfigProtocolStructure.
    TYPE_BYNARY)) {
2       /*
3         * Translate dicom message
4         */
5       DicomTranslate translator = new DicomTranslate(alarm, destination
            , validate);
6       return translator.build(protocol, message, ie);
7 }
```

O processo *Translate* é análogo ao processo *Untranslate*, baseando-se ambos nas definições do protocolo para conseguir efectuar a conversão da mensagem para formato neutro, e desta última de volta para uma mensagem DICOM.

Mais uma vez, relembramos que até ao momento apenas suportamos a codificação tipo *Implicit Little Endian*, onde a estrutura de cada elemento DICOM deverá ser:

```
1 <level name="Element Implicit" dictionaryreference="Tag">
2         <level name="Tag" size="4"/>
3         <level name="Length" size="4"/>
4         <level name="Value" sizelocation="Length"/>
```

```
5 </level>
```

Mais uma vez vamos converter a tabela de integração para uma MetadataTree, para melhor estruturar a mensagem que pretendemos contruir, tendo como resultado será uma árvore como a apresentada na Figura 3.23. Esta conversão é realizada invocando o método *buildMetadataTrees* da classe DicomTranslate.

```
1 // build one metadataTree for each iEntry
2 LinkedList<MetadataTree> trees = buildMetadataTrees(iEntries);
```

Com todos os elementos DICOM estruturados em árvore torna-se mais fácil percorrer recursivamente a árvore e converter o conteúdo de cada elemento num *byte array*.

Recorrendo mais uma vez à definição do protocolo DICOM do tipo *Implicit Little Endian* supra-mencionada vamos prosseguir com a conversão invocando o método *treesToByteArrays* da classe DicomTranslate:

```
1 // Convert MetadataTrees to byte[], following a protocol
2 byte[] dicomMessages = treesToByteArrays(protocol, message, trees);
```

O *byte array* dicomMessages contém a mensagem DICOM C-FIND-RSP a ser enviada para à modalidade que efectuou o pedido.

O processo normal de envio das mensagem do Healthy Talkie para exterior é realizado com o módulo *Send* (ver Figura 3.6), para o que as mensagens a serem enviadas deverer colocadas na SQueue. No entanto, esta arquitectura não se adapta às necessidades de comunicação entre *Modality Worklist* SCU e *Modality Worklist* SCP, visto que a mensagem C-FIND-RSP deverá ser enviada pelo mesmo *socket* utilizado para a recepção da mensagem C-FIND-RQ, como demonstra a Figura 3.25. Por este motivo, em vez da mensagem ser colocada na fila SQueue será introduzida na fila RQueue para ser enviada pela *thread* Receive, pelo que o *socket* criado no início da comunicação com a modalidade não deverá ser fechado em nenhum momento até ao envio da resposta ao pedido.

3.3.7 Envio da mensagem DICOM (P-DATA-TF PDUs)

O envio da mensagem DICOM C-FIND-RSP é realizado na classe DicomExchangeThread com a ivocação do método *writePDataPDUs* que necessita como argumentos a referência do *socket* TCP e da mensagem a enviar.

```
1 /*
2  * Write PDataPDUs
3  */
4 DicomMessage.writePDataPDUs(socket, responses);
```

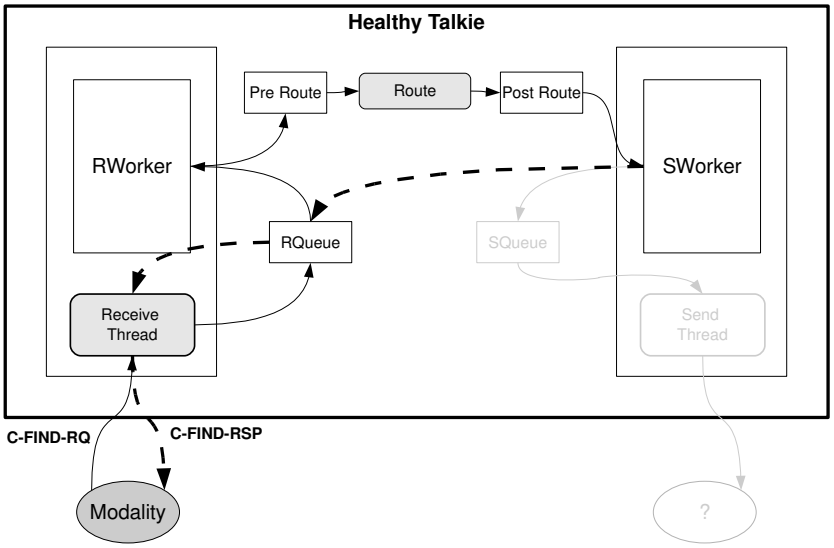


Figura 3.25: Fluxo de informação no envio de um C-FIND-RSP

Este método realiza três operações fundamentais:

- Separação do *Command Set* e do *Data Set* da mensagem com base nas definições existentes no ficheiro de configuração protocols.xml;
- Preenchimento mínimo de dois PDUs, um com a parte *Command Set* e um com a parte *Data Set*. O tamanho de cada PDU não deverá exceder o tamanho máximo negociado durante a fase de associação entre DICOM SCU e DICOM SCP, devendo-se subdividir por mais PDUs caso seja necessário;
- Escrita no *stream* do *socket* TCP do *byte array* portador dos PDUs contruídos.

3.3.8 Fecho da associação entre DICOM SCP e SCU

Para concluir correctamente a comunicação entre a modalidade e o Healthy Talkie é fundamental trocar duas mensagens (ver apêndice A).

- A-RELEASE-RQ
- A-RELEASE-RSP

O Healthy Talkie, como *Modality Worklist* SCP aguarda que lhe seja solicitado o fecho da comunicação pela entidade *Modality Worklist* SCU através de uma mensagem A-RELEASE-RQ. O *socket* TCP poderá ser fechado para terminar a comunicação quando o Healthy Talkie enviar a resposta a este pedido com uma mensagem A-RELEASE-RSP (ver Figura 3.26).

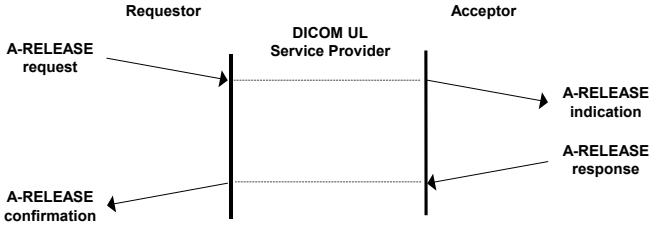


Figura 3.26: Mensagens trocadas durante fecho de associação

O mecanismo de fecho da comunicação encontra-se implementado na classe DICOMExchangeThread:

```
1  ...
2  /*
3    * Receive A-Release-RSP
4    */
5  DicomMessage.receiveAReleaseRQ(socket);
6  /*
7    * Send A-Release-RQ
8    */
9  DicomMessage.sendAReleaseRP(socket);
10
11 /*
12   * Close socket
13   */
14 socket.close();
15 ...
```

3.4 Testes e funcionamento

O testes efectuados ao Healthy Talkie com o módulo DCM4HT consistem na sua funcionalidade como *Modality Worlist* SCP. Foi utilizada a aplicação ModalitySCU disponibilizada pela TIANI, e base de dados MySQL com a informação a ser recolhida pelo Healthy Talkie. O fluxo de informação deste teste é apresentado na Figura 3.27.

Neste cenário de teste, o utilizador pretende conhecer o nome do operador responsl por uma modalidade. Usando a aplicação ModalitySCU, o utilizador deverá preencher o campo *Modality* com a identificação da modalidade e seleccionar o campo *ScheduledPerformingPhysicianName* deixando-o sem qualquer valor.

Os campos seleccionados sem conteúdo serão preenchidos pelo Healthy Talkie e retornados numa mensagem C-FIND-RSP.

A base de dados MySQL pretende simular um RIS e contém uma tabela para testes como apresenta a Figura 3.29

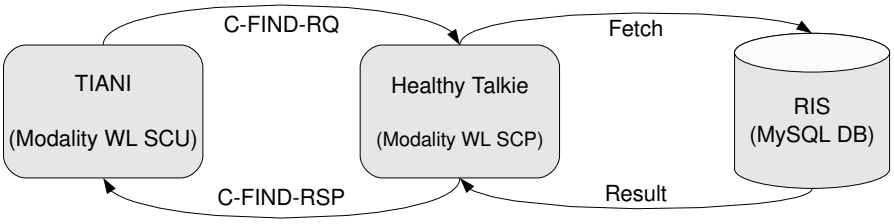


Figura 3.27: Fluxo de informação do cenário de teste

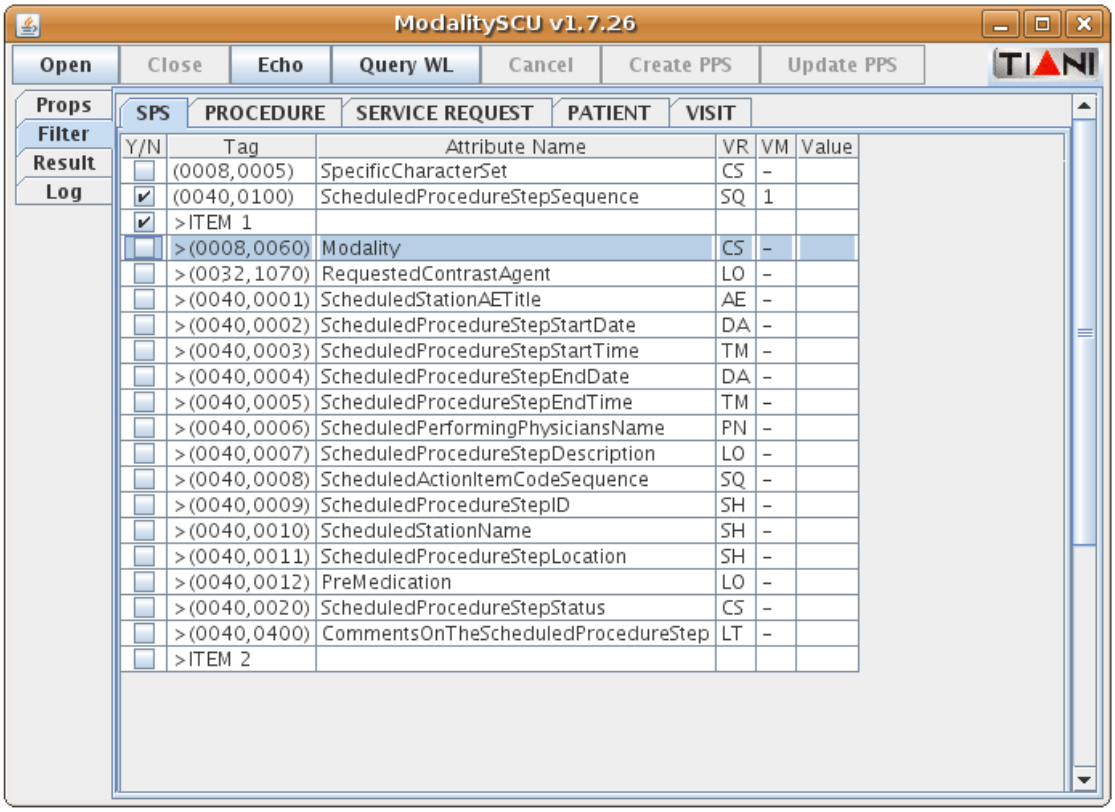


Figura 3.28: Aplicação TIANI Modality SCU

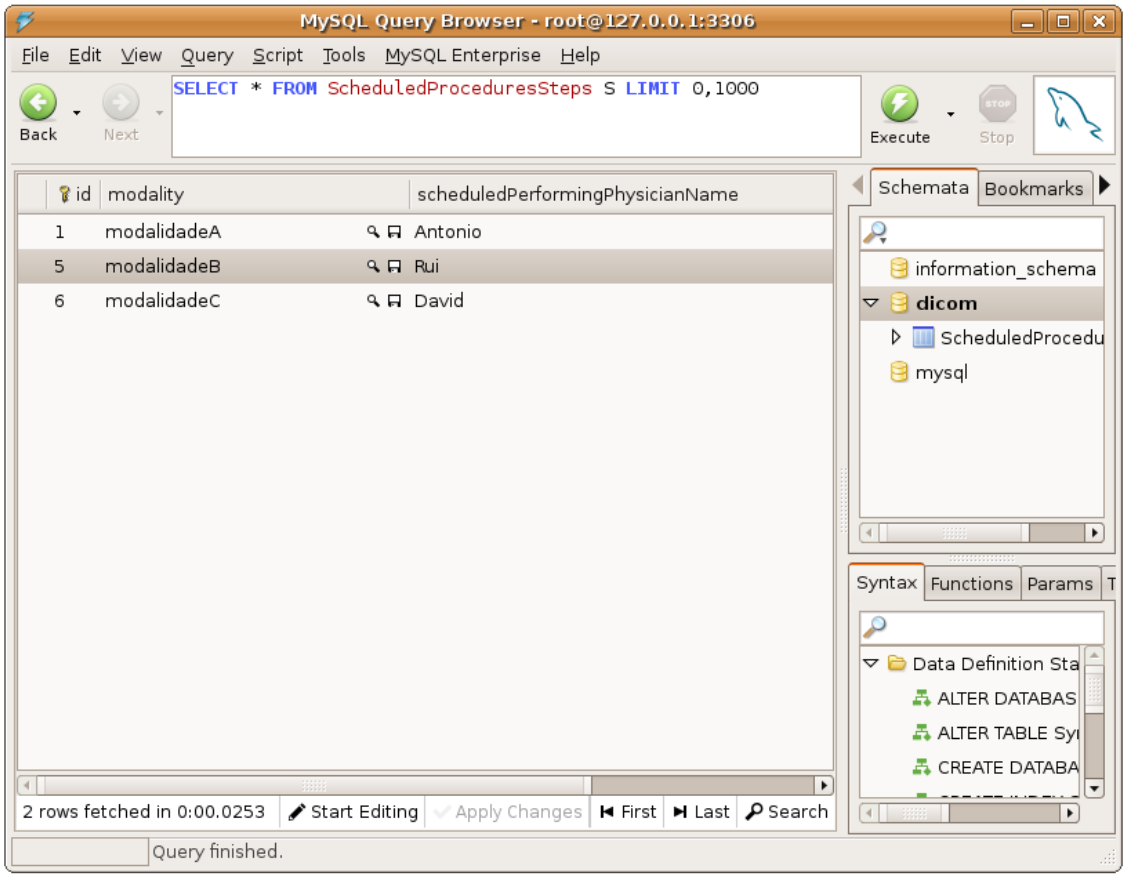


Figura 3.29: Tabela Base de dados MySQL

Para que o Healthy Talkie seja capaz de comunicar com esta base de dados é necessário adicionar um *peer* ao ficheiro de configuração peers.xml do seguinte modo:

```
<!-- ris database -->
<database id="RIS" hostname="127.0.0.1" port="3306" databasename="dicom
    " vendor="MySQL" version="5.0.51a-3ubuntu5.1" username="root"
    password="password"/>
```

E para que o sistema DCM4HT consiga construir uma mensagem de resposta a um pedido C-FIND-RQ é essencial associar as *tags* DICOM aos campos da base de dados no ficheiro connections.xml:

```
<!--These rules are adapted to Dicom. The goal is just to associate a
    dicom tag to an database field-->
<fetchbinding type="string" destination="00400006">
    <origin type="external" peer="RIS" database="dicom" table="
        ScheduledProceduresSteps" field="
        scheduledPerformingPhysicianName" variabletype="string"/>
</fetchbinding>
<fetchbinding type="string" destination="00080060">
    <origin type="external" peer="RIS" database="dicom" table="
        ScheduledProceduresSteps" field="modality" variabletype="string
        "/>
</fetchbinding>
```

Desta forma estamos a associar o atributo DICOM com *tag* "00400006" à coluna "scheduledPerformingPhysicianName" e o atributo com *tag* "00080060" à coluna "modality" da tabela "ScheduledProceduresSteps".

Tendo sido efectuadas todas as configurações necessárias, podemos executar o Healthy Talkie, neste exemplo recorrendo ao IDE Eclipse versão Europa. O produto Healthy Talkie não disponibiliza de momento uma interface gráfica, pelo que cesso consultar as mensagens texto apresentadas na consola. Devemos entguardar que a *thread* Receive esteja a correr, o que nota pelo aparecimento da mensagem:

```
[Receiver - Dicom MWL In] - INFO  - Receive starting.
```

Podemos efectuar neste momento o pedido de informação por meio da aplicação ModalitySCU, escolhendo como identificador da modalidade o valor "modalidadeB". Para concretizar o pedido basta pressionar o botão "Query WL".

De imediato é apresentado no ecrã o resultado da pesquisa como exemplificado na Figura 3.32.

Como podemos constatar este teste foi concluído com sucesso, visto que os dados retornados são os esperados. A mensagem de *log* disponibilizada pela aplicação ModalitySCU descreve com detalhe todas as operações realizadas durante a comunicação.

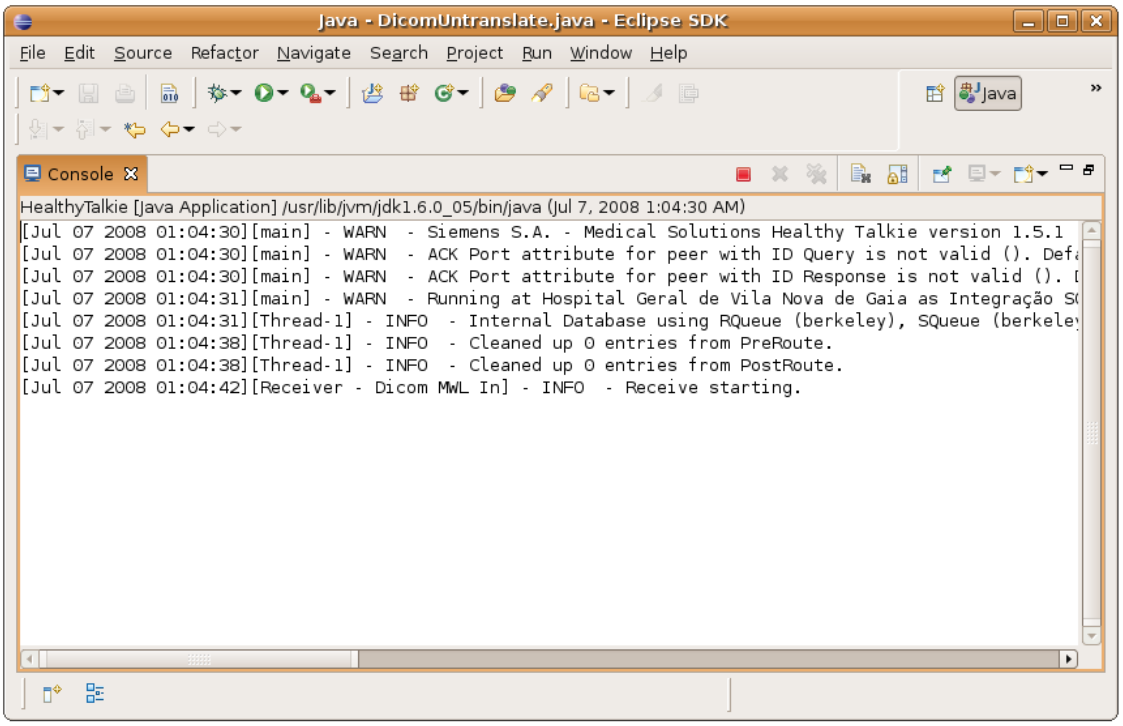


Figura 3.30: Execução do Healthy Talkie

```
1  jdicom: #18:RisServer << A-ASSOCIATE-RQ PDU
2  jdicom: *** request ***
3  application context UID: null
4  called title: RisServer
5  calling title: ModalitySCU
6  max pdu size: 32768
7  max operation invoked: 1
8  max operation performed: 1
9  implementation class UID: 1.2.826.0.1.3680043.2.60.0.1
10 implementation version Name: softlink_jdt103
11 abstract syntax                  scu scp
12 1.2.840.10008.1.1                -1  -1
13 1.2.840.10008.5.1.4.31           -1  -1
14 1.2.840.10008.3.1.2.3.3          -1  -1
15 nr   abstract syntax                  pcid  description
16 0    1.2.840.10008.1.1                1     Verification SOP Class
17   ts-0     1.2.840.10008.1.2                Implicit VR Little Endian Transfer Syntax
18 1    1.2.840.10008.5.1.4.31           3     Modality Worklist Information Model - FIND SOP Class
19   ts-0     1.2.840.10008.1.2                Implicit VR Little Endian Transfer Syntax
20 2    1.2.840.10008.3.1.2.3.3          5     Modality Performed Procedure Step SOP Class
21   ts-0     1.2.840.10008.1.2                Implicit VR Little Endian Transfer Syntax
22 ***************
23 Waiting for AssociationRsp
24 ASSOCIATE_ACKNOWLEDGE detected
25 jdicom: #18:RisServer >> A-ASSOCIATE-AC PDU
26 jdicom: *** acknowledge ***
27 max pdu size: 32768
28 max operation invoked: 1
29 max operation performed: 1
30 implementation class UID: 1.2.826.0.1.3680043.2.60.0.1
31 implementation version name: softlink_jdt103
32 abstract syntax                  scu scp
33 nr   pcid  result                          transfer syntax
34 0    1     unsupported abstract syntax
35 1    3     accepted                        1.2.840.10008.1.2
36 2    5     unsupported abstract syntax
37 ******************
```

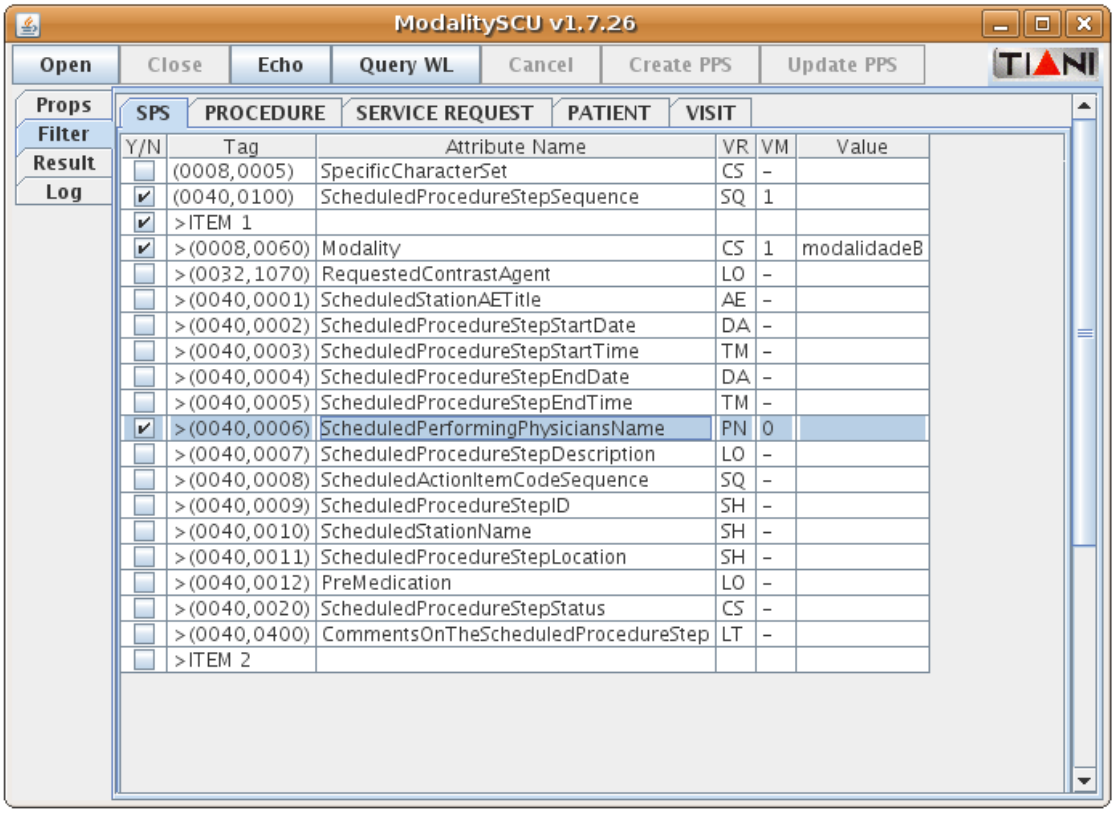


Figura 3.31: Aplicação TIANI Modality SCU com valores do pedido

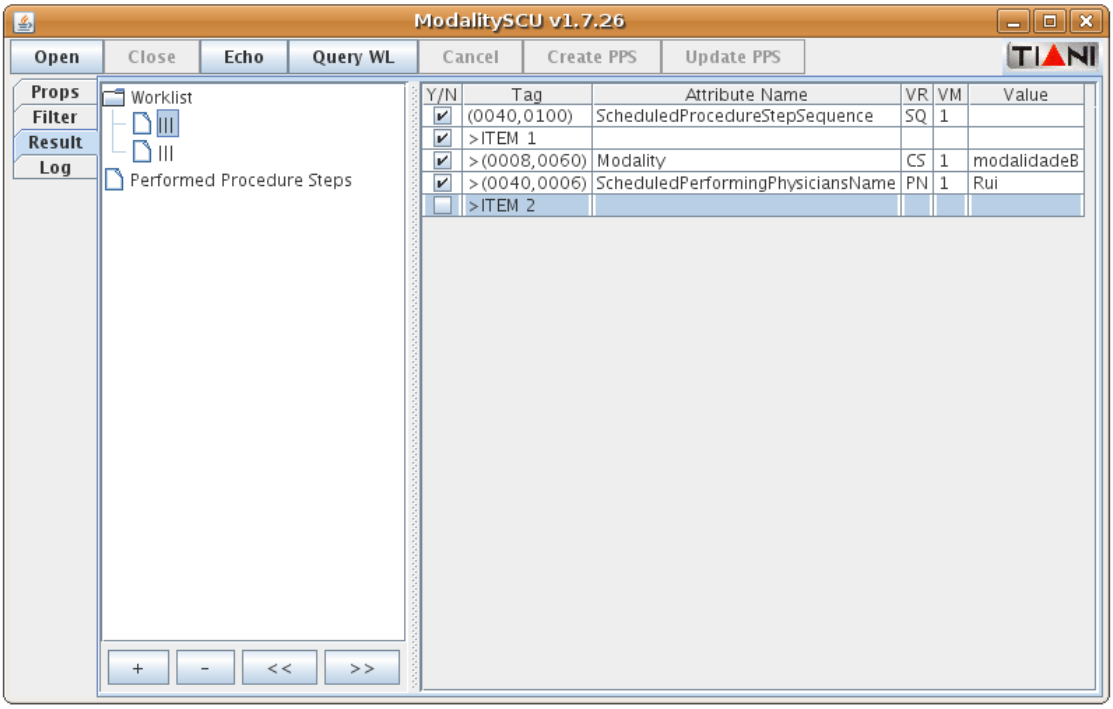


Figura 3.32: Aplicação TIANI Modality SCU com o resultado da pesquisa

```
38 jdicom: RisServer Enter DimseExchange.run()
39 jdicom: RisServer Waiting for PDU
40 jdicom: #18:RisServer << C-FIND-RQ Modality Worklist Information Model - FIND SOP Class
41 (0000,0002) UI [1.2.840.10008.5.1.4.31]                    #  22       1  AffectedSOPClassUID
42 (0000,0100) US [32]                                        #  2        1  CommandField
43 (0000,0110) US [1]                                         #  2        1  MessageID
44 (0000,0700) US [0]                                         #  2        1  Priority
45 (0000,0800) US [65278]                                     #  2        1  DataSetType
46 jdicom: #18:RisServer << Dataset
47 (0040,0100) SQ [
48       (0008,0060) CS [modalidadeB]                         #  12       1  Modality
49       (0040,0006) PN []                                    #  0        0
             ScheduledPerformingPhysiciansName
50 ]                                                          #  52       1
     ScheduledProcedureStepSequence
51 jdicom: RisServer PDU received
52 jdicom: #18:RisServer >> C-FIND-RSP Modality Worklist Information Model - FIND SOP Class, status
       #0000H[Success]
53 (0000,0002) UI [1.2.840.10008.5.1.4.31]                    #  22       1  AffectedSOPClassUID
54 (0000,0100) US [32800]                                     #  2        1  CommandField
55 (0000,0120) US [1]                                         #  2        1
     MessageIDBeingRespondedTo
56 (0000,0800) US [65278]                                     #  2        1  DataSetType
57 (0000,0900) US [0]                                         #  2        1  Status
58 jdicom: #18:RisServer >> Dataset
59 (0040,0100) SQ [
60       (0008,0060) CS [modalidadeB]                         #  12       1  Modality
61       (0040,0006) PN [Rui]                                 #  4        1
             ScheduledPerformingPhysiciansName
62 ]                                                          #  56       1
     ScheduledProcedureStepSequence
63 jdicom: RisServer Waiting for PDU
64 jdicom: #18:RisServer << A-RELEASE-RQ PDU
65 jdicom: RisServer PDU received
66 jdicom: #18:RisServer >> A-RELEASE-RP PDU
67 jdicom: #18:RisServer closing socket
68 jdicom: RisServer Leave DimseExchange.run()
```

Capítulo 4

Conclusões e Trabalho Futuro

O Projecto durou vinte semanas que foram passadas numa instituição que disponibilizou todos os recursos possíveis e conhecimentos numa área (Saúde) que para nós seria desconhecida. Para a conclusão deste projecto é importante realizar uma reflexão muito objectiva sobre todo o trabalho desenvolvido, analisando de que forma os objectivos estipulados inicialmente pela Siemens MED foram cumpridos com a solução DCM4HT e qual a orientação a ser tomada em trabalho futuro.

4.1 Satisfação dos Objectivos

Os objectivos inicialmente propostos para a solução DCM4HT são orientados para responder às necessidades do Healthy Talkie como produto em desenvolvimento. Assim, as decisões tomadas visam a obtenção de resultados a médio ou longo prazo, não se reduzindo à longevidade deste projecto do Mestrado Integrado.

O protocolo DICOM, devido à sua complexidade, exige um esforço inicial que permita ao programador conhecer o vasto conjunto de definições desta norma: os objectos que podem ser alvo de comunica, e os serviços que podemos usufruir, mas pricipalmente, como estes deverão ser implementados. Só posteriormente a esta fase, que envolveu aproximadamente nove semanas de trabalho, é que foi possível determinar quais seriam as nossas prioridades para o restante período de desenvolvimento.

A solução desenvolvida, fruto deste projecto e apresentada neste relatório, pode ser sumarizada nos seguintes pontos:

- Implementação de todos os mecanismos de comunicação de rede do protocolo DICOM de raíz, sem recurso a APIs externas, nxistindo qualquer tipo de dependências;

- Através do processo de associação das entidades DICOM SCU e SCP, garantimos que apenas os dispositivos reconhecidos pelo utilizador poderão comunicar com o Healthy Talkie;
- Adaptação do núcleo do Healthy Talkie de modo a ser capaz de processar uma mensagem DICOM com base em regras definidas em XML;
- Implementação e teste do serviço DICOM *Modality Worklist*. Este inclui a recepção e processamento da mensagem C-FIND-RQ e a constru da resposta (C-FIND-RSP) com recurso a uma base de dados externa para simulação de um RIS.

4.2 Trabalho Futuro

Os possíveis melhoramentos ao sistema DCM4HT consistem na sua maioria no suporte a um maior número de serviços DICOM como:

- *Store*
- *Query/Retieve*
- *Storage Commitment*
- *Modality Performed Procedure Step*

Para além da implementação destes serviços é necessário garantir que a conversão de uma mensagem DICOM para um outro protocolo como HL7 ou CSV é correctamente realizada. Sendo o Healthy Talkie uma interface de sistemas clínicos é vital que a integridade da informação seja mantida e portanto é um mecanismo que deverá ser revisto e testado no futuro.

Referências

[1] Steven C. Horiil, Fred W. Prior, W. Dean Bidgood, Charles Parisot, and Geert Claeys. The dicom standard, 2008. http://www.dicomanalyser.co.uk/html/introduction.htm, acedido a última vez em 2 de Junho de 2008.

[2] Keith J. Dreyer, David S. Hirschorn, James H. Thrall, and Amit Mehta (Editor). *PACS: A Guide to the Digital Revolution*. Springer;, Second edition, 2005.

[3] Jonathan Whitby. (white paper)the dicom standard, 2007. www.barco.com/barcoview/downloads/TheDICOMstandard_v2.pdf.

[4] H. K. Huang. *PACS and Imaging Informatics: Basic Principles and Applications*. Wiley-Liss;, First edition, 2004.

[5] Estelle M. Philips and Derek S. Pugh. *Digital Imaging and Communications in Medicine (DICOM): A Practical Introduction and Survival Guide*. Springer;, First edition, 2008.

[6] H. K. Huang. *PACS: Basic Principles and Applications*. Wiley-Liss;, First edition, 1998.

[7] NEMA. The dicom standard, 2008. ftp://medical.nema.org/medical/dicom/2008/, acedido a última vez em 1 de Julho de 2008.

Anexo A

Detalhes do protocolo DICOM

A.1 Estrutura das principais mensagens

A-ASSOCIATE-RQ PDU/A-ASSOCIATE-AC PDU

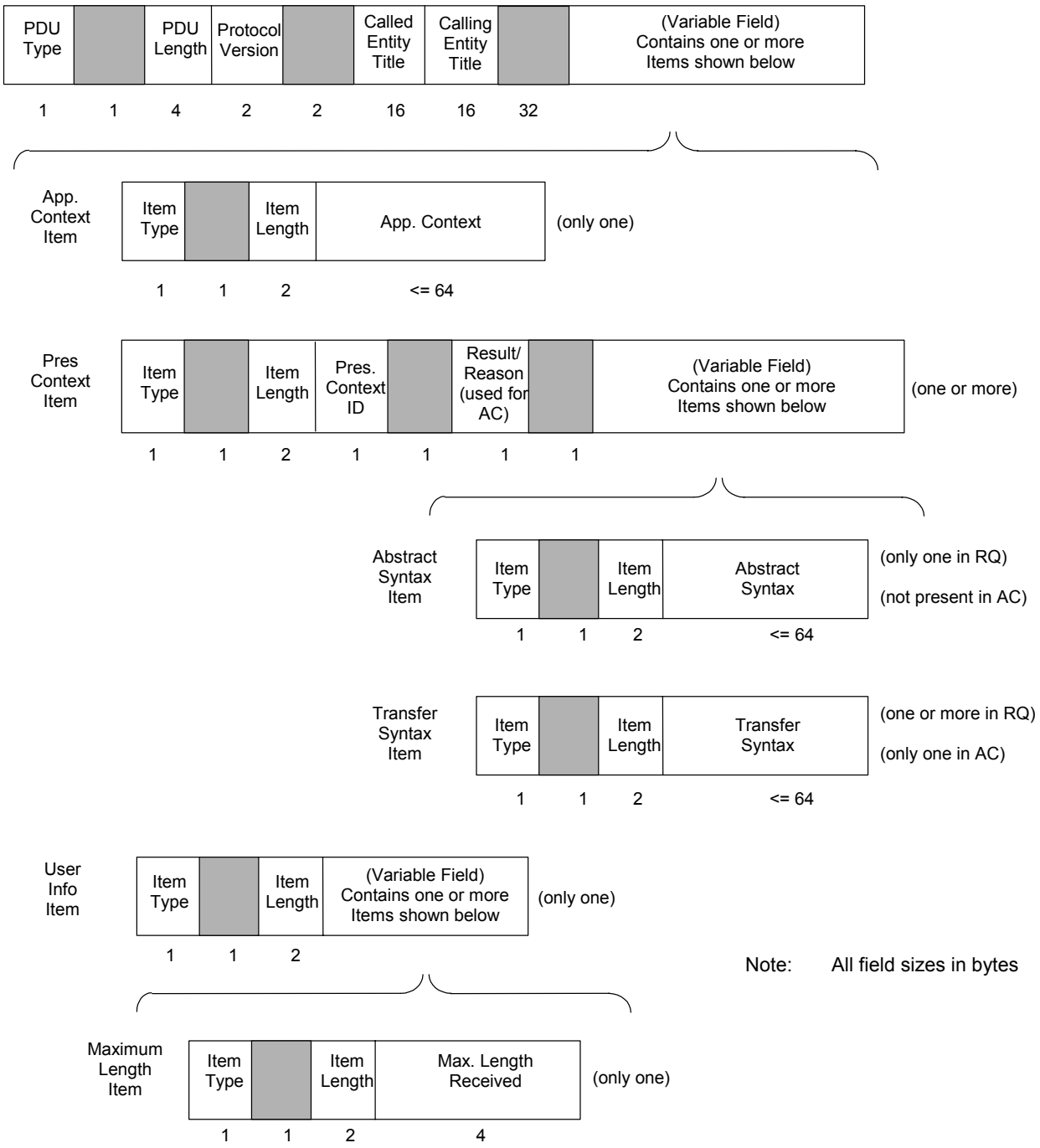
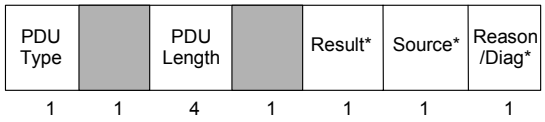


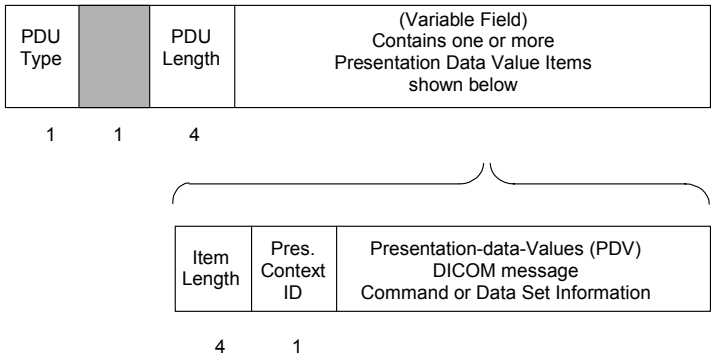
Figure 9-1
PROTOCOL DATA UNITS STRUCTURE AND ENCODING

A-ASSOCIATE-RJ PDU/A-RELEASE-RQ PDU/A-RELEASE-RP PDU/A-ABORT PDU



*Note: Depending on the specific PDU this field may be used or reserved.

P-DATA-TF PDU



Note: All field sizes in bytes

Figure 9-2
PROTOCOL DATA UNITS STRUCTURE AND ENCODING

9.3.2 A-ASSOCIATE-RQ PDU STRUCTURE

An A-ASSOCIATE-RQ PDU shall be made of a sequence of mandatory fields followed by a variable length field. Table 9-11 shows the sequence of the mandatory fields.

The variable field shall consist of one Application Context Item, one or more Presentation Context Items, and one User Information Item. Sub-Items shall exist for the Presentation Context and User Information Items.