Universidade de Aveiro Departamento de
2018 Eletrónica, Telecomunicações e Informática

**Tiago
Marques Godinho**

Métodos computacionais para otimização de desempenho em redes de imagem médica

Computer methods for performance optimization in medical imaging networks

Universidade de Aveiro Departamento de
2018 Eletrónica, Telecomunicações e Informática

Tiago Marques Godinho

Métodos computacionais para otimização de desempenho em redes de imagem médica

Computer methods for performance optimization in medical imaging networks

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Engenharia Informática, realizada sob a orientação científica de Carlos Manuel Azevedo Costa, Professor do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Apoio financeiro da FCT e do FSE, MEC, POCH, MCTES no âmbito do III Quadro Comunitário de Apoio.

Mais uma vez, dedico este trabalho à minha irmã, por todo o carinho, compreensão e amizade incondicional que fazem com que a próxima etapa seja apenas mais uma.

agradecimentos / acknowledgements

Um agradecimento muito especial à minha família, por me terem sempre proporcionado alcançar mais esta etapa.
A todos os membros do grupo de bioinformática, pelo espetacular ambiente de cooperação.
Um agradecimento a todos os professores, que durante toda a minha vida contribuíram para o meu crescimento enquanto pessoa. Em particular ao professor Carlos Costa e ao professor José Luís Oliveira por mais uma vez terem aberto as portas à realização deste trabalho.

Este programa doutoral foi financiado pela Fundação para a Ciência e a Tecnologia (FCT) no âmbito do contrato de bolsa de doutoramento SFRH/BD/104647/2014.

A very warm-hearted acknowledgement to my family for their unconditional support, which has been fundamental.
To all the IEETA Bioinformatics group members, for their cooperation and outstanding working environment.
An acknowledgement to all the professors who throughout my whole life have contributed to my growth as a human being. Most notably, to professor Carlos Costa and professor José Luis Oliveira for, once again, opening the doors to this work.

This doctoral program has been financed by the Portuguese National Science Foundation (Fundação para a Ciência e a Tecnologia - FCT) under the grant agreement SFRH/BD/104647/2014.

Palavras-chave

Redes de Imagem Médica, Pesquisa, Distribuição, Desempenho e Optimização

Resumo

As diversas modalidades de imagem médica têm vindo a consolidar a sua posição dominante como meio complementar de diagnóstico. O número de procedimentos realizados e o volume de dados gerados aumentou significativamente nos últimos anos, colocando pressão nas redes e sistemas que permitem o arquivo e distribuição destes estudos. Os repositórios de estudos imagiológicos são fontes de dados ricas contendo dados semiestruturados relacionados com pacientes, patologias, procedimentos e equipamentos. A exploração destes repositórios para fins de investigação e inteligência empresarial, tem potencial para melhorar os padrões de qualidade e eficiência da prática clinica.

No entanto, estes cenários avançados são difíceis de acomodar na realidade atual dos sistemas e redes institucionais. O pobre desempenho de alguns protocolos standard usados em ambiente de produção, conduziu ao uso de soluções proprietárias nestes nichos aplicacionais, limitando a interoperabilidade de sistemas e a integração de fontes de dados.

Este doutoramento investigou, desenvolveu e propõe um conjunto de métodos computacionais cujo objetivo é maximizar o desempenho das atuais redes de imagem médica em serviços de pesquisa e recuperação de conteúdos, promovendo a sua utilização em ambientes de elevados requisitos aplicacionais. As propostas foram instanciadas sobre uma plataforma de código aberto e espera-se que ajudem a promover o seu uso generalizado como solução vendor-neutral. As metodologias foram ainda instanciadas e validadas em cenários de uso avançado. Finalmente, é expectável que o trabalho desenvolvido possa facilitar a investigação em ambiente hospitalar de produção, promovendo, desta forma, um aumento da qualidade e eficiência dos serviços.

Keywords Medical Imaging Networks, Discovery, Retrieval, Performance and Optimization

Abstract

Over the last few years, the medical imaging has consolidated its position as a major mean of clinical diagnosis. The amount of data generated by the medical imaging practice is increasing tremendously. As a result, repositories are turning into rich databanks of semi-structured data related to patients, ailments, equipment and other stakeholders involved in the medical imaging panorama. The exploration of these repositories for secondary uses of data promises to elevate the quality standards and efficiency of the medical practice.

However, supporting these advanced usage scenarios in traditional institutional systems raises many technical challenges that are yet to be overcome. Moreover, the reported poor performance of standard protocols opened doors to the general usage of proprietary solutions, compromising the interoperability necessary for supporting these advanced scenarios.

This thesis has researched, developed, and now proposes a series of computer methods and architectures intended to maximize the performance of multi-institutional medical imaging environments. The methods are intended to improve the performance of standard protocols for medical imaging content discovery and retrieval. The main goal is to use them to increase the acceptance of vendor-neutral solutions through the improvement of their performance. Moreover, it intends to promote the adoption of such standard technologies in advanced scenarios that are still a mirage nowadays, such as clinical research or data analytics directly on top of live institutional repositories. Finally, these achievements will facilitate the cooperation between healthcare institutions and researchers, resulting in an increment of healthcare quality and institutional efficiency.

List of contents

List of contents **i**

List of figures **v**

List of tables **vii**

List of acronyms **ix**

1 Introduction **1**

1.1 Overview . . . 1
1.2 Motivation . . . 2
1.3 Objectives . . . 3
1.4 Outline . . . 4
1.5 Scientific Results . . . 5
1.5.1 International Journals . . . 5
1.5.2 International Conferences . . . 5

2 Medical Imaging Laboratories **7**

2.1 Medical Imaging Workflows . . . 8
2.2 Picture Archive and Communication Systems . . . 9
2.3 Standards . . . 11
2.3.1 DICOM . . . 11
2.3.2 HL7 . . . 22
2.3.3 IHE . . . 23
2.4 Current Trends . . . 23

3 Service Performance in Medical Imaging Networks **27**

3.1 Performance and Scalability . . . 27
3.2 Big Data Scenarios . . . 32
3.3 Service Provision Issues . . . 34
3.3.1 Storage . . . 35

3.3.2 Content Discovery . . . 35
3.3.3 Content Retrieval . . . 38
3.4 Technologies for Service Optimization . . . 38
3.4.1 Communications Optimization . . . 39
3.4.2 Cache Mechanisms . . . 40
3.4.3 Information Retrieval . . . 42
3.4.4 Content discovery . . . 44

4 Proposed Methodology **51**
4.1 Medical Imaging Repositories . . . 51
4.1.1 Dicoogle Open-source PACS . . . 52
4.1.2 Results . . . 57
4.2 Content Distribution in Medical Imaging Networks . . . 58
4.2.1 Cache Architecture . . . 59
4.2.2 DICOM Routing Platform with Caching Management . . . 62
4.2.3 Workflow Management . . . 66
4.2.4 Controlled Channels . . . 72
4.2.5 Validation . . . 75
4.3 Content Discovery . . . 79
4.3.1 Performance and Scalability Testing Framework . . . 80
4.3.2 Datasets . . . 82
4.3.3 Test cases . . . 83
4.4 Summary . . . 86

5 Advanced Usage Scenarios **87**
5.1 Digital Pathology . . . 87
5.1.1 Related Work . . . 89
5.1.2 Methods . . . 91
5.1.3 Validation . . . 95
5.2 Research PACS . . . 98
5.2.1 Background and Related Work . . . 99
5.2.2 Architecture . . . 101
5.2.3 Results . . . 105

6 Conclusions **109**

References **111**

Annexes

A Intelligent Generator of Big Data Medical Imaging Repositories **125**

List of figures

1.1 A user oriented perspective of advanced medical imaging networks. 4

2.1 Traditional Workflow in medical imaging laboratories. 9
2.2 Typical information flow in PACS. 10
2.3 DICOM Data Elements structure. 13
2.4 Illustration of nested objects in DICOM. 13
2.5 Simplistic illustration of the DICOM Information Model. 15
2.6 DICOM Storage service. 17
2.7 DICOM Query service. 18
2.8 DICOM Query-Retrieve service. 19
2.9 Single frame format vs Tiled format, and image pyramid. 21

3.1 Performance, Scalability and Cost of ownership. 30
3.2 Illustration of the system's demand throughout its lifetime. 30
3.3 Venn diagram with the logical distribution of results in searches. 37

4.1 Simplified class diagram of Dicoogle SDK's main interfaces. 53
4.2 Dicoogle full-stack architecture. 54
4.3 DICOM Storage service as handle by Dicoogle. 55
4.4 DICOM Query-Retrieve service as handle by Dicoogle. 56
4.5 Caching system architecture. 60
4.6 Caching system software components. 60
4.7 Illustration of content retrieval in the distributed environment. 65
4.8 WMPe architecture and lifecycle. 68
4.9 DICOM Query-Retrieve service from multiple sources. 75
4.10 Analysis of speed-up introduced by our caching system. 78
4.11 Analysis of the impact of increasing load in our distributed PACS environment. 79
4.12 Overview of the different stages of the validation framework. 81
4.13 Illustration of the index test case steps. 84
4.14 Illustration of the query test case steps. 85

5.1 Overview of the proposed architecture. 92
5.2 Information flows in the proposed Pathology PACS architecture. 93
5.3 Business Intelligence framework architecture. 102
5.4 Overview of the Dicoogle BI dashboard. 106
5.5 Sample of rules applied to a view of the dataset. 107
5.6 Overview of the Dicoogle BI visualisation interface. 107

A.1 Illustration of nested objects in DICOM. 128
A.2 Architecture and information flow of the dataset generator. 129
A.3 Distribution of ultrasound studies on the generated datasets 132

List of tables

2.1 Most common DICOM Transfer Syntaxes. 16

4.1 Data profile of the dataset A. 73
4.2 Evaluation of the Controlled Channels method. 76
4.3 Routing and cache mechanism performance evaluation. 77
4.4 DICOM datasets. 83

5.1 Storage size of different WSI formats. 96
5.2 Whole-slide imaging viewer performance trials. 97
5.3 Speed-up achieved by the proposed Pathology PACS architecture. 97

A.1 Comparison of productivity statistics between real and generated datasets . . . 133

List of acronyms

ACR	American College of Radiology
AD	Anomally Detection
AETitle	Application Entity Title
API	Application Interface
ARMA	Autoregressive Moving Average Model
BI	Business Intelligence
CAD	Computer-assisted diagnosis
CBIR	Content Based Imaging Retrieval
CDA	Clinical Document Architecture
CR	X-Rays
CR	Computer Radiography
CSS	Cascading Style Sheets
CSV	Comma separated values
CT	Computed Tomography
DA	Data Analitics
DBMS	Database Management System
DC	Data Cleansing
DICOM	Digital Imaging and Communications in Medicine
DIM	DICOM Information Model
DIMSE	DICOM Message Service Elements
DM	Data Mining
DSS	Digital Slide Scanners
DX	Digital Radiography
EHR	Electronic Heath Records
HDFS	Hierarchical Data Format Store
HIS	Hospital Information System

HL7	Health Level 7
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
ICT	Information and Communication Tecnologies
IE	Information Entities
IHE	Integrating the Healthcare Enterprise
IOD	Information model definition
IP	Internet Protocol
IR	Information Retrieval
IT	Information Technologies
JPIP	JPEG 2000 Interactive Protocol
JS	JavaScript
JSON	JavaScript Object Notation
JSPF	Java Simple Plugin Framework
LRU	Least Recently Used
MPPS	Modality Performed Procedure Step
MR	Magnetic Resonance
MWL	Modality Worklist
NEMA	National Electrical Manufacturers Association
NLP	Natural Language Processing
NM	Nuclear Medicine
NoSQL	Not only SQL
OLTP	Online Transaction Processing
OO	Object-Oriented
PACS	Picture Archive and Communications Systems
PIX	Patient Identifier Cross-Referencing
PT	Positron emission Tomography
QoS	Quality of Service
RDBMS	Relational Database Management System
REST	Representational State Transfer
RIS	Radiology Information System
SCP	Service Class Provider
SCU	Service Class User

SDK	Software Development Kit
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SR	Structured Report
TCP	Transmission Control Protocol
TLV	Tag-Length-Value
UID	Unique Identifier
UL	Upper Layer
URI	Universal Resource Identifier
US	Ultrasounds
VR	Value Representation
WADO	Web Access to DICOM Objects
WMPe	Workflow Management for PACS Environments
WSI	Whole-slide imaging/image
XA	X-Ray Angiography
XDS-I	Cross-Enterprise Document Sharing for images
XPATH	XML Path Language

Chapter 1

Introduction

This chapter presents an introduction to this doctoral thesis. It gives a glimpse of the main issues found in the medical imaging environment and how they influenced the proposed methods. It describes the main goals of this doctorate and summarizes the remaining document structure.

1.1 Overview

Information and Communication Tecnologies (ICT) have been major drivers of the progress of western societies. The progress obtained with introduction of ICT are significant, including the reduction of operational costs, the improvement of services quality and accessibility. These benefits are transversal to distinct areas of society and the healthcare sector is no exception. In the recent years, many information systems such as Electronic Heath Records (EHR), Hospital Information System (HIS), and Radiology Information System (RIS) have been proposed and deployed. Although their adoption by medical institutions is still increasing, these systems have already enabled the migration of most hospital workflows into the digital era. The outcomes are perceptible in provided services quality, with reduced upkeep and extended population coverage.

During the last decades, the use of digital medical imaging modalities in healthcare institutions has increased expressively [1], being nowadays a valuable support tool for medical diagnosis and treatment. Healthcare practitioners rely on information extracted from medical imaging systems for providing a more efficient clinical practice. ICT have been fundamental in this process [2]. The reality of medical imaging departments was shaped by the coupling of two main efforts; the emergence of multiple information systems for supporting the clinical practice and the proliferation of distinct digital acquisition modalities. The result is a rich ecosystem that includes patient, imaging and pathology related information. Nowadays, the data volume generated by medical imaging procedures has reached unimaginable values [3–6]. For instance, it was estimated that the USA produced over 1 Exabyte of medical imaging data in 2016 [7]. New methods for processing and extraction of information from those data sources are being proposed by the academy and industry, with direct impact on healthcare services quality.

Picture Archive and Communications Systems (PACS) is the common designation for information systems that manage the storage and distribution of medical images [2]. Digital Imaging and Communications in Medicine (DICOM) is the standard used for storage and exchange of structured medical imaging data [8]. DICOM is a crucial element for interconnecting archives, acquisition devices and visualization workstations [2, 9]. The workflows of a modern digital imaging laboratory are associated with two major phases; the archiving of images for posterior use and the distribution of images through the network to satisfy the distinct consumers' requirements. Currently, it is common to have acquisition devices, storage systems and visualization workstations managed by physicians in geographically dispersed places [10, 11]. The result is a constant pressure over PACS communications layer for providing highly efficient mechanisms for storage, search and retrieval of medical imaging examinations and associated data.

The integration of distinct information domains has suffered major incentives through the usage of Cross-Enterprise Document Sharing for images (XDS-I) and the Integrating the Healthcare Enterprise (IHE) initiative [12]. They aim at the interoperation of multiple institutions on a broader level, promoting the sharing of performed studies, patient records, and other resources. By doing so, institutions may benefit from increased quality, efficiency and cost reduction [13].

In the last few years, the healthcare industry has dedicated special attention towards the optimisation of its business processes, particularly, in the provision of medical imaging services. Besides the common usage of a PACS as a centralized image repository, continuous technological evolution has led to the emergence of other intensive data usage scenarios, such as clinical research, Business Intelligence (BI) and Computer-assisted diagnosis (CAD) systems.

1.2 Motivation

Despite presenting clear advantages, when compared to their analogic contenders [14], PACS architectures are still far from perfect. Regardless of the standardisation efforts, medical institutions still act as isolated islands, allowing limited mobility of patients, studies, and professionals, which contradicts the patient-centred care model [13, 15]. The lack of integration capabilities in state of the art solutions may be explained, not only by vendor's business strategies but also by some inefficiency of current standards in shared environments (outside institutions) [12, 13, 16].

The performance constraints imposed by distributed medical imaging workflows is a crucial issue that may condition the system acceptance among the medical personnel [13]. The huge amount of data involved in medical imaging procedures raises many technical and technological challenges, commonly referenced as Big Data issues [3, 5]. These problems are even more notorious in multi-institutional environments not only because the absolute amount of data is larger, but also because of the overhead introduced in communications [4].

Nonetheless, small-scale imaging repositories are not effective for research purposes. In addition, automated business analytics processes are still absent in the medical imaging field. Not fully exploring the potential held by medical imaging data, inevitably leads to inferior quality healthcare services.

Our research tackles the main causes behind the delayed adoption of advanced imaging networks (as illustrated in Figure 1.1); the poor performance of standard protocols and the absence of validated technologies that enable affordable and riskless deployment of these networks.

1.3 Objectives

This doctoral program researched **new computer methods and scalable architectures** for supporting efficient data access in distributed medical imaging scenarios, fulfilling the requirements raised by **advanced usage scenarios**, such as **digital pathology**, **research**, and **business intelligence**. The proposal is compliant with nowadays medical imaging workflows and manages the big-data issues raised by advanced usage scenarios.

The research must be focused on the **optimization of storage, and content discovery and retrieval processes**. It must provide a highly modular and scalable architecture capable of encompassing different vendor components and satisfying distinct categories of functional and integration requirements. On the other perspective, a methodology to **facilitate the research, development, and validation of new components** must also be proposed. Our goal is to use the researched methods to **increase the acceptance and usage of vendor-neutral solutions**. Ultimately, the proposed methods should be able to **support advanced usage scenarios**, such as research and other secondary uses of data directly on top of live institutional repositories.

In the functional point of view, the resulting architecture must be optimized for supporting distributed workflows, including the ubiquitous deployment of PACS infrastructures external to medical institutions, such as cloud providers. Moreover, the proposal must be scalable, concerning the number of entities interacting and the volume of data manipulated, without compromising the system performance.

Concerning the integration issues, the architecture should rely on vendor-neutral protocols, allowing the reuse of existing institutional resources, infrastructures and applications. Furthermore, the proposed computational methods should be decoupled into various modules, allowing them to be independently used in currently deployed architectures.

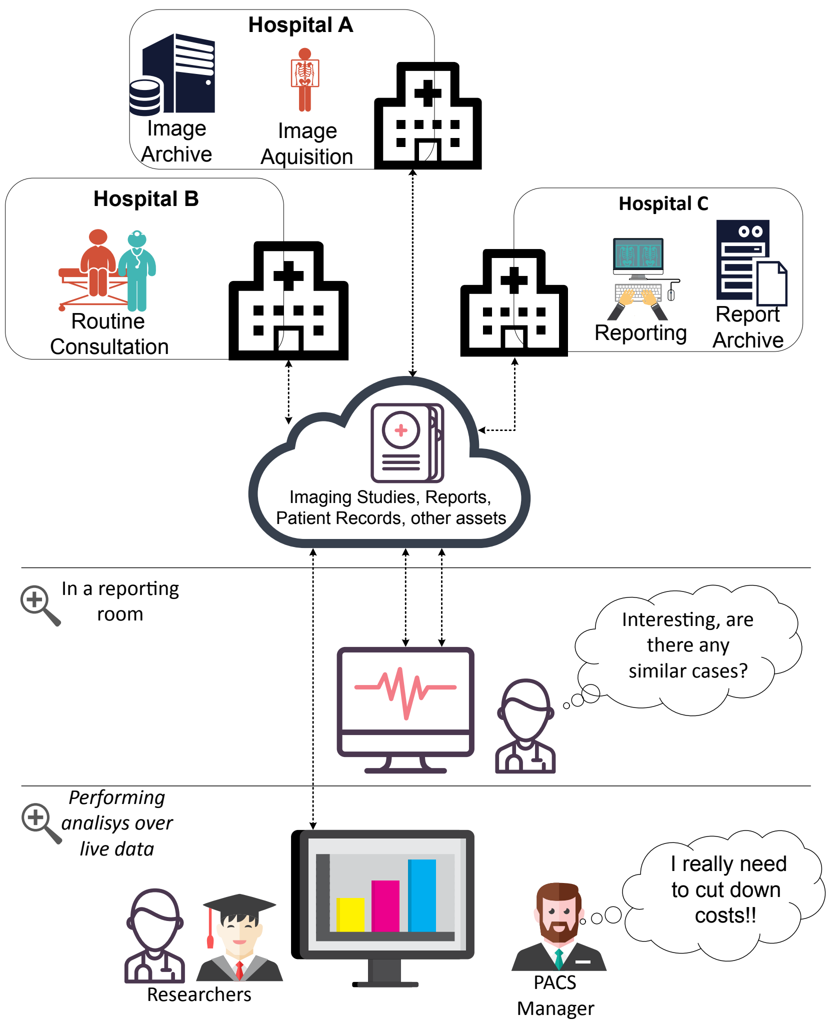

Hospital A
Image Archive
Image Aquisition
Hospital B
Routine Consultation
Hospital C
Reporting
Report Archive
Imaging Studies, Reports, Patient Records, other assets
In a reporting room
Interesting, are there any similar cases?
Performing analisys over live data
Researchers
PACS Manager
I really need to cut down costs!!

Figure 1.1: A user oriented perspective of advanced medical imaging networks.

1.4 Outline

The remaining document is organized into the following chapters:

- Chapter 2 — Provides a careful description of the current paradigm in medical image laboratories with special emphasis to the information systems used in such environments.
- Chapter 3 — Describes the concepts of performance and scalability. Provides a thorough description of the issues raised by the exploitation of information systems in the medical image panorama. Moreover, important technologies for the course of this thesis are described. Finally, the related work research is presented.
- Chapter 4 — Details the contributions of this doctoral program regarding the performance optimization of PACS. This chapter is organized according to the main

tasks associated with the PACS operation, namely content discovery and retrieval. An emphasis is also given to the framework that powered the research, development and validation of methods proposed, the Dicoogle.

- Chapter 5 — Describes and validates the proposed methodology in two highly demanding PACS usage scenarios.
- Chapter 6 — presents the thesis final remarks and suggests future work directions.

1.5 Scientific Results

This doctoral program has contributed to the fields of medical informatics and computer science, more especially to performance engineering, with the following publications:

1.5.1 International Journals

- T. M. Godinho, R. Lebre, L. B. Silva, and C. Costa, “An efficient architecture to support digital pathology in standard medical imaging repositories,” Journal of Biomedical Informatics, vol. 71, pp. 190–197, 2017.
- T. M. Godinho, C. Costa, and J. L. Oliveira, “Intelligent generator of big data medical imaging repositories,” IET Software, Feb. 2017.
- F. Valente, L. A. B. Silva, T. M. Godinho, and C. Costa, “Anatomy of an Extensible Open Source PACS,” Journal of Digital Imaging, vol. 29, no. 3, pp. 284–296, Jun. 2016.
- E. Pinho, T. Godinho, F. Valente, and C. Costa, “A Multimodal Search Engine for Medical Imaging Studies,” J Digit Imaging, pp. 1–10, Aug. 2016.
- T. M. Godinho, C. Viana-Ferreira, L. A. Bastiao Silva, and C. Costa, “A Routing Mechanism for Cloud Outsourcing of Medical Imaging Repositories,” IEEE Journal of Biomedical and Health Informatics, vol. 20, no. 1, pp. 367–375, Jan. 2016.

1.5.2 International Conferences

- J. M. Silva, T. Marques Godinho, D. Silva, and C. Costa, “Web Validation Service for Ensuring Adherence to the DICOM Standard,” Stud Health Technol Inform, vol. 235, pp. 38–42, 2017.
- M. Santos, J. Pavao, T. Godinho, and N. P. Rocha, “DICOM metadata aggregation from multiple healthcare facilities,” 2017, pp. 1–4.

- M. Valerio, T. M. Godinho, and C. Costa, “User Oriented Platform for Data Analytics in Medical Imaging Repositories,” Stud Health Technol Inform, vol. 228, pp. 717–721, 2016.
- P. Matos, L. A. Bastiao Silva, T. M. Godinho, and C. Costa, “A Dynamic Approach to Support Interoperability for Medical Reports Using DICOM SR,” Stud Health Technol Inform, vol. 228, pp. 461–465, 2016.
- T. M. Godinho, C. Costa, and J. L. Oliveira, “Generating Big Data repositories for research in medical imaging,” in 2016 11th Iberian Conference on Information Systems and Technologies (CISTI), 2016, pp. 1–5.
- T. M. Godinho, E. Almeida, L. A. B. Silva, and C. Costa, “Integrating multiple data sources in a cardiology imaging laboratory,” 2016, pp. 1–6.
- A. P. Alves, T. M. Godinho, and C. Costa, “Assessing the relational database model for optimization of content discovery services in medical imaging repositories,” 2016, pp. 1–6.
- T. M. Godinho, L. M. Silva, and C. Costa, “An automation framework for PACS workflows optimization in shared environments,” 2015, pp. 1–7.
- T. M. Godinho, L. A. Bastião Silva, C. Costa, and J. L. Oliveira, “Multi-provider architecture for cloud outsourcing of medical imaging repositories,” Stud Health Technol Inform, vol. 205, pp. 146–150, 2014.
- T. M. Godinho, L. A. B. Silva, C. Viana-Ferreira, C. Costa, and J. L. Oliveira, “Enhanced regional network for medical imaging repositories,” in 2013 8th Iberian Conference on Information Systems and Technologies (CISTI), 2013, pp. 1–6.

Chapter 2

Medical Imaging Laboratories

This chapter provides an insight into the current paradigm in digital medical imaging laboratories, including a description of its stakeholders, systems and most important workflows. From a technological perspective, a detailed description of the standards and system architectures is provided. A special focus is given to PACS and the DICOM standard due to their current role in the field. The information provided in this chapter is crucial in order to correctly understand the proposed methods.

Medical imaging procedures are common elements in nowadays clinical practice and even the small diagnostic centres have a set of basic modalities. They figure as a powerful mean of diagnosis for a variety of health conditions. Medical imaging is formally defined as the technique and process used to acquire visual representations of the human body [17]. This field is tightly coupled with clinical areas such as Radiology and Nuclear Medicine, which are also common image acquisition techniques. In the western countries, medical imaging modalities such as X-Rays (CR), Computed Tomography (CT), Magnetic Resonance (MR) and Ultrasounds (US) are commonly recognized by the general public.

Medical imaging has been used for quite a long time. More than a century has passed since the discovery of X-Rays, the pioneering technique of acquiring images of the human body. Since then, the advancements in the medical imaging field have been astonishing, especially in the late 20th century with the advent of digital modalities. In the beginning, operating medical imaging laboratories was a hard manual enterprise, the acquisition devices were very expensive and the images printed in films. The availability of medical imaging studies was limited by the physical access to analogic storage mediums and associated administrative processes. Moreover, the medical imaging centres were located only in major cities and sharing studies was a manual process.

The proliferation of medical imaging modalities was driven by the emergence of modern ICT that were progressively explored by manufacturers of medical imaging modalities and information systems. On the one hand, PACS and RIS platforms started to appear in the market, thus, making the exploitation of medical image laboratories more financially appealing [18]. On the other hand, modern digital modalities, such as CT, have pushed up the quality of diagnostic imaging to a new level supported by powerful computational tools that included 3D imaging manipulation.

The improvements introduced by the digital era in the exploitation of medical imaging laboratories are evident. However, the technical and operational requirements, including specialized human resources, also increased considerably as evidenced in [19]. Nowadays, the PACS industry is focused not only on the performance of regular operations, but also in the added value to the quality of diagnosis and treatment [3, 4, 12, 13, 20]. In this picture, the efficient provision and integration of multiple institutional PACS is a major target. However, this issue faces complex challenges, including the technical ones associated with the tremendous amount of data generated by medical imaging laboratories [20].

2.1 Medical Imaging Workflows

This section provides a brief explanation of the main workflows in a medical imaging laboratory. The knowledge of actors and workflows in a traditional radiology department (Figure 2.1) is of major importance, since the workflows associated to the data production, exchange and usage, greatly influence the planning of the underlying information systems, most notably PACS. A major aspect of these workflows is the high usage of query, storage and retrieval services involving distinct PACS components but also third systems like, for instance, the RIS [21].

The basic workflow in a radiology department involves three major stages: Admission, Examination and Reporting.

- In the admission stage, information related to the patient is registered in the RIS and the requested examination is scheduled. The registered information has a more administrative nature. Nonetheless, added value may be generated by exploring these data.

- During the examination stage, the imaging procedure is conducted. Initially, the acquisition equipment is calibrated by technicians according to the patient profile and the procedure specifics. Then, the equipment produces the images, which are then stored in the institution's repository.

- The last stage is the revision process. During this process, a physician retrieves the desired study to its workstation according to his worklist. The produced artefacts, such as other images or reports, are stored afterwards in the PACS archive or in the RIS. When PACS are not available, the reviewing process is based on analogic devices, i.e., radiology film (Figure 2.1).

2.2 Picture Archive and Communication Systems

PACS defines a set of hardware, software and communications technologies for acquiring, storing, distributing and analysing digital medical images [2]. It is one of the most important advances in the medical imaging field, allowing to support the digital workflows in radiology departments. As illustrated in Figure 2.2, there are three main types of components: Aquisition equipment, Repositories and Viewer Applications that support three important groups of processes: Acquisition, Distribution and Visualization [2].

As described in the previous section, the acquisition process is ensured by specialised equipment. There are two common methodologies associated with the acquisition of medical images. The first method involves acquiring images directly from digital equipment. The second involves scanning of analogic films printed by other equipment. This method is traditionally used to make sure that previously performed studies are not lost. However, as medical imaging studies are increasingly stored and exchanged electronically, the usage of this method tends to disappear. After being acquired, medical imaging studies are stored in repositories commonly denominated as PACS archives.

Distribution is the process of exchanging images or studies among the different PACS components. As previously stated, one of the key aspects of radiology workflows is that they demand a heavy exchange of information between different actors and systems. Therefore,

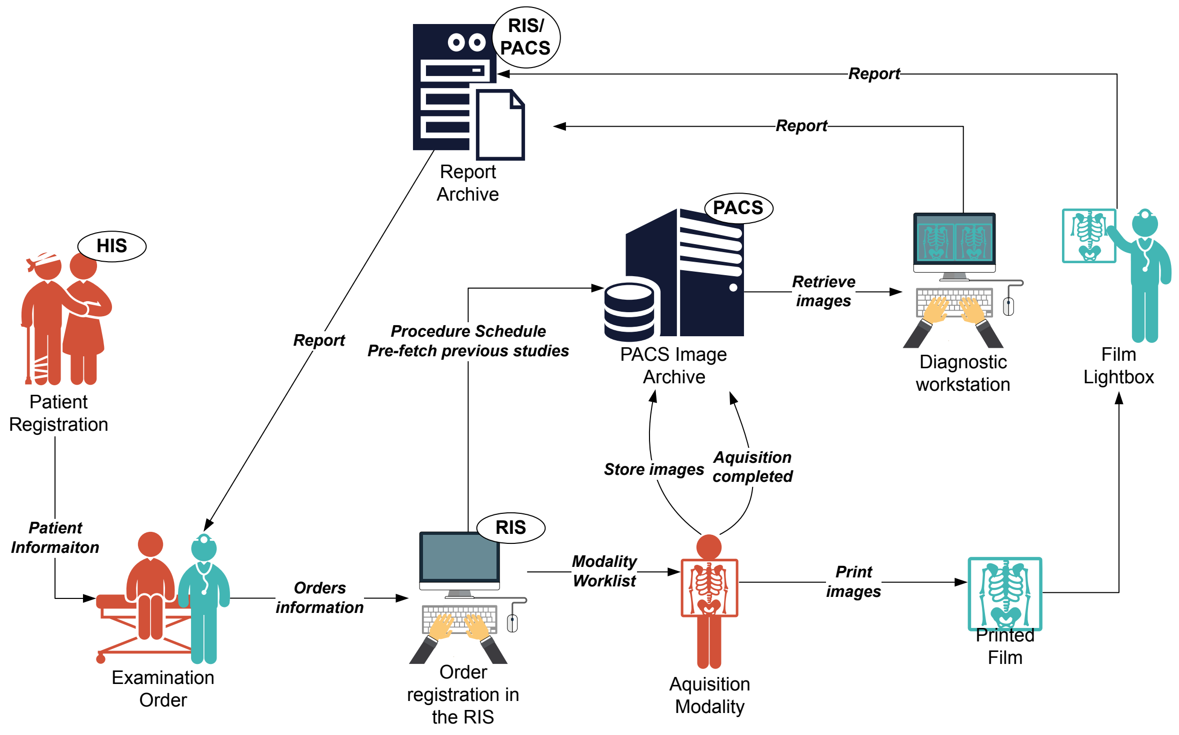


Figure 2.1: Traditional Workflow in medical imaging Laboratories. Interaction between the PACS, RIS and HIS.

medical imaging studies need to be moved along the different PACS components. As a result, the distribution layer performance is key to the general PACS Quality of Service (QoS).

The visualization of medical imaging studies is of paramount importance since physicians rely on this process to make accurate decisions. Viewer workstations must provide efficient tools for content discovery and retrieval. Moreover, visual representations of data must be done according to standards and medical guidelines. The modularity of PACS architectures should enable the usage of the best breed applications for each task. For instance, the usage of third-party viewer applications, such as, Osirix[1] or K-PACS[2], which are not provided by acquisition equipment manufacturers, is rather common.

The implementation of PACS in medical imaging laboratories provides better integration for the different stages of the radiology workflow. Moreover, the medical practice performance is improved by facilitating access to relevant clinical information [18]. The integration of different PACS components, especially PACS archives, in the institutional workflows may vary greatly from an institution to another. Nonetheless, three general architectures for the integration of the institutional workflows within the PACS are presented in [2]: Stand-alone, Client-Server and Web-based architecture.

[1] Osirix Viewer: http://www.osirix-viewer.com/ ; accessed in Jan-2015.
[2] K-PACS: http://www.k-pacs.net/ ; accessed in Jan-2015

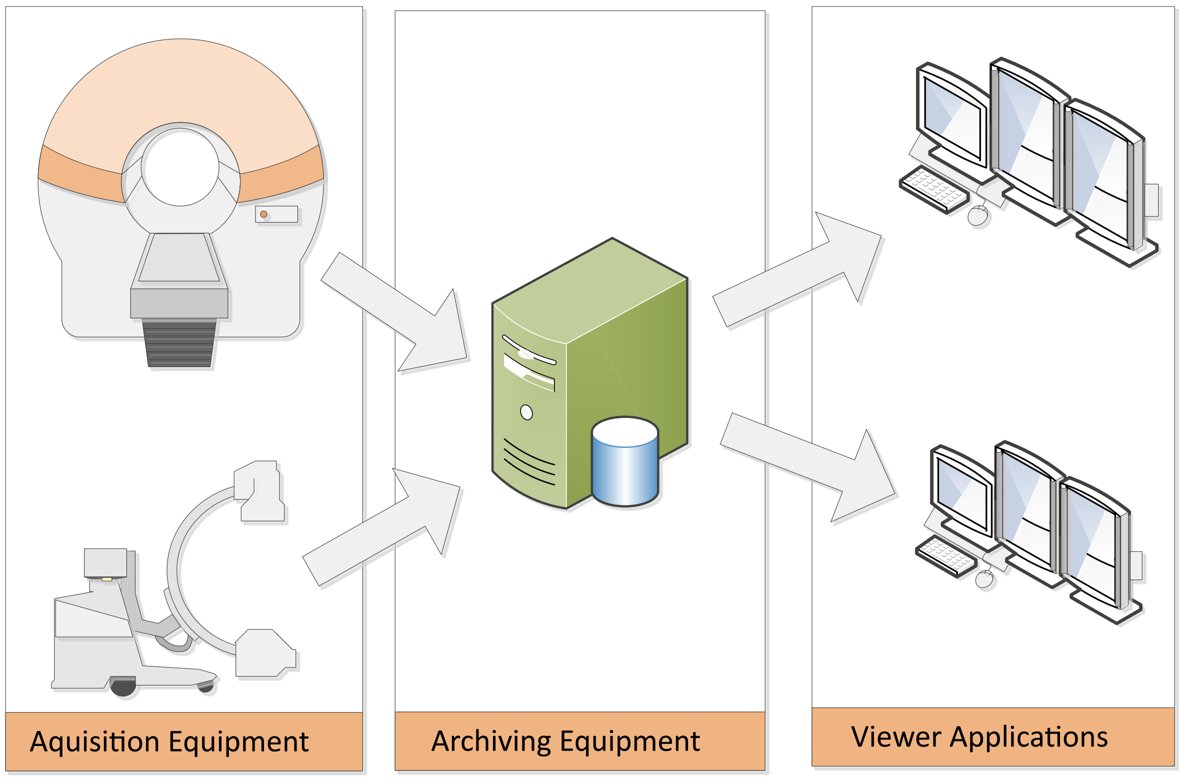


Figure 2.2: Typical information flow in PACS.

The stand-alone architecture involves a store-and-forward approach, as images acquired during procedures are immediately sent to the main archive and immediately forwarded to the PACS workstations without requiring any interaction. A physician would then analyse and report the studies in these workstations. Although very simple, this architecture is efficient because studies are typically moved to the physician's workstations earlier than required. Nonetheless, this architecture raises many security considerations since unauthorized personnel may have access to the patient's studies as they are automatically broadcasted instead of being requested on demand by a credentialed physician.

On an intermediate level of complexity, there is the Client-Server Architecture, which is currently the most widely used approach. In this architecture, studies are uploaded from acquisition equipment to a central repository automatically. Physician's workstations retrieve studies only when needed, without any automatic pre-fetching strategies. This approach may introduce a relative delay into the study revision process since studies will not be moved until requested by physicians. Nonetheless, it provides a more efficient access control to studies, when compared with the previous architecture.

The latest trend in PACS architectures is the Web-based [2, 22, 23]. In this case, PACS integrates the repository and the visualization applications into the same infrastructure, typically the same datacentre in a private or public Cloud environment. This solution is powerful both in performance terms and business model. Despite generally enabling multiple affiliated institutions to use the PACS via Internet connectivity, these architectures usually have low levels of modularity [24]. Moreover, these solutions are normally vendor locked, which disables the integration of the PACS with other vendor's solutions such as viewer workstations, archives, or even other institutional PACS. This is a major problem, as it limits medical institutions from benefiting from state of the art application features [25], and from accessing patient information stored in other systems (HIS, RIS, PACS).

2.3 Standards

2.3.1 DICOM

In the beginning of digital medical imaging equipment, manufacturers started by developing their own communication protocols and data formats. Therefore, exchanging images between different vendors of equipment was impossible. As result, the mobility of medical imaging studies was severely constrained. Circa 1980, the National Electrical Manufacturers Association (NEMA) and the American College of Radiology (ACR) formed a consortium with the intent of standardizing the technologies involved in the medical imaging field, most notably communications between different equipment and image data formats. The first draft of the standard was named ACR-NEMA 300 (or ACR-NEMA 1.0). It is considered a major contribution to the expansion of PACS. However, the guidelines presented in this first version were not sufficient and inclusively incurred in some errors. Consequently, later versions of the standard were introduced [9].

The definitive version of the standard, called DICOM 3.0, was published in 1993 [8]. Commonly known as the DICOM standard, it figures as the most important standard in the medical imaging field, it is fundamental for nowadays PACS. The standard is a multi-part document that is continuously updated through the addition of new parts and supplements that support the compatibility with newer acquisition equipment and technologies. Nonetheless, the compatibility with previous versions has always been maintained.

The proliferation of DICOM compliant[3] equipment has enabled the exchange of medical imaging data among different vendors' equipment, thus, providing the possibility of implementing modern PACS. Moreover, extensions to the standard have introduced new ICTs, such as Web services, that have further shaped the working environment of digital medical imaging laboratories.

DICOM data format

Medical images formatted under the DICOM standard are binary files formed by two logical sections: a meta-data header and the actual image, i.e., the pixel data. DICOM files are composed of many DICOM Data elements[4]. Each one of these elements carries meta-information about the different stakeholders related to the image, such as the patient, clinical staff, acquisition equipment, and medical institutions.

DICOM Data elements are encoded using a Tag-Length-Value (TLV) structure. The *tag* field is supposed to identify the data element. In DICOM, it is formed by two subfields: the group identifier and the element identifier within the group, both encoded using 16 bit unsigned values. DICOM Data Elements are grouped by their relation with real-world entities, i.e., Information Entities (IE) that represent, for instance, the Patient (`0x0010`), the Study (`0x0008`) and the Series (`0x0020`). Therefore, elements holding information related to the patient are encompassed in the Patient group (`0x0010`) and so on. As an example, the *Patient's Name tag* is represented by (`0x0010`, `0x0010`). Apart from the *tag*, DICOM Data Elements have also an identifier for their length in bytes, the *length* field. Lastly, the *value* field holds the actual element's data. A simple illustration of a DICOM Data element is provided in Figure 2.3.

The information enclosed in DICOM objects[5] is very heterogeneous. There are Data Elements for representing names, measures, dates, among others. Therefore, in order to

[3] DICOM Compliant or DICOM Ready is the common designation of equipment or software that support the DICOM Standard.

[4] the terms Elements and Attributes are interchangeable

[5] DICOM Objects an umbrella term to describe DICOM files, which could be images, structure reports, among others

express all these data types, the encoding of the *value* field changes according to the element's type. The Part 5 of DICOM standard [26] defines 27 different encoding formats for the *value* field. These are the most basic data types in the standard and are called Value Representation (VR). A Data Element can be composed of other elements, having the SQ (Sequence) VR. This creates a hierarchical document structure similar to many contemporary data models. This structure is illustrated in Figure 2.4.

The DICOM standard provides two methods for matching an element to its VR. The explicit method involves inserting a VR field into the element's TLV structure, thus turning

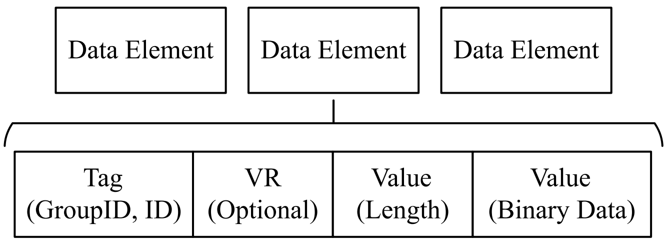


Figure 2.3: DICOM Data Elements structure.

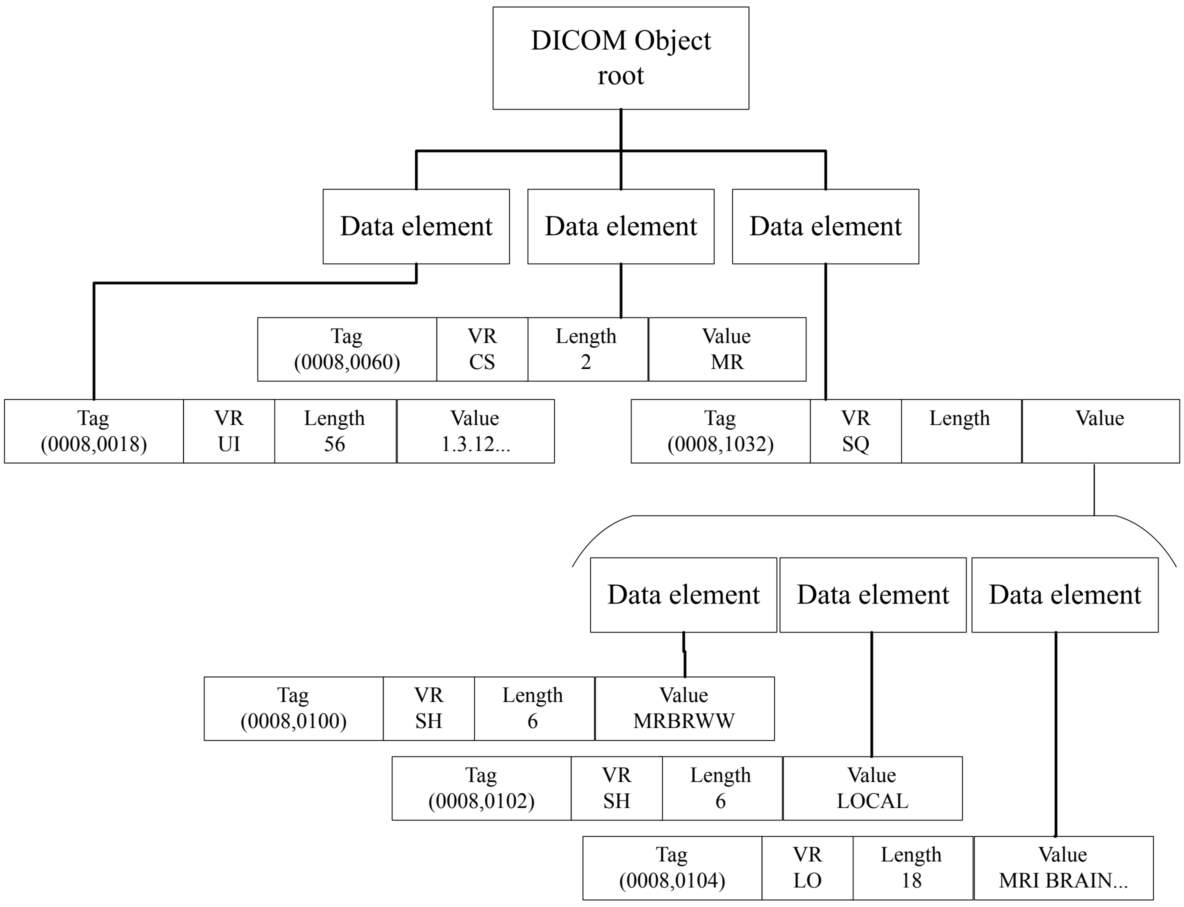


Figure 2.4: Illustration of nested objects in DICOM.

it into *Tag-VR-Length-Value*. In this method, the VR code is directly associated with the element. An example is provided in Figure 2.3. Alternatively, it can be implicitly declared in the DICOM standard dictionary [27].

The implicit method is more commonly used. DICOM Data dictionaries make possible the association of a DICOM element's tag with its VR, multiplicity and description. The multiplicity of an element is how many values are encoded in the element.

The DICOM standard defines, by default, a dictionary containing over 2000 data elements [9]. These elements cover very well the general requisites of medical imaging environments. Nonetheless, the standard is extensible and private data elements may be added by manufacturers in order to support their latest features. As a result, private dictionaries may extend or even replace the default dictionary. By doing so, the standard has the capability of keeping up with state of the art solutions, which is a crucial feature for its prevalence in the field.

DICOM Information Object Definitions and Information Hierarchy

DICOM Objects are capable of representing multiple medical imaging artefacts, such as images from a wide range of modalities or Structured Reports (SRs). These objects are composed of several modules. These modules are associated with an IE that represents real-world stakeholders such as patients, studies and so on. They are composed of multiple data elements, for instance, the *patient module* is composed by the patient's name, sex, birthday among other attributes. Therefore, these modules are closely related to the data elements groups described above.

DICOM also defines which modules shall be included for each DICOM Object class, according to specific Information model definition (IOD) [9]. IOD are collections of modules that describe each particular object. They define which information must be included in the DICOM file, according to its type. These modules can be mandatory, conditional or user optional.

DICOM includes the concept of instances that are IOD templates filled with real-world data and are identified by Unique Identifiers (UIDs). As the name suggests, UIDs are data elements encoded with the *UID* VR. Every DICOM Object's Instance UID must be unique. Any change in an object instance obligates to create a new one with a different UID. For example, if a physician needs to introduce a new overlay in an image. In a global scenario, since instances must be unique, a common practice is to split the UIDs into a series of prefixes identifying equipment manufacturers, types, institutions and other specific information. This practice is well described in [9].

A final remark will be given to how DICOM organizes the IE. The DICOM Information Model (DIM), follows the hierarchical organization in Patient-Study-Series-Image. This approach resembles the real-world organization since a Patient may perform multiple studies;

these studies may have image series from multiple modalities, and lastly, each modality may produce a large collection of images. DICOM images are typically spread by multiple files for convenience. The linkage between these entities is achieved using the instances UID. As a result, the DIM may be reconstructed even when images are spread across multiple repositories. The UID attribute is generally named after its Stakeholders name, however, there are some UIDs that are not named so obviously, like the SOPInstanceUID (Service-Object Pair) which is the UID of the whole DICOM object. Figure 2.5 provides a simplistic modulation of the DIM. The different levels of the hierarchy are well evidenced, as well as their linkage using the different levels UIDs.

DICOM Communications

Normalization of communication processes is of paramount importance to PACS. As such, the DICOM communication protocol figures as a relevant contribution to the field. DICOM provides a Upper Layer (UL), an application-level protocol that uses the TCP/IP protocol in the transport layer [28]. This approach provides major compliance with nowadays infrastructures commonly found in healthcare institutions. Moreover, they are compliant with

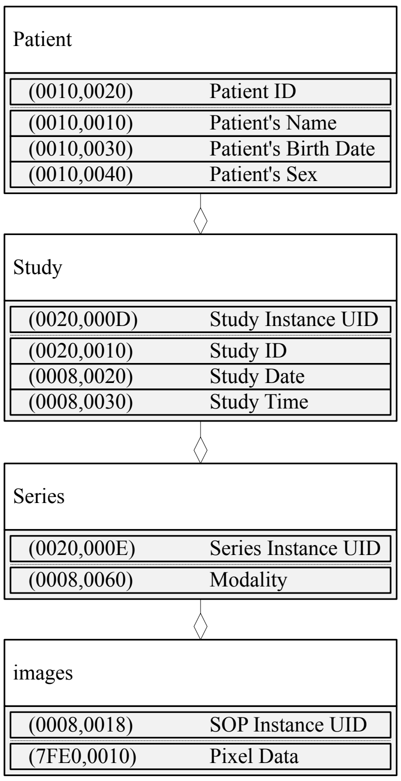

Patient
(0010,0020) Patient ID
(0010,0010) Patient's Name
(0010,0030) Patient's Birth Date
(0010,0040) Patient's Sex
Study
(0020,000D) Study Instance UID
(0020,0010) Study ID
(0008,0020) Study Date
(0008,0030) Study Time
Series
(0020,000E) Series Instance UID
(0008,0060) Modality
images
(0008,0018) SOP Instance UID
(7FE0,0010) Pixel Data

Figure 2.5: Simplistic illustration of the DICOM Information Model.

Table 2.1: Most common DICOM Transfer Syntaxes (Adapted from *www.dicomlibrary.com*).

Transfer Syntax UID	Transfer Syntax name
1.2.840.10008.1.2	Implicit VR Endian: Default Transfer Syntax for DICOM
1.2.840.10008.1.2.1	Explicit VR Little Endian
1.2.840.10008.1.2.1.99	Deflated Explicit VR Little Endian
1.2.840.10008.1.2.2	Explicit VR Big Endian
1.2.840.10008.1.2.4.50	JPEG Baseline: Lossy JPEG 8-bit Image Compression
1.2.840.10008.1.2.4.51	JPEG Baseline: Lossy JPEG 12-bit Image Compression
1.2.840.10008.1.2.4.57	JPEG Lossless, Nonhierarchical
1.2.840.10008.1.2.4.70	JPEG Lossless, Nonhierarchical, First-Order Prediction
1.2.840.10008.1.2.4.80	JPEG-LS Lossless Image Compression
1.2.840.10008.1.2.4.81	JPEG-LS Lossy (Near-Lossless) Image Compression
1.2.840.10008.1.2.4.90	JPEG 2000 Image Compression - Lossless
1.2.840.10008.1.2.4.91	JPEG 2000 Image Compression
1.2.840.10008.1.2.4.92	JPEG 2000 Part 2 Multicomponent Image Compression Lossless
1.2.840.10008.1.2.4.93	JPEG 2000 Part 2 Multicomponent Image Compression
1.2.840.10008.1.2.4.94	JPIP Referenced
1.2.840.10008.1.2.4.95	JPIP Referenced Deflate

Internet technologies, which are the current trend for sharing information in ICT.

DICOM Applications are identified by Application Entity Title (AETitle). As a result, a DICOM application is addressed by the triplet (IP Address, Port, AETitle), which makes possible running multiple applications on the same computational node. DICOM UL also incorporates the concept of associations, i.e. DICOM communications sessions. In these associations, DICOM applications always begin by negotiating the transference properties such as endiness, image encoding or compression, which ensures correct communication between heterogeneous applications [9]. These parameters are defined according to the *TransferSyntaxUID* attribute. The default and most common syntax is the *DICOM Implicit VR Little Endian Transfer Syntax* (1.2.840.10008.1.2), which uses no compression, no explicit VR and the little-endian byte orientation. Table 2.1 provides an overview of the most common Transfer Syntaxes used in DICOM.

After the association establishment, the involved applications exchange a set of predefined commands for storing, searching and retrieving data [27].

On a higher level, DICOM defines the set of operational services such as Storage, Query, or Printing. They are denominated as DICOM Message Service Elements (DIMSE) and follow a client-server architecture on which a given application offers a predefined service to a set of possible clients. Depending on their role in the association, these applications are called Service Class Provider (SCP) and Service Class User (SCU) [2].

The DICOM Storage service is probably the most used in a PACS network, allowing the SCU application to store DICOM objects in the SCP. The most typical use case is when acquisition devices store the acquired images in the PACS archive. The service starts when the SCU sends a *C-Store-Request* command to the SCP. This request holds the object to be stored in the SCP. Afterwards, the SCP sends back a *C-Store-Response* signalling the status of the operation. Figure 2.6 provides an illustrative example of the interaction involved in the Storage service.

The discovery (searching) of images in a repository is supported by the DICOM Query service. An illustration is provided in Figure 2.7. The SCU, for instance, a physician's workstation, sends a *C-Find-Request* to the SCP. It carries a query used to filter the images presented in the repository, including the projection attributes and selection criteria. This query is composed of several DICOM Objects. The SCP will search in the repository taking into account the *tag* and values of each element. It is possible to use wildcards such as `A*`, or date ranges. For each object matching the query, the SCP sends back a *C-Find-Response* with the returned fields.

Moving images from a repository to another equipment is accomplished using the DICOM Retrieve service through the *C-Move* command. This service makes use of the Storage services, described previously. The SCU sends a *C-Move-Request*, which holds information about the target's repository (i.e., the AETitle) along with the reference to the instances that should be sent to destination. Retrieval can be performed at the Study, Series or Image (instance) level. When the SCP receives this request, it starts a storage session with the target repository. Therefore, it issues a *C-Store-Request* for each object and waits for the respective confirmation. When the *C-Store* process is complete, a *C-Move-Response* is sent back to the Retrieve SCU

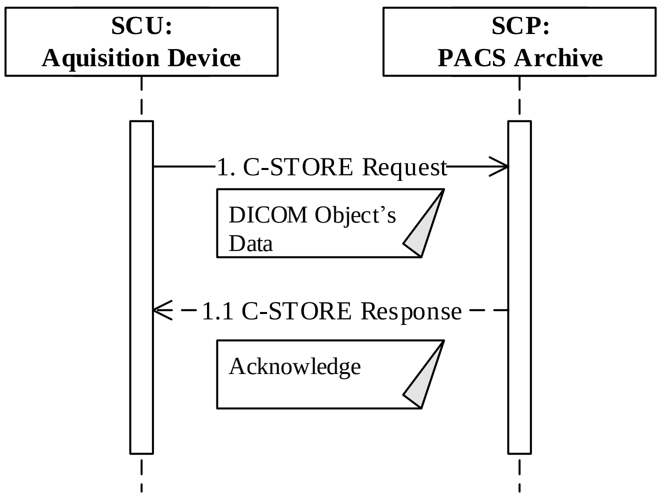


Figure 2.6: DICOM Storage service.

with the notification of the transference status. This process is represented in Figure 2.8.

The advent of Web-based technologies and the necessity of interconnecting multiple medical imaging laboratories across the Internet has led to the addition of several content discovery and retrieval services supported by Web services. Initially, DICOM Web Access to DICOM Objects (WADO) was defined in supplement 148 [29]. It enabled access to the elements of DICOM Objects using Simple Object Access Protocol (SOAP) Web services. These services mapped the functionality provided by the traditional *C-Move* command [30].

However, SOAP is considered a heavyweight technology for Web services, because the verbose messages make them difficult to be included in lightweight applications, such as Web sites or mobile applications. Therefore, the community demanded the inclusion of simpler services based on the Representational State Transfer (REST) technology for searching and storing objects. In 2014, the standard introduced three REST Web services: STOW-RS[6] , QIDO-RS[7] , and WADO-RS[8].

Besides the traditional DICOM services and Web services, DICOM also contemplates storage and streaming of *JPEG2000* images. Currently, *JPEG2000* is the most advanced format for general purpose images. The standard defines a lossless compression algorithm, as

[6] URL: service-endpoint/studies ; HTTP – POST
[7] URL: service-endpoint/studies/[StudyInstanceUID]/series/[SeriesInstanceUID]/instances ; HTTP - GET
[8] URL: service-endpoint/studies/[StudyInstanceUID]/series/[SeriesInstanceUID]/instances /[SOPInstanceUID]/frames/[FrameList] ; HTTP - GET

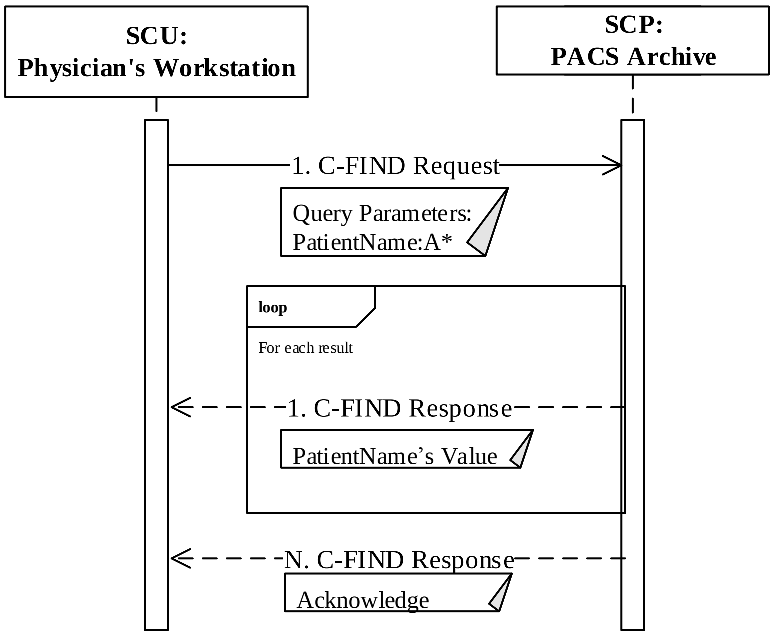


Figure 2.7: DICOM Query service.

well as a protocol for "interacting with *JPEG2000* based images in an efficient and effective manner" [31]. In fact, the *JPEG2000* standard has a special focus towards image interactivity, supporting interesting features such as resolution scalability, progressive refinement and spatial randomness [31, 32]. The JPEG 2000 Interactive Protocol (JPIP) protocol allows viewer applications to interact with *JPEG2000* images over the network. By using JPIP, viewers do not have to download the whole image from the server, instead, it is possible to request only a particular region that best fits their visualization purposes [31, 33]. This image streaming strategy allows the optimization of the visualization process itself, as well as the reduction of the required bandwidth to support remote visualization of images [33].

The DICOM standard supports both *JPEG2000* images and the JPIP protocol. The supplement 61 [34] introduced the *JPEG2000* format into regular DICOM images. The *JPEG2000* pixel data is stored in the DICOM image pixel data attribute, just like any other compression format such as *JPEG Baseline* or *JPEG-LS*. As such, this supplement introduced two new transfer syntaxes: *1.2.840.10008.1.2.4.90*, and *1.2.840.10008.1.2.4.91*. The support for JPIP was introduced by supplement 106 [35]. In this case, the image pixel data is replaced by the JPIP server URL. As a result, the viewer application can request the image directly from the JPIP server, without mediation from the PACS archive. This behavior is supported by the transfer syntaxes: *1.2.840.10008.1.2.4.94*, and *1.2.840.10008.1.2.4.95*.

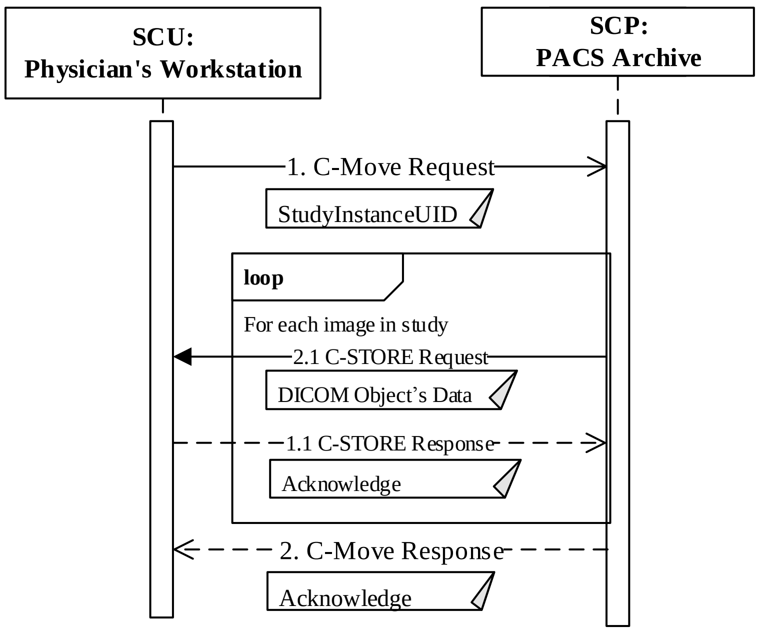


Figure 2.8: DICOM Query-Retrieve service.

Modality Worklist

The DICOM standard also defines a Modality Worklist (MWL) service that is typically fed by the RIS. This system provides information about the scheduled procedures for a given acquisition equipment. For instance, information about the patient such as name, id or weight. As previously stated, in the basic radiology workflow, medical imaging procedures are typically registered beforehand in the RIS. Therefore, in order to keep coherence, reduce entry errors and improve the workflow efficiency [36], the DICOM MWL allows equipment to get this information automatically, via the DICOM *C-Find command*.

Whole-slide imaging in DICOM

This section gives a special focus on whole-slide imaging (WSI). This modality has been kept outside of the PACS scope due to the major challenges to the PACS content discovery and retrieval services. As a result, it was selected for the validation of the proposed methods as an advanced usage scenario for PACS.

The rising interest in WSI motivated the constitution of DICOM Workgroup 26 with the intention of supporting whole-slide imaging in DICOM. This required alterations to the standard, not only at the image storage level to handle images with very high resolutions but also to the information model [37]. The traditional DICOM information model (see section 2.3.1.2) is patient-centric as opposed to the microscopy environment where the specimen is the most relevant subject [37]. The DICOM supplement 122 [38] was the first contribution of this workgroup and introduces the concept of a specimen and container. Moreover, it also includes new data elements necessary to more accurately describe the workflows associated with WSI preparation and acquisition [39].

WSI presented major challenges for medical imaging visualization. These images cannot be fully displayed in their maximum resolution, not only because there are no displays supporting such resolutions, but also because their size is much greater than the amount of memory available in most computers [40, 41]. As a result, pathologists screening these images use a pan & zoom strategy. This involves browsing and selecting sub-regions with diagnostic interest in a lower resolution image and then zooming into a higher resolution whenever more detail is needed. Nowadays, most Internet users are accustomed to this process because of the map browsers available online, such as Google Maps.

The arrangement of the image pixel data in the storage data structure has significant performance implications for the visualization processes. The simplest and most common, way of storing two-dimensional images is in a single frame/page, where the image pixels are stored in a sequential array. Normally, pixels are oriented horizontally, by rows, although they could follow a vertical orientation just as easily. However, this organization has limitations in the WSI screening paradigm, because it does not provide direct access to 2D sub-regions. As a result, all the regions' overlapping rows must be loaded completely, which due to the gigantic size of WSI may not fit into the system capabilities [40]. An illustration of this limitation is provided on the left of Figure 2.9. In order to retrieve the green region, all the pixels in red must be retrieved because they are arranged sequentially.

A more efficient way of storing high-resolution WSI is using a tiled organization, illustrated in Figure 2.9 (middle), where the entire image is stored in rectangular regions sequentially. Despite being a more complex organization, it enables direct access to these sub-regions. As a result, in order to load the green region, it is only necessary to retrieve a smaller image subset composed only of the overlapping tiles [42], as shown in Figure 2.9. The tile size has a significant effect on the retrieval performance and its optimum value is influenced not only by the physical storage medium but also by the viewer display resolution. While a tiled organization facilitates the panning processes, the zooming process, on the other hand, is still a challenge. At lower resolutions, larger areas of the image must be accessed in order to display the requested region, as shown in Figure 2.9 (middle). In the extreme case, a thumbnail requires the entire image data to be processed. This would probably be a very intensive task.

In order to optimize zooming, a lower resolution version of the full image can be pre-calculated and stored alongside the full resolution image. Thus, the typical image pyramid organization arises, as shown in Figure 2.9 (right). According to this scheme, the WSI consists of multiple images at different magnifications where the pyramid provides distinct zoom level. The base of the pyramid contains the highest resolution, while the top contains the lowest resolution image, typically a thumbnail. The thumbnail is a very low-resolution version of the image, making it easy to see the entire image. One or more levels may be created, at intermediate resolutions, to facilitate the loading of arbitrary magnification levels. Each pyramid level follows the tiled organization described above. These strategies are valid for open and proprietary image formats.

DICOM Workgroup 26 also adopted the image pyramid concept and introduced it in the standard. The DICOM Supplement 145 [43] takes advantage of the existing multi-frame objects where each sub-resolution image is stored in a separate multi-frame object, while the

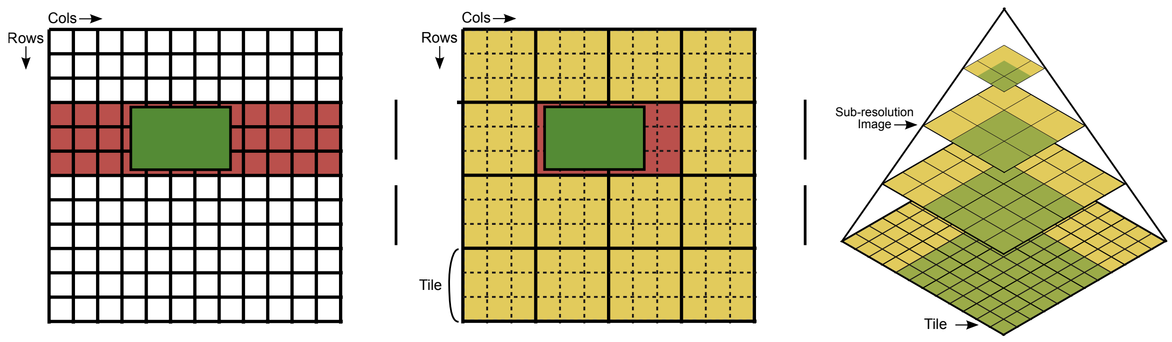


Figure 2.9: Single frame (left) format vs Tiled format (middle), and image pyramid example (right).

individual tiles are stored as separated frames. Each resolution level is assigned a different DICOM series. The standard also supports 3D microscopy, a technique that acquires multiple planes of the slide at different depths [44]. Therefore, each resolution level may contain several Z-planes, which represent acquisition at different focal points. The Z-planes can be stored as separate objects within the DICOM series or, alternatively, in the image object with the corresponding magnification level.

2.3.2 HL7

Although PACS are the most important systems associated with the medical imaging practice, they do not stand alone in the typical healthcare institutional environment. In fact, PACS are focused on dealing with medical imaging specifics. Other aspects of the radiology practice are typically supported by the RIS. Moreover, healthcare institutions typically enclose a broader scope system called HIS, which is used to manage hospital-wide workflows. If DICOM is the utmost standard in the PACS panorama, Health Level 7 (HL7) has a similar role in HIS and RIS. Therefore, the cooperation between DICOM and HL7 systems is a common requirement, in order to achieve full coverage of the medical practice.

Functionally, HL7 protocol works similarly to DICOM as it is based on message exchange between different components in the system. However, HL7 messages have an underlying event. For instance, a *Register* event may be used to capture a patient admission in the system.

Integrating DICOM and HL7 systems often raises several technical challenges. These challenges are often split into two categories: Data integration or Application integration. Regarding data integration, the problems are often associated with differences in how data is stored in the different systems. For instance, there are a variety of date formats that are misleading in the absence of a clear definition. While DICOM encodes dates in YYYYMMDD, other systems might do the opposite DDMMYYYY. These errors are very hard to detect despite having the potential to cause serious problems.

An application integration layer requests multiple conversion points for messages under both protocols. Usually, HL7 based systems are required to implement a subset of the DICOM protocol to be used in communication. In the scope of this thesis, the most important cooperation between HL7 and DICOM compliant systems relies on DICOM Modality Performed Procedure Step (MPPS) and DICOM MWL. MPPS enables the communication between the acquisition devices and the RIS for event announcement, such as ongoing or already concluded procedures. This service is particularly important since it enables an active scheduling management, billing and labour accounting processes to be synchronised with the actual products of the medical imaging practice.

The other great point of integration between DICOM and HL7 systems is clinical reports. The DICOM part 20 [45] defines the guidelines for encoding reports in the HL7 Clinical

Document Architecture (CDA) format. It defines the mapping between DICOM attributes and CDA elements. By doing so, reporting applications are allowed to reference image artefacts directly, which leads to more accurate reporting.

2.3.3 IHE

Interoperability between medical imaging institutions has received a major incentive with the IHE initiative[9]. Circa 1998, the IHE initiative has gathered medical personnel and equipment vendors with the intent of improving the interoperability of medical information systems across multiple institutions, namely HIS-RIS-PACS. In fact, the standardization introduced by DICOM and HL7 proved not to be sufficient for integrating workflows across these systems, especially in multi-institutional environments. IHE defined a set of integration profiles for performing common processes that rely on multiple systems. It introduces the concept of actors, who perform a certain predefined role in the integration profile. Moreover, the interactions between actors are also defined. As a result, performing a process across multiple systems is easily achieved if the systems support the desired integration profile. IHE does not define any additional standard and relies on pre-existing standards, such as DICOM and HL7, for communication and data encoding.

In the scope of medical imaging laboratories, the XDS-I profile is of utmost importance. It provides a centralized discovery of medical imaging studies and their reports. Another good example of an integration profile is the Patient Identifier Cross-Referencing (PIX). It provides linkage between patient identifiers across distinct institutional domains [46].

2.4 Current Trends

The tremendous evolution of ICT is promoting a revolution in the traditional PACS workflows. Nowadays, the usage paradigm for PACS is slowly changing from the local PACS, which supported a single institution or department, to a paradigm where the PACS has to accommodate multiple institutions. The outsourcing of PACS repositories to the Cloud and the establishment of centralized archives that could be shared by multiple institutions has powered this paradigm shift. In this new scenario, the different PACS resources are spread by multiple locations. As a result, we have introduced the concept of distributed PACS to describe these systems. This work has given a special emphasis to performance issues associated with distributed PACS environments.

The appearance of several public Cloud providers has raised medical institutions' interest in outsourcing their PACS infrastructure [47]. This presents itself as a financially attractive option as these institutions often lack Information Technologies (IT) resources to maintain a fully deployed in-house data centre with high availability [48]. Moreover, with the expected

[9] Many resources published online at: www.ihe.net

proliferation of zero-footprint viewers[10] [49], physicians can work remotely without affecting productivity.

Silva et al. propose a PACS architecture that takes advantage of Cloud computing capabilities [1]. Cloud providers offer huge amounts of storage space, as well as optimal availability of data. Those characteristics make them very attractive for storing large volumes of data, key features of any PACS archive. In [50] it is claimed that the costs of storing large data on Cloud providers tend to be lower than regular in-house storage, as it does not require infra-structural upgrades associated with storage scaling. Another argument in favour of Cloud-based PACS archives is that they can be effectively shared among different institutions, promoting study exchange and cost reductions [21]. Nevertheless, the consequence of not storing data locally is the increased time spent on retrieval. Moreover, in some countries, storage of patient data on a public Cloud provider's infrastructure is not allowed.

In [15], the authors suggest that many problems occur when trying to share medical images across multiple hospitals. An architecture based on Hadoop and Cloud computing is described. Nonetheless, it required the usage of a proprietary Web interface for accessing the system. On the same research line, SmartWADO [51] provides a middleware capable of aggregating multiple PACS archives in the same server, a proxy-like system. Furthermore, similar attempts had also been successfully used in the past, namely in [1, 52–54].

DIPACS [23] is another example of a distributed PACS architecture. It is able to federate multiple affinity domains across the Internet. A proxy-like system is also used to provide seamless integration with other DICOM compliant applications. Although the system complies with the proposed functional requirements, it does not rely on Web 2.0 compliant communications and has a complex setup, which helps to justify the need for expert technical staff. Moreover, based on the provided evidence, it is difficult to evaluate the performance of the solution.

Grid solutions have also been targeted to address issues related to the federation of multiple medical image resources. VirtualPACS [55] offers a gateway to access federated resources using caGRID as a distribution layer. In [55], the problems related to the federation of distributed PACS resources are well evidenced, namely, the difficulties in achieving good performance and full connectivity across different network configurations. Despite being able to federate multiple resources, study retrieval performance has proved to be slightly worse than the original version of the standard. Globus MEDICUS [56] uses the same strategy by federating multiple PACS resources under a Grid infrastructure. This Grid environment also provides rich authentication features based on the Grid's existing infrastructure. This solution has better performance than the DICOM protocol for large files. Despite presenting a rich environment for federating multiple PACS resources, GRID solutions generally require extensive IT expertise to set up. Moreover, as the solutions are tightly coupled with the

[10] Visualisation applications that use standard Web-browsers and do not require specific hardware

infrastructure itself, they are not portable and supporting them requires the deployment of the GRID infrastructure, which may be an encumbrance in most use cases.

The healthcare industry in general and the medical imaging field, in particular, are constantly interested in exploring IT for optimization of business practices. Nowadays, it is expected that recent technological trends such as Cloud computing, Ubiquitous computing and Big Data analysis, will help us to expand our knowledge in disease treatment, diagnostics and personalized healthcare [3, 57]. Moreover, such technologies are expected to provide better insights into the healthcare business practices and help to reduce the spiralling healthcare costs by better using the available medical resources [3].

Perez et al [3] reports that there is a unique opportunity in the integration of heterogeneous systems that hold medical data; not only medical information systems, which generally belong to healthcare institutions but also with social and personal health appliances. Performing large-scale population-based analysis over such heterogeneous data sources is expected to bring unprecedented research opportunities transversal to all kinds of diseases. On the same note, Viceconti et al corroborate the perspective that Big Data analysis of medical information resources, such as medical images, opens new opportunities for new epistemological studies [20]. Such approaches are commonly referenced in the literature as secondary uses of data [58]; in contrast with the primary use of medical data which is to directly support the medical practice.

Unfortunately, the requirements raised by this new environment imposes tremendous technical challenges that still need to be addressed. Some of them are transversal to all Big Data appliances such as providing efficient access or supporting heavy analysis over large data repositories, which both require expensive computational resources [20, 57, 58].

The tremendous amount of data generated by the medical practice [7, 50] obligate us to carefully design PACS to deal with the storage of large volumes of data and have a fast distribution layer to keep the communications fluid and acceptable. Moreover, searching content in these huge repositories is not always trivial.

However, the medical field imposes additional challenges due to their intricate requirements in terms of security, data ownership, governance and privacy [3, 20, 58]. In fact, Griebel et al report that only a very small portion of publications about cloud-based medical systems are referent to successful implementations [58]. With all these, the integration of multiple institutional domains is severely constrained at best. Hence, there is need to research efficient methods to support both the standard radiology practice and advanced usage scenarios, such as research, with acceptable Quality of Service.

Chapter 3

Service Performance in Medical Imaging Networks

This chapter provides all the required insights about performance and scalability of PACS components designed to work in distributed environments and manage huge volumes of data. This section starts by familiarizing the reader with the concepts, terminology and general approaches used when addressing the performance thematic. Afterwards, the technological requirements and operational issues associated with the main services of PACS are identified. The technologies that will have a major impact on the proposed subject will be described along with their impact on the problem statement. Attention will be given to the ecosystem where the doctoral work was developed and evaluated, namely the issues related to the distributed nature of modern PACS solutions and the requirements of big-data analytics scenarios. Finally, a related work section will provide the required information about previous research in the thematic proposed by this thesis.

3.1 Performance and Scalability

In terms of computer systems, performance and scalability are very broad concepts. Moreover, there are many misconceptions about what these two concepts are and how and when a given system may be classified as performant and scalable. So, this section is dedicated to introducing these concepts in the scope of the proposed work. Performance refers to the system's ability to fulfil its functional requirements. It measures how fast and efficiently a system completes a certain task [59]. In other words, the system performs well the tasks it was designed to. Taking the PACS as example, the systems must support the radiology practice by storing all the medical imaging studies produced in the institution and by allowing the physicians access to studies for diagnosis and reviewing purposes. If the system handles all the studies correctly and follows the established protocols, it may be classified as performant. However, in most situations performing a certain task is not the only requirement. Usually, the clinical protocols also impose certain time requirements for each task. For instance, when a physician requests a study, he expects the study to be available in a determined tolerable time window, usually less than 2s [60]. Otherwise, he might consider that the system is unable to retrieve the study, even though the system would eventually be able to do so in the future.

There are two major performance metric each one related to the type of task at hand. The throughput is the metrics used to quantify the performance of "non-interactive batch jobs", i.e., tasks that run unattended and do not require user intervention. It measures the number of tasks completed over a time period. On the other hand, tasks consisted of interactive user activities, which are known as Online Transaction Processing (OLTP), are better measured by response time, since it measures how fast the system is able to respond to the user's requests [59].

In computer systems, the fact that most tasks are performed deterministically, i.e., either the task is successfully accomplished, or it is not, has contributed greatly to the close correlation between the performance concept and speed perception associated with the performance metrics described above. Nevertheless, it is important to remember that since the system is intended to store all the produced studies, it must comply with this requirement by storing all the studies. Failing to store a single study, despite managing a big storage throughput, would, in theory, classify the system as not performant. The medical imaging field generally imposes very hard performance requirements to its information systems. This is understandable since mistakes are often translated into severe financial loses or even into life-threatening events.

On its hand, the scalability concept refers to the ability of a system to sustain, or not, its performance indicators during its lifespan. Load is an important concept to understand the implications of scalability, it is an indicator of user's demands towards the system. According to the type of task, load may refer to the number of users, for OLTP, or the number of non-interactive tasks. The rate at which the system's performance decays with increasing load is the best indicator of the system ability to scale [59]. The higher this rate is, the worst the system scales. It means that the system's performance is highly influenced by load factors [59]. In the limit, such systems may reach the point of not performing if the actual load demands overreach the projected levels. Conversely, the lower this rate is, the better. It means that even with severe fluctuations in load, the system's performance would not be affected, thus being considered a scalable system.

Liu [59] suggests the classification of scalability issues in two groups, regarding their harshness. Scalability issues of Type I are inherent to systems highly influenced by load factors. Nevertheless, these can, in fact, be overcome with optimizations, performance tunings and increasing the system's available resources. Such systems may be classified as poorly scalable. In contrast, Type II scalability issues are immune to such improvements, thus, cannot be overcome without major architectural interventions. Such systems would simply be classified as not scalable.

Over time, there are numerous external factors that may increase the system's load demands. For instance, the population covered by a medical institution may increase during a time window or the institution may acquire new acquisition equipment to their resource pool. Under these circumstances, the PACS will have to deal with much more processes, which will stress all the system's components. Whereas some situations might be predictable beforehand, leaving time to account them at the systems implementation, others are completely unpredictable. As a result, most systems should be adaptable to both situations.

Apart from external factors, there are also internal factors that may affect the system's performance over time. This is the case of databases and other storages components. In the system's early ages, it is perceivable that these components will have little load. However, the amount of data stored in these components will increase along its lifetime. It is of utmost importance that these components perform satisfactorily until the systems end life. This is particularly important for medical systems such as PACS and RIS because most countries demand that patient information must be maintained at least for seven years [4, 60].

Of major interest for any computer scientist or engineer is to understand the roles for achieving the optimal balance between performance, scalability and cost of ownership. In fact, these concepts are part of a triangle, where each vertex may have an antagonistic perspective over the information system (Figure 3.1). Considering the performance, it may be generally translated into computational resources as they are essential for the system to comply with its functional requirements. These resources have an associated financial cost, which is translated either in acquisition costs or maintenance costs. Therefore, increasing the system's cost of ownership. In a similar fashion, a typical approach for keeping the system scalable and compliant with stressful scenarios is to overfit the system computational resources. This translates into an increased cost of ownership since some of the available computational resources are not required at some point in the system's lifetime. Consequently, to optimize the cost of ownership, the available computational resources most come hand to hand with the system's demand. Lastly, in order to increase the system scalability, a common approach is to leverage from replicated resources. This strategy does not only increase the cost of ownership of the system but may also have a negative impact on performance due to the overhead of orchestrating those resources.

At this point, the reliability concept comes into play. This concept is a direct consequence of trying to optimize the triangle described above. It refers to the system's probability of being performant at a certain point in time. In other words, it is the probability of a random request to be handled according to the system functional requirements. It is a direct consequence of trying to minimize the system's cost of ownership by reducing the available computational resources.

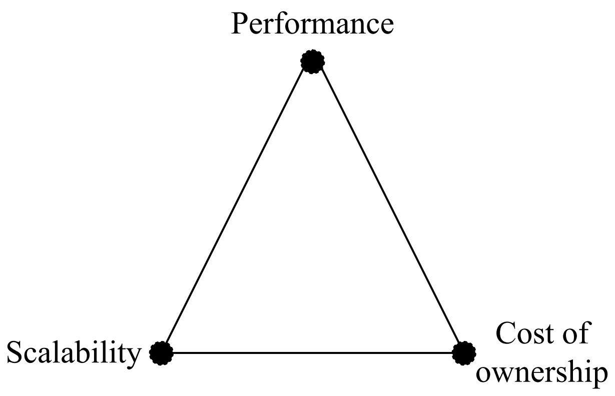


Figure 3.1: Performance, Scalability and Cost of ownership.

Figure 3.2 provides an illustrative example of a system's lifetime. The system's required resources are consistently rising, despite some highs and lows episodes. Meanwhile, the available resources for the system had been stable. These factors have caused three distinct episodes; initially, the system is overfitted, i.e., the available resources significantly exceed the system's requirements to handle the request's load (A). At some point in the system's lifetime, the number of requests exceeded meaningfully the projected mean (B) and the system becomes not performant. When the flow of requests dropped back to normal, the system resumed its operation normally. The third episode occurs further in the system's lifetime; the requests clearly outgrew the system's available resources. At this point, the system has become completely not functional and the solution is to append more resources to the system.

The previous description allows us to draw out the major importance of the underlying computational resources, i.e., hardware, to the performance of a given system or application. In fact, Liu et al. reports that hardware has the greatest impact on the performance of application-level systems. Therefore, understanding the capabilities of the available hardware

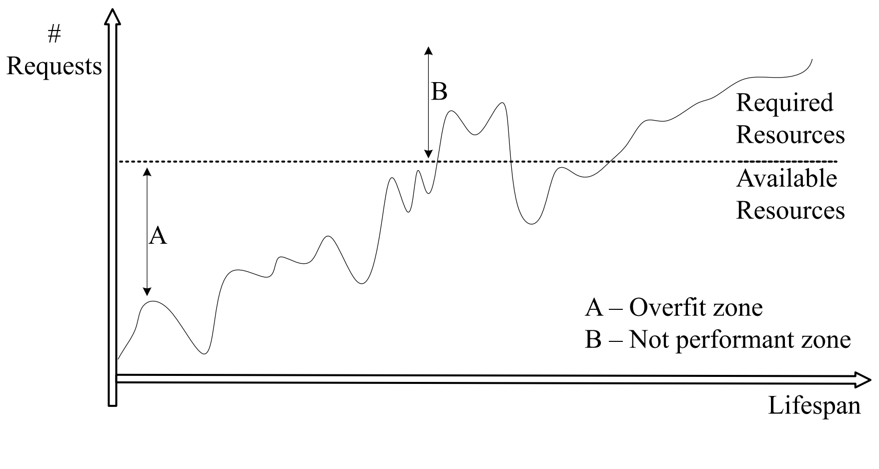


Figure 3.2: Illustration of the system's demand throughout its lifetime.

resources is crucial to develop performant and scalable systems, as it comes down to maximize the usage of the underlying hardware resources.

Despite not being as noticeable as the impact of hardware in the performance of applications systems, the software stack also constitutes an environmental factor of performance and scalability. The concept of software stack refers to the multiple layers of software that are orchestrated, not only to enhance the applications development but also their interaction with the underlying hardware resources. Operating systems, virtual machines, compilers and external libraries are all encompassed in this definition of software stack. However, the impact of the software stack is not significant compared to its benefits, therefore, it is important to keep in mind that some peculiarities in the software running in beneath our applications might affect their performance in rather unexpected ways.

The solution is to use a systematic approach to improve the performance of information systems in general. It involves identifying the critical tasks performed by the system and consequently applying a modification to induce the desired improvement. If the modifications are carefully drawn out to induce the maximum improvement with the minimum impact on the systems cost of ownership, the result will be an overall better system. This is exactly why the critical tasks performed by the system must be selected beforehand. Since they have a greater impact in the system, improvements induced into these tasks will also have a great impact, as opposed to meaningless tasks, which even with a huge improvement would not affect the overall system's performance. The term bottleneck is commonly used to describe the task or component that has the major limiting effect on the system's performance or scalability.

In this domain, there are two recurring concepts that are often marketed by system vendors; Vertical and Horizontal Scalability. These concepts were derived from the scale-up and scale-out concepts presented firstly by IBM researchers in [61]. They defined scale-up as the deployment of an application in a single large shared memory server; as opposed to scale-out, which defines the deployment of applications in clusters of smaller interconnected machines. However, it should be noted that these concepts, vertical and horizontal scalability, lack a proper scientific definition. Moreover, they have been majorly used by software vendors to market their products, which resulted in a multitude of not always concordant meanings and claims. Despite our lack of confidence in such classification, we felt the need to present the most coherent definition of these concepts, due to their extended references in both research and engineering literature.

Therefore, according to the concepts described above, there are two distinct approaches to improve the performance and scalability of application systems. The vertical approach involves improving the performance of each critical task directly, which includes the task's algorithm, any middleware application in the software stack or even the available hardware resources like, for instance, upgrading the CPU or introducing more storage space. In case the improvement is achieved in the software level, it translates into a lower decay in the performance *versus* load relation. Therefore, this approach also enables to improve the long-term performance of the system and, consequently, its scalability. In such case, it also indirectly reduces the system's cost of ownership by reducing the required computational resources, a vital condition for accomplishing the task. The major drawback of the vertical approach is having limited applicability since a certain task, algorithm, procedure or a single hardware resource may only be improved up to a certain degree.

At this point, the system's performance and scalability may only be improved using a horizontal approach. This approach relies on introducing replicated resources into the system. It involves introducing replicated content or computational nodes into the system and managing cooperation with the pre-existing ones. For instance, more servers may be added to deal with more clients' requests or prevent sudden failures, thus shifting from a single machine to a cluster. Moreover, replicated content may be introduced into the system, for instance, to reduce the communications overhead created by moving them; this is the case of caches. This approach is frequently combined with divide and conquer strategies to achieve load balancing.

The most evident drawback of the horizontal approach is its clear impact on the system's cost of ownership since additional resources must be acquired and maintained. Moreover, a subtle impact on the system's performance may be induced by the overhead of managing the replicated resources. Consequently, this technique should be only used after applying the vertical approach optimization upon the system

3.2 Big Data Scenarios

This section provides an overview of state-of-the-art methods for managing content discovery and retrieval in Big Datasets. This scenario particularly affects the PACS used in the research context, where many technical issues arise when attempting to perform these operations on truly enormous datasets. This thesis aims to study and propose solutions compliant with such scenario.

Nowadays, Big Data, Cloud computing and Internet technologies have together created favourable conditions for computational processes that disclosure hidden information contained in Big Datasets. Firstly, these technologies have enabled the gathering of multiple data sources into the same system. Secondly, Cloud computing has provided infrastructural and computational power to support such data-intensive systems. Lastly, the recent rush in the research of efficient methods to deal with Big Data related issues has brought up new technologies, which promise themselves capable of performing these tasks.

In this scenario, scalable Database Management Systems (DBMSs) perform a major role since they are responsible for dealing directly with content storage, discovery and retrieval [62]. Since most systems, including PACS, are moving towards the inclusion of Cloud computing technologies into their architectures, scalable DBMSs figure as an important branch of Cloud computing [62]. Moreover, Cloud computing has also changed the paradigm in content discovery and retrieval. Its adoption caused a shift in the requirements for content discovery, thus the birth of new database designs such as Key-Value stores [63, 64], Document stores [65, 66] and the MapReduce paradigm [67]. A description of these are provided in section 3.4.4.

What Big Data is and why traditional database systems do not perform properly under those scenarios? There are many different interpretations of what Big Data issues are and, consequently, when a given dataset may be considered so. Personally, the most compelling definition of Big Data is characterized by the three Vs: Volume, Velocity and Variety. Volume means the dataset is large in volume; how large? It depends on the other Vs. Velocity refers to how fast user transactions are processed. Lastly, Variety means that creating a data model for the whole dataset is either impossible or unfeasible. In fact, it is usual to have heterogeneous sources of data that may even suffer alterations over time. In conclusion, Big Data is the combination of these factors, which makes general purpose technologies unfit to perform content discovery and retrieval under such conditions [68].

The second question that arises at this point is: does any framework for dealing with this heterogeneity of data in a context of PACS exist? Yes, the Dicoogle open-source PACS [69, 70]. This project started circa 2007 with the intent of extending the general functionality of PACS with Peer-to-Peer communication capabilities and document-based indexing techniques. Although the peer-to-peer capabilities have seen to be outdated in nowadays picture, the document-based indexing technologies were very innovative at that time when most, if not all PACS architectures, had been using relational database technologies. Even nowadays, such contributions remain valid and pertinent as demonstrated in this document.

Besides the original approach regarding the document-based indexing capabilities, what also makes Dicoogle special is its plugin-based architecture. The Dicoogle's research group had felt the need of a framework for supporting clinical and technological research in PACS environments. Traditionally, the researchers had to develop basic PACS and DICOM functionalities to access their contributions. This had resulted in a lot of frustration and prolonged times for testing a new functionally. To solve this problem, Dicoogle provides a plugin based architecture and a Software Development Kit (SDK). It supports the research and development of new PACS components and applications in form of plugins, which may leverage from the existing capabilities of Dicoogle such as storage, indexing of DICOM metadata, or even other plugins functionality. Therefore, it is possible to orchestrate services, creating complex added value workflows between multiple plugins. There are three types of plugins that we are particularly interested in: Storage, Index and Query. They allow us to develop customized storage and content discovery services on top of the image repository.

3.3 Service Provision Issues

The quantification of PACS performance requires knowledge about the major tasks supported by those systems and how they integrate into the medical practice workflows. As described before, there are three main operational services in medical imaging networks: storage of produced studies, content searching and retrieval. These operations involve the movement of large amounts of data between the different PACS components. Therefore, the performance of storage components, database systems and communication infrastructure is of extreme importance.

An important aspect to consider when defining the requirements of a PACS is its usage scope. In regular medical imaging practice, a PACS must excel at moving studies efficiently and supporting queries based either on scheduled procedures, or patients with appointments. Moreover, these requests usually refer to a defined time frame. This scenario presents a much-limited scope when compared to a PACS that is used in research or supports BI. The last scenarios require the PACS to be able to search for all the images metadata. As a result, the range of queries that must be supported is tremendously wide. Moreover, these queries will much probably be focused on the repository as a whole, rather than in a specific time window. In this case, the movement of studies does not have the same significance as in a regular PACS. From this point onward, we will reference these two application scenarios as enterprise PACS and research PACS.

Despite seeming a major encumbrance for PACS developers and administrators, this divergence in usage scopes presents itself as a fertile ground for someone interested in improving the performance of PACS and optimizing its cost of ownership. By focusing on the critical tasks of each scenario, the PACS performance may be upgraded while limiting its effect on the cost of ownership. For instance, in case of a regular institutional PACS, it may be a waste of resources to have expensive technologies to deal with Big Data issues. These are not likely to be felt in such usage scenario as opposed to a research scenario. A comprehensive description of big data issues was provided in section 3.2.

The most evident conclusions are that the PACS adaptability is of utmost importance, and that different technologies may be used in different scenarios or even in different stages of the PACS lifetime. Moreover, PACS that are intended to be used both in research and enterprise scenarios must excel in every task. This imposes serious implications on the development of those systems, especially in the performance and scalability point of view. Notably, this has been one of the major interests in the development of the Dicoogle PACS (see section 4.1.1). A description of these technologies will start in the next section.

3.3.1 Storage

Storing is the act of saving resources for later use. In the context of PACS, these resources are normally medical image studies, Structured Reports or patient records. Nonetheless, other resources may be DICOMized and stored in a PACS repository. Since storage is normally based on the presumption that stored resources will be required in the future, providing content discovery mechanisms are also crucial. However, in this section, we are only interested in the direct access to medical imaging data, rather than in the discovery of medical imaging content.

Nowadays, the storage of medical imaging resources in a repository is not considered a limiting factor for the PACS performance in most scenarios. There are two major reasons justifying this perception. Firstly, the performance of nowadays storage media is sufficient [71, 72]. Moreover, systems with slow media devices can use other techniques to improve its performance, for instance, caching systems [71]. Secondly, the storage service provision is generally more limited by the communications infrastructure, particularly in geographically distributed environments [60]. As described previously, the different components of a PACS are usually hosted on different computational nodes. Therefore, the performance of the communication infrastructure is an important factor transversal to all tasks that involve an exchange of data between distributed nodes.

On the scalability point of view, the limiting factor is the volume of data involved in the medical services provision. As expressed before, the radiology practice generates a tremendous amount of data and the trend is to continue growing with the inclusion of novel acquisition techniques, which are shifting gradually to the 4th dimensional space [3]. Another aspect considered is the legal obligations in many countries, which require the medical institutions to keep performed procedures during a considerable time window [4, 60].

Comparing the scope of the storage service in enterprise and research PACS, we have not found significant differences between them. Both deal with huge amounts of data and both are required to maintain such data over long periods of time, either because of legal constraints or business practices.

The provision of storage services is usually associated with other procedures, namely content indexing. Moreover, they are generally used to enable faster content discovery. However, they could implicate the perceived performance of the storage service because indexing operations are often triggered by the storage services [72–74]. For simplicity reason, these procedures will be referred in the content discovery section.

3.3.2 Content Discovery

Content discovery is a major functionality in any information system. However, the requirements of such service are very demanding in PACS. Nowadays, the overwhelming amount of data produced by medical imaging procedures is conditioning the provision of search services as never before [4].

As stated previously, research and enterprise PACS have very different requirements in terms of content discovery. Enterprise PACS generally rely on more systematic searches, on which specific information about patients and studies are constrained to a certain time window. This presents a much more constrained scenario, where queries may even be anticipated and the data involved completely modelled [75, 76]. Therefore, the heterogeneity of medical imaging data does not present itself as a limiting factor in enterprise PACS. On the other hand, the research PACS must have a greater concern with the heterogeneity of medical imaging data but also with the usage patterns of content discovery services. In this case, the systems must be prepared for much broader searches and even whole repository analysis. Such scenario usually requires data analysis techniques to provide greater depth to the extracted information.

The complexity of search algorithms is directly related to the dataset volume. Therefore, the amount of data involved in content discovery is a great limiting factor [72, 77].

The most important concepts related to the performance of content discovery services are: precision and recall. They provide a metric for evaluating content discovery operations based on the relevance of the results provided [77]. In this case, relevance measures how well a result item is connected to the input query, i.e., the user's need. As an illustrative example, the system should not deliver studies from John Doe when the physician requests all the studies from a patient named Ana. The precision concept refers to the relation between the number of relevant items and the total number of items, both in the result set. Therefore, to maximize precision, the discovery service most not present irrelevant results. Notice the implications of this definition in the example above. Even if the service did not present all the studies from Ana it would be classified as precise if it did not return studies from any other patient. The recall concept refers to the relation between the number of relevant items in the result set, against the number of relevant items in the whole dataset. Therefore, a system with perfect recall would indeed present all the studies from Ana. These concepts are presented in more depth in [77].Figure 3.3 provides an illustration of the distribution of the result set items and their relation with precision and recall. Usually, discovery services are designed to find the best trade-off between precision and recall, including the systems that support fuzzy queries like free text or based on natural language.

Typically, the communications infrastructure is not a limiting factor for the performance of discovery services since only the summaries of the result items are exchanged between client and provider [78]. Nonetheless, in federated environments where the queries are sent to many distributed end-points, the communications infrastructure may introduce some limitations to the system's performance. In this case, the different partial results must be assembled before being presented to users. These processes also raise many technical difficulties such as identifying similar result items across multiple endpoints [78, 79]. Such operations tend to be computationally heavy.

In this context, medical imaging repositories must use state-of-the-art database technologies to increase the efficiency of search operations. They provide searching facilities by organizing the data in a way that facilitates its analysis. Besides the usage of databases, a common practice to speed-up content discovery is by creating specialized indexes. This technique provides an alternative view of the information contained in the database and can be compared to books indexes. They facilitate the query procedures because the information is sorted in advance. Another technique often encompassed in indexation is the organization of the information in clusters. In this case, the goal is to reduce the complexity of the complete dataset and, thus, let the algorithms perform over less quantity of data.

There are two major kinds of information that may be indexed, content and data structure. Firstly, content indexes aim the improvement of the search operation in the store values [77]. Taking, for instance, a book, the content index will improve search operations about keywords contained directly in the paragraphs. On the other hand, the structural indexes are focused on the data structure, how it is organized [80]. They enable more powerful searches, rather than optimizing the performance of content searches. For instance, in the book analogy, a structural index would enable to find a section related to content discovery inside another section about performance and scalability.

Finally, it is also important to refer the importance of query languages in content discovery services because they are the main interface between servers and users. The term expressiveness refers to the different types of queries supported by the query language, for instance, Range, Free-text and Boolean [77]. The query language generally enables to perform content searches, i.e., filter the dataset based on the document's content. However, some languages allow structural queries, which enable filtering documents by their structure [80].

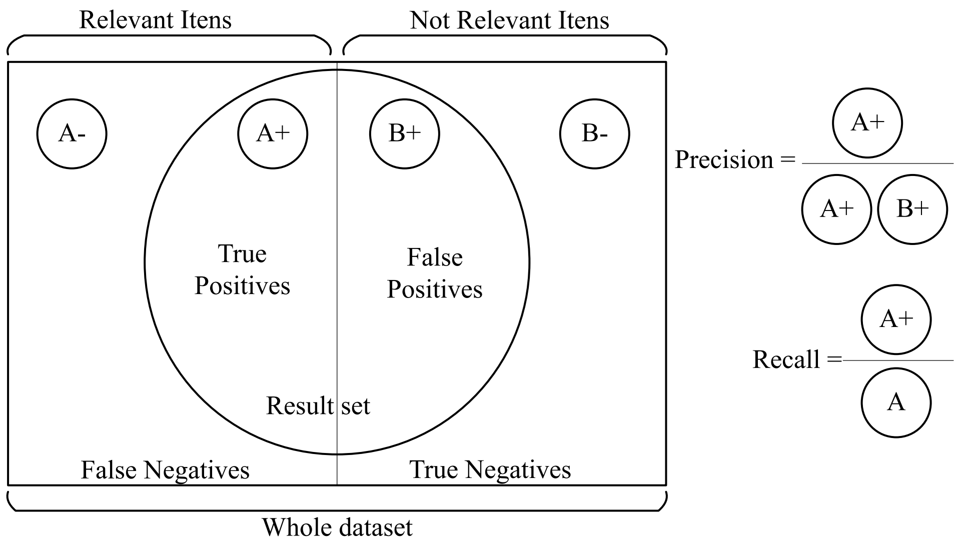


Figure 3.3: Venn diagram with the logical distribution of results in searches.

Another important factor for choosing a query language is the user's familiarity. Ideally, all the users would want to use their natural language for querying the system. However, technologies that enable Natural Language Processing (NLP) are still in early stages, not providing the same expressiveness as languages with a stricter format and controlled vocabulary.

3.3.3 Content Retrieval

The provision of content retrieval services in PACS is mainly limited by the network infrastructure. Medical imaging examinations can reach the gigabyte-scale in some modalities and the workflows resort frequently to the movement of studies between the different PACS components. As a result, the performance of the PACS content retrieval services is directly related to the speed of communications infrastructure. In fact, a major requirement for enterprise PACS is to have great performance in study retrieval. Either in revision or emergency scenarios, the physicians usually demand an immediate access to studies. Such position is easy to understand since any time wasted waiting for studies may represent financial losses or even life-threatening risks for patients.

Nevertheless, the content retrieval service may be also limited by the number of users and requests. Nowadays, LAN and WAN technologies both offer very fast connectivity. However, the number of users in a hospital environment during business hours has the potential to overload any communications setup [16].

Nowadays, there is a clear tendency of moving PACS into distributed environments (as supported by this study[58]), either by deploying cloud archives or by sharing resources between multiple institutions. In both scenarios, the communication infrastructure creates significant bottlenecks, which impact directly the study retrieval performance. Since these technologies represent great contributions to other services, namely storage and content discovery, it is crucial to minimize their constraints to the retrieval services. In this regard, improvements to the communications protocol and the usage of multiple repositories seem to be acceptable approaches.

3.4 Technologies for Service Optimization

In the medical imaging panorama, PACS repositories are rich databanks composed of visual and metadata information typically stored within the DICOM files. Exploring this information efficiently is not only required to accurately support the medical practice, but it is also important in research scenarios around the radiology practice or BI in healthcare institutions. However, the difference between a databank and an efficient database is the ability to discover and retrieve content. In this regard, there is still much to do to bring PACS repositories into this stage. Particularly, state of the art technologies must be associated with PACS architectures to support efficient content discovery and retrieval at the infrastructural level.

3.4.1 Communications Optimization

Optimization of the communications processes is a core theme of this thesis proposal. This section presents the main technologies used in such endeavour. The study of communication technologies used in PACS is required for improving the data exchange protocols, especially in distributed environments.

The lack of state of the art solutions that promote inter-institutional workflows may be explained, not only by vendor's business strategies but also by some inefficiency of current standards in shared environments (outside institutions) [12, 13]. For instance, the DICOM Standard has demonstrated some performance inefficiency when ported to Internet environments [16, 81].

In this regard, a few methods had been proposed previously with the intent of optimizing the performance of standard protocols without breaking its compatibility. In [16, 82], Maani et al. claim that, although DICOM states various transference syntaxes with different levels of compression (both lossy and lossless), applications frequently support distinct syntaxes from each other, leading to the frequent usage of the default non-compressed syntax. This results in a significant lack of efficiency, especially in Internet environments. Therefore, the authors suggested an interface between the DICOM applications in the transfer.

In [83], the author tackled the performance problems associated with remote study retrieval by decoupling image processing from viewer application. This involves the outsourcing of such processing tasks to nodes deployed in the same location as the image repository. As a result, these approaches severely reduce the communications overhead associated with a remote PACS environment. However, they are not generalizable to other distributed PACS scenarios, especially because this method is not even applicable to all DICOM applications.

Other approaches have been focused on limiting the communications overhead in distributed PACS environments by adapting existing workflows. In [84], the authors propose a rearrangement of their applications deployment locations so that resources could be fetched directly from the file system instead of retrieved via DICOM. However, this approach is not feasible in every scenario, especially if distinct institutions are involved.

Some approaches have implemented the PACS storage on top of distributed file systems [15]. However, the reports evidence the difficulties associated with managing communications in a distributed PACS. Moreover, it is stated that the quality of the acquisition devices has outpaced the underlying infrastructure for communications.

Nowadays, there is a generalized trend to the usage of data replication mechanisms (see section 3.4.2). In a distributed PACS, image repositories are natural candidates for replication. Moreover, the studies may be stored in multiple locations simultaneously for enhancing reliability and the scalability of the system. Consequently, this scenario creates the opportunity to leverage from all these sources to retrieve data from multiple locations simultaneously. By doing so, the overall study retrieval times could be reduced at the expense of a more complex control mechanism. This could be accomplished by extending the data transference protocols with additional capabilities, like the approach presented in [53].

3.4.2 Cache Mechanisms

Cache mechanisms are widely used in distinct types of IT systems. Cache concept consists of creating a temporary memory to store certain objects that have a high probability of being accessed in a short period of time compared with the normal repository. The idea is based on the assumption that retrieving an object from this temporary memory should be significantly faster than retrieving the same object from its original source. Cache mechanisms have been largely used in many ICT scenarios, namely in computer architectures, and even before the proliferation of Web 2.0 applications.

Caching systems are frequently associated with multiple hierarchical levels, as different storage devices often have different specifications, namely storage capacity, persistence, retrieval speed and latency [85]. For example, hard disk-based caches have greater capacity, but severely lack performance when compared to memory (RAM) based cache.

Distributed systems have taken the use of caching technologies very seriously. Since communications inside a computational system are much faster than in LAN environments and those are also much faster than WAN environments, it was introduced the cache levels concept that combines cache instances deployed at different nodes of the system.

Distributed systems can be split into two major groups: client caches and distributed caches. Client caches (or local caches) are deployed locally in each distributed system user. The goal is to provide a significant acceleration for objects' retrieval, although they are often very limited considering the storage volume. Thus, they are only able to cache a very small portion of the distributed system data. Distributed caches are deployed on distinct infrastructure points, instead of its end users. Although they do not provide much acceleration when compared with local caches, they usually have a bigger data repository.

Nevertheless, the exploration of caches solutions also imposes difficult engineering problems associated with the integrity of cached objects in the different hierarchical levels of the system and with strategies associated with the population and eviction of those objects.

According to [85], there are two major trends in data management for hierarchical multi-level caches. The inclusive management, where the higher levels, i.e., with greater capacity, enclose the cached objects in lower levels. The exclusive management where cached objects are generally only present in a single level, providing better usage of cache capacity at the expense of harder population/eviction strategies.

Cache population strategies are associated with the insertion of objects into the cache. Cache population can be done expressly, resorting to pre-fetching techniques, or it can be executed along with the lifetime of the cache, taking advantage of the client's workflow. In its turn, cache eviction is associated with the removal of less important objects, making room for more important ones. It is normally associated with eviction policies, although users can perform it expressly. These policies are used to select the less important objects which are the best candidates to be evicted. A Cache eviction policy example is the Least Recently Used (LRU) that evicts the least used objects [75].

Pre-fetching is the technique for cache population that is likely to bring best results. In this strategy, resources such as medical imaging studies are moved beforehand to the cache, even before any request is made. Thankfully, the workflows associated with the radiology practice are rather biased. There are countless clues that may indicate that a given study is likely to be requested in the near future. For instance, a physician usually requests studies from a worklist previous appointed in the RIS. Moreover, during the revision process, the physician will likely be interested to look at previous studies from the same patient.

Taking the specificities of the radiology workflows, it is possible to apply pre-fetching techniques using two approaches; attended and unattended. The attended approach involves using static rules, which select the studies that need to be moved. The unattended approach involves using machine learning techniques to predict whether a study will be requested in the future or not and afterwards perform the operation accordingly. In this regard, there have also been some approaches [75, 86].

In [87], Bui et al. describe a pre-fetching mechanism for medical imaging based on a dynamic list of rules. Although they did not change the communication latency, they were able to decrease its impact, fetching the data before they were needed. Nevertheless, their mechanism is focused on Intranet communications. Also, the authors did not describe what happens when the images are no longer needed, i.e. the eviction policies. Another disadvantage is that there is no automatic system to infer pre-fetch rules, this being defined by the administrator.

Descampe et al. describe pre-fetching and caching strategies for client/server architectures of JPEG2000 image retrieval services [88]. When the user browses a Web portal, the system evaluates the window of interest of the page and requests the images before being needed. So, the images are pre-fetched and stored in the caching system for future use.

A few cache solutions have been tested, to maximise the performance of existing solutions, although most of them only in specific scenarios [89, 90]. In [89], the author described a scenario in which the PACS archive was outsourced and a caching system was deployed in each department to minimize the latency impact. SmartWADO is another example of how caches may be orchestrated in a point to point environment to obtain a meaningful performance benefit [51]. Nonetheless, the methods proposed are very basic. Nowadays caches must not only provide significant benefit for a single client, but also for a community of clients.

Langer et al. propose a few modifications to medical image workflows inside institutions [84]. It is identified that medical imaging studies are often moved inside radiology departments. Although it does not seem a serious problem with in-house repositories, it could impose serious performance constraints when studies are in outsourced locations. The need for a "tank" component in PACS is also identified, which behaves somehow like a cache, to be deployed along with the PACS. This component is primarily intended to reduce the number of study requests to remote locations.

Nonetheless, these contributions are unfit to be used in our distributed PACS architecture because they are tightly bound to their working environment. We have identified some generic solutions for implementing distributed caches, namely, Memcached[1], Ininispan[2], Ehcache[3], Riak[4], and Redis[5]. All these solutions offer a key-value data model suitable for our scenario. The major difference in these solutions is the replica management protocol used to distribute the cached objects along the multiple cache instances. Each implementation provides certain benefits and drawbacks that must be taken into account when choosing the most suitable for a specific scenario. In the context of this thesis, we decided to propose and implement a new system that could use the already implemented communication layer for supporting the replica management protocol (section 4.2.1). By doing so, the communications on the distributed PACS could be minimized to some extent by taking advantage of the non-cache specific movement of studies.

3.4.3 Information Retrieval

Information Retrieval (IR) technologies have been popularized by Internet search providers such as Google or Yahoo. Everyone has become intensive users of such systems due to their ability to provide us with relevant information. However, IR is not a recent topic in computer sciences. In fact, it has been, for a long time, a fertile research ground in many applied sciences, including the health area. The main goal of IR is to provide access to unstructured information, including text and multimedia content. This section is mainly focused on textual IR since this type of information is predominant in medical imaging metadata. Nonetheless, the methods are analogous to other types of content like, for instance, visual findings [91, 92].

A predominant example in content retrieval lectures is the dataset of Shakespeare's plays. In order to help us explain the IR related concepts, let's assume that we want to know in which plays the characters Brutus and Caesar perform. An easy approach would be to search in the entire corpus, which is the designation given to a dataset of texts, therefore analysing each word to find the desired terms. Next, register their occurrence and, in the end, record whether or not the play contained both characters. In an algorithmic point of view, this approach is

[1] www.memcached.org
[2] infinispan.org
[3] www.ehcache.org
[4] basho.com/products/#riak
[5] redis.io

equivalent to build a matrix matching plays (columns) to character's names (rows), and fill each cell with true or false according to character's name occurrence in the given play. This matrix is generally called incidence matrix. Afterwards, to retrieve the result, it would only require a logical AND between the matrix rows corresponding to Brutus and Caesar.

The above description is an illustrative example of the Boolean Retrieval Model. This model perceives each document in the corpus as a collection of words. In IR, words are often called terms since terms may not match exact words; they may also match expressions, composite names, numerical expressions and other relevant aggregation of characters. This model conveys user searches via Boolean Queries. Moreover, they allow the aggregation of multiple terms with the logical operators *AND*, *OR*, *NOT*. For instance, it is possible to search for plays starred by Brutus *AND* Caesar *AND NOT* Calpurnia. Formally, Queries are the vehicle used to retrieve the requested information. Therefore, they are not the information needed itself. It may require multiple queries to extract the desired information or it may even be impossible to do so, due to query language limitations.

Despite being incredibly simple, incidence matrix scales badly since the search algorithm complexity grows linearly with the dataset size. Therefore, such approach is not feasible for large datasets. However, as stated in [77], it is generally very sparse, considering that documents normally have a limited number of terms, which may differ greatly from each other. Therefore, with the intent of optimizing the incidence matrix, the concept of inverted index has come up. The goal of inverted indexes is to reduce the required space for the incidence matrix by only recording the terms appearances. They are among of the most important IR tools. An inverted index resembles an associative memory, rather than a matrix, where terms are associated with a list of documents where they appear in. The whole set of terms is often called dictionary, the associated list is the posting list and the individual entries of these lists are the postings.

Inverted indexes are generally built offline, i.e., before queries are made, and stored in efficient data structures. By doing so, the query processing may benefit from increased performance. The process of creating the index is composed of three major stages. Firstly, a document must be fetched. Secondly, the terms of the document must be extracted. This stage requires the parsing of all documents for extracting the individual terms, a process called tokenization. Each term may pass through various transformations before being indexed. For instance, plurals may be removed and the term's case may be normalized. After these transformations, the term is finally ready to be appended to the index, i.e., append the document identifier to the postings list. If the term does not exist in the dictionary, it must be added. There is also additional information that may be recorded in the dictionary, such as the term's document frequency, or in the posting. A key feature of inverted indexes is to facilitate searches in the postings lists, therefore, the new insertions in the lists are usually ordered.

3.4.4 Content discovery

Database Management Systems

Traditionally, most information systems are supported by relational databases [78]. The relational model provides a structured way of storing information that minimizes redundancy and ensures integrity. Data attributes are organized into relations, which act as predicates in the relational algebra, allowing restricting the range of possible attributes and values. As a result, the database itself constraints the attribute space, thus, contributing to its own quality. By understanding the complete data model, Relational Database Management Systems (RDBMSs) provide more efficient storage and access to data. Moreover, auxiliary indexes are often introduced in these systems for increasing the speed of searching operations [78].

From the engineering point of view, data attributes are grouped in tables, which reflect a relation between them, such as belonging to the same entity. Each row in a table represents a data object. Although tables are a linear data structure, it is possible to define intricate data models by establishing relationships between tables. All these restrictions are defined beforehand in a database schema. By having this knowledge, users can make accurate queries regarding the data model. Virtually, all the RDBMSs support the Structured Query Language (SQL) for defining the database schema, storing data and querying the database.

As opposed to common knowledge, RDBMS perform very well with very large datasets, even petabyte-scale volumes. This advantage is associated with the usage of an underlying strict data model. Ironically, the greatest strength of relational systems is also the source of their weaknesses in Big Data scenarios. It constraints their fitness for velocity and variety in two distinct ways [68]. Firstly, the conversion between the data models used by client applications and the underlying data model of the database is time-consuming [78, 93]. Moreover, the complexity of this process grows linearly with the complexity of the data models involved. This is the paradigm in nowadays emergent systems, which rely heavily on integrating multiple data sources. Therefore, even in cases where it is possible to model completely the data environment, the complexity of managing such model might not fit into the performance requirements of the system. Notice the usage of the term "even" because it introduces the next problem. Secondly, it can be impossible to model completely the data of a given system. Moreover, the data models may change over time, which is aggravated by the rising of data heterogeneity in nowadays systems, which in its turn has been promoted by the increasing importance given to unstructured data sources.

In addition, two other factors have been contributing to the loss of fashion of RDBMSs. Firstly, the software engineering community has been gradually changing its preference from proprietary to open-source software; and most RDBMSs are proprietary. Secondly, the relational database model is not easy to scale horizontally, which may cause problems in extreme scenarios requiring very high throughput on very large datasets [78]. Unfortunately, the community tends to look to these extreme cases as a norm because of their mediatic interest.

In this scenario, a new class of semi-structured data appeared. It differs from structured data since it does not follow a predefined schema, i.e., data model. Sometimes is called “self-describing”, because the data schema is often contained within the actual data. Although some formats may support an external schema to describe the data model, these schemas are not strictly followed and thus, it imposes only loose constraints to the data [94]. This clearly opposes the tight data models of traditional relational databases.

The content discovery and retrieval of semi-structured data imposes serious difficulties in multiple stages of the process. Firstly, the query language must support queries from clients who lack the full knowledge of the underlying schema. Moreover, clients may want to discover more about the actual data schema. Three examples of this level of expressiveness were extracted from [94]:

- Where does the string "Casablanca" can to be found?
- Are there integers greater than 2 in the database?
- What objects have an attribute name that starts with "act"?

These types of queries are easily not handled by the query languages of traditional database systems such as SQL. This has led to the emergence of the NoSQL concept.

NoSQL

Not only SQL (NoSQL) is the common designation of databases that support semi-structured data. In the absence of a tight data model, many databases and management systems have appeared, each one designed for serving in a specific usage scenario. The proliferation of these types of databases was very fast. Many products, following distinct paradigms, are continuously appearing on the market. Nowadays, it is not simple to select which database best fits the desired use case.

The best effort at organizing the NoSQL world is by database type. It groups databases by the query and retrieval facilities offered and, consequently, by the level of perception that the database has about the actual model of the semi-structured data. Examples of types of NoSQL database are Key-value, Column-Family, Document Stores and Graph.

Key-value stores are essentially associative memories that allow us to store key-value pairs. Insertions, content discovery and retrieval are based on keys, disregarding completely the value’s content or data model. In the database’s perspective, values are simple blobs of uninteresting data. Examples of Key-Value databases are: Riak[6] , Redis[7] and Memchached[8].

[6] Riak’s Web site: http://basho.com/products/
[7] Redis’s Web site: http://redis.io/
[8] Memcached Web site: http://memcached.org/

Although very simple, these types of databases suit various scenarios. Moreover, due to their simplicity, the performance of storage, querying and retrieval is a strong argument in favour of these databases. Due to their lack of features, their usage as a primary database in a medical imaging repository is compromised. Nonetheless, these types of databases may be useful to support other systems or services like, for instance, cache mechanisms.

Column-Family stores provides an increased awareness of the stored values data model when compared to Key-Value stores. In this case, the stored value is a tuple. The database provides methods to retrieve and/or query about specific values of the stored tuple, i.e., columns that clients may address by its name. These features increase also the database complexity since multiple indexes are inserted to facilitate queries. Examples of Column-Family stores are Amazon's Dynamo DB [95] or Apache Cassandra[9].

Document Stores are databases designed for storing entire semi-structured documents. The most common formats for these documents are XML and JSON. Nonetheless, other document types may be supported. Ideally, these databases should allow the level of query expressivity described above. In such scenario, database users might search in the indexed document structure, as well as in the values range. However, most databases do not offer these features. Notably, XML databases are the most compliant with the full range of expressiveness. Thanks to XML Path Language (XPATH) which is a query language that allows selecting and filtering document nodes according to their position and content. This query expressiveness is required for taking full advantage of the PACS archive contents. Examples of document-oriented databases are MongoDB[10] and XML databases such as eXist [66] and BaseX[11].

Graph databases are intended to capture a wide variety of relationships between the stored values. The logical view of the database is a graph where the stored values are represented as nodes and their relations as labelled edges. Due to the internal structure used for storage, graph databases are often called triple stores since each database entry is the typical triplet (NodeA, relation, NodeB). Many graph databases support the SPARQL query language, which is a standard in semantic appliances. It enables inference of knowledge by traversing the database graph while applying restrictions to the possible paths. The usage of this type of database alone in a medical imaging repository is not feasible. However, their usage must be taken into consideration in some specific scenarios, like capturing semantic relations within DICOM SRs or other documents.

Heterogeneously Structured Documents

The management of heterogeneously structured documents containing medical imaging data requires the usage of data-structures that are not strictly bound to a predefined document structure. This is usually referred in the literature as schema-less data structures. These

[9] Apache Cassandra's Web site: http://cassandra.apache.org/
[10] MongoDB Web site: https://www.mongodb.org/
[11] Basex's Web site: http://basex.org/home/

structures are generally represented by a tree where each node is an attribute whose children are directly below, the document tree.

Each attribute is unequivocally identified by the combination of its name, the attribute tag in DICOM, and position within the document tree, since multiple instances of the same attribute may appear in the document. Syntaxes, such as XPATH, have been proposed to address this issue. XPATH uses a path expression to navigate in the hierarchical structures of the document [66]. In general, queries which provide the complete path to an attribute are relatively easy to address. However, when the path expression itself contains relations between attributes other than parent-child, the attributes path resolution is not trivial. The simplest approach used to deal with this problem requires the analysis of every attribute in the document. However, this is not practical because the complexity of the process grows linearly with the average document size in the collection.

The most common strategy to improve the performance of these processes is indexing the structure of the document. As a result, the structural relationship, parent-child/ancestor-descendent/sibling, between different attributes in the document may be exposed. This will avoid parsing the whole document for most queries. The effective creation of a structural index requires the existence of an order relation between the nodes that must reflect the relations between each attribute in the document. For this reason, a numbering scheme is often proposed. The scheme must assign a unique numeric identifier to each attribute. A naive example of a numbering scheme would be traversing the document tree in order or pre-order direction.

In [96], a numbering scheme composed by a 3-tuple was proposed. The identifier is composed of the document number, the start and end position of the attribute in the document and its nesting level (D,S:E,N). This scheme modulates the ancestor-descendant relation the following way: The attribute x (Dx,Sx:Ex Nx) is a descendant of attribute y (Dy, Sy:Ey, Ny) if Dx = Dy, i.e., the attributes belong to the same document, and Sx < Sy and Ex > Ey. The drawback of this system is the requirement for multiple numbers to store the identifier.

The XISS system was proposed in [80]. It identifies attributes by the tuple (Order, Size). The order is mapped by traversing the document tree in pre-order, while, the Size is a major of the number of the attributes descendants. This numbering scheme enables ancestor-descendant relations to be found using the proposition: x is an ancestor of y if order(x) < order(y) <= order(x)+size(x). Therefore, this check is feasible in constant time.

Another numbering scheme was proposed by [80]. The scheme assigns an identifier by traversing the document in level-order. However, it assumes that the document tree is a complete k-ary tree, i.e., the maximum number of child attributes, which are present in each node of the tree. This scheme, although very simple, is able to establish the parent-child relation between attributes very easily.

Concluding, as explained in [66] the completeness constraint of the document tree does not fit every scenario. As a result, multiple sparse identifiers must be inserted which means a greater requirement for the identifier capacity.

$$parent(i) = \left\lfloor \frac{i-2}{k} + 1 \right\rfloor \tag{3.1}$$

Literature reports

There are several studies reported in the literature aiming to find the most suitable database technology for PACS. In [73], a new database model for DICOM images is proposed. The solution is based on a decomposed storage model, which is similar to the document-oriented model. This proposal was compared to a relational database model. In order to overcome the limitation of the model to store heterogeneous documents, the DICOM elements not included in the database schema were stored as an extra BLOB field in the information entity table.

This proposal performance was compared with the relational model that stored only a predefined set of DICOM elements. The trials demonstrated that the relational database version was faster than the proposed model 78.6%, with, and 57.9% without using the BLOB field.

Regarding the query performance, the authors found that both approaches were similar in spite of the overhead of parsing the BLOB data. Summarizing, the relational database model was found more suitable to handle queries which involved filtering a large number of attributes and a low selectivity[12], whereas the proposed model performed better for cases with high selectivity. It is important to mention that these trials were conducted using a dataset of only 67 studies, with a combined size of 4GB. This dataset is very small and not representative the real-world datasets, considering that the annual volume of data produced by imaging centres in on the order of 10TB [60]. Therefore, the definition of high and low selectivity in context of this dataset is much different than in real-world scenarios. As such, we cannot draw any conclusions of how the above-mentioned models would perform in a real institutional PACS.

Multiple document-oriented databases for PACS archives have been put through a comparative laboratory trial in [97]. The authors showed concerns with the performance of the databases in the short, medium and long-term by using three different datasets. However, their largest dataset might not be representative enough of the institutional reality, since it contains only around 50 thousand images. For instance, a dataset collected at a medium size Portuguese institution held over 7 million images in a single year (see section 4.3.2). More evidences of this issue are found in [98–101].

Langer et. al. shown how cloud computing can help solve some problems faced by medical imaging researchers whose work include the Data Mining (DM) and processing of DICOM data

[12] Inverse relation between the number of returned records and the total number of records in the database

[102]. The solution is backed by a relational database, made extensible to support new Data Elements. However, in a posterior article [103], the authors point out the difficulty of indexing a very heterogeneous data source, which ended up leaving some DICOM elements unindexed. The result is that some repository content is not searchable nor retrievable, a sensitive issue in medical context. The authors stated that NoSQL approaches may be required to handle this issue.

Summarizing the literature review, it is possible to state that previous proposals have difficulties handling the flexible data-structure of DICOM objects and were not sufficiently validated with datasets that truly simulate the conditions found in real institutions and research centres.

Chapter 4

Proposed Methodology

This chapter presents this thesis' core contributions to the optimization of modern PACS architectures. More precisely, it starts describing our contributions to the architecture of a medical imaging repository. Then, the contributions to efficiently support distributed medical imaging networks are presented. Lastly, it describes how this doctoral program has been supporting the optimization of PACS content discovery services.

4.1 Medical Imaging Repositories

The optimization of PACS architectures should start with the analysis and refactoring of the most important component, the PACS archive. In the scope of this thesis, we have been deeply enrolled in the research and development of Dicoogle PACS. Although this system is a not a direct output of this thesis, the solution was extremely important to the success of developed methods. It was used as working framework and significant contributions were made to its conceptual architecture and implementation. These contributions endowed Dicoogle with the capabilities to support our research, most notably the validation of new methodologies and processes.

Dicoogle is not only a fully fledge PACS archive, but also a software framework that enables developers and researchers to quickly prototype and deploy new functionality taking advantage of the embedded DICOM services, data indexing and retrieval capabilities. it promotes the exploration and application of innovative technological approaches while providing the required robustness of a production PACS archive.

Summarizing, the Dicoogle platform was the ideal foundation for the prototyping, experimentation, and validation of methods proposed in this thesis. On the other hand, our contributions were made available to the community through its inclusion in the Dicoogle open source distribution.

4.1.1 Dicoogle Open-source PACS

Dicoogle is an open source PACS. Its implementation started in 2007 and has since been tested in multiple contexts. Besides supporting standard DICOM semantics, Dicoogle leverages peer-to-peer communication models and document-based indexing techniques within a DICOM network, diverging from the common architecture employing relational databases [104]. This replacement has resulted in a paradigm change since it has enabled much more information to be extracted from medical imaging repositories contrasting heavily with the standard DICOM query and retrieve services (see section 3.4.4). The project has evolved into a catalyst for research and development of new PACS applications. With the project growing organically, several forks and branches were created for providing distinct functionality. Dealing with multiple implementations [91, 105–107] quickly became unattainable and cumbersome to scale. The merge of branched code with the stable branch has proven not to be free of issues and led to minor and sometimes major duplication of code having slightly different interfaces, to the need to adapt existing functionality to changes introduced elsewhere in the code, or even caused the introduction of race conditions. This is a common problem faced by many software developers, and in our view, it stated the need for a more decoupled and modular architecture and a more integrated development culture. Given the very active research effort in PACS systems, we felt the need for an environment where an idea may quickly be prototyped, tested, and validated. On the other hand, the intention of supporting advanced usage scenarios, such as research and BI led us to refactor the application's architecture, with the goal of streamlining and easing third-party development, a process that benefited from the field expertise obtained from previous iterations and deployments of Dicoogle.

The Dicoogle framework enables the development of plugins to modify or enhance its core functionality. Plugin communication, services, and lifetime are all managed by the core application. A plugin-based approach maximizes the decoupling between components, enabling orthogonal features to be developed separately and deployed easily. Custom builds are done by packing only the desired plugins, which minimizes the area of impact of experimental components.

There are innumerous scenarios in healthcare institutions and academia where clinical practitioners are working together with computer scientists to develop innovative solutions to improve the quality of provided services in the medical imaging field. However, the researching and developing of new software pieces, such as the methods developed in the scope of this thesis, would be very difficult to perform in traditional PACS-DICOM solutions. By facilitating the access to a fully-fledged and extensible PACS archive, Dicoogle has permitted us to bridge the gap between state-of-the-art investigation, prototyping, and validation, which usually is a complex and time-spending endeavour.

Dicoogle Framework

Our analysis of DICOM and PACS usage and trends coupled with the field experience obtained from previous iterations of the Dicoogle project have led us to separate and streamline the functionalities provided into multiple categories: storage, indexing, query, service, and presentation. Each category is associated with a particular interface, the implementation of which is loaded at runtime, as a plugin (see Figure 4.1). It is up to the core application to orchestrate the operations provided by the various plugins.

Although the various plugin categories embody an orthogonal semantic context often, in practice, there is a strong degree of intermingling that needs to be explored to provide a fast and robust solution. For instance, an indexing plugin will likely have a query plugin counterpart backed by the same technology. As such, the entry point for the plugin framework is a class representing a set of plugins; the PluginSet. This data structure aggregates plugins from multiple categories into a functionally consistent unit simplifying both development and deployment as shown in Figure 4.1.

The plugin's lifecycle is managed by the Dicoogle core allowing each module to be enabled or disabled per user request. During the application start-up, the plugin directory is scanned and identified plugins are loaded according to their configuration file.

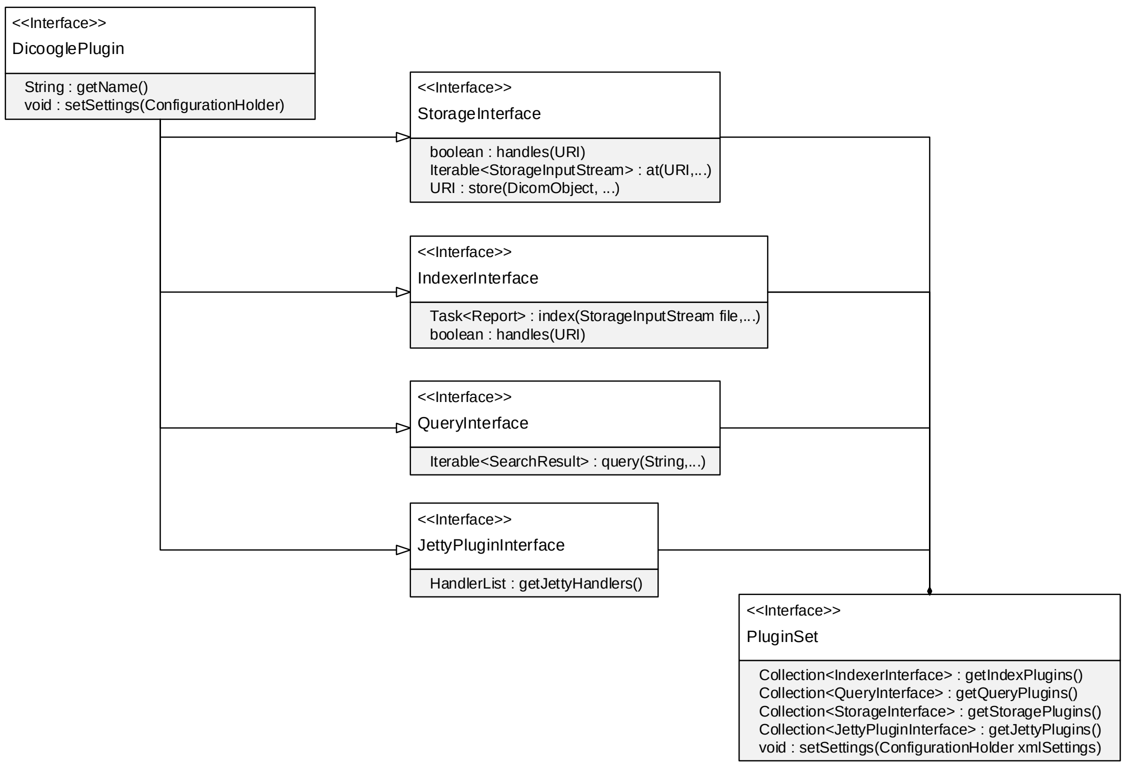

<<Interface>>
DicooglePlugin
String : getName()
void : setSettings(ConfigurationHolder)
<<Interface>>
StorageInterface
boolean : handles(URI)
Iterable<StorageInputStream> : at(URI,...)
URI : store(DicomObject, ...)
<<Interface>>
IndexerInterface
Task<Report> : index(StorageInputStream file,...)
boolean : handles(URI)
<<Interface>>
QueryInterface
Iterable<SearchResult> : query(String,...)
<<Interface>>
JettyPluginInterface
HandlerList : getJettyHandlers()
<<Interface>>
PluginSet
Collection<IndexerInterface> : getIndexPlugins()
Collection<QueryInterface> : getQueryPlugins()
Collection<StorageInterface> : getStoragePlugins()
Collection<JettyPluginInterface> : getJettyPlugins()
void : setSettings(ConfigurationHolder xmlSettings)

Figure 4.1: Simplified class diagram of Dicoogle SDK's main interfaces.

From the core application's point of view, the various plugins within a set are independent of each other, accessed only via the respective interfaces. However, internally, complex plugins may need to have their state shared. These types of dependencies are easily solved inside the PluginSet constructor which must properly instantiate its child plugins and share any data structure required to achieve communication between its internal components. From the external user point of view, functionality is accessible through the DICOM protocol, the exported Web services or the Web frontend, depending on purpose and configuration.

In order to facilitate plugin development by third parties, Dicoogle provides a SDK. By using this SDK, a plugin may be created and distributed as a jar file. The SDK provides common functionalities to unify and ease the development of various plugins. For instance, a settings management system and a logging interface are both supported by Dicoogle and exposed to the developer. The SDK further provides a message-based inter-plugin communication infrastructure and exposes an interface to the application task manager allowing plugins to dispatch asynchronous tasks without having to deal with boilerplate code. Convenience methods to handle DICOM objects and access indexed data are also provided. Figure 4.2 frames the Dicoogle SDK and the distinct types of plugins in the broader PACS archive picture.

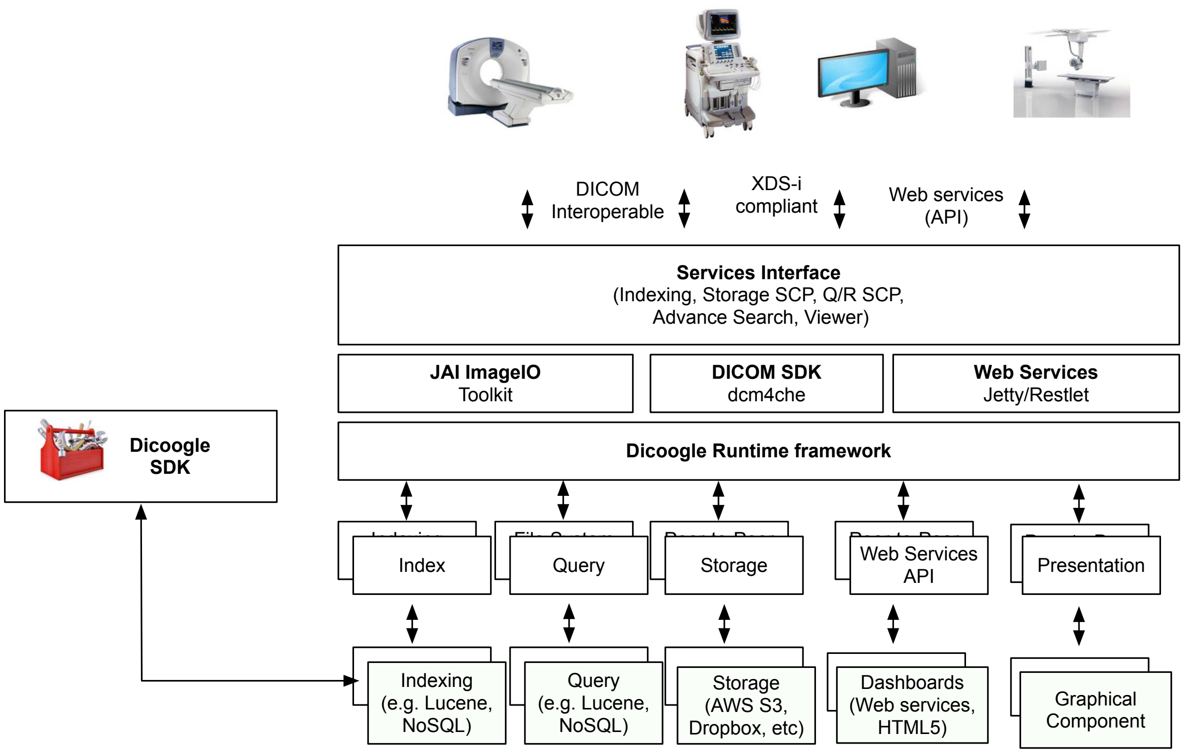


Figure 4.2: Dicoogle full-stack architecture (adapted from [70]).

Storage

A crucial part of any PACS is the persistence of medical imaging studies. A default Dicoogle deployment relies on the local file-system as the backend of choice to store persistent DICOM objects. However, with the coming of age of cloud storage technologies, peer-to-peer distribution mechanisms and grid networks, a distinct set of storage policies come into play. Data may not, and needs not, in all cases to be locally available to the system. The distinct types of storage policies are fertile research ground. However, they must to offer a normalized interface in order to provide storage functionality in a consistent manner regardless of the underlying technology.

In Dicoogle, the access to DICOM data is required for responding to services and when extracting information for fast querying and indexing (Figure 4.3 and Figure 4.4). Hence, it is necessary to unambiguously identify each file so that a relation may be established between separate indexed data and the actual physical location of the DICOM object. Moreover, given the wide set of potential technologies that may be employed for storage, the solution does not particularly care where the files are stored, as long as we can retrieve them on demand.

Dicoogle employs storage plugins to provide the persistence mechanisms, and universal resource of the SDK's main interfaces locators (URIs) to uniquely identify a resource, i.e., DICOM file, and the means to retrieve it, i.e., a storage plugin that knows how to decode and retrieve the specified data. For instance: *cloud:///dicomsrv1/file1.dcm* or *file:///folder0/a.dcm*. Storage Plugins must implement the *store* method that takes either a DICOM object or stream and returns a Universal Resource Identifier (URI) for the stored resource. The converse operation must also be supported. Given an URI, the underlying resources must be delivered. This is achieved by defining the method *at(URI)*, where the

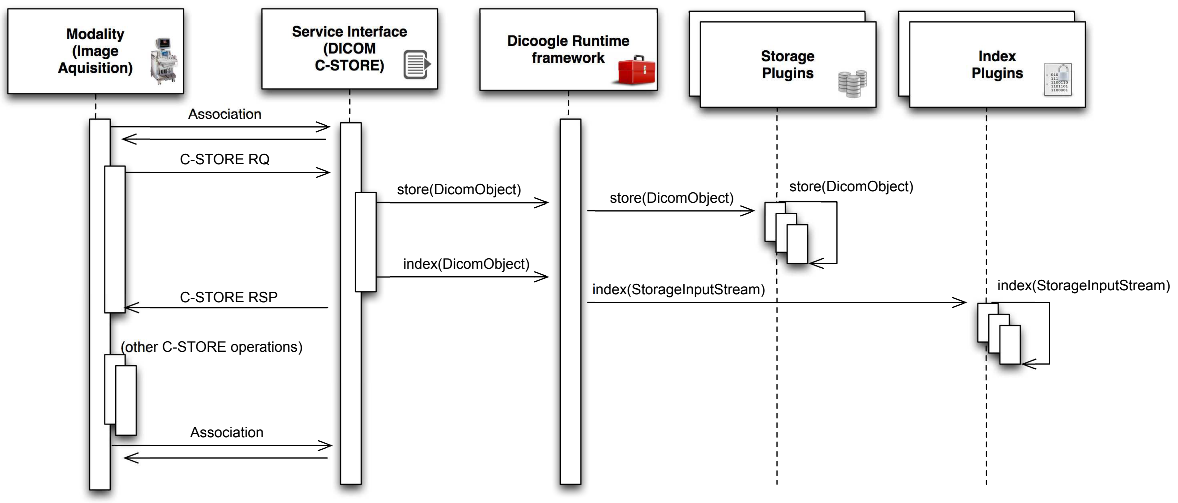

Modality (Image Aquisition)
Service Interface (DICOM C-STORE)
Dicoogle Runtime framework
Storage Plugins
Index Plugins
Association
C-STORE RQ
store(DicomObject)
store(DicomObject)
store(DicomObject)
index(DicomObject)
index(StorageInputStream)
index(StorageInputStream)
C-STORE RSP
(other C-STORE operations)
Association

Figure 4.3: DICOM Storage service as handle by Dicoogle (adapted from [70]).

URIs refer to collections of resources; accordingly. This method returns an iterator into a set of *StorageInputStream* objects, allowing access to the raw DICOM data. Specializations of this class may be used to provide data prefetching and caching in a transparent manner to the core.

Query and Indexing Plugins

Dicoogle aims to provide a set of functionalities oriented to practitioners and researchers that usually need access to a significant volume of data for decision support or statistical analysis. To achieve this goal, Dicoogle initially relied on the extraction of metadata present in the objects of DICOM repositories and its respective indexation in Lucene [104]. This proved a successful approach, validated in the field, where it provided insights about service quality and efficiency as well radiology dosage surveillance [108].

Indexing plugins are the components responsible for organizing data in a format that allows quick access to the stored information, while query plugins mediate access to that information. This solution harmonizes the processes of extracting, storing, and retrieving information, hence allowing the exploration of other data representation and IR mechanisms [109]. Due to the variety of data that can be indexed (textual, visual, hierarchical), the indexing policy and information extraction mechanisms are left entirely to the indexing plugin, which is free to contain additional dependencies to specific databases or libraries not part of the core and to define its own internal data representation. The indexer interface requires as input a StorageInputStream which has the effect of decoupling the indexing plugins from the storage policy.

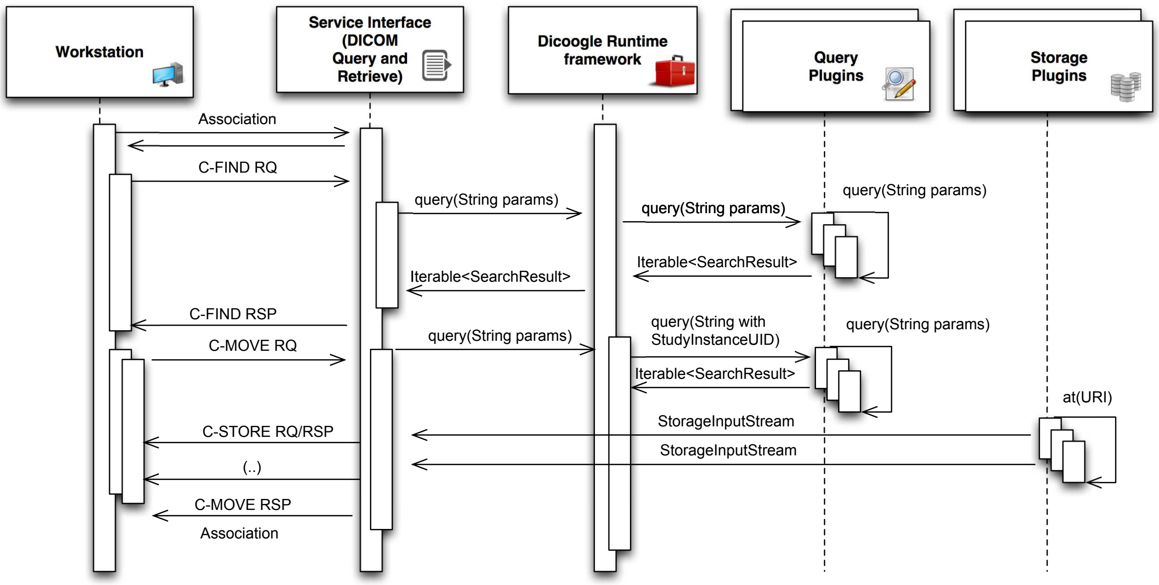


Figure 4.4: DICOM Query-Retrieve service as handle by Dicoogle (adapted from [70]).

The output of a set of index operations is a report data structure with a status, successful or not, and some additional information. However, instead of directly returning a report, the plugin must return a task. The reasons are twofold. Firstly, while the indexing process is fairly quick when handling the textual component of a DICOM file, the same is not true when extracting other image features (for instance, in Content Based Imaging Retrieval (CBIR) plugins or any kind of image analysis). Hence, the task is performed asynchronously. Each index method is scheduled for execution on new data as it arrives, independently of its source, which may be a DICOM service or a user-initiated action.

The indexed information can be accessed programmatically using the SDK's Application Interface (API), which has a query language similar to Lucene, or through the Web interface. Queries are automatically performed whenever any DICOM service requests them, such as DICOM query/retrieve. This is fully interoperable with workstations that support the DICOM standard. Results returned by query plugins can be retrieved or forwarded using the storage plugins. Nonetheless, it is not mandatory to support the Lucene query syntax for implementations which do not need to interact with the native services. Query plugins are the natural counterpart to the indexing plugins and provide the means for information to be retrieved. They provide the bridge from an index's internal data representation into an object understood by Dicoogle's frontend. Typically, for each index plugin, we have at least a query plugin that knows how to leverage the indexed data to provide answers quickly to a user's query.

Web-services

Dicoogle supports some of the most important DICOM services, such as C-STORE, C-FIND, and C-MOVE, and WADO, and has some degree of support for other services such as XDS-I [110]. More recently, the QIDO-RS and STOW-RS services were integrated. These operations invoke functionality provided by the storage and indexing plugins.

The Dicoogle SDK facilitates the integration of new Web services through an API that enables Servlets to be loaded into the embedded Web server. The Web services are enabled, configured and orchestrated by Dicoogle's core leaving to the application only the semantics of the service.

4.1.2 Results

Dicoogle provides several key features to extract meta-data and imaging information for retrospective assessments, which is useful for research, statistics, management, and reporting tasks. It can be used for wide-ranging clinical studies requiring, for example, dose metrics that are now increasingly available in DICOM persistent objects created by recent models of digital image equipment [107, 111]. By enabling multiple views over the medical data repository in a flexible and efficient way, and with the possibility of exporting data for further statistical analysis, Dicoogle allows identification of inconsistencies in data and processes. This platform can be used to audit PACS information data and contribute to the improvement of radiology department's practices [107, 108]. In the context of this thesis, Dicoogle has supported the development and validation of the proposed methods. Compared with the previous architecture, this architecture is easily adjustable to the multiple usage scenarios of Dicoogle, which contributed to the Dicoogle's popularity rise.

In the scope of this thesis, we have been enrolled in the architectural planning and development of the new Dicoogle architecture described above. Latter, we have improved the Dicoogle's performance and scalability, most notably, the storage, content discovery and retrieval services leveraging the framework presented in Section 4.3. Finally, in the context of this thesis, we also developed new functionality, namely, the Digital Pathology support and the Dicoogle BI extension.

4.2 Content Distribution in Medical Imaging Networks

Web-based technologies have been increasingly used in PACS, in services related to storage, distribution, and visualization of medical images. Nowadays, many healthcare institutions are outsourcing their repositories to the cloud. However, managing communications between multiple geo-distributed locations is still challenging due to the complexity of dealing with huge volumes of data and bandwidth requirements. It is crucial that these technologies do not impose constraints on the medical practice workflow. Moreover, standard methodologies still do not take full advantage of outsourced archives, namely because their integration with other in-house solutions is troublesome.

In order to improve the performance of distributed medical imaging networks, a smart routing mechanism is proposed. It includes an innovative caching system based on splitting and dynamic management of DICOM objects. The idea is to endow the system with the capability of temporarily storing the studies more likely to be needed. The system can correctly cache any portion of a study, even if it is composed of only one image. So, it may not have all study fragments, making it possible to request only specific fragments of a study from remote locations. This approach allows us to start providing data to a local client while, in parallel, the remaining fragments are retrieved from the remote caching system. The percentage of study cached can change dynamically according to the cache occupation and network conditions.

A method to reduce the data transfer footprint in distributed PACS environments is also proposed. It provides a high-level control of transfer parameters, such as the number of connections, number of fragments and size of each fragment. The result is a highly tuneable data stream that can be used in any distributed environment. Pairing with it, a method to enable the retrieval of medical imaging studies from multiple archives simultaneously was devised. The objective is to provide performance improvements, load balancing and higher data availability.

The proposed methodologies were successfully deployed in a regional PACS. The results obtained proved that it is better than conventional approaches, as it reduces remote access latency and the required cache storage space.

4.2.1 Cache Architecture

In order to be useful in a distributed PACS-cloud environment, the proposed cache architecture must excel in supporting four main features: query, retrieve, population, and eviction. First, the query service allows an external application to query the cache about the existence of objects (studies, series, and images). Second, the retrieval service allows fetching the objects that are in the cache. Moreover, the caching system must embed mechanisms that populate and evict the cache, i.e., choose which objects should be kept and which should be removed from the cache, respectively.

There are some problems related to the development of the cache mechanism to support medical imaging scenarios. The retrieval, caching, and transfer of medical imaging studies must deal with huge amounts of data, but also with its heterogeneity [73]. Different modalities produce data with distinct characteristics, such as image matrix, number of frames, and average image size [89]. Some modalities, such as CT, may produce thousands of image files per study up to 1 GB. Other modalities, like cardiac US, can produce several cine-loop files with hundreds of megabytes. Caching huge files is a complex issue, not only in terms of storage space but also in data transfer latency.

The strategy adopted to manage this problem, i.e., increasing the router's cache storage capacity and reducing latency, is based on splitting DICOM objects. So, each file is logically divided into fragments of a predefined chunk size. Moreover, the cache may not have all study fragments and it is possible to implement remote retrieval processes of only specific fragments of a study. Moreover, if the information is available in more than one archive, the client router can retrieve blocks from multiple data sources (i.e., provider routers), increasing the system performance in some network conditions. Those mechanisms will be discussed further.

System Architecture

The proposed caching system is composed of four modules (Figure 4.5): storage management, metadata management, service layer, and caching system interface. This highly decoupled architecture enables the system to take full advantage of heterogeneous technologies in the different tasks associated with the caching system. Moreover, it is easy to switch the implementation of any module to satisfy specific scenario requirements, thus making the system easily tunable.

The system was developed in Java. It takes advantage of several tools like Java Simple Plugin Framework (JSPF), dcm4che, MapDB and DB4O (Figure 4.6). JSPF was used to

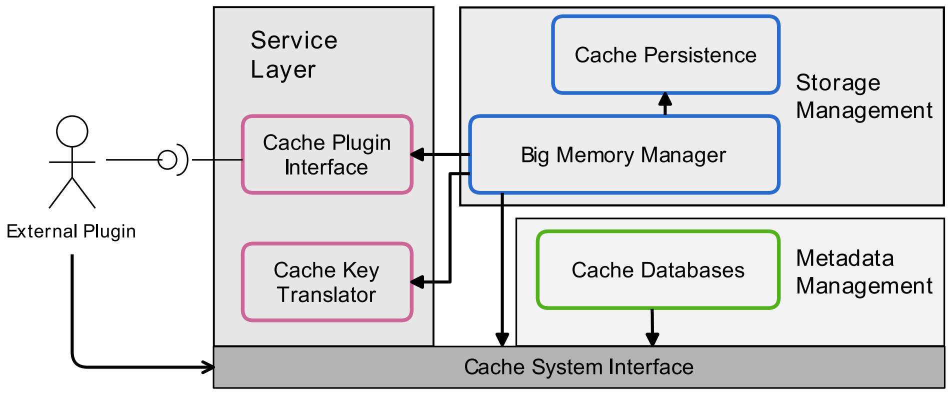


Figure 4.5: Caching system architecture.

detect, register, load, and execute the plugins, for instance, the intelligence module that provides algorithms to manage the cache. In turn, the dcm4che framework is responsible for parsing and building DICOM communication messages. MapDB proved to be a valuable tool in order to save storage space while keeping good performance in storing and retrieving fragments. Since MapDB is a key-value database, it only supports loading complete values for each key. However, the DICOM information model has four entity levels (patient, study, series, image). As a result, MapDB does not support operations based on attributes from all these levels, for instance, StudyUID and Series. Therefore, DB4O was used as a conventional database that has aggregated information of the study and series to allow direct queries for these attributes.

The metadata management module is responsible for indexing metadata related to the

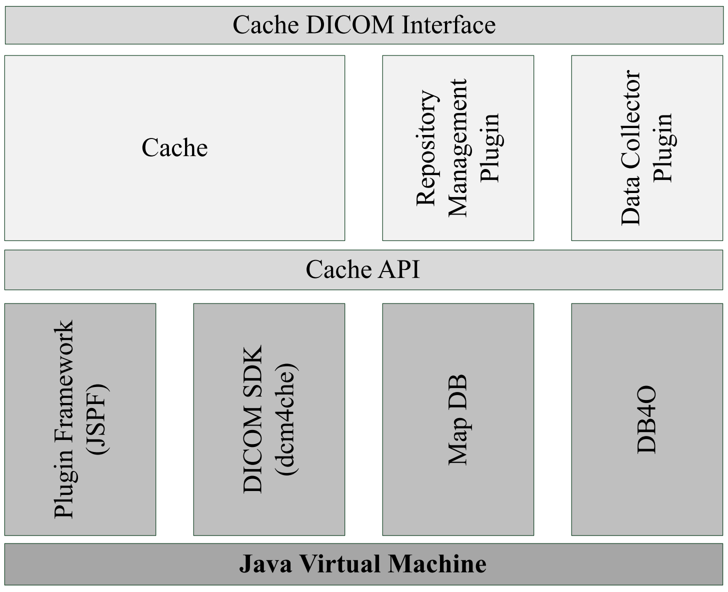


Figure 4.6: Caching system software components.

objects stored in the caching system, meaning images can be related to study, series, etc. This strategy enables fast query answering since it avoids analysing stored DICOM objects each time a query must be processed. This database is based on DB4O, an API for object databases.

Besides providing the persistence functionality, this module also provides low-level methods for querying about the existence of specific objects (or fragments), mainly to perform integrity checks. There are two components in this module: cache persistence and big memory manager.

The cache persistence is the lowest level component in the system, working only with object keys and binary data. It provides simple methods (put, get, remove, contain) to manage objects in the persistence medium. To do so, it uses MapDB, relating an object to its key.

The big memory manager component, however, provides an abstraction for DICOM objects (e.g., images) on top of the cache persistence module. It uses the service layer cache key translator to translate the objects to keys, prior to storage or retrieval. Conceptually, the big memory manager is also responsible for triggering events via the cache plugin interface module.

The elements enclosed in the service layer are intended to provide extended functionalities to other modules. The architecture includes two service components, the cache key translator and the cache plugin interface.

The cache key translator provides a centralized object-key translation. For that, this module uses a hash function to translate the object into a key. In this way, the storage management module does not need to deal with object metadata, but only with the seemingly homogeneous key generated by the cache key translator for identification of each piece of information.

When elaborating the requirements of the proposed caching system architecture, we noticed that different clients would expect different behaviours from their caching system. These behaviours are associated with different population and eviction policies. Moreover, along with different policies comes the need to manage metadata associated with the stored objects. Therefore, instead of building a static module for these policies, the proposed architecture incorporates a plug-in system where caching system clients can supply their own policy implementation plugin. They can acquire metadata through an event listener interface to receive notifications of events occurring inside the caching system (such as insertion, eviction or retrieval of an object). If necessary, they can take actions on the caching system using the public caching system interface (such as removing an object or cache one).

The proposed architecture provides an interface that acts as a wrapper, linking all system modules. There are two main reasons for bundling all specification in a unique interface. First, it is easier for cache clients (i.e., the applications that use the caching system) to interact with a single interface, than with different modules separately. Second, a common interface for cache users and developers (i.e., the developers of the external cache plugins) is less prone to create inconsistencies between modules.

The caching system interface provides methods to store objects, to query the existence of objects in the system, and to retrieve and evict objects. They are low-level methods mainly associated with the repository management module. Besides these methods, the interface also enables queries about studies (following the DIM and mimicking the C-FIND Command) and explicit registration of metadata. The metadata management modules mainly issue those methods. Moreover, the provided interface is able to receive DICOM protocol messages and respond using the same communication protocol, maintaining compatibility with the standard.

Plugins

One important feature of the proposed system is its ability to easily change the cache prediction module. For that, it was used an architecture based on plugins. There are two kinds of plugins: the repository management and the data collectors. The system requires one repository management plugin. However, concerning data collector plugins, they are optional and the system supports more than one plugin of this kind.

By definition, a caching system is a temporary storage area with limited capacity. For that reason, it is necessary to have management mechanisms to control the stored volume of data. Therefore, this plugin is responsible for managing the occupation of the cache repository, in order to maximize the probability of a needed piece of information being in cache. There are two main actions in the repository management process: cache population and cache depopulation. Finally, this plugin must register information in a database about the images, studies and series stored in the cache.

Distinct population and eviction strategies need to collect distinct measures to discover which studies will be discarded or pre-fetched. So, it was necessary to have a flexible data collector mechanism. The data collector plugins are responsible for collecting the measures needed by the repository management plugin. For instance, studying user patterns requires access to the user's request data. Besides, sometimes there are other external data sources that could have important information for the caching system.

4.2.2 DICOM Routing Platform with Caching Management

The DICOM routing platform makes use of a previously developed work that aimed to outsource medical imaging repositories to the cloud [52]. The main goal of this work was to provide interoperability, namely, query/retrieve and storage services between remote locations over the Internet using the HTTP protocol for communications. This approach enhances study exchange among institutions and facilitates the integration of repositories and the setting up of teleradiology scenarios. The architecture contemplates an application, called DICOM router, which is placed inside a DICOM island (i.e., a DICOM network without connectivity to other DICOM networks). For each local PACS, routers are viewed as DICOM nodes that provide services published by other routers (islands). Therefore, services provided by DICOM applications inside the islands will then be accessible from outside applications, through the router. The communications between routers are carried out through the bridge relay which can be located on a cloud provider.

The platform communications are Web 2.0 compliant. The processes are supported by Web services and the messages are encapsulated in the HTTPS protocol. This provides communications security and ensures the setup of DICOM services in restrictive private LAN environments, even without IT network expertise or administrator privileges. In this way, our system is able to run on most network configurations present in healthcare institutions.

This DICOM routing platform supports standard interoperability between remote medical imaging devices. Physicians can work at home, as if they were in the healthcare institution, without changing their workflow. However, the solution has some issues regarding access to medical images in scenarios with limited bandwidth, which is especially critical when retrieving large size studies.

The next sections will describe the new version of this DICOM routing platform, developed in the scope of this thesis. It improves the previous version with the proposed caching system, and new search and retrieval processes. The cache eviction algorithm with object splitting will be also described.

Searching

One of the main features of the proposed system is related to its ability to provide search functionality in a seamless way. For instance, visualization workstations can send DICOM requests (i.e., C-FIND-Request) and receive a response from the DICOM router, as if it was an intranet PACS archive.

The DICOM client application sends a request command to the router, which forwards it to its local cache. In parallel, the request is also forwarded to the bridge relay. At the bridge, the request is forwarded to the router that serves the destination AETitle. When a response is returned, the bridge terminates the session with the destination router and reroutes the response to the original router. Upon receiving the response, the router forwards it to the requester DICOM application.

All this process of consulting the remote archive before answering is much more efficient than responding directly from the cache database. By doing this, the router can check if its cache metadata is updated, enabling more accurate responses

Retrieval

The implementation of C-MOVE Command in this distributed environment is very similar to the C-FIND Command. However, the cache deployment in the router increases the overall availability of archived data and reduces studies' retrieval time.

Figure 4.7 presents a scenario where a physician wants to retrieve a study that is stored in an outsourced archive hosted, for instance, in a private cloud. A public cloud provider hosts the bridge relay. Nevertheless, the resources involved in this setup could be deployed in other cloud environments, either public or private. It illustrates the transfer of a study composed of six blocks of data. To take advantage of cached objects, the C-MOVE command workflow is carried out as follows.

1. The router receives the request (i.e., C-MOVE-Request) that will be first forwarded to the router's local cache. If the study is in the cache, the router can start serving the requester. Even if the local cache satisfies the request, the request is always forwarded to the bridge, because the archive may have received new images or series (Step 1). Along with this message, a "prune work-list" is also transmitted containing the objects stored in the router's local cache (hit-list). In the example, the SCU router contains blocks 1, 2, and 3 in its local cache, i.e., the "prune work-list."

2. The bridge receives the C-MOVE request and the "prune work-list", forwarding it to the router that advertises the destination AETitle. A C-MOVE session is created in the bridge, in order to track the response from the destination router, i.e., the SCP router.

3. The remote router forwards the C-MOVE request to the PACS archive (Step 2). The request is also forwarded to its cache. If the archive is unreachable, the set A will be empty.

4. In order to respond to the bridge, the SCP router produces a list of DICOM objects that exist in the archive (or cache), but are not yet present in the issuer's router cache (i.e., "prune work-list"). In the example, Step 3, the result will be the "upload work-list" enumerated as R (4, 5, and 6 blocks) that is sent back to the SCU Router (Step 4). The 4 and 5 are already locally in its local cache, and 6 was moved from the PACS archive.

5. The SCU router merges the response from its cache, and the received new blocks, (Step 5) into the final requested result.

6. Finally, the SCU router starts uploading the worklist by transmitting the objects (or fragments) using the data channels (Step 6), as explained in [81].

Cache Eviction Policy

The routing platform uses a cache eviction algorithm based on LRU, which is commonly used for different cache purposes [112], combined with DICOM objects splitting technique. In a PACS archive outsourcing scenario, the studies produced at each institution are cached

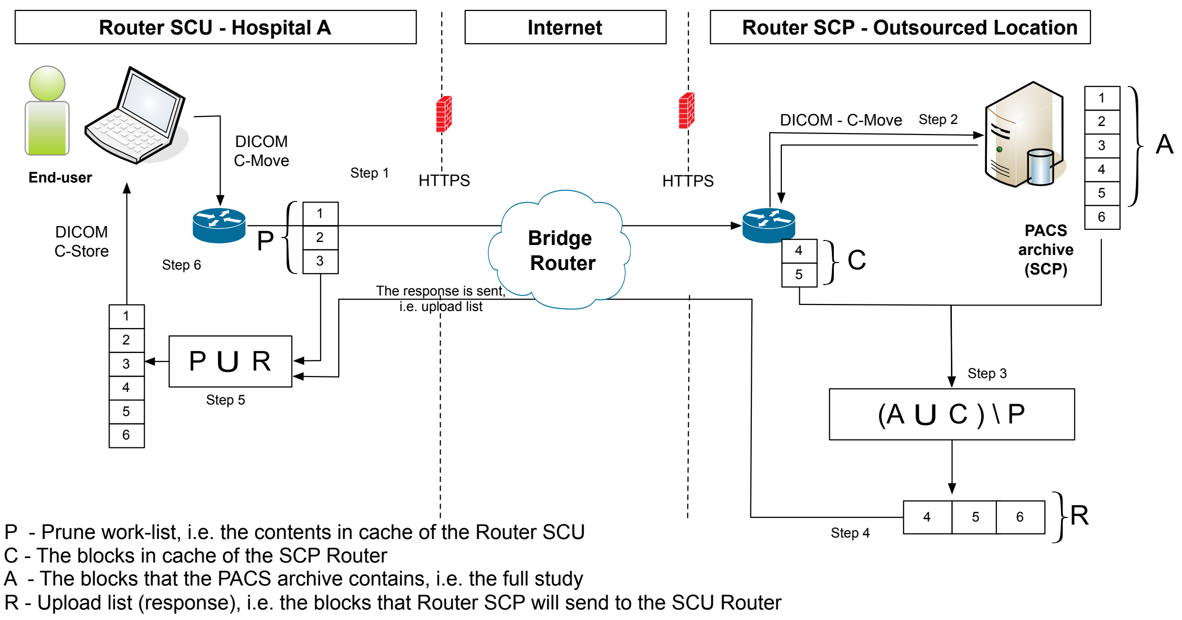

Router SCU - Hospital A
Internet
Router SCP - Outsourced Location
End-user
DICOM C-Move
Step 1
HTTPS
HTTPS
DICOM - C-Move
Step 2
A
DICOM C-Store
Step 6
P
Bridge Router
C
PACS archive (SCP)
The response is sent, i.e. upload list
P U R
Step 5
Step 3
(A U C) \ P
Step 4
R
P - Prune work-list, i.e. the contents in cache of the Router SCU
C - The blocks in cache of the SCP Router
A - The blocks that the PACS archive contains, i.e. the full study
R - Upload list (response), i.e. the blocks that Router SCP will send to the SCU Router

Figure 4.7: Illustration of content retrieval in the distributed environment.

immediately in the respective local router. However, the studies cannot be cached forever due to capacity limitations. As a result, the standard LRU algorithm evicts objects (studies) which are least used. We adapted this algorithm to explore the splitting technique. Our version evicts objects by a combination of their “age,” date of acquisition, and the Least Recently Used information. By doing so, higher priority is given to recently produced studies.

The proposed splitting technique enables our caches to store incomplete studies in order to save storage space. Therefore, the router does not always evict complete studies from the cache, but instead, it gradually evicts portions of a study taking into account the amount of storage space available in cache and the priority of the study. The amount of each study in our caches is given by (equation (4.1)), as shown at the bottom of the page, where n is the nth most high-priority study. The percentage of free cache is calculated using equation (4.2), where CacheSize is the cache storage space and size(i) is the size of the ith most high-priority study. Notice that the cache storage space was intentionally fragmented into three regions with distinct Quality of Service. The regions definition is based on the maxRegionStart and minRegionEnd values defined according to user/institution requirements. For instance, it is possible to define distinct regions per medical imaging modality (equation (4.1)) . The optimal maxThreshold and minThreshold are values obtained for each routing platform setup.

$$\%study(n) = \begin{cases} maxThreshold, & if\%freeCache(n) \in [maxRegionStart; 1, 0[\\ \%freeCache(n), & if\%freeCache(n) \in [minRegionEnd; maxRegionStart[\\ minThreshold, & if\%freeCache(n) \in]0, 0; minRegionEnd[\end{cases} \tag{4.1}$$

$$\%freeCache(n) = \frac{CacheSize - \sum_{i=1}^{n-1} Size(i) \times \%Study(i)}{CacheSize} \tag{4.2}$$

The high-quality region provides optimal retrieval times for cached objects. The low-quality region reflects the tradeoff situation. Studies cached in this region do not benefit from optimal retrieval performance. However, this enables our caches to store more studies, covering a longer time span. Between these two regions, the portion of each study cached follows an adaptive function, based on the amount of space still available in the cache. As a result, the Quality of Service offered for these studies is slightly degraded as the priority of the study decreases. However, this region further enables more studies to be cached with controlled performance losses.

4.2.3 Workflow Management

As expressed, one of the most common approaches used to minimize the communications latency in distributed systems is to avoid fetching resources from remote locations through the usage of cache mechanisms. The idea is to store a representative image of the distributed system's resources in a cache, serving these resources with an increased QoS and hoping that requests for cached resources outnumber requests for those which are not. Therefore, a pertinent question arises: how to populate caches with resources that will be most requested?

The answer to this question required an in-depth analysis of the medical imaging workflows. There are some indicators for the likelihood of a study being requested in the near future. For instance, physicians are likely to request previous studies from the same patient, or specific studies could be referenced by a worklist in the RIS. These patterns and prefetching rules must be tailored for each institution, taking into account its workflow, software and clinical staff behavior.

Aiming to mitigate of the previously described issues by increasing the effectiveness of caches, this section proposes a new module, called Workflow Management for PACS Environments (WMPe), that enables the automation of tasks within a distributed medical imaging scenario.

Architecture and Lifecycle

The proposed methods aim to improve the radiology workflow, especially in distributed scenarios. Repetitive tasks can be automated and delegated to off-duty hours, freeing the PACS from handling them during business hours when infrastructural usage is at its peak. Initially, the proposal was based on the definition of static rules for caches population. However, the scope of this work was broadened to support distinct usage scenarios. So, a novel module-based architecture was developed to support the orchestration of heterogeneous tasks. It makes possible to define static rules that match PACS resources and then perform a given procedure on them. The resources may be medical imaging studies, patients, worklists or any other asset in the PACS-DICOM universe. For instance, it is possible to move all the studies performed yesterday, in a remote repository, to a local one. Another illustrative example is to move previous studies from patients which are present in today's worklist. Moreover, the system was designed to assist critical tasks in the PACS lifecycle, such as migration of studies or population of caches.

WMPe splits the PACS components into two categories: resource holders and targets; The first type holds some abstract resources, the most flagrant example is the main PACS archive which contains studies the previously acquired studies. On the other hand, a cache may act as a resource target when receiving contents from the main archive. Resources are an abstraction used to map any PACS object in the proposed rule system. WMPe's modus operandi consists of scheduling and launching tasks, which are composed of two distinct stages: Monitoring and Action. In the monitoring stage, a resource holder, such as a PACS archive, is scanned. A new action event is triggered for each resource found matching the defined rules. The action phase consists in the actual execution of the desired task. The monitoring stage is triggered by a scheduled event. The time, date and periodicity which these events are fired are configurable according to the user requirements.

To demonstrate the methods underlying the WMPe, the following deployment scenario will be used: "a central archive shared by multiple institutions, being possible to schedule movements of studies to local caches through the usage of rules". In the monitoring stage, the WMPe will search in the central repository for studies belonging to patients with scheduled procedures for the next day. Conversely, in the action stage, the found studies would be moved to the local cache archive. Although this illustrative scenario is very restive in terms of resource holders and targets, the WMPe supports virtually every holder, target and resource triplet due to its module system. WMPe's tasks are implemented separately from the core itself and bundled together when the system is deployed. Furthermore, the monitoring and action phase can be spread by different modules, enabling the orchestration of new or existent functionalities. This plugin-like system enables a complete abstraction between tasks functionality and the scheduler environment.

The WMPe architecture is composed by three major components, described in Figure 4.8: the Module Manager, which is responsible for managing the module's lifecycle; the Event Scheduler, responsible for firing the tasks; and the Configuration Manager which provides an interface for configuring the system. This is illustrated in Figure 4.8 along with their interactions within the WMPe's lifecycle.

Workflow Manager lifecycle starts by loading the configuration files encoded in XML (Extensible Markup Language) format. It contains information about the schedules, tasks and modules directory. After this, the other two main components are initialized. The Module Manager starts by searching module implementations in the respective directory.

Next, the modules classes are loaded into the execution environment and the tasks defined in configurations are launched.

The Scheduler Module is responsible by processing the scheduled entries that are associated to tasks. Therefore, when an event is ready to be fired, the Scheduler sends a signal along with the tasks identifier to the Module Manager. As a result, this module will redirect the signal to the corresponding module, triggering the monitoring phase.

The Rule Engine is a very important component since it enables the selection of resources requested by the Task. However, the concept behind the Rule Engine is simple. It defines an interface that offers a decision based on an input resource. Since, in our scenario, different modules may have different representations of resources, this procedure must be implemented by monitoring modules. Rules are formatted similarly to the query requests in DICOM, in which a tag is coupled with its desired value or filter, for instance, `StudyDate:20150215`. Despite most users being familiar with this kind of queries, they did not completely fit our needs since dates must be absolute. For instance, it is not possible to specify a floating date such as today's, similarly to SQL's `now()` function. To mitigate this problem, the Rule Engine provides a property list that extends the regular wildcards, adding more keywords such as `$today`, `$yesterday` or `$last-week`. These keywords are translated into their absolute value right before the matching procedure. In addition to these properties, it is also possible to replace a property with the output of a module, simply by using `${n}{property-name}` where n is the module number which is followed by the property name. As a result, our method is capable of orchestrating heterogeneous modules thus enabling WMPe to be extensible and

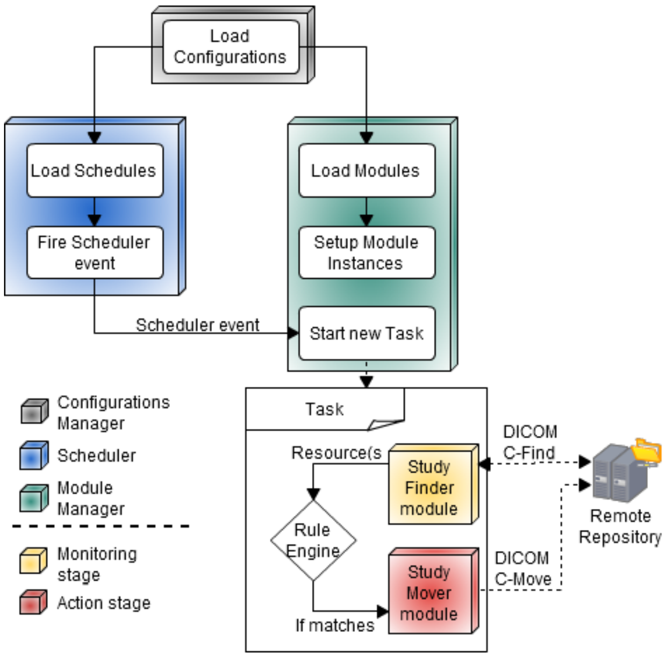


Figure 4.8: WMPe architecture and lifecycle.

adaptable to every scenario.

Bundled Modules

The operation of WMPe system was demonstrated through the implementation of three modules: Study-Finder, Study-Mover and Patient-Finder. The Study-Finder leverages the DICOM C-Find command for searching studies in a repository. Therefore, this is a typical case of a monitoring stage module. A Unique Identifier, i.e. the *StudyInstanceUID*, is retrieved for each study matching the defined rules. This is accomplished by sending a C-Find-Request to the repository, requesting the *StudyInstanceUID* of the studies that match the defined rules. The number of images in the study is also retrieved. Although this property is not required in every scenario, it has proven useful for some use cases.

The Study-Mover is an action stage module that receives a StudyInstanceUID from Study-Finder and issues a C-Move command, to transfer a study between source and target repositories. Finally, the Patient-Finder evidences the facilities of orchestrating modules. It searches patients in a DICOM MWL SCP with appointments, according with a timespan configurable through the rules. This module is able to extend the properties provided by the Rule Engine. For instance, it is possible to enter the procedure's date as a floating date (`StudyDate:$today`). However, it cannot be used to directly feed the Study-mover module since it only outputs the PatientID. This must be done through the module manager by replacing the PatientID tag in the rules of the Study-Finder module with the output resource of the Patient-Finder Module.

On the other hand, we developed the Study-Mover as an action stage module. This module receives a StudyInstanceUID from the last module as the input and issues a C-Move command to the source repository. This repository then initiates the study transference procedure to the target repository. Both source and target repositories are defined in the configurations file.

In order to evidence the facilities of orchestrating modules, we have also developed the Patient-Finder module. This module searches in a DICOM MWL SCP for patients with appointed procedures. The timespan of the procedures is configurable via the rules. Moreover, this module leverages heavily the extended properties provided by the Rule Engine since it is possible to enter the procedure's date as a floating date, for instance, `StudyDate:$today`. This module only outputs the PatientID, therefore it cannot be used to directly feed the Study-mover module. However, thanks to the module manager, we are able to orchestrate this module with the Study-Finder. This is done by replacing the PatientID tag in the rules of the Study-Finder module with the output resource of the Patient-Finder Module

Demonstrative use cases

The proposal of WMPe was driven by the need to bring automation tools into the PACS universe. Two main scenarios were identified as important use cases, both referring to affiliated institutions.

In the first case, the institution is spread across distinct geographic locations. Medical imaging studies are produced in any facility and their revision can be done in any point of the distributed network, including at home. The central archive is located at one specific facility, while the remaining locations have cache archives.

Concerning the operational requirements, it is expected to ensure that the cache archive contained the most recently produced studies by all modalities connected to the network. The task's configurations are shown in 4.1 and includes two execution modules. Firstly, the Study-Finder module is executed to find the yesterday's *StudyInstanceUIDs*. After, the Study-Mover module is called for each StudyInstanceUID found, moving the study to the ``CACHE-REPO''.

Listing 4.1: WMPe's task configuration for the first scenario.

```
<task name="import-0">
 <module id="1" class="StudyFinder">
  <origin-repository>
   <hostname>XXXXX.XXXXX.XX</hostname>
   <port>XXXX</port>
   <aetitle>CENTRAL-REPO</aetitle>
  </origin-repository>
  <rules>
   <dicom tag="StudyDate" id="00080020">\${yesterday}</dicom>
  </rules>
 </module>
 <module id="2" class="StudyMover">
  <origin-repository>
   <hostname>XXXXX.XXXXX.XX</hostname>
   <port>XXXX</port>
   <aetitle>CENTRAL-REPO</aetitle>
  </origin-repository>
  <destination-aetitle>CACHE-REPO</destination-aetitle>
 </module>
</task>
```

The second use case involved the migration of studies to the central archive. Although studies produced by local modalities were stored in the cache and then progressively uploaded to the remote archive, it was necessary to introduce a verification process since one facility had Internet connectivity problems. Therefore, it necessary to deploy an automatic method to check if all images present in the local repository were also present in the central archive.

The configurations for this scenario are shown in Listing 4.2. This task involves a more complex setup. Firstly, the Study-Finder is executed for filtering the studies in the local repository by yesterday's date. Then, for each study founded, another instance of the Study-Finder is executed. This time, the studies in the central repository are not only filtered by date but also by StudyInstanceUID and number of images. These values come directly from the previous execution of the Study-Finder module. For each study found that does not have the same number of images in both repositories, a Study-Mover module is called to perform the transference. Note how the properties are linked between the different module executions.

Listing 4.2: WMPe's task configuration for the second scenario.

```
<task name="export-0">
 <module id="1" class="StudyFinder">
  <origin-repository>
   <hostname>localhost</hostname>
   <port>1045</port>
   <aetitle>CACHE-REPO</aetitle>
  </origin-repository>
  <rules>
   <dicom tag="StudyDate" id="00080020">${yesterday}</dicom>
  </rules>
 </module>
 <module id="2" class="StudyFinder">
  <origin-repository>
   <hostname>XXXXX.XXXXX.XX</hostname>
   <port>XXXX</port>
   <aetitle>CENTRAL-REPO</aetitle>
  </origin-repository>
  <rules>
   <dicom tag="StudyDate" id="00080020">${yesterday}</dicom>
   <dicom tag="StudyInstanceUID" id="0020000D">${1}{StudyInstanceUID}</dicom>
   <property not name="NumberOfImages">${1}{NumberOfImages}</property>
  </rules>
 </module>
 <module id="3" class="StudyMover">
  <origin-repository>
```

```
   <hostname>localhost</hostname>
   <port>1045</port>
   <aetitle>CACHE−REPO</aetitle>
  </origin−repository>
  <destination−aetitle>CENTRAL−REPO</destination−aetitle>
 </module>
</task>
```

Finally, in order to fire the scheduler events accordingly, a separate entry was configured for each task. Both tasks are executed once a day, during the night, to avoid perturbation of normal workflows. The format used to enter the dates is the DateTime, Time and Duration defined in the XML's standard. The initial date is omitted since the task starts when the WMPe is executed. The task configurations are shown in Listing 4.3.

Listing 4.3: WMPe's schedulers configuration used in the described scenarios.

```
<schedules>
 <schedule>
  <task>import−0</task>
  <start unit="time">01:00:00</start>
  <interval unit="duration">P1D</interval>
  <end unit="date"></end>
 </schedule>
 <schedule>
  <task>export−0</task>
  <start unit="time">03:30:00</start>
  <interval unit="duration">P1D</interval>
  <end unit="date"></end>
 </schedule>
</schedules>
```

4.2.4 Controlled Channels

This section proposes a mechanism to improve the communications in the distributed PACS environment described above. It provides a high-level control framework for the routers' point-to-point communications, aiming to improve its performance on top of other transpor layers network flow and congestion control algorithms. It provides a generic implementation that makes no assumption about the actual transport protocol beneath it, nor the transferred data.

The Controlled Channels' design started by identifying the transference properties that have a higher impact on the PACS QoS and might be more perceivable to users; The

Table 4.1: Data profile of the dataset A.

Modality	Number of images	Volume (MB)	Average image Size (KB)
NM	1	1	1 000
NM	5	2	400
NM	6	8.2	1 367
PT	244	16.3	67
MR	223	47.1	211
MR	369	206.1	559
XA	15	401.6	26 773

transference delay is the actual time needed to complete the transference. The *communication channel's throughput*, which is the relevant data transfer rate, discarding any control overhead. Lastly, the *channel establishment delay*, which is the time taken by the channel to start transferring relevant data. Improving these three properties should improve the performance of the system itself.

The transference of medical image studies raises certain issues related to the potentially huge volume of data and its heterogeneity. Distinct modalities produce studies with different data characteristics such as total volume, number of images and average image size [113]. This is shown in Table 4.1 which encloses 5 different modalities, Nuclear Medicine (NM), Positron emission Tomography (PT), MR, and X-Ray Angiography (XA).

As a result, the controlled channels architecture needed to be adaptable to provide the best improvement for each possible data profile. For instance, studies with many small files generate more control traffic overhead but benefit from a small transference delay. Since a Bridge is used in the DICOM Routers mechanism [52], the establishment of Router-to-Router connections has a higher delay due to the multiple points involved, i.e. Router-Bridge and Bridge-Router. On the other hand, in studies with a few large files, the connection establishment delays are meaningless and a maximum throughput is desirable. Considering the previous issues, the Controlled Channels mechanism implements a method to normalize the studies data profiles.

Data Normalization

The data profiles normalization contemplates two main processes. Firstly, splitting the image data into chunks, i.e. dividing the binary data into portions. To ease the chunk management and keep the control overhead at the minimum, each chunk includes a descriptor that identifies it unequivocally. It contains the chunk size, its position in the binary data (the chunk number) and the total number of chunks produced from the file. Holding both the chunk's data and the chunk descriptor, any application can recombine the data and reproduce the original file. Chunks are also compressed, using the bzip2 format [114], to optimize both the storage space and the transference performance. The chunk splitting process was implemented in the DICOM Routers before the data transference.

In addition, a bulk transference method was also applied where a predefined number of files are transferred sequentially as a group. To accomplish this, the Controlled Channels have an inner queue where chunks wait for its turn to be transferred. When the queue reaches the required number of chunks, a new bulk is dispatched. There is also a maximum waiting time in which an incomplete bulk will be sent. This prevents chunks from waiting too long in the queue and minimizes the *transference delay*.

Combined these two methods create a normalized view of the different studies because the chunk size is the same for all modalities. The difference relies on the number of chunks and their arrival rate to the channels inner queue; studies such as XA will produce a few bursts of many chunks, while studies with many small images will produce a constant stream of few chunks.

The solution to deal with both scenarios resides on the configuration of the Controlled Channels parameters, namely the *bulk size* and the *maximum pool time*. Increasing the *bulk size* reduces the control overhead and theoretically increases the throughput of the channel. This should be done when the channel has many chunks in the queue. However, there is a cut-off point where the throughput of the channel cannot be increased further. The solution then is opening another parallel channel. Decreasing the *maximum pool time* and *bulk size* will reduce latency between completed bulk transferences, thus providing lower *transference delay* to smaller studies and images.

Dividing DICOM images into chunks has another advantage; more flexibility in managing the images data. For instance, it is possible to divide images into two distinct repositories or store only a portion of the image in a provider. Conventional archives do not support incomplete images but the proposed cache mechanism (section 4.2.1) can take advantage of it.

Use Case

Figure 4.9 shows the integration of the proposed Controlled Channels in the routing mechanism described above when a DICOM Query Retrieve service request is handled by multiple repositories. The process is initiated by a normal client application (e.g. workstation of the physician at home) which sends a C-Move RQ to the Intranet Router. This message is immediately forwarded to the Bridge that broadcasts the request to every DICOM Router advertising the requests' SCP AETitle. This procedure initiates a small C-Move session, which orchestrates the transference. Namely, it defines which chunks each archive must upload. This is performed in parallel, so the overhead introduced by the increasing number of archives is minimized. See stage 1.

When a SCP Router receives a request, it queries its archive about the images contained in the desired study, receiving a list of results. If the archive does not have an entire image,

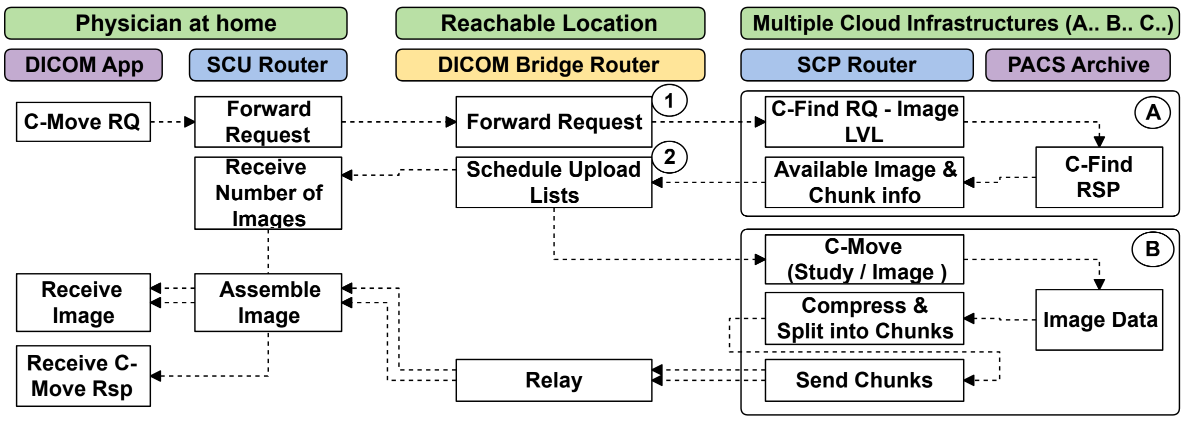


Figure 4.9: DICOM Query-Retrieve service from multiple sources.

the router only sends information of the available chunks in the cache. The list is sent back to the Bridge. This procedure is executed simultaneously on every router. See the box A.

After receiving all the Routers' responses, the Bridge has all the necessary information to schedule the transference, i.e. to assign which router will upload which chunk. The scheduling algorithm is implemented in an external plugin. By doing so, we virtually support every scheduling behaviour possible. This is very important, as different institutions have different intentions towards the platform and may want to prioritise certain studies or repositories. Upon finishing the scheduling, a message is sent to every router with a list of images or chunks that they need to upload. In parallel, the Bridge also sends an informative message to the SCU Router with information about the images which will be transferred. See stage 2.

When each SCP Router receives the upload list, it immediately starts processing the data. If the Router has all the requested chunks cached, it starts splitting, compressing and transmitting the data to the SCU Router. Once again, this procedure is parallel to every router (see block B). While receiving the data, the SCU Router immediately assembles the completed images and sends them to the SCU application. When every image is moved, the router finally closes the association sending the C-Move RSP message.

4.2.5 Validation

Performance

Concerning the quantification of the performance improvements obtained exclusively with the usage of controlled channel methods, trials were performed with a control dataset enclosing multiple studies from different modalities representing different data profiles. Dcm4che[1] was used to deploy a storage server and a storage client simulating an environment of a distributed PACS archive. This simulation represents a usual real-world scenario. The elapsed time

[1] http://www.dcm4che.org/ (accessed: 26/10/2017)

Table 4.2: Evaluation of the Controlled Channels method.

Modality	Number of images	Volume(MB)	VPN (s)	Controlled Channels(s)
NM	1	1	2.9	2.6
NM	5	2	5.7	3.6
NM	6	8.2	12.2	5.5
PT	244	16.3	27.4	33.5
MR	223	47.1	59	32.8
MR	369	206.1	264.4	82.4
XA	15	401.6	622.8	350.4

accounted by the dcm4che application is shown in the results in table 4.2.

The deployment scenario involved a datacentre where the storage command was called to an externally located Storage Provider. This scenario offered a 30 mbps downstream bandwidth for the SCU and a 10 mbps upstream for the SCP.

The trials aimed to compare the performance of the often used approach, i.e. VPN connection against the proposed method. In the first scenario, the VPN was used to link both networks holding the SCP and SCU as suggested in [52] (note that this version as some limitation expressed in [52]). The other scenario used our distributed PACS with the Controlled Channels. The Controlled Channels were used with parameters of bulk size = 10, maximum waiting time = 1s, and chunk size = 50KB and a maximum of three channels were used in parallel.

The Controlled Channels solution presents a general improvement of performance compared to the other possible scenarios. Better improvements are achieved for studies with higher average image size where there is also more need for improvements. For smaller studies, the archived improvements are not significant when the system is only supporting a single user.

In order to demonstrate the feasibility and the performance benefit of the DICOM routing platform with cache mechanism included, the solution was deployed in a real-world environment. The scenario involved two geo-distributed institutions sharing a private PACS-cloud repository. They belong to the same owner and a radiologist can practice in both institutions and report examinations remotely. Those institutions deal with different modalities, handling an average of 3000 examinations monthly, with a combined volume around 60 GB. Each location has its own DICOM router with the proposed caching system implemented. Previously, without using the presented methods, these caches were able to hold approximately eight months of PACS usage (500 GB). The latency associated with remote retrieval of medical imaging studies is a critical issue.

Before evaluating the performance gains associated with the proposed solution, the retrieval times without using the routing platform were measured. The trials consisted of recording the delays of moving several medical imaging studies between the PACS-cloud archive and the remote DICOM client application. The different locations were connected through a 30 Mb/s channel. The average download times are presented in table 4.3.

The baseline in our trials is the study retrieval time when using the standard DICOM transfer through a VPN connection. Next, the DICOM router platform is used, and even without caching, this platform considerably speeds up study retrieval times, on average 1.55 times, with the best case study up to 2.4 times (MR-3 study).

The justification is in the inefficiency of the DICOM protocol in WAN, which introduces severe performance penalties. Finally, results of the DICOM router platform with complete study cached are also presented in table 4.3. This trial is equivalent to accessing the local network PACS archive since no image data have to be retrieved from a remote location. In this scenario, the DICOM router platform performs optimally, speeding up the transfer up to 5.8 times (XA-4 study), 4.17 on average, when compared to the standard VPN usage.

The next step was to evaluate the performance of the DICOM router platform with cache, using the splitting technique. The trials were repeated for multiple percentages of each cached study, from 5% to 100% with hops of 5%. The results are presented in Figure 4.10. As expected, the usage of cache reduces the latency. However, the improvement introduced by the proposed caching system is not limited to the enhancement of retrieval times, but on finding the minimum percentage of caching that assures the same retrieval time of a full cached study (i.e., the optimal retrieval speed-up — maxThreshold). Knowledge of this threshold allows the proposed system to actually save storage space, permitting more studies in the cache, without impacting the retrieval performance. This maxThreshold level can be automatically adjusted according to the network's conditions. As expressed, the threshold is the result of fetching the image data from the remote environment while moving, simultaneously, the available images in the cache to the client application (i.e., the consumer). So, if the Internet connection bandwidth increases, less amount of study is required to be present in the router cache, and vice versa.

Analysing Figure 4.10 charts, we realize that caching 70% of the studies is enough to achieve a near optimal speed-up. This situation represents an increase of 43% in cached exams. On the other hand, the vertical line at 45% represents a frequent trade-off situation (i.e., the minThreshold). The optimal speed-up in retrieval times is not achieved, but the

Table 4.3: Routing and cache mechanism performance evaluation.

			Retrieval Time (s)		
				DICOM Router Platform	
Modality	Number of Files	Volume (MB)	VPN Connection	0% Study Cached	100% Study Cached
PT-1	244	16.3	14.3	13.8	7.4
MR-2	223	47.1	31.1	19.1	8.9
MR-3	369	206.1	115.5	48.1	21.2
XA-4	15	401.6	228.7	202.8	39.4

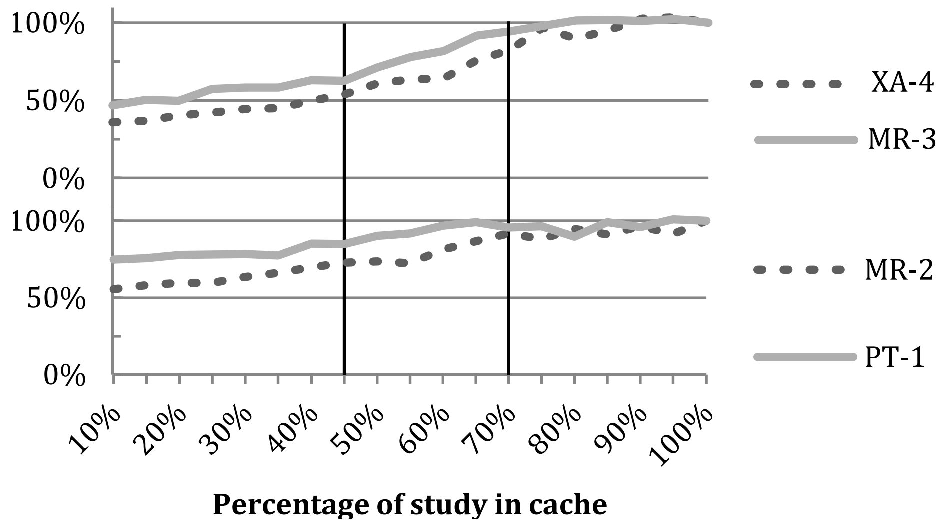


Figure 4.10: Percentage of the achieved speed-up (Y-axis) with multiple percentages of the study in cache (X-axis), taking the complete study in cache as reference (in inverse order).

required storage space is reduced to 45% of the study. In this case, larger studies tend to perform a little worse than smaller ones, due to control overhead.

The reduction of study retrieval times is evident in teleradiology sessions where physicians are remotely reporting. Moreover, institutional routers have now more studies in the cache, more than 16 months, of which five months are serviced with optimal retrieval performance.

Stress Trials

Besides calculating the impact of the routing mechanism with the proposed caching system, in terms of performance and storage space usage; we also performed a stress test to assess the impact of multiple institutions using simultaneously and intensively a cloud-based archive served by our platform. As in the performance trials, we measured the average retrieval times for each study in our dataset. However, in this case, we introduced two new institutional routers in the setup, i.e., noise routers.

The average time for retrieving the reference dataset was measured in each cache region. A degradation factor, which gives a measure of how much the average retrieval times decreased when compared to the reference, is presented in Figure 4.11. This degradation factor is essentially the inverse of the speed-up. The chart shows the degradation factor in three network situations: *"Noise = 0"* when no other router is requesting studies besides the one assigned to our reference dataset, i.e., the same scenario as the performance trials; *"Noise = 1"* when another router is active besides our reference; and *"Noise = 2"* when two more routers are active. Analysing the results, we can check that the degradation factor is residual when we cache 70% of the study and there are no simultaneous requests (*Noise = 0*). Nevertheless, when the noise levels rise, the degradation factor does not increase in the same proportion, less than 2x and 3x, which introduces another added value for our system. This behaviour is transversal to both regions (i.e., caching 70% and 45%). Moreover, the DICOM router platform is shown to perform well even without the study in the cache (0%), when compared with the standard VPN solution. Therefore, we confirm that our system holds its performance improvements even in intensive scenarios.

In the trials, each client router was programmed to continuously request studies. Initially, their local cache was prepopulated with a separate dataset of studies, all of them contained in the central PACS archive. The trials were performed with three distinct percentage of study cached (0%, 45%, and 70%), according to the regions identified in section 4.2.2. The optimum speed-up scenario, where each study is completely cached, was used as the reference time.

4.3 Content Discovery

Our research in the optimization of content discovery services started with a careful evaluation of the available database technologies. This research is presented in section 3.4.4. In this initial phase, the database technologies with potential to be used in the PACS context were identified. After analysing all these technologies, the most evident conclusion is that everyone has strengths and weaknesses. Consequently, the choice of a definitive database model for PACS depends on the usage scenario in which the PACS operates.

Therefore, the next steps were focused on better characterizing the possible usage scenarios

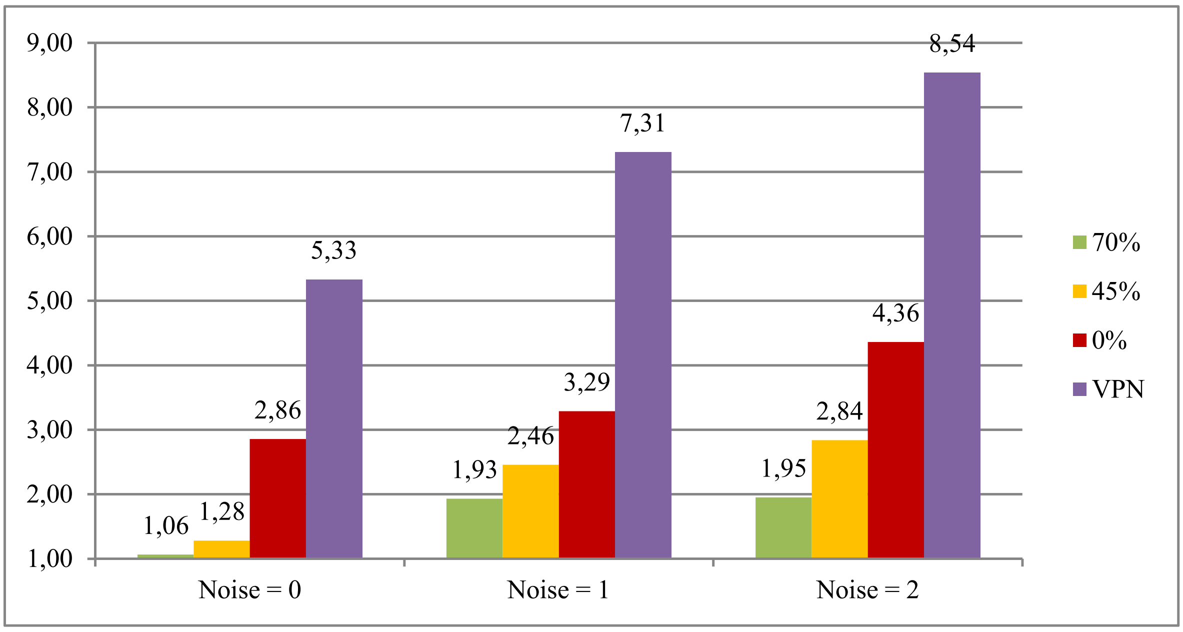


Figure 4.11: Degradation factor for the reference dataset under different conditions: Noise = 0, only the reference router; Noise = 1 another router besides the reference; Noise = 2, two noise routers besides the reference router.

for PACS and their requirements. Two major groups of usage scenarios were identified. The enterprise PACS, which mainly supports directly the medical practice, and the research PACS, which supports clinical research, BI, and other analytical tasks. These broad usage scenarios have very different requirements that are presented in greater detail in section 3.4. Moreover, in a finer level, the PACS workflows vary a lot from institution to institution, and even within different modalities, thus changing the solution's requirements as well. Those findings were also supported by the experience acquired in previous participations in medical imaging projects. Moreover, this difference in requirements was the major cause for digital pathology not being supported by PACS alongside other modalities. Considering the previous conclusions, it was decided not to propose nor develop a database model for content discovery in PACS. Since the solution would simply not be able to cover the requirements of all the possible usage scenarios for PACS.

Instead, this section presents a framework for validating the performance and scalability of PACS content discovery and retrieval services. In combination with Dicoogle, it offers a full-stack solution for the prototyping, development, and validation of new PACS services. The proposed framework provides automatic deployment of test instances, assists the definition of test cases and the collection of relevant metrics.

Great attention was given to the major issue in the validation of content discovery solutions (see section 3.4.4), the lack of meaningful datasets. Besides the exhaustive work of collecting, anonymizing and organizing datasets obtained in real-world institutions, a method for generating larger datasets was also proposed. It is able to extend small datasets representative of the real-world scenario in study. This generator is particularly important to assess the long-term performance of a solution.

The proposed framework has been supporting the research on the two advanced usage scenarios for PACS presented in the next chapter, as well as in the development of new functionality for Dicoogle.

4.3.1 Performance and Scalability Testing Framework

This thesis presents distinct methodologies for improving the performance of PACS services. These contributions were initially supported by theoretical evidence, which indicated that there was margin for improving the targeted service or task. However, this thesis aims the application of its contributions to actual systems. Therefore, it was necessary to demonstrate that our proposals introduced, in fact, a practical improvement through comparative analysis with other solutions. In this context, validation and testing are major tasks of performance engineering.

In this regard, it was developed a performance testing framework for PACS services. This tool is not intended to replace the common integration and unitary tests frameworks that have a very important role in the software development. Instead, it is intended to validate the

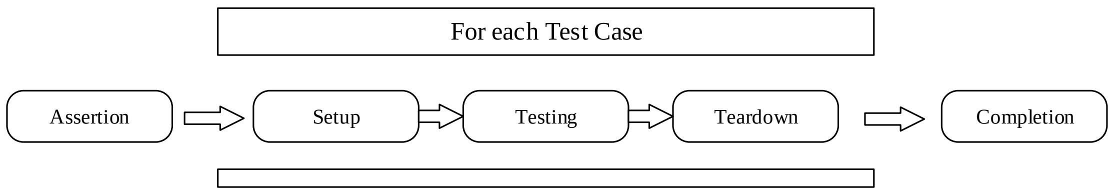


Figure 4.12: Overview of the different stages of the validation framework.

non-functional requirements related to the software's performance.

There are many performance testing frameworks available. Some of the most well-known are Gatling[2] and Locust[3]. These two are focused on Web service testing, with special interest towards load testing. Lately, Apache JMeter[4] has been slowly gaining more acceptance of the performance engineering community. It enables testing more services, like TCP/IP, rather than just Web services because it supports the development of new adapters.

However, after trying to use them in our scope it became clear why tailor-made solutions are still predominant in the performance engineering field. It is very hard to find a solution that works well for all the test subjects. For instance, in the scope of this thesis, it was required to evaluate the performance of a large variety of services, ranging from the low-level DICOM services to DICOM Web-Services, and even the performance of the pathology viewer application presented in the next chapter. Each case required different metrics to be recorded, and different processes for preparing the test cases and analysing the outputs. It was possible to develop new adapters for JMeter to support the low-level DICOM services, however, it would not support the automatic deployment of test instances.

In this picture, it was decided to create our own framework for supporting the validation of the most important services of PACS, focusing on the Dicoogle PACS. However, it may be used for validating any other PACS solution. It was written in Python, as well as the test case definitions. The test definitions are essentially class modules. Therefore, their functionality can be extended and higher level test cases can be composed by other test cases using the Object-Oriented (OO) paradigm.

Framework Lifecycle

The developed framework defines five sequential stages: Initial, Assertion, Setup, Test, and Teardown, and Completion. These stages are illustrated in Figure 4.12.

In the Initial stage, the test definitions are imported by the framework. They extend a base class that defines four methods that map the remaining stages. Test cases are encouraged

[2] https://gatling.io
[3] https://locust.io
[4] http://jmeter.apache.org

to override these methods with proper implementations. The definitions are written in python and they can also import any third-party dependency required by test case definitions.

The Assertion stage guarantees that all the deployment requirements for the upcoming tests are satisfied. For instance, during the development of new plugins for Dicoogle, the tests required a Dicoogle instance deployed in a specific folder, some tests also required specific plugins to be in an auxiliary folder for deployment in the Dicoogle instance during the test setup stage. This stage provides a fail-fast paradigm to the framework. Often, our test cases could not be run due to errors in the deployment setting that had anything to do with the performance of the tested services. Creating the correct deployment setting is normally a very tricky and error-prone process. Therefore, this fail-fast paradigm was very useful. The assertions can be defined globally or on a per-test basis.

The Setup stage is intended to prepare the current deployment for running the actual test cases. Therefore, this phase is executed before each test. Each test case may implement its own setup procedure. In the above example, our tests used the setup stage to deploy the necessary plugins to the Dicoogle instance and to start it. Conversely, the proposed framework also implements a Teardown stage, which will be described briefly.

After the Setup stage, the actual test procedure is run. This procedure must be implemented by every test case. The proposed framework offers several convenience callbacks that ease the test cases development and the collection of metrics. For instance, the *set/getTimer* that allows recording the running time of a specific part of the test, and the *log* callback conveniently records the desired metrics in a Comma separated values (CSV) file for posterior analysis. A test case might use multiple timers and record metrics in multiple log files according to the user convenience.

After the test is finished, the framework enters in the Teardown stage. In this phase, all the processes related to the test case are terminated, and any temporary or specific changes to the deployment scenario are reverted. This process is necessary for running multiple test cases, and to make sure that the system deployment is always in a consistent state.

When all the test cases are finished, the framework enters the Completion stage. This phase can be used to notify the user that tests have finished or to start the analysis of the output metrics in the log files. The analysis of the output results is not included in the framework, although we consistently used the Python data analysis stack to perform it. It involves using the Pandas[5] Library and its companions to parse, compute statistics, and output the log file data in a human-readable format, like spreadsheets.

4.3.2 Datasets

As mentioned in section 3.4.4, one of the major flaws in previous research about the performance of PACS content discovery and retrieval services is the absence of datasets

[5] https://pandas.pydata.org

Table 4.4: DICOM datasets.

Name	Nº Images	Nº Patients	Nº Studies	Nº Modalities
A	863	7	4	4
B	147 859	162	169	8
C	7 524 561	63 790	153 645	11
D	14 795 248	134 918	339 482	12
E	21 595 695	225 601	467 199	11
F	35 476 975	279 392	1 268 336	20

representative of the volume and diversity of data produced by real-world medical institutions.

One of the most time-consuming tasks of this doctoral program was related to the collection of datasets. In this regard, it was necessary to establish partnerships with many medical imaging researchers and institutions which granted us access to the datasets referred throughout this document. Most of these datasets were collected in real-world institutions using the Dicoogle PACS. It was also necessary to develop multiple applications for helping us in this task, including anonymization tools. The most important is a plugin for Dicoogle which allows reconstructing the DICOM files from the indexed metadata. Moreover, we also proposed a method for extending datasets [115], allowing us to generate artificial datasets with the same statistical characteristics as a model dataset, for instance, one of the datasets collected at an institution.

The datasets used throughout this doctoral program are presented in Table 4.4. The smallest datasets (A and B) are not representative of any particular institution. They are compilations of studies from common modalities that are useful for supporting the development lifecycle. The DICOM objects in these datasets were acquired in publicly available servers, such as DICOM library[6]. From dataset C onwards, all the datasets were collected in Portuguese institutions. Dataset C represents a single year of one of these institutions; dataset D represents roughly 2 years, and E represents roughly 3 years. The largest dataset, F, contains over 35 million DICOM images. These datasets are much larger than the ones used by previous studies referenced in section 3.4.4.

4.3.3 Test cases

The proposed framework was extensively used to validate the new developments in the Dicoogle project. In the particular case of the content discovery services, two test case were defined to access the performance and scalability of the Dicoogle's index and query processes.

Regarding the index service, it is important to understand how much time is required to index a DICOM file, how it varies with the increasing volume of previously indexed data and, lastly, how large is the resulting index in terms of data volume.

[6] https://www.dicomlibrary.com/

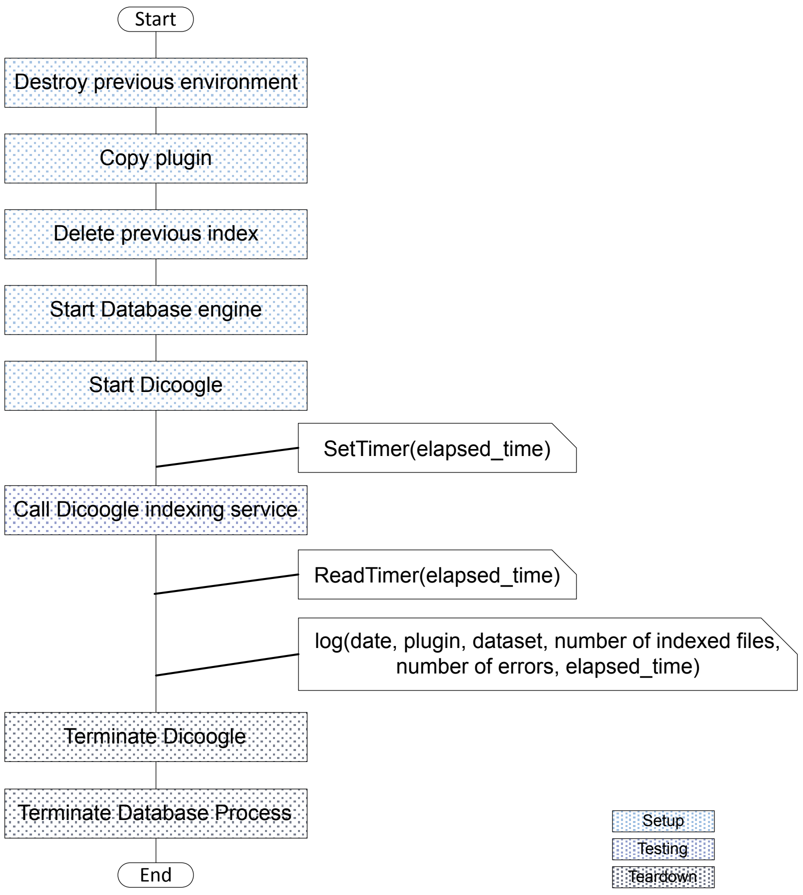

Start
Destroy previous environment
Copy plugin
Delete previous index
Start Database engine
Start Dicoogle
SetTimer(elapsed_time)
Call Dicoogle indexing service
ReadTimer(elapsed_time)
log(date, plugin, dataset, number of indexed files, number of errors, elapsed_time)
Terminate Dicoogle
Terminate Database Process
End
Setup
Testing
Teardown

Figure 4.13: Illustration of the index test case steps.

Figure 4.13 provides an overview of the indexer test case. In the Setup stage, the test starts by destroying the previous deployment settings, including eventual indexes and deployed plugins. Then, the specific plugin targeted by the test is deployed into Dicoogle environment and the database engine process is started, finishing this way the Setup stage. The main test consists in calling the Dicoogle index service for a specific dataset location and recording the indexing time and the storage space of the resulting database. The Dicoogle and the database engine processes are terminated in the Teardown stage.

Normally, an instance of this definition was created for each plugin (technology) evaluated. On a lower level, several test cases were created for distinct dataset sizes. The objective is to access how the required time for index and query procedures are influenced by the already existing volume of data. Comparing the experimental results for the different datasets, it is possible to infer how the volume of data influences these services and their performance over time.

The query test case is pictured in Figure 4.14. The query test definitions gathered data about how distinct implementations of the query plugins cope with different types of DICOM objects metadata, and how the volume and selectivity of the indexed datasets influences the

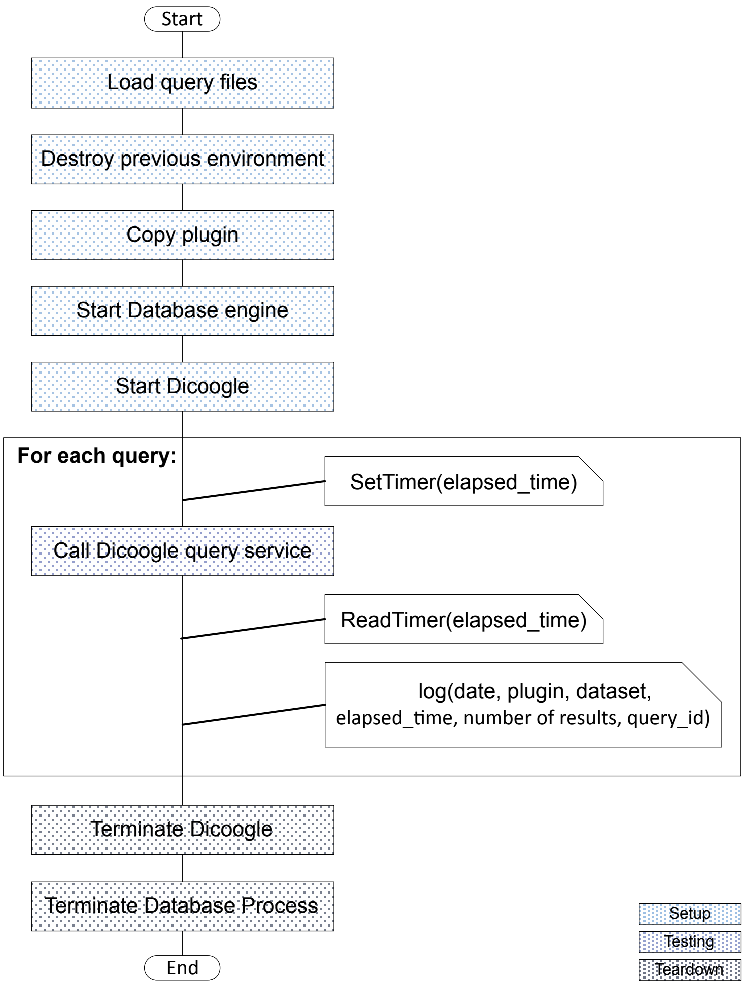


Figure 4.14: Illustration of the query test case steps.

required time for query and retrieval.

The Setup stage is analogous to the index test cases with the operational difference that the database's contents are not erased. In the Setup stage, configurations files describing the queries to be performed are loaded. These files contain a mapping of queries per each dataset and the number of expected results per query. The queries were created to filter the dataset using the following DICOM attributes; the SOPInstanceUID, StudyInstanceUID, SeriesInstanceUID, Modality, and PatientName. The SOPInstanceUID is expected to match a single image. While the StudyInstanceUID and the SeriesInstanceUID are expected to match multiple images. Therefore, different queries would result in different selectivity levels, and thus more information can be extracted from the test logs. The Modality was selected for representing very simple queries, which yield a high number of results. Querying for the different modalities in the dataset also produced the diversity in the different queries selectivity observed above. Lastly, the PatientName was selected to be representative of more complex queries because it is usually a large string field. This large range of queries per test case allows more information to be extracted from the test logs. Resulting in a better understanding of how the technologies are performing at the expense of more time required to complete the test cases.

These test cases have been used by many contributors of the Dicoogle project. Many plugin implementations have been subjected to validation based on technologies, such as Lucene, ElasticSearch, MongoDB, CouchDB, and PostgreSQL among others.

In the scope of this thesis, this framework and the collected datasets made possible the validation of the advanced research scenarios presented in the next chapter. More information about the test cases used is provided in the respective section.

4.4 Summary

This chapter purposes several methods for optimizing the performance of content distribution and discovery in PACS environments. The Dicoogle framework was the base for the research supporting this thesis. Most of the proposed methods were developed and validated in the context of this PACS. Moreover, the contributions were integrated into this open-source project, enabling the fast prototyping and development of real-world PACS applications.

After describing the contributions to the Dicoogle PACS, a smart routing mechanism for efficient support of distributed medical imaging environments is proposed (section 4.2.2). It enables better support to teleradiology and multi-institutional collaboration, including the ubiquitous deployment of PACS components. The solution includes a novel caching system (section 4.2.1) and a workflow management module that optimizes data retrieval according to institutional profiles (section 4.2.3). Moreover, the controlled channels method was proposed to improve the routers point-to-point communications performance (section 4.2.4).

Regarding content discovery, it is proposed a framework for validating the performance and scalability of technologies associated with these services (section 4.3.1). It provides automatic deployment of test instances, assists the definition of test cases and in the collection of relevant metrics. Moreover, it addresses the problem of having datasets representative of real institutional environments. A heterogeneous collection of datasets are provided, including a methodology for extending user samples into larger datasets (section 4.3.2).

These methodologies fulfil the objectives presented in section 1.3. They focused on the optimization of the PACS storage, content discovery and retrieval services. The integration of the Dicoogle PACS into the smart routing mechanism provides a scalable distributed PACS architecture that is ready to support multi-institutional environments and other advanced usage scenarios mentioned in this document. Moreover, these contributions supported the research and development of the use cases presented in the next chapter.

Chapter 5

Advanced Usage Scenarios

This chapter presents two advanced usage scenarios which were used as a proof of concept and validation for the proposed methodologies. They are novel real-world scenarios that would be extremely difficult to support using state of the art PACS solutions without additional optimization processes.

In the field of digital pathology, this thesis purposes an architecture of a Web Pathology PACS fully compliant with the DICOM standard, including a Web-based viewer. This scenario severely strains the storage and content retrieval services because the PACS components must deal with images with extremely high resolution.

Concerning the usage of PACS for research purposes, a framework based on Data Analitics (DA), BI, and Web technologies is proposed. It provides the necessary environment for conducting research on top of live institutional repositories.

5.1 Digital Pathology

Pathology focuses mainly on the identification of structural anomalies, through the naked eye or a microscope, and on the detection of possible relationships with functional disorders of tissues, therefore, identifying diseases. The aim of pathology has remained unchanged over time, focused on the analysis and comparison of tissue specimens on specific glass slides. For this, the use of optical microscopes has been fundamental since it was the only available instrumentation for centuries [116]. Despite using very methodical analysis workflows, it is possible for the same professional to draw different conclusions about the same specimen at different times. Moreover, asking for second opinions is common practice, and specific cases could be part of conferences or external quality assurance programs [117]. Consequently, there is the requirement for glass slides storage and delivery infrastructures. However, the specimen storage process is expensive, requiring accessibility, cleaning, and protection, which entails greater care by specialized staff. In contrast, digital storage and distribution cuts these costs and increases the throughput of pathology laboratories.

In this context, the branch of digital pathology and whole-slide imaging arises. These recent concepts refer to the digital capture of an image from a classic glass slide as well as to the field of information systems for managing the associated data. The arrival of Digital Slide Scanners (DSS) has introduced the concept of WSI. As the name suggests, these images capture the slide as a whole, rather than specific artifacts found by pathologists. As such, they can be captured unattended and screened later. These large images have characteristics, forms of handling and operation similar to images produced by optical microscopes [42]. It is worth mentioning that a typical slide scanned at x40 (approx. 1,600 megapixels) produces a file with several gigabytes [118, 119] and requires a viewer application with special functionality to fulfill the pathologist's needs [120].

In the past decade, digital pathology and WSI have been gaining momentum with the proliferation of digital scanners from different manufacturers. The literature reports significant advantages associated with the adoption of digital images in pathology, namely, improvements in diagnostic accuracy and better support for telepathology. Moreover, it also offers new clinical and research applications. However, numerous barriers have been slowing the adoption of WSI, among which the most important are performance issues associated with storage and distribution of huge volumes of data, and lack of interoperability with other HIS, most notably PACS based on the DICOM standard.

Digital pathology and WSI have been gaining momentum with the proliferation of digital scanners from different manufacturers [119, 121]. The literature reports significant advantages associated with the adoption of digital images in pathology [122–125], namely, improvements in diagnostic accuracy, promotion of distributed work processes (e.g. telepathology), integration of images with HIS and economic efficiency gains. Moreover, WSI offers new clinical and research applications to the pathology community [119, 126]. However, numerous barriers have been slowing this process, including performance issues, workflow efficiency, infrastructure, and integration with other software [122, 123].

Despite being a medical imaging modality, pathology studies have been kept away from standard PACS, in proprietary formats and information systems [119, 121]. In part because the DICOM standard did not address all the necessary requirements, but also due to the technical challenges raised by whole-slide imaging. As a result, major DDS vendors preferred to use proprietary archives and image formats (e.g. NDPI, SVS, and TIFF) [41, 119]. However, in the last few years, whole-slide imaging has been incorporated in the DICOM standard, and vendors are slowly starting to support DICOM in their scanners. This opens the door to the use of vendor-neutral DICOM-compliant archives and services for storage processes and for exchanging data with viewer applications [120]. Moreover, such a system would support third-party applications to access the WSI images directly. Nonetheless, WSI in general purpose PACS is still a mirage, because traditional solutions are not ready to handle the remarkable image resolution and volumes of data generated from each study, requiring new architectures for storage, efficient distribution, and visualization across heterogeneous systems [42].

This section presents an architecture of a Web Pathology PACS fully compliant with DICOM standard communications and data formats. The solution includes a PACS archive responsible for storing WSI data in DICOM WSI format and offers a communication interface based on the most recent DICOM Web services. The second component is a zero-footprint viewer that runs in any Web-browser. It consumes data using the PACS archive standard Web services. Moreover, it features a tiling engine especially suited to deal with the WSI image pyramids. These components were designed with special focus on efficiency and usability. The performance of our system was assessed through a comparative analysis of the state-of-the-art solutions. The results demonstrate that it is possible to have a very competitive solution based on standard workflows.

5.1.1 Related Work

In the scope of this thesis, an exhaustive search was performed for PACS-DICOM solutions that support digital pathology workflows, not only in the scientific literature but also in commercial applications. It was searched in the Scopus using the following keywords: DICOM Pathology Viewer, Web Pathology Viewer, and DICOM WSI. The queries matched 144 articles, of which 33 were found relevant to the topic. A detailed examination of these documents allowed us to conclude that there is no solution, commercial or not, for supporting DICOM WSI in a general purpose PACS resorting only to DICOM compliant communications. Moreover, only a much-reduced number of solutions provided a centralized repository compliant with Cloud Computing requirements, i.e. a network archive and a zero footprint Web-viewer ready to run in any common Web-browser without third-party software requirements or setup processes. Another important aspect is that many of these solutions required very complex server-side setups in order to achieve good performance [41]. However, the most important conclusion is that, despite being specified in the DICOM Standard, WSI is not currently efficiently supported by vendors [40].

Repositories and Data Formats

Currently, there is a multitude of image formats used by WSI equipment vendors, most notably TIFF, JPEG, JPEG2000, SVS or ndpi [119, 127]. These formats represent the most commonly accepted way of storing large images using a pyramidal and tiled structure. In this picture, the TIFF format comes as a convenient container, since it supports multiple tiles and multiple compression algorithms. Moreover, it allows storing multiple images in a single file, which is useful for storing multiple pyramid levels for faster browsing. TIFF also enables semi-structured meta-data to be recorded within the image.

Another promising format for WSI is JPEG2000. In [120], the authors report that previously to the DICOM Standard, encoding images in JPEG2000 and serving them as JPIP was a workaround for supporting WSI in standard PACS. Such an approach was applied in a solution for browsing mammography images [33]. In [128], a solution with a JPIP viewer is used for supporting education in virtual microscopy. A study addressing the use of virtual microscopy in mobile devices also applied the JPEG2000 image format along with several optimization strategies [41].

A study assessing the performance of JPEG2000 and JPIP protocol for whole-slide imaging was presented in [129]. An optimal combination of JPEG2000 compression algorithm parameters was evaluated for fulfilling the WSI usage requirements. However, there is no discussion about the method that enabled them to draw such conclusions. An empirical process, rather than a quantifiable one, may compromise the validity of such conclusions. The authors also compared their solution performance against a Zoomify[1] server. The JPIP Server reportedly managed 33% more users than the Zoomify server.

In order to overcome the lack of a de facto standard format for WSI images and as an effort to manage the wide variety of proprietary protocols, the OpenSlide project [130] provides a C library capable of reading and manipulating slides from different vendor formats. Currently, it supports Aperio, Leica, and Hamamatsu among others but, unfortunately, DICOM is not supported. This library provides an abstraction to the applications developer, who may shift his focus from the multitude of vendor formats to the OpenSlide's unified API. However, a unified API reduces the extractable information to a subset of common information elements. While this does not hinder the extraction of pixel data itself, it affects the metadata related to the image, which in fact has been crucial to the success of the DICOM standard. OpenSlide also provides a Web-based viewer that consumes WSIs converted to Microsoft's Deep Zoom format. In contrast, our architecture resorts exclusively to the DICOM standard for data format and communications. We provide a full-featured Web viewer platform compliant with DICOM. As a result, third-party applications can access DICOM WSIs, including their metadata.

The adoption of DICOM by the major scanner vendors has been delayed due to various reasons, most notably patent ownership and concerns about the DICOM standard not being able of delivering sufficient performance for use in pathology laboratories. We hope that our work may contribute to foster the adoption of the standard by demystifying the supposedly negative impact of DICOM format and communications on delivering WSI services. Meanwhile, while the standard is not widely adopted by vendors, a compromise solution is to "DICOMize", i.e. convert proprietary formats into DICOM, for instance leveraging the OpenSlide library to read the proprietary formats. However, this approach has a major drawback when it comes to fulfilling the DICOM requirements in terms of metadata, namely because it cannot be completely extracted from proprietary formats by third-party applications

[1] http://www.zoomify.com/

or simply because these formats do not contain that information.

Content Distribution and Visualization

In [131], a distributed collaborative system for pathologists is described. The article refers to [43], stating that their system is able to export lower-resolution pixel data into DICOM files. The system is supported by a Zoomify Web viewer that uses its own file format rather than DICOM. Moreover, the database model described in the article does not include the necessary information to build DICOM compliant WSI. The PAIS project, described in [42], hosts a database for supporting research using WSI. The article describes a comprehensive data model that is considered suitable for supporting a Pathology PACS system. However, the system's interfaces are not DICOM, but custom-made Web services. Moreover, the images seem to be stored in different TIFF files, separated by regions. In their Web site, at least some images are in Asperio's SVS format.

A distribution platform based on JPEG2000 and JPIP is presented in [132]. It includes a JVS DICOM Workstation that interacts with the PACS archive using traditional DICOM Query/Retrieve services. The images are provided to the workstations by a JPIP Server deployed alongside the PACS archive. These workstations receive a JPIP reference and open an external JPIP viewer to display the WSI images. This article also presents the compressor application that converts the proprietary WSI format to DICOM-compatible JPEG2000 transfer syntax. However, this platform is only compatible with Windows OS and is no longer supported.

Dcm-Ar is a Web solution based on Adobe Flash [133]. It uses a proprietary protocol that implements progressive image transition, a process similar to JPIP. Because it does not use DICOM services for image distribution, a proxy server is required to support DICOM communications with the PACS archive. This viewer is claimed as being able to display large DICOM images. However, the images used in the validation study only ranged from 10 to 30 MB. The article does not refer to any specific modality. In [126], the authors developed a browser-based WSI viewer using Microsoft Silverlight. Similarly, there are many other visualization solutions reported in the literature built entirely using HTML and JavaScript (JS) technologies, such as [134], but they do not seem to support whole-slide microscopy.

5.1.2 Methods

The proposed architecture has two major components: a PACS archive and a Web viewer application (Figure 5.1). The PACS archive, based on Dicoogle, is the central component that provides services for storage, content discovery and retrieval of medical images. An efficient DICOM WSI pyramid generator and a DICOM Web services API were developed, capable of providing tiles at distinct zoom levels. A zero-footprint viewer application, with its own tiling engine, was developed exclusively based on HTML5 and JS technologies. This client is a multi-platform Web application that allows pathologists to search over the Pathology PACS archive, retrieve WSI studies and manipulate them in a general-purpose Web-browser (e.g. Chrome or Safari). Its tiling engine uses the streaming paradigm to avoid downloading whole images and makes use exclusively of the DICOM Standard for communications with the PACS archive, namely through its DICOM Web services interface.

The viewer workstation is a fundamental component of the proposed architecture. To be useful in real-world environments, a set of common tools required by pathologists to review the slide images is provided, such as annotations and filters, for example. Our viewer uses the pan & zoom navigation paradigm described in the background section. In order to be efficient, it makes use of the image pyramids available in the PACS archive and only downloads and maintains in memory portions of the WSI required to display the current viewport region of interest. To do this, it must retrieve some information about the image, namely, its dimension, resolution, z-planes, and the image pyramid configuration. Additionally, for each sub-resolution level, it needs information about its dimension, resolution and tile configuration.

Image Acquisition

The journey to support WSI in general purpose PACS begins with the image acquisition itself. Digital scanners are able to capture digital images directly from glass slides. After the acquisition, images must be moved to the PACS archive for long-term storage. This procedure is analogous to other medical imaging modalities. Our architecture contemplates two services to accomplish this task. The images can be uploaded to the PACS archive via the traditional DICOM C-Store service or by its Web service counterpart, the STOW-RS. As digital scanner manufacturers gradually start to support DICOM in their equipment, exporting the DICOM WSI directly from the acquisition equipment to the archive should be straightforward. In fact, we have already sent images to our Pathology PACS directly from a whole-slide scanner of a major vendor. This stage is illustrated in Figure 5.2, step 1. This process is entirely compliant with the distributed PACS architecture presented in section 4.2. This architecture could be used to improve the communications between the WSI scanners and a remote PACS archive.

Nonetheless, this process can raise two major issues related to image compression and to the configuration of the image pyramid. These factors have a considerable impact both on the PACS long-term performance, and on the image visualization procedure itself. However, they

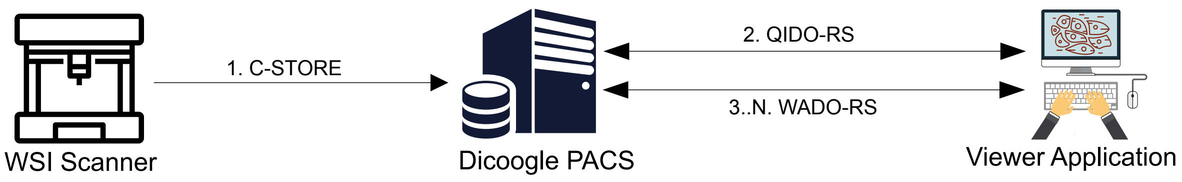


Figure 5.1: Overview of the proposed architecture.

are subject to the acquisition equipment capabilities and configurations. Regarding image compression, the algorithm used by the acquisition equipment may not be suited to fulfill the PACS requirements, thus creating larger images that would ruin the PACS long-term storage capacity. Conversely, the acquisition equipment could apply an irreversible compression algorithm that produces images unfit for medical diagnosis. As a result, adequately configuring the acquisition equipment is a major requirement for an efficient PACS. Our archive can be configured to recompress the incoming images with a fitter algorithm according to the PACS administrator's preference. This stage is illustrated in Figure 5.2, step 2.

Regarding the image pyramid configuration, this thesis already discussed its major importance in the screening procedure. The acquisition equipment could be configured to produce only the larger resolution image or, alternatively, to create the image pyramid and send several sub-resolution images to the archive. Since the image pyramid configuration influences the performance of the visualization procedure, we argue that it should be generated in the PACS to achieve an optimal compromise between storage space and efficient access. In fact, our PACS archive is able to generate the sub-resolution images solely based on the largest resolution image. When new images arrive at our PACS, the procedure for generating the image pyramid is scheduled. Depending on the image dimensions and compression used, it could be an intensive process, therefore it is performed as a background process to avoid overloading. This stage is illustrated in Figure 5.2, step 3.

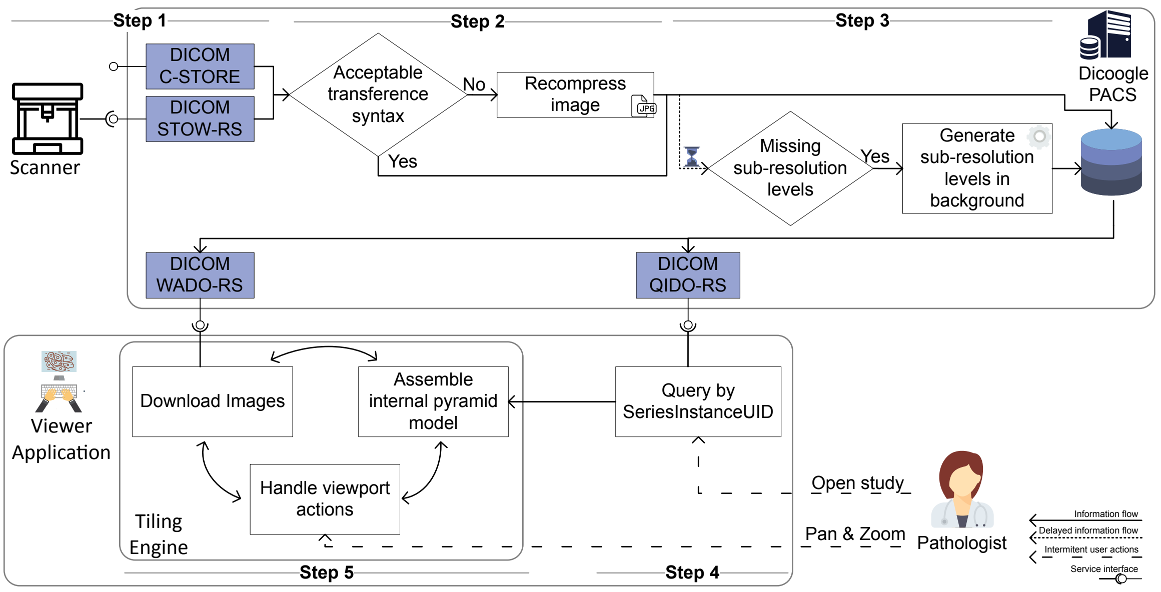


Figure 5.2: Information flows in the proposed Pathology PACS architecture.

Content Discovery and Visualization

The next step in the workflow requires finding a certain image, or specimen, in the archive. In order to discover DICOM WSI in the Pathology PACS, the traditional DICOM Query and Retrieve services (i.e. C-FIND and C-MOVE), as well as the more recent DICOM QIDO-RS service [29], can be used. These services allow searching in the DICOM images metadata. As such, information about the available images can be retrieved, as well as objects of interest described within it, such as Patients or Specimens. Orchestration of these services enables the development of advanced query interfaces that could be implemented by multiple vendors. We have tested our architecture using Dicoogle's own search interface, as well as BMD's software PACS Center, which is an enterprise solution.

Our viewer makes use of existing image pyramids in the PACS archive because this method is more efficient than generating the image pyramids on demand. In order to do this, it has to retrieve some information about every resolution level of the image, namely, its dimension, resolution, z-planes, and last but not the least, the image pyramid configuration. This information is retrieved from the PACS archive using the DICOM QIDO-RS, i.e. the standard DICOM Web service for querying. The different sub-resolution images can be found by submitting a query for "SeriesInstanceUID", for which the archive returns a list of instances that belong to the specified series. In order to organize the image pyramid in terms of resolution, the viewer firstly looks for the instance with the attribute "Image Type (0008,0008)" equal to "ORIGINAL/PRIMARY/VOLUME". This instance has the maximum resolution. In contrast, instances of this attribute ending in "../VOLUME/DERIVED" are sub-resolution images. For each of these images, the viewer computes their magnification rate based on the image size (cols x rows). As a result, the visualization platform builds an internal representation of the image pyramid, as illustrated in Figure 5.2, step 4.

At this point, the viewer is ready to download the required image tiles and start displaying them. Initially, it identifies the resolution level in the image pyramid's that is closest to the viewport window resolution. This minimizes the amount of pixel data exchanged between the PACS archive and the remote viewer, as well as the complexity of the resizing operation required to fit the pixel data to the viewport window. The image pixel data is retrieved from the archive using the DICOM WADO-RS service. The viewer specifies the image SOP, Study, and Series UIDs, as well as the list of frames to be retrieved. Remember that the image tiles are stored in individual frames, as described in section 2.9. After receiving the frames, the viewer copies their pixel data into the window viewport, thus displaying the image. If the retrieved frames do not fit the viewport requirements exactly, the viewer also performs a resizing and cropping operation on the tiles' pixel data. This is completely performed client-side, to avoid loading the PACS archive with additional image operations which could compromise the platform's performance and scalability.

Panning and zooming operations are resolved analogously by the viewer. In each event the user performs on the viewport, the tiling engine recalculates the closest resolution level and the underlying image tiles. These tiles are requested via DICOM WADO-RS and displayed on the window's viewport. As expressed, the application supports several filters to the original image that are performed client-side and therefore have no additional impact on the solution's server-side performance.

5.1.3 Validation

This section provides several controlled laboratory trials aiming to validate the proposed architecture. The goal is to show that it is possible to support digital pathology efficiently in a general purpose PACS with fully DICOM compliant communications, and without compromising the system's overall Quality of Service. To do this, several trials were made targeting the challenges faced by our architecture, namely the storage space required and the content retrieval performance. When possible, the trials included a comparison of our architecture with the current state of the art alternatives. We also deployed an online demonstration with publicly available studies[2].

Storage

Regarding the required storage space to hold WSI studies, our interest lays in knowing how the DICOM format influences this variable, especially when compared to other formats. We used a dataset composed of 100 WSI studies collected by a major vendor scanner. The images were stored in DICOM WSI format, encoded with JPEG lossy compression (factor 85). Sub-resolution images were also supplied by the scanner. The maximum resolution images of this dataset have a combined size of 13.9 GB. Together with the sub-resolution images, the dataset amounts to 19.4 GB. Regarding the image dimensions, the largest images have 26 GPixels, while the lowest image resolution is 16 MPixels.

For validation purposes, the studies were replicated in other formats, namely Pyramidal TIFF, JPEG2000 Lossy (factor 85) and JPEG2000 Lossless. The JPEG2000 images were generated using JVSDicom Compressor with parameters recommended in [129]. These parameters are supposed to enable better visualization of the WSIs. Table 5.1 shows a comparison of the required storage space for the multiple image formats. For simplicity reasons, only the values of the five largest images are shown, together with the mean and standard deviation values for the whole dataset.

By analysing table 5.1, it is possible to conclude that TIFF pyramids are on average 94% the size of the same image stored in DICOM. This was expected due to the amount of important metadata stored alongside the DICOM images, which is not shipped in the TIFF files. Another conclusion drawn from this table is that JPEG2000 reversible compression tends to generate very large files when compared with irreversible JPEG compression (factor=85) of

[2] demo.dicoogle.com/pathology

Table 5.1: Storage size of different WSI formats.

Sample	Max	Pyramid	Image size (GB)						
N°	resolution	Levels	DICOM	TIFF	vs DICOM	JP2	vs DICOM	JP2 Lossy	vs DICOM
21	14.8	10	2.874	2.648	0.92	10.907	3.80	1.550	0.54
99	25.8	10	2.663	2.520	0.95	12.400	4.66	2.720	1.02
61	4.5	9	1.276	1.205	0.94	4.277	3.35	0.480	0.38
23	5.1	9	1.152	1.068	0.93	4.224	3.67	0.542	0.47
64	5.4	9	1.075	0.984	0.92	3.661	3.41	0.566	0.53
Average					0.94 ± 0.02		3.29 ± 0.41		0.43 ± 0.26

the DICOM pyramids. These images are on average three times the size of the reference for this study. Finally, comparing the DICOM image pyramids with the best JPEG2000 irreversible compression parameters, we found that on average the images produced are 43% the size of our reference. We think that such a figure is relevant in the context of an institutional PACS since it represents meaningful storage savings. However, as will become clear in this document, this saving in storage space will come at the expense of performance in the visualization process, which is a negative effect.

Visualisation

Performance in image visualization is of paramount importance for any digital pathology laboratory since it influences directly the Quality of Service perceived by pathologists. Therefore, it is crucial to validate the proposed architecture on this point, as well as comparing its performance with the previous alternatives described in this document. In order to assess our architecture's content discovery and retrieval capabilities, a trial was developed to evaluate how quickly images could be presented to the user. This consists of executing a predefined workflow on the viewer application. A workflow with seven tasks was designed for each image. The tasks, shown below, reflect common operations performed by users and require additional data to be retrieved from the image server.

1. Open the image
2. Zoom to 10x in the current position (centre)
3. Pan to a point 'a'
4. Zoom to 20x centred on point 'a'
5. Pan to point 'b'
6. Zoom to 40x centred on point 'b'
7. Pan to point 'c'

The time required for the application to fulfill each task was measured. This process involved displaying the image regions in their highest quality. Two viewers were submitted to this trial, the solution proposed here and JVSView, which was selected because it was presented in the literature as the best solution using JPEG2000 and JPIP, namely in [129]. Consequently, it allowed establishing a comparison between the performance of a fully DICOM architecture, and its JPEG2000 & JPIP counterpart. Both applications and servers were deployed on the same computer (Intel Core i7-3770 CPU, 16GB DDR3 RAM, and a 7200rpm hard-drive), and the same monitor and resolution were used by both applications. The framework presented in section 4.3 was used to automatically perform the task in our solution.

The tables 5.2 and 5.3 show the output of this trial as well as the speed-up provided by our architecture. The results demonstrate that the proposed solution has significant performance gains. For anyone using both applications, this perception is immediate due to the usability discrepancy between them. The proposed architecture managed to complete the workflow on average 18x faster than its counterpart. A key factor is that it managed to complete every task in less than 2 seconds, which we think is a manageable waiting period for a reviewing pathologist.

In the previous experiments, the time required for retrieving each tile from the proposed Pathology PACS archive was also recorded. On average, each tile was loaded in 20 ms, with a

Table 5.2: Whole-slide imaging viewer performance trials. Absolute time required to complete the workflow (in seconds).

Task	21		99		61		23		64	
Nº	JVSView JPIP	Proposal DICOM	JVSView JPIP	Proposal DICOM	JVSView JPIP	Proposal DICOM	JVSView JPIP	Proposal DICOM	JVSView JPIP	Proposal DICOM
2	22.94	1.14	23.12	1.12	27.24	0.92	27.43	1.07	31.85	1.27
3	30.68	1.36	18.55	1.59	26.27	0.97	29.53	1.06	31.05	1.16
4	15.83	0.47	10.14	0.82	12.83	0.47	13.81	0.40	19.09	0.42
5	29.07	1.46	16.70	2.69	29.95	1.03	23.96	1.15	35.59	1.01
6	12.08	0.04	10.29	0.03	13.82	0.80	10.00	0.03	17.35	0.43
7	19.90	1.09	22.14	1.87	26.69	0.91	19.22	0.91	29.61	1.13
All	133.41	7.44	104.37	10.03	140.87	6.76	128.42	6.18	168.08	7.02

Table 5.3: Speed-up achieved by the proposed Pathology PACS architecture.

Task Nº	21	99	61	23	64	Average
2	20.09	20.73	29.71	25.68	25.02	24.25
3	22.58	11.70	26.97	27.88	26.69	23.16
4	33.69	12.43	27.14	34.38	45.30	30.59
5	19.86	6.21	29.13	20.83	35.09	22.22
6	283.78	338.90	17.18	317.09	39.99	199.39
7	18.19	11.86	29.25	21.04	26.11	21.29
All	17.93	10.41	20.84	20.77	23.96	18.78

standard deviation of 15 ms. We estimate that the proposed visualization solution is capable of loading around 50 tiles per second, which corresponds to an area of 3620 square pixels. This area is larger than the high-resolution 4K displays which are expected to improve the WSI screening efficiency [135]. Those are interesting values that validate the content retrieval capabilities of a solution based on standard DICOM services. Moreover, the improvements in the visualization process and image distribution completely compensate for the greater storage space requirements.

The development of the contributions presented in this section was based on the Dicoogle PACS framework presented in section 4.1. This development was also supported by the performance and scalability validation framework presented in section 4.3. Although the content distribution methods are not specifically targeted in this section, they would enable the deployment of this architecture in a distributed, multi-institutional environment.

5.2 Research PACS

Nowadays, medical imaging repositories contain a wide-range of valuable metadata that describes thoroughly all the stakeholders involved in medical imaging practice. Despite being mostly used for supporting medical diagnosis and treatment, many recent initiatives claim the utility of medical imaging studies in clinical research scenarios and to improve the medical institutional business practices.

The current paradigm of medical imaging repositories fits well in the definition of Big Data [136]. The continuous production of huge volumes of data, its heterogeneous nature and the increasing number of performed examinations makes the analysis of medical imaging repositories very difficult for conventional tools. Moreover, initiatives, like the distributed PACS architecture presented in section 4.2, will contribute to the integration of multiple institutions in the same PACS, thus creating larger and more useful datasets. As a result, DA and BI techniques applied to this scenario have potential to increase the efficiency and quality of the medical practice.

This section proposes a DA and BI framework for medical imaging repositories that aims to provide the necessary environment for conducting research on top of live institutional repositories. It leverages all the metadata stored in a medical imaging repository without requiring predefined data models or imposing rigid data flows. Specifically, the developed system takes advantage of Dicoogle's Data Mining features [137] for extracting data from existing PACS. It provides a series of exploratory techniques and visualization tools for a deep understanding of the working dataset and extraction of valuable information. Moreover, its design facilitates the usage of analytics tools, since the users do not need the programming skills commonly required for data analysts and scientists , such as Python and R. It provides an intuitive Web-based interface that empowers the usage of novel DM techniques, namely a variety of Data Cleansing tools, filters, and clustering functions. Moreover, it features an extensible dashboard with customizable charts and reports.

5.2.1 Background and Related Work

The analysis of the PACS archives contents has been demonstrated to withdraw positive outcomes for many research endeavours, namely radiation dosage surveillance [111], performance analysis of institutional business practices processes [138, 139], cost-effectiveness of diagnostic procedures [140], among many others [3, 20, 140, 141].

Nevertheless, the complexity of medical imaging data has increased tremendously [142] since the volume and heterogeneity of data generated by the medical practice also increase considerably. As a result, researchers must rely on IT tools to be able to perform their analytical tasks. However, traditional PACS do not allow exploring the imaging metadata for extraction of relevant knowledge. This leads to the use of third-party applications to perform data analysis, including proprietary solutions that only work with specific PACS.

BI consists of a pipeline that integrates a series of tasks. When combined, these are responsible for acquiring, transforming and translating raw data into useful information for improving the business practice [143]. This process encapsulates a multitude of capabilities such as reporting, dashboards, Data Mining, among many others [144].

Nowadays, the data generated by medical imaging laboratories DICOM is highly heterogeneous and inconsistent. Although all equipment implements the same DICOM standard, they might use different configurations. This generates irregular data, which occurs, for instance, when two different equipment report the same value but using different metrics [102]. Furthermore, the interaction between technical staff and equipment is another inconsistency factor. For these reasons, applying analytical procedures on inconsistent data may generate unreliable results. So, one of the most important and crucial steps in the whole BI process is the Data Cleansing (DC) stage. It ensures the reliability of the data in a given repository, by detecting and correcting inaccurate records [145]. DC is a complex process, which can be further divided into a series of iterative operations. First, the Data Auditing step, where the working dataset is analysed to determine which anomalies it contains. Next, the Workflow Specification step defines a set of operations required for fixing the previously identified anomalies (or, in some critical cases, exclude them). After this, the operations are executed in the Workflow Execution step. Finally, the applied operations are validated in the Post-Processing and Controlling step. Each of these steps can be performed manually, supervised or unsupervised according to the level of user input system required.

There are some references in the literature relative to the usage of DA and BI systems applied to medical imaging repositories. Nagy et al. [141] developed a tool that implements a fully-fledged BI stack. It starts by aggregating data from the institution's systems, namely DICOM metadata from the PACS and HL7 data from the RIS, among others. Data is extracted on a periodical basis and stored in a MySQL database that feeds a dashboard. This graphical tool includes the most relevant chart types, such as histograms, bubble charts, and others. Whenever a database's entry is added or updated, the corresponding chart is automatically rendered to reflect these changes, as well as the detailed currently selected reports.

Kallman et al. [146] developed a framework that provides a similar set of functionalities. However, there are some important differences. For instance, the second framework separates the DICOM metadata header from the image's pixel data, storing the first in a separate repository. Moreover, the user interface is based on the usage of SQL queries. Although it is a more powerful and flexible language, it makes it harder to perform data analysis for less experienced users, such as radiology researchers.

The two previous frameworks make use of separate repositories, i.e. they do not work directly over institutional PACS archives. The advantage is that they can be used for statistical purposes with no risk of interfering with the regular clinical workflow, however, they will always work with an outdated snapshot of the PACS archive's content. In our point of view, the major limitation is that they are bound to a strictly relational database model. This means that researchers cannot derive knowledge from metadata fields that were not previously encompassed in the database schema, not satisfying this way the requirements of most research endeavours

Wang et al. [111] developed a database solution for controlling patient's radiation exposure by analysing the DICOM images metadata. It includes reporting, alerts capabilities, and a Web interface for interacting with the system. The most relevant module, when compared with the previous tools, is the Knowledge Base that is capable of unifying distinct vendor data by enforcing the measured attributes to use the same units, which were defined statically. The major limitation of this solution is to be strictly related to the radiation dose thematic. It is focused on the detection of irregular radiation dosages through the use of several filters and, it can analyse information at study, patient and institutional level.

In [102], the authors present a DICOM Data Warehouse for arbitrary Data Mining. It enables automated data analytics tasks on top of DICOM metadata databases. The authors claim that despite existing some previous efforts in the literature, namely [111] and [138], none shared the purpose of enabling completely arbitrary DM capabilities. The solution is backed by a SQL database, which data model needs to be extended to support new Data Elements. It also targets the discrepancies between data attribute's values from different vendors by creating static mappings for different attributes that represent the same measurements. This framework does not provide a graphical interface for creating reports, alerts or browsing the data. In an updated article [103], the authors point out the difficulty of indexing a very heterogeneous data source, such as DICOM, which resulted in leaving some DICOM attributes unindexed. They state that NoSQL approaches may be required to handle it.

5.2.2 Architecture

The proposed architecture follows a classic client-server model, segmented into three distinct layers: presentation, business, and persistence. In this architectural pattern, the client acts only as the presentation layer of the developed system, displaying the interface to the user and resorting to the server when required. In turn, the server is responsible for implementing both the business and persistence layer of the application. The business layer encompasses most of the application's logic, handling the client's requests and providing an adequate response. The persistence layer is responsible for storing and maintaining data across multiple sessions.

The server's BI functionalities are provided by the Python Scipy stack (which includes the NumPy, SciPy and pandas libraries) and the scikit-learn library. The methods developed using the previous libraries are exposed through a REST API, using Django and the Django REST framework toolkit[3]. The persistence layer is based on PostgreSQL RDBMS, which persists the application's logic related data, in conjunction with panda's Hierarchical Data Format Store (HDFS), responsible for bulk data storage.

The client is a Web application that follows the single page application pattern, in which, all the necessary client code (HyperText Markup Language (HTML), JS, and Cascading Style Sheets (CSS)) is loaded in a single page load. The application was developed using the *React* framework, with the help of the *Redux* state container. Furthermore, it makes use of the *Bootstrap* framework to manage most of the interface as well as the *plotly.js* library to handle charts rendering.

This framework leverages the Dicoogle's DM features for extracting data from the DICOM objects in the PACS. The communication is supported by the Dicoogle's content discovery and retrieval services, namely its Web service query end-points. In turn, these services rely on the technologies implemented by plugins, as described in section 4.1. In order to support research scenarios more efficiently, a distributed PACS architecture that is able to support multiple institutions because it enables the collection of larger and more representative datasets. In this regard, this analytics framework leverages the contributions presented in section 4.2.

In order to support larger queries, the Dicoogle's content discovery and retrieval services had to be optimized. These end-points relied too much on in-memory processing, a recurring problem found in data-intensive applications. When dealing with larger result sets, in-memory processing may fail because all the temporary results may not fit into memory, instead, the stream processing paradigm was used. Besides the Dicoogle query services, this thesis also focused on the performance of the Lucene plugin. In both cases, the validation framework and the test cases, described in section 4.3, were used to support the development.

Figure 5.3 presents the developed functional modules, as well as their interactions, that will be described in the next sections.

[3] www.djangoproject.com

Rule-based Data Cleansing

One of this application's main concern was to provide the operator with a tool that allowed manipulation of multiple irregular records simultaneously, because usually, a detected inaccuracy occurs multiple times in the dataset. Consequently, manually correcting every one of those records would be impractical.

In order to provide these capabilities, a Rule-based control system was devised. It is responsible for implementing the system's Data Cleansing features. It works in two phases: the Matching and Action. In the first phase, the different rules are applied in the dataset to detect inaccurate records. The action phase performs the corrections to the records.

A small set of basic rules and actions were developed to provide a more powerful service, keeping at the same time the ease of use. The rules cover most use cases defined in the context of this thesis but it can cover others by adding additional modules, written in *Python*, which are interpreted at runtime.

So far, the following five rules are available:

- Empty Field – Allows the detection of empty values on a set of fields. This issue is the most common anomaly when working with data. Unfortunately, most of these fields cannot be inferred, usually leading to their removal from the dataset.
- Expected Value – For detecting of one (or more) values on a given group of fields. It works by defining a set of values and the fields where those values are expected.
- Regular Expression – A more generic and advanced version of the Expected Value rule, requiring Regular Expression's knowledge.
- Filter Date – Selects a set of records on which the Date field matches the defined interval.

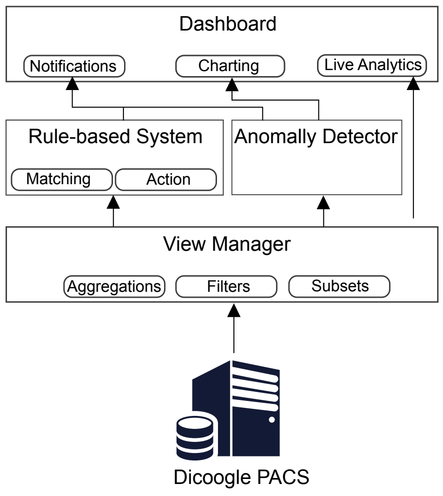


Figure 5.3: Business Intelligence framework architecture.

- Expression – Enables the execution of statements or even small scripts, allowing to detect more complex cases, impossible to perform with the previous rules. It should only be used as a fallback.

These rules can be combined with the following actions:

- Fill Empty Field – It replaces the empty values in the defined fields with a chosen value.
- Replace by Value – It replaces a defined set of values in a given range of fields.
- Replace by Regex – A specific case of the previous action when the value to be replaced is a Regular Expression.
- Date to Age – It converts a specified field (usually the PatientDateofBirth field) to the corresponding Age (most often the PatientAge field) according to the DICOM standard.
- Normalize Age – Convenience action to normalize the Age field, enforcing the measure to be in Years, Months, Weeks or Days.
- Expression – It uses a python script as an action.

It is important to note that the defined rules might not be mutually exclusive since a rule's triggered action might update the values that match another rule. For this reason, every rule has a priority attribute, an integer value. This allows the rules to be executed in-order, according to their priority.

Anomaly Detector

This component allows performing Anomally Detection (AD) by associating a given field to one of two categories: ordinal or nominal data, based on the field's datatype. As the names imply, if the selected field's datatype is numerical (either integer or float), then it is provided with an ordinal description; otherwise, it is provided with a nominal description.

An ordinal description returns a count attribute, with the total number of non-empty entries in the field, as well as some of the most common statistics for a numerical set of values, such as mean, standard deviation, minimum and maximum values, and percentiles (25%, 50%, and 75%). On the other hand, a nominal description returns the number of entries in which a given value appears. Also, both descriptions feature an empty values' count, if applied

Unfortunately, the automatic detection of a field's description based on its datatype is sometimes flawed. Often, a field is composed of an integer set of values. However, those values do not represent a continuous variable, meaning that they are supposed to be interpreted as a category. Therefore, it is possible to manually enforce a nominal description for fields composed entirely of integer values.

View Manager

The concept of Views was implemented to ease the process of working with large and heterogeneous datasets. Most use cases do not work with all repository data. Particularly, when the size and heterogeneity of data would affect significantly the performance of most operations, even the most basic ones. Moreover, very often the analysis targets a specific subset of the repository, such as analysing data related to a specific modality or analysing reports performed in a given date range. Taking this into consideration, it was developed a view concept that intends to represent a smaller and more specific dataset from the original PACS repository.

Therefore, Views are very important because they determine the execution context of the other components, i.e. rules and anomaly detectors will only be applied to their assigned Views, and consequently sub-datasets. It allows the user-created Views to inherit the transformations of their parent views, much like the concept of OO inheritance. User-created Views can inherit either from the default (root) View or from other user-created View. This provides flexibility and extensibility to the system. The Views support all previously referred data manipulations. For supporting this, a system was developed that allowed merging a view's transformations with its parent's, thus optimizing the usage of the Dicoogle's DM capabilities.

To create new views, three transformations are provided:

- Aggregation – It mimics the SQL `groupby` operation, defining the set of aggregation fields and functions to be performed. It is possible to provide only one function as a parameter. In this case, it will be applied to all the fields that are not part of the aggregation fields. However, this is rarely the desired output. For this reason, it is also possible to define one or more functions per column. This possibility is particularly useful taking into consideration the Information Entities Hierarchy (Patient, Study, Series, Image). The allowed functions are: `cumulative sum`, `cumulative product`, `maximum`, `minimum`, `median`, `standard deviation`, `mean`, `size` and `sum`.

- Subset – It enables the creation of a static subset, defining the interval of either rows and/or fields (i.e. selection and projection restrictions).

- Filter – It makes possible to use the above rules, as transformations.

The defined transformations can be applied using the Dicoogle Data Mining capabilities in two situations. Firstly, when an Expected Value filter is specified, it is translated exactly to the Dicoogle's query language. For instance, a filter by Modality and PatientSex, and expected values CR and F respectively, would be translated to *"Modality:CR AND PatientSex:F"* in the Dicoogle's query service.

Secondly, when the fields of a subset transformation are defined they are also mapped to the Dicoogle's query interface. For instance, the subset with fields `Modality` and `PatientAge`, are translated to the query's return fields `fields=[Modality,PatientAge]`.

Dashboard

The developed solution provides extensive dashboard capabilities. The dashboard enables the development of a fully customizable client page that may include any of the available visualisation components, every one inserted in a fully resizeable and sortable panel. Visualizations also depend on the defined views. So, the system not only stores the layout of the dashboard, including every panel's coordinates and size, but also the instructions necessary to populate the component with the desired data.

Given its integration with the underlying Dicoogle PACS, the proposed system provides real-time analytics capabilities. This means that the platform is immediately notified when new data arrives at the archive, avoiding the necessity of creating repository snapshots, acquired on a periodical basis. New images are automatically added to the necessary Views and are properly analysed by the previously defined rules. Afterwards, a notification is also sent to the Dashboard with updated information.

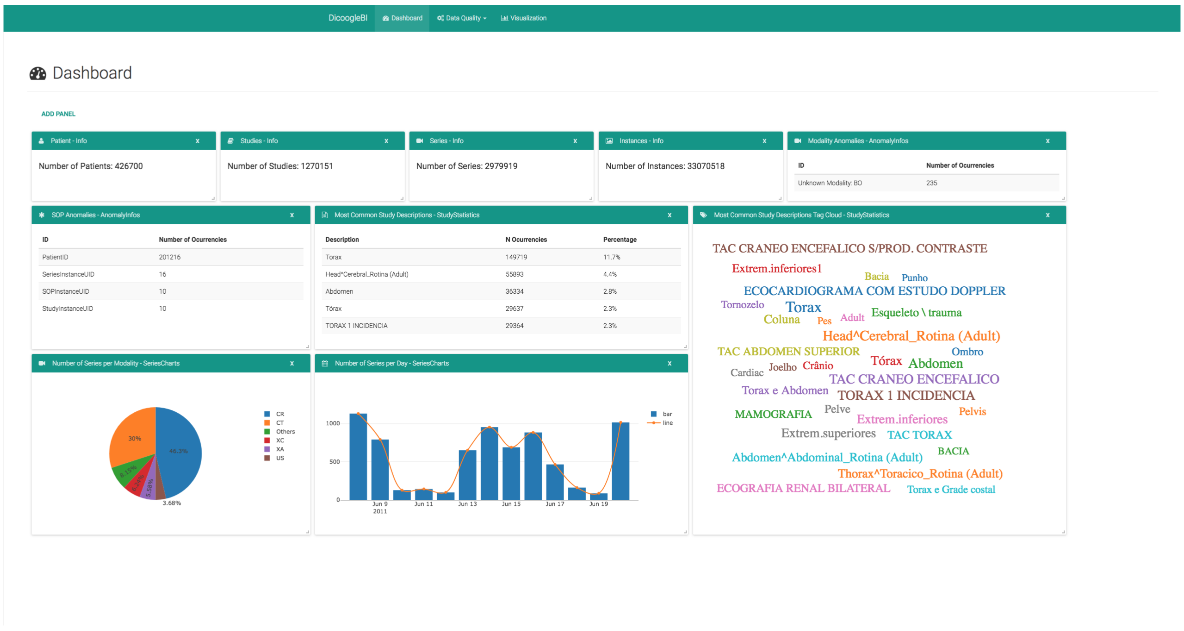
5.2.3 Results

This section shows the content discovery capabilities of the developed framework for extracting knowledge from medical imaging repositories. For demonstrative purposes, it was used the real-world *dataset F* collected at an affiliated institution, as described in section 4.3.

An overview of the Dicoogle BI dashboard is showed in figure 5.4, including several widgets. On the top, there are four informative widgets that count the number of patients (426 700), studies (12 070 151), series (2 979 919), and instances (33 070 518) in the dataset. Notice that there is an inconsistency between this summary and the summary provided in Table 4.4. This discrepancy in the number of images was caused by duplicated DICOM files in the repository. These files were copied without having their SOPInstanceUID updated, which violates the standard, and caused the previously overestimation of the number of images in the repository. It also had an impact on the estimation of the number of the other stakeholders. The discrepancies found at the patient level were also caused by duplicated identifiers for clearly different records. For instance, the same *PatientID* was given for images with different *(PatientName, PatientBirthDate, PatientSex)* tuples.

The dashboard includes also two widgets related to anomalies in the dataset. The first finds *Unknown Modalities*, i.e. modalities that are not defined in the standard. It identified 235 different series from a modality called *BO*. The second widget identifies duplicated identifiers in the dataset. In this case, over 200 000 duplicated *PatientID*s where identified[4]. Although many

[4] The same PatientID was given for records with different PatientName, PatientBirthDate, PatientSex tuples



Number of Patients: 426700

Number of Studies: 1270151

Number of Series: 2979919

Number of Instances: 33070518

ID	Number of Ocurrencies
Unknown Modality: BO	235

ID	Number of Ocurrencies
PatientID	201216
SeriesInstanceUID	16
SOPInstanceUID	10
StudyInstanceUID	10

Description	N Ocurrencies	Percentage
Torax	149719	11.7%
Head^Cerebral_Rotina (Adult)	55893	4.4%
Abdomen	36334	2.8%
Tórax	29637	2.3%
TORAX 1 INCIDENCIA	29364	2.3%

Figure 5.4: Overview of the Dicoogle BI dashboard.

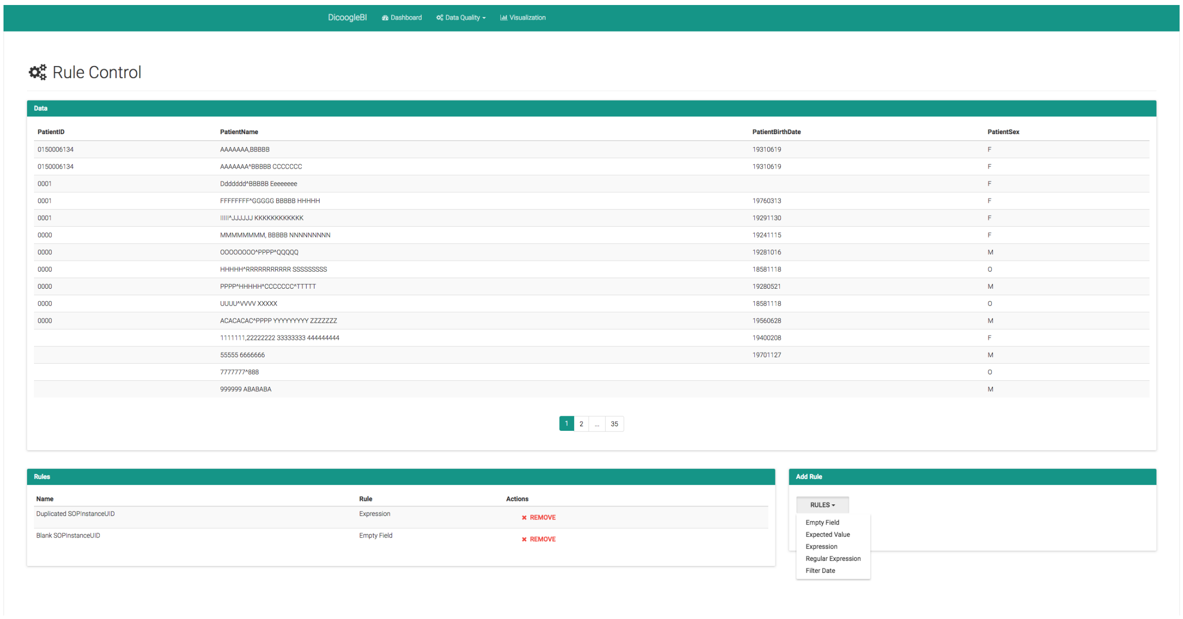
proprietary PACS can workaround these inconsistencies, they do not favour any analitical endeavour that relies on the quality of the collected data.

Other widgets with statistics were also presented in the Dashboard. For instance, there is a pie chart summarizing the contents of the dataset according to the series modality. There is also a bar chart with the production of series per day. Notice the reduced productivity on weekends ((11,12) & (18,19)) compared to regular weekdays. It is also perceivable the impact of the 10 Jun holiday on the hospital productivity level, and an unexplainable decrement on Friday compared to the other weekdays.

Lastly, a tag cloud widget with study descriptions was included. It is possible to observe that the *StudyDescription* attribute has a lack of normalization. The previous statistics required the aggregation of the dataset by *SeriesInstanceUID* and then by *Modality*, before the *count* function could be applied

Figure 5.5 shows the interface for configuring the rules of a given view. The sample data shows the duplicated *PatientID* records referred in the previous paragraphs. The records are obfuscated by privacy reasons, but a real name will be always replaced by the same obfuscated version. As a result, you can see clear different *PatientNames* with the same identifier. Very suspicious *PatientBirthDates* can also be seen, for example `18581118`.

For finalising, an overview of the application charting interface is shown in Figure 5.6. The attributes and chart type can be customized for a given view. The chart itself is interactive, being possible to resize the visualisation window and apply transformations to the axis. The presented graphical functionalities allow any user to perform DA and BI tasks on their PACS content even if they do not have programming skills.



Rule Control

Data

PatientID	PatientName	PatientBirthDate	PatientSex
0150006134	AAAAAAA,BBBBB	19310619	F
0150006134	AAAAAAA^BBBBB CCCCCCC	19310619	F
0001	Dddddd^BBBBB Eeeeeeee		F
0001	FFFFFFFF^GGGGG BBBBB HHHHH	19760313	F
0001	IIIII^JJJJJJ KKKKKKKKKKKK	19291130	F
0000	MMMMMMMM, BBBBB NNNNNNNNN	19241115	F
0000	OOOOOOOO^PPPP^QQQQQ	19281016	M
0000	HHHHH^RRRRRRRRRRR SSSSSSSSS	18581118	O
0000	PPPP^HHHHH^CCCCCCC^TTTTT	19280521	M
0000	UUUU^VVVV XXXXX	18581118	O
0000	ACACACAC^PPPP YYYYYYYYY ZZZZZZZ	19560628	M
	1111111,22222222 33333333 444444444	19400208	F
	55555 6666666	19701127	M
	7777777^888		O
	999999 ABABABA		M

1 2 ... 35

Rules

Name	Rule	Actions
Duplicated SOPInstanceUID	Expression	✖ REMOVE
Blank SOPInstanceUID	Empty Field	✖ REMOVE

Add Rule

Figure 5.5: Sample of rules applied to a view of the dataset.

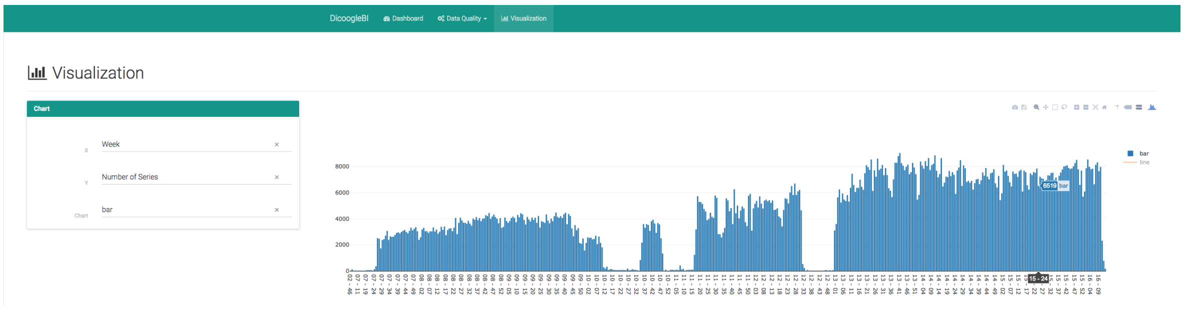


Figure 5.6: Overview of the Dicoogle BI visualisation interface.

The capabilities of the Dicoogle's BI module can also be applied to other third-party PACS thanks to the Dicoogle's ability of indexing other PACS content. Moreover, it could be used to support these research endeavours in non-production PACS. Although, the benefit of having a real-time analytical pipeline would be lost in both these cases.

Chapter 6

Conclusions

The medical practice, in general, is increasingly reliant on information systems. Most notably, the PACS plays a central role in supporting all the major workflows associated with the medical imaging practice. PACS are fundamental to improve the quality of healthcare services and the efficiency of institutional business practices. They can empower the medical personnel to be more efficient, providing better tools for clinical decision support, computer-aided diagnosis or simply by providing readily all the necessary imaging artifacts. Moreover, PACS can also be used for exploring the imaging metadata and support advanced usage scenarios, such as BI and research directly on top of production repositories. To be successful, these scenarios require multi-institutional PACS architectures and technologies to perform data analysis over large datasets.

The production of medical imaging data has grown tremendously in the last decades. Furthermore, the general trend in new imaging modalities is to produce more data per examination. Nowadays, many institutions are interested in outsourcing their systems to Cloud infrastructures and in using Web-based technologies in services related to storage, distribution, and visualization of medical images.

However, the design and implementation of distributed PACS architectures are still challenging due to the complexity of dealing with big data issues. Unfortunately, the research on this matter is still in its early stages, mostly due to the difficulty of implementing and validating new technologies in real environments, without interfering with clinical practice. Moreover, standard methodologies still do not take full advantage of outsourced components and their integration with other in-house solutions is troublesome. All these reasons have been hindering the support for advanced usage scenarios in medical imaging.

This thesis researched new computer methods for supporting efficiently advanced usage scenarios in distributed medical imaging environments. It focused on the performance optimization of storage, content discovery and retrieval services which are essential for distributed PACS architectures. An efficient distributed PACS architecture supported by a routing mechanism was proposed. It encompasses a caching system and a workflow management module that optimizes data retrieval according to institutional profiles. Complementary, the Controlled channels mechanism was developed to improve the router-to-router communications performance.

The issues related to the prototyping, development, and validation of PACS components were also addressed. This thesis has been enrolled in the architectural planning and development of the Dicoogle PACS. In addition, a framework for validating the performance and scalability of PACS services was designed. Together with Dicoogle, it offers an integrated solution for the prototyping, development, and validation of new PACS applications and services. It also provides a heterogeneous collection of datasets representative of real institutional environments and a method for extending any dataset maintaining statistical accuracy.

Finally, two very demanding advanced usage scenarios were used as proof of concept and validation for the proposed methodologies. In the field of digital pathology, the first Pathology PACS compliant with the DICOM standard, other medical imaging modalities, and Web-based technologies was proposed. It is an efficient solution able to store, distribute and display extremely high-resolution images in common Web browsers. For the Research PACS usage scenario, it was developed a Web platform for performing Data Analysis and BI directly on top of live institutional repositories.

The methodology applied successfully in this thesis to improve the performance and appeal of PACS can be applied to computer systems of virtually every area. It is itself a contribution to the computer science community in general, and more especially to all the fellows who share the passion for the field of performance engineering.

Along with the methods, solutions, and results that were addressed in this thesis, there are several research lines that could be explored in future work, such as:

- In the digital pathology field, the proposal of an automatic method for pre-fetching WSI segments based on predictive techniques, such as machine learning.
- In the research PACS scenario, the extension of the platform for supporting DA over multimodal sources, allowing the correlation of data patterns found in distinct media.
- Research the hypothetical benefits of using a native hierarchical document-oriented database in medical imaging repositories.

References

[1] L. A. B. Silva, C. Costa, and J. L. Oliveira, "A PACS archive architecture supported on cloud services", *International Journal of Computer Assisted Radiology and Surgery*, vol. 7, no. 3, pp. 349–358, 2012. DOI: 10.1007/s11548-011-0625-x.

[2] H. K. Huang, *PACS and Imaging Informatics: Basic Principles and Applications*, 2nd ed. Wiley-Blackwell, 2010, 978 pp.

[3] J. Andreu-Perez, C. C. Y. Poon, R. D. Merrifield, S. T. C. Wong, and G.-Z. Yang, "Big Data for Health", *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 4, pp. 1193–1208, 2015. DOI: 10.1109/JBHI.2015.2450362.

[4] Z. Jin and Y. Chen, "Telemedicine in the Cloud Era: Prospects and Challenges", *IEEE Pervasive Computing*, vol. 14, no. 1, pp. 54–61, 2015.

[5] E. A. Mohammed, B. H. Far, and C. Naugler, "Applications of the MapReduce programming framework to clinical big data analysis: Current landscape and future trends", *BioData Mining*, vol. 7, p. 22, 29, 2014. DOI: 10.1186/1756-0381-7-22.

[6] I. Ojog and B. Mill, "A cloud scalable platform for DICOM image analysis as a tool for remote medical support", 2013.

[7] B. Myers. (2012). Frost & Sullivan: U.S. Medical Imaging Informatics Industry Reconnects with Growth in the Enterprise Image Archiving Market.

[8] "Digital Imaging and Communications in Medicine (DICOM)", NEMA, Standard, 2017.

[9] O. S. Pianykh, *Digital Imaging and Communications in Medicine (DICOM): A Practical Introduction and Survival Guide.* Springer Science & Business Media, 26, 2009, 424 pp.

[10] D. Salas-Lopez, S. J. Weiss, B. Nester, and T. Whalen, "Physician Clinical Alignment and Integration: A Community-Academic Hospital Approach", *Journal of Healthcare Management*, vol. 59, no. 3, pp. 195–209, 2014.

[11] L. F. Donnelly, D. J. Merinbaum, M. Epelman, L. E. Grissom, K. E. Walters, *et al.*, "Benefits of integration of radiology services across a pediatric health care system with locations in multiple states", *Pediatric Radiology*, vol. 45, no. 5, pp. 736–742, 2015. DOI: 10.1007/s00247-014-3222-7.

[12] B. Gibaud, “The quest for standards in medical imaging”, *European Journal of Radiology*, vol. 78, no. 2, pp. 190–198, 2011. DOI: 10.1016/j.ejrad.2010.05.003.

[13] L. Faggioni, E. Neri, C. Castellana, D. Caramella, and C. Bartolozzi, “The future of PACS in healthcare enterprises”, *European Journal of Radiology*, vol. 78, no. 2, pp. 253–258, 2011. DOI: 10.1016/j.ejrad.2010.06.043.

[14] M. Srinivasan, E. Liederman, N. Baluyot, and R. Jacoby, “Saving time, improving satisfaction: The impact of a digital radiology system on physician workflow and system efficiency”, *Journal of healthcare information management: JHIM*, vol. 20, no. 2, pp. 123–131, 2006.

[15] C.-T. Yang, W.-C. Shih, L.-T. Chen, C.-T. Kuo, F.-C. Jiang, *et al.*, “Accessing medical image file with co-allocation HDFS in cloud”, *Future Generation Computer Systems*, vol. 43-44, pp. 61–73, 2015. DOI: 10.1016/j.future.2014.08.008.

[16] R. Maani, S. Camorlinga, and N. Arnason, “A Parallel Method to Improve Medical Image Transmission”, *Journal of Digital Imaging*, vol. 25, no. 1, pp. 101–109, 2012. DOI: 10.1007/s10278-011-9387-9.

[17] S. P. Parker, *McGraw-Hill Dictionary of Scientific and Technical Terms*. McGraw-Hill Education, 2003, 2400 pp.

[18] B. Mansoori, K. K. Erhard, and J. L. Sunshine, “Picture Archiving and Communication System (PACS) Implementation, Integration &amp; Benefits in an Integrated Health System”, *Academic Radiology*, vol. 19, no. 2, pp. 229–235, 2012. DOI: 10.1016/j.acra.2011.11.009.

[19] V. Joshi, V. R. Narra, K. Joshi, K. Lee, and D. Melson, “PACS Administrators’ and Radiologists’ Perspective on the Importance of Features for PACS Selection”, *Journal of Digital Imaging*, vol. 27, no. 4, pp. 486–495, 1, 2014. DOI: 10.1007/s10278-014-9682-3.

[20] M. Viceconti, P. Hunter, and R. Hose, “Big Data, Big Knowledge: Big Data for Personalized Healthcare”, *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 4, pp. 1209–1215, 2015. DOI: 10.1109/JBHI.2015.2406883.

[21] M. Benjamin, Y. Aradi, and R. Shreiber, “From shared data to sharing workflow: Merging PACS and teleradiology”, *European Journal of Radiology*, vol. 73, no. 1, pp. 3–9, 1, 2010. DOI: 10.1016/j.ejrad.2009.10.014.

[22] C. Bolan, “Cloud PACS and mobile apps reinvent radiology workflow”, *APPLIED RADIOLOGY*, 2013.

[23] T. U. Onbay and A. Kantarcı, “Design and implementation of a distributed teleradiaography system: DIPACS”, *Computer Methods and Programs in Biomedicine*, vol. 104, no. 2, pp. 235–242, 2011. DOI: 10.1016/j.cmpb.2011.05.006.

[24] K. C. Wang, R. W. Filice, J. F. Philbin, E. L. Siegel, and P. G. Nagy, "Five Levels of PACS Modularity: Integrating 3D and Other Advanced Visualization Tools", *Journal of Digital Imaging*, vol. 24, no. 6, pp. 1096–1102, 2011. DOI: 10.1007/s10278-011-9366-1.

[25] E. Bellon, M. Feron, T. Deprez, R. Reynders, and B. Van den Bosch, "Trends in PACS architecture", *European Journal of Radiology*, vol. 78, no. 2, pp. 199–204, 1, 2011. DOI: 10.1016/j.ejrad.2010.05.025.

[26] "Digital Imaging and Communications in Medicine (DICOM) Part 3: Information object definitions", NEMA, Standard, 2017.

[27] "Digital Imaging and Communications in Medicine (DICOM) Part 7: Message Exchange", NEMA, Standard, 2017.

[28] "Digital Imaging and Communications in Medicine (DICOM) Part 8: Network Communication Support for Message Exchange", NEMA, Standard, 2017.

[29] "Digital Imaging and Communications in Medicine (DICOM) Supplement 148: Web Access to DICOM Persistent Objects by Means of Web Services", NEMA, Standard, 2011.

[30] P. Lipton, P. Nagy, and G. Sevinc, "Leveraging Internet Technologies with DICOM WADO", *Journal of Digital Imaging*, vol. 25, no. 5, pp. 646–652, 2012. DOI: 10.1007/s10278-012-9469-3.

[31] D. S. Taubman and R. Prandolini, "Architecture, philosophy, and performance of JPIP: Internet protocol standard for JPEG2000", vol. 5150, 2003, pp. 791–805. DOI: 10.1117/12.502889.

[32] D. Dragan and D. Ivetić, "Request redirection paradigm in medical image archive implementation", *Computer Methods and Programs in Biomedicine*, vol. 107, no. 2, pp. 111–121, 2012. DOI: 10.1016/j.cmpb.2011.06.001.

[33] J.-F. Pambrun and R. Noumeir, "Streaming of Medical Images Using JPEG2000 Interactive Protocol", *Int. J. Innov. Comput. Appl.*, vol. 4, pp. 135–148, 3/4 2012. DOI: 10.1504/IJICA.2012.050058.

[34] "Digital Imaging and Communications in Medicine (DICOM) Supplement 61: JPEG 2000 Transfer Syntaxes", NEMA, Standard, 2002.

[35] "Digital Imaging and Communications in Medicine (DICOM) Supplement 106: JPEG 2000 Interactive Protocol", NEMA, Standard, 2006.

[36] K. S. Mann and A. Bansal, "HIS Integration Systems Using Modality Worklist and DICOM", *Procedia Computer Science*, vol. 37, pp. 16–23, 2014. DOI: 10.1016/j.procs.2014.08.007.

[37] C. Daniel, M. García Rojo, K. Bourquard, D. Henin, T. Schrader, *et al.*, “Standards to Support Information Systems Integration in Anatomic Pathology”, *Archives of Pathology & Laboratory Medicine*, vol. 133, no. 11, pp. 1841–1849, 1, 2009. DOI: 10.1043/1543-2165-133.11.1841.

[38] “Digital Imaging and Communications in Medicine (DICOM) Supplement 122: Specimen Identification and Revised Pathology”, NEMA, Standard, 2008.

[39] B. A. Beckwith, “Standards for Digital Pathology and Whole Slide Imaging”, in *Digital Pathology*, Springer, Cham, 2016, pp. 87–97. DOI: 10.1007/978-3-319-20379-9_9.

[40] D. Haak, Y. Z. Filmwala, E. Heder, S. Jonas, P. Boor, *et al.*, “An ImageJ Plugin for Whole Slide Imaging”, in *Bildverarbeitung Für Die Medizin 2014*, T. M. Deserno, H. Handels, H.-P. Meinzer, and T. Tolxdorff, Eds., Springer Berlin Heidelberg, 2014, pp. 415–420.

[41] G. Corredor, E. Romero, and M. Iregui, “An adaptable navigation strategy for Virtual Microscopy from mobile platforms”, *Journal of Biomedical Informatics*, vol. 54, pp. 39–49, 2015. DOI: 10.1016/j.jbi.2015.01.013.

[42] F. Wang, T. W. Oh, C. Vergara-Niedermayr, T. Kurc, and J. Saltz, “Managing and Querying Whole Slide Images”, *Proceedings of SPIE*, vol. 8319, 16, 2012.

[43] “Digital Imaging and Communications in Medicine (DICOM) Supplement 145: Whole Slide Imaging in Pathology”, NEMA, Standard, 2009.

[44] T. Kalinski, R. Zwönitzer, F. Grabellus, S.-Y. Sheu, S. Sel, *et al.*, “Lossy compression in diagnostic virtual 3-dimensional microscopy—where is the limit?”, *Human Pathology*, vol. 40, no. 7, pp. 998–1005, 2009. DOI: 10.1016/j.humpath.2008.12.010.

[45] “Digital Imaging and Communications in Medicine (DICOM) Part 20: Imaging Reports using HL7 Clinical Document Architecture”, NEMA, Standard, 2017.

[46] L. S. Ribeiro, C. Viana-Ferreira, J. L. Oliveira, and C. Costa, “XDS-I Outsourcing Proxy: Ensuring Confidentiality While Preserving Interoperability”, *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 4, pp. 1404–1412, 2014. DOI: 10.1109/JBHI.2013.2292776.

[47] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, *et al.*, “A view of cloud computing”, *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[48] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility”, *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 1, 2009. DOI: 10.1016/j.future.2008.12.001.

[49] F. Valente, C. Viana-Ferreira, C. Costa, and J. L. Oliveira, "A RESTful Image Gateway for Multiple Medical Image Repositories", *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 3, pp. 356–364, 2012. DOI: 10.1109/TITB.2011.2176497.

[50] "Prepare for disasters, tackle terabytes when evaluating medical image archiving", Frost & Sullivan, 2008. [Online]. Available: http://www.ironmountain.com/forms/drcmi/prepare-for-disasters-tackle-terabytes-when-evaluating-medical-image-archiving.pdf (visited on 10/27/2017).

[51] L. Liu, L. Liu, X. Fu, Q. Huang, Y. Zhang, *et al.*, "SmartWADO: An Extensible WADO Middleware for Regional Medical Image Sharing", *Journal of Digital Imaging*, vol. 28, no. 5, pp. 547–557, 2015. DOI: 10.1007/s10278-015-9790-8.

[52] L. A. B. Silva, C. Costa, and J. L. Oliveira, "DICOM relay over the cloud", *International Journal of Computer Assisted Radiology and Surgery*, vol. 8, no. 3, pp. 323–333, 1, 2013. DOI: 10.1007/s11548-012-0785-3.

[53] L. S. Ribeiro, C. Costa, and J. L. Oliveira, "Clustering of distinct PACS archives using a cooperative peer-to-peer network", *Computer Methods and Programs in Biomedicine*, vol. 108, no. 3, pp. 1002–1011, 2012. DOI: 10.1016/j.cmpb.2012.05.013.

[54] ——, "A proxy of DICOM services", presented at the Medical Imaging 2010: Advanced PACS-Based Imaging Informatics and Therapeutic Applications, vol. 7628, International Society for Optics and Photonics, 11, 2010, p. 76280L. DOI: 10.1117/12.843572.

[55] A. Sharma, T. Pan, B. B. Cambazoglu, M. Gurcan, T. Kurc, *et al.*, "VirtualPACS—A Federating Gateway to Access Remote Image Data Resources over the Grid", *Journal of Digital Imaging*, vol. 22, no. 1, pp. 1–10, 1, 2009. DOI: 10.1007/s10278-007-9074-z.

[56] S. G. Erberich, J. C. Silverstein, A. Chervenak, R. Schuler, M. D. Nelson, *et al.*, "Globus MEDICUS-federation of DICOM medical imaging devices into healthcare Grids", *Studies in Health Technology and Informatics*, vol. 126, p. 269, 2007.

[57] W. Wang and E. Krishnan, "Big Data and Clinicians: A Review on the State of the Science", *JMIR Medical Informatics*, vol. 2, no. 1, 17, 2014. DOI: 10.2196/medinform.2913.

[58] L. Griebel, H.-U. Prokosch, F. Köpcke, D. Toddenroth, J. Christoph, *et al.*, "A scoping review of cloud computing in healthcare", *BMC Medical Informatics and Decision Making*, vol. 15, p. 17, 19, 2015. DOI: 10.1186/s12911-015-0145-7.

[59] H. H. Liu, *Software Performance and Scalability: A Quantitative Approach.* John Wiley & Sons, 20, 2011, 453 pp.

[60] J. Philbin, F. Prior, and P. Nagy, "Will the Next Generation of PACS Be Sitting on a Cloud?", *Journal of Digital Imaging*, vol. 24, no. 2, pp. 179–183, 1, 2011. DOI: 10.1007/s10278-010-9331-4.

[61] M. Michael, J. E. Moreira, D. Shiloach, and R. W. Wisniewski, "Scale-up x scale-out: A case study using nutch/lucene", in *2007 IEEE International Parallel and Distributed Processing Symposium*, IEEE, 2007, pp. 1–8.

[62] D. Agrawal, S. Das, and A. El Abbadi, "Big Data and Cloud Computing: Current State and Future Opportunities", in *Proceedings of the 14th International Conference on Extending Database Technology*, ACM, 2011, pp. 530–533. DOI: 10.1145/1951365.1951432.

[63] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, *et al.*, "Bigtable: A Distributed Storage System for Structured Data", *ACM Trans. Comput. Syst.*, vol. 26, no. 2, 4:1–4:26, 2008. DOI: 10.1145/1365815.1365816.

[64] B. F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, *et al.*, "PNUTS: Yahoo!'s Hosted Data Serving Platform", *Proc. VLDB Endow.*, vol. 1, no. 2, pp. 1277–1288, 2008. DOI: 10.14778/1454159.1454167.

[65] H. V. Jagadish, S. Al-Khalifa, A. Chapman, L. V. S. Lakshmanan, A. Nierman, *et al.*, "TIMBER: A Native XML Database", *The VLDB Journal*, vol. 11, no. 4, pp. 274–291, 2002. DOI: 10.1007/s00778-002-0081-x.

[66] W. Meier, "eXist: An Open Source Native XML Database", in *Web, Web-Services, and Database Systems*, Springer, Berlin, Heidelberg, 7, 2002, pp. 169–183. DOI: 10.1007/3-540-36560-5_13.

[67] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008. DOI: 10.1145/1327452.1327492.

[68] S. Madden, "From Databases to Big Data", *IEEE Internet Computing*, vol. 16, no. 3, pp. 4–6, 2012. DOI: 10.1109/MIC.2012.50.

[69] C. Costa, C. Ferreira, L. Bastião, L. Ribeiro, A. Silva, *et al.*, "Dicoogle - an Open Source Peer-to-Peer PACS", *Journal of Digital Imaging*, vol. 24, no. 5, pp. 848–856, 2011. DOI: 10.1007/s10278-010-9347-9.

[70] F. Valente, L. A. B. Silva, T. M. Godinho, and C. Costa, "Anatomy of an Extensible Open Source PACS", *Journal of Digital Imaging*, vol. 29, no. 3, pp. 284–296, 2016. DOI: 10.1007/s10278-015-9834-0.

[71] F. Wang, R. Lee, X. Zhang, and J. Saltz, "Towards Building High Performance Medical Image Management System for Clinical Trials", *Proceedings of SPIE*, vol. 7967, 79670C 2011. DOI: 10.1117/12.877838.

[72] A. Jacobs, “The Pathologies of Big Data”, *Commun. ACM*, vol. 52, no. 8, pp. 36–44, 2009. DOI: 10.1145/1536616.1536632.

[73] A. Savaris, T. Harder, and A. von Wangenheim, “DCMDSM: A DICOM decomposed storage model”, *Journal of the American Medical Informatics Association*, vol. 21, no. 5, pp. 917–924, 1, 2014. DOI: 10.1136/amiajnl-2013-002337.

[74] E. G. M. Petrakis and A. Faloutsos, “Similarity searching in medical image databases”, *IEEE Transactions on Knowledge and Data Engineering*, vol. 9, no. 3, pp. 435–447, 1997. DOI: 10.1109/69.599932.

[75] C. Viana-Ferreira, L. Ribeiro, S. Matos, and C. Costa, “Pattern recognition for cache management in distributed medical imaging environments”, *International Journal of Computer Assisted Radiology and Surgery*, pp. 1–10, 2015. DOI: 10.1007/s11548-015-1272-4.

[76] C. Viana-Ferreira, A. Guerra, J. F. Silva, S. Matos, and C. Costa, “An Intelligent Cloud Storage Gateway for Medical Imaging”, *Journal of Medical Systems*, vol. 41, no. 9, p. 141, 1, 2017. DOI: 10.1007/s10916-017-0790-8.

[77] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, Reprinted. Cambridge Univ. Press, 2009, 482 pp.

[78] M. Kleppmann, *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems.* O’Reilly Media, 14, 2017, 590 pp.

[79] W. Su, J. Wang, and F. H. Lochovsky, “Record Matching over Query Results from Multiple Web Databases”, *IEEE Trans. on Knowl. and Data Eng.*, vol. 22, no. 4, pp. 578–589, 2010. DOI: 10.1109/TKDE.2009.90.

[80] Q. Li and B. Moon, “Indexing and querying XML data for regular path expressions”, in *VLDB*, vol. 1, 2001, pp. 361–370.

[81] T. M. Godinho, L. A. B. Silva, C. Viana-Ferreira, C. Costa, and J. L. Oliveira, “Enhanced regional network for medical imaging repositories”, in *2013 8th Iberian Conference on Information Systems and Technologies (CISTI)*, 2013, pp. 1–6.

[82] R. Maani, S. Camorlinga, N. Arnason, and R. Eskicioglu, “A practical fast method for medical imaging transmission based on the DICOM protocol”, B. J. Liu and W. W. Boonn, Eds., 4, 2010, p. 76280M. DOI: 10.1117/12.843896.

[83] L. He, L. Xu, X. Ming, and Q. Liu, “A Web Service System Supporting Three-dimensional Post-processing of Medical Images Based on WADO Protocol”, *Journal of Medical Systems*, vol. 39, no. 2, 2015. DOI: 10.1007/s10916-015-0196-4.

[84] S. G. Langer, K. Persons, B. J. Erickson, and D. Blezek, "Towards a More Cloud-Friendly Medical Imaging Applications Architecture: A Modest Proposal", *Journal of Digital Imaging*, vol. 26, no. 1, pp. 58–64, 2013. DOI: 10.1007/s10278-012-9545-8.

[85] R. Appuswamy, D. C. van Moolenbroek, and A. S. Tanenbaum, "Cache, cache everywhere, flushing all hits down the sink: On exclusivity in multilevel, hybrid caches", in *2013 IEEE 29th Symposium on Mass Storage Systems and Technologies (MSST)*, IEEE, 2013, pp. 1–14.

[86] C. Viana-Ferreira, S. Matos, and C. Costa, "Long-term prefetching for cloud medical imaging repositories", *Studies in health technology and informatics*, vol. 210, pp. 1028–1030, 2014.

[87] A. A. Bui, M. F. McNitt-Gray, J. G. Goldin, A. F. Cardenas, and D. R. Aberle, "Problem-oriented Prefetching for an Integrated Clinical Imaging Workstation", *Journal of the American Medical Informatics Association : JAMIA*, vol. 8, no. 3, pp. 242–253, 2001.

[88] A. Descampe, C. D. Vleeschouwer, M. Iregui, B. Macq, and F. Marques, "Prefetching and Caching Strategies for Remote and Interactive Browsing of JPEG2000 Images", *IEEE Transactions on Image Processing*, vol. 16, no. 5, pp. 1339–1354, 2007. DOI: 10.1109/TIP.2007.894258.

[89] S. Langer, "Issues Surrounding PACS Archiving to External, Third-Party DICOM Archives", *Journal of Digital Imaging*, vol. 22, no. 1, pp. 48–52, 2009. DOI: 10.1007/s10278-008-9125-0.

[90] J. Gutiérrez-Martínez, M. A. Núñez-Gaona, H. Aguirre-Meneses, and R. E. Delgado-Esquerra, "A Software and Hardware Architecture for a High-Availability PACS", *Journal of Digital Imaging*, vol. 25, no. 4, pp. 471–479, 2012. DOI: 10.1007/s10278-012-9494-2.

[91] F. Valente, C. Costa, and A. Silva, "Dicoogle, a Pacs Featuring Profiled Content Based Image Retrieval", *PLoS ONE*, vol. 8, no. 5, 6, 2013. DOI: 10.1371/journal.pone.0061888.

[92] F. Valente, A. Silva, and C. Costa, "Content-based Image Retrieval for Clinical Applications: An Overview of Current Approaches and Challenges", *Current Medical Imaging Reviews*, vol. 9, no. 4, pp. 250–262, 1, 2013.

[93] S. Harizopoulos, D. J. Abadi, S. Madden, and M. Stonebraker, "OLTP Through the Looking Glass, and What We Found There", in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ACM, 2008, pp. 981–992. DOI: 10.1145/1376616.1376713.

[94] P. Buneman, “Semistructured Data”, in *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, ACM, 1997, pp. 117–121. DOI: 10.1145/263661.263675.

[95] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, *et al.*, “Dynamo: Amazon’s Highly Available Key-value Store”, in *Proceedings of Twenty-First ACM SIGOPS Symposium on Operating Systems Principles*, ACM, 2007, pp. 205–220. DOI: 10.1145/1294261.1294281.

[96] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman, “On Supporting Containment Queries in Relational Database Management Systems”, in *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, ACM, 2001, pp. 425–436. DOI: 10.1145/375663.375722.

[97] L. A. B. Silva, L. Beroud, C. Costa, and J. L. Oliveira, “Medical imaging archiving: A comparison between several NoSQL solutions”, in *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, IEEE, 2014, pp. 65–68.

[98] T. C. Prado, D. D. J. de Macedo, M. A. R. Dantas, and A. von Wangenheim, “Optimization of PACS Data Persistency Using Indexed Hierarchical Data”, *Journal of Digital Imaging*, 9, 2014. DOI: 10.1007/s10278-013-9665-9.

[99] A. Savaris, T. Härder, and A. v Wangenheim, “Evaluating a Row-Store Data Model for Full-Content DICOM Management”, in *2014 IEEE 27th International Symposium on Computer-Based Medical Systems*, 2014, pp. 193–198. DOI: 10.1109/CBMS.2014.61.

[100] S. J. Rascovsky, J. A. Delgado, A. Sanz, V. D. Calvo, and G. Castrillón, “Informatics in Radiology: Use of CouchDB for Document-based Storage of DICOM Objects”, *RadioGraphics*, vol. 32, no. 3, pp. 913–927, 1, 2012. DOI: 10.1148/rg.323115049.

[101] D. D. De Macedo, A. Von Wangenheim, M. Dantas, and H. G. Perantunes, “An architecture for DICOM medical images storage and retrieval adopting distributed file systems”, *International Journal of High Performance Systems Architecture*, vol. 2, no. 2, pp. 99–106, 1, 2009. DOI: 10.1504/IJHPSA.2009.032027.

[102] S. G. Langer, “A Flexible Database Architecture for Mining DICOM Objects: The DICOM Data Warehouse”, *Journal of Digital Imaging*, vol. 25, no. 2, pp. 206–212, 2012. DOI: 10.1007/s10278-011-9434-6.

[103] ——, “DICOM Data Warehouse: Part 2”, *Journal of Digital Imaging*, vol. 29, no. 3, pp. 309–313, 2016. DOI: 10.1007/s10278-015-9830-4.

[104] C. Costa, F. Freitas, M. Pereira, A. Silva, and J. L. Oliveira, “Indexing and retrieving DICOM data in disperse and unstructured archives”, *International Journal of Computer Assisted Radiology and Surgery*, vol. 4, no. 1, pp. 71–77, 1, 2009. DOI: 10.1007/s11548-008-0269-7.

[105] C. Viana-Ferreira, C. Costa, and J. L. Oliveira, "Dicoogle relay - a cloud communications bridge for medical imaging", in *2012 25th IEEE International Symposium on Computer-Based Medical Systems (CBMS)*, 2012, pp. 1–6. DOI: 10.1109/CBMS.2012.6266402.

[106] C. Viana-Ferreira, D. Ferreira, F. Valente, E. Monteiro, C. Costa, *et al.*, "Dicoogle Mobile: A medical imaging platform for Android", *Studies in Health Technology and Informatics*, vol. 180, pp. 502–506, 2012.

[107] M. Santos, L. Bastião, C. Costa, A. Silva, and N. Rocha, "DICOM and Clinical Data Mining in a Small Hospital PACS: A Pilot Study", in *ENTERprise Information Systems*, Springer, Berlin, Heidelberg, 5, 2011, pp. 254–263. DOI: 10.1007/978-3-642-24352-3_27.

[108] L. A. B. Silva, L. S. Ribeiro, M. Santos, N. Neves, D. Francisco, *et al.*, "Normalizing Heterogeneous Medical Imaging Data to Measure the Impact of Radiation Dose", *Journal of Digital Imaging*, vol. 28, no. 6, pp. 671–683, 1, 2015. DOI: 10.1007/s10278-015-9805-5.

[109] A. P. Alves, T. M. Godinho, and C. Costa, "Assessing the relational database model for optimization of content discovery services in medical imaging repositories", IEEE, 2016, pp. 1–6. DOI: 10.1109/HealthCom.2016.7749484.

[110] L. S. Ribeiro, R. P. Rodrigues, C. Costa, and J. L. Oliveira, "Enabling outsourcing XDS for imaging on the public cloud.", in *MedInfo*, 2013, pp. 33–37.

[111] S. Wang, W. Pavlicek, C. C. Roberts, S. G. Langer, M. Zhang, *et al.*, "An Automated DICOM Database Capable of Arbitrary Data Mining (Including Radiation Dose Indicators) for Quality Monitoring", *Journal of Digital Imaging*, vol. 24, no. 2, pp. 223–233, 2011. DOI: 10.1007/s10278-010-9329-y.

[112] J. Wang, "A survey of web caching schemes for the Internet", *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 5, pp. 36–46, 1999. DOI: 10.1145/505696.505701.

[113] K. J. Dreyer and D. S. Hirschorn, Eds., *PACS: A Guide to the Digital Revolution*, 2. ed, Springer Science+Business Media, 2006, 579 pp.

[114] M. Burrows and D. J. Wheeler, "A block-sorting lossless data compression algorithm", 1994.

[115] T. M. Godinho, C. Costa, and J. L. Oliveira, "Intelligent generator of big data medical imaging repositories", *IET Software*, 2017.

[116] A. Huisman, A. Looijen, S. M. van den Brink, and P. J. van Diest, "Creation of a fully digital pathology slide archive by high-volume tissue slide scanning", *Human Pathology*, vol. 41, no. 5, pp. 751–757, 2010. DOI: 10.1016/j.humpath.2009.08.026.

[117] J. Ho, A. V. Parwani, D. M. Jukic, Y. Yagi, L. Anthony, *et al.*, "Use of whole slide imaging in surgical pathology quality assurance: Design and pilot validation studies", *Human Pathology*, vol. 37, no. 3, pp. 322–331, 1, 2006. DOI: `10.1016/j.humpath.2005.11.005`.

[118] S. Park, L. Pantanowitz, and A. V. Parwani, "Digital Imaging in Pathology", *Clinics in Laboratory Medicine*, vol. 32, no. 4, pp. 557–584, 1, 2012. DOI: `10.1016/j.cll.2012.07.006`.

[119] N. Farahani, A. V. Parwani, and L. Pantanowitz. (11, 2015). Whole slide imaging in pathology: Advantages, limitations, and emerging perspectives.

[120] V. Punys, A. Laurinavicius, and J. Puniene, "A data model for handling whole slide microscopy images in picture archiving and communications systems", *Studies in Health Technology and Informatics*, vol. 150, pp. 856–860, 2009.

[121] M. Garcia-Rojo, A. Sanchez, G. Bueno, and D. de Mena, "Standardization Of Pathology Whole Slide Images According To DICOM 145 Supplement And Storage In PACs", 2016. DOI: `10.17629/www.diagnosticpathology.eu-2016-8:175`.

[122] E. S. Patterson, M. Rayo, C. Gill, and M. N. Gurcan, "Barriers and facilitators to adoption of soft copy interpretation from the user perspective: Lessons learned from filmless radiology for slideless pathology", *Journal of Pathology Informatics*, vol. 2, 7, 2011. DOI: `10.4103/2153-3539.74940`.

[123] L. Pantanowitz, P. N. Valenstein, A. J. Evans, K. J. Kaplan, J. D. Pfeifer, *et al.*, "Review of the current state of whole slide imaging in pathology", *Journal of Pathology Informatics*, vol. 2, 13, 2011. DOI: `10.4103/2153-3539.83746`.

[124] J. C. Caicedo, F. A. González, and E. Romero, "Content-based histopathology image retrieval using a kernel-based semantic annotation framework", *Journal of Biomedical Informatics*, vol. 44, no. 4, pp. 519–528, 2011. DOI: `10.1016/j.jbi.2011.01.011`.

[125] G. Bueno, M. M. Fernández-Carrobles, O. Deniz, and M. García-Rojo, "New Trends of Emerging Technologies in Digital Pathology", *Pathobiology*, vol. 83, pp. 61–69, 2-3 26, 2016. DOI: `10.1159/000443482`.

[126] T. T. Brunyé, E. Mercan, D. L. Weaver, and J. G. Elmore, "Accuracy is in the eyes of the pathologist: The visual interpretive process and diagnostic accuracy with digital whole slide images", *Journal of Biomedical Informatics*, vol. 66, pp. 171–179, 2017. DOI: `10.1016/j.jbi.2017.01.004`.

[127] M. G. Rojo, G. B. Garcia, C. P. Mateos, J. G. Garcia, and M. C. Vicente, "Critical Comparison of 31 Commercially Available Digital Slide Systems in Pathology", *International Journal of Surgical Pathology*, vol. 14, no. 4, pp. 285–305, 1, 2006. DOI: `10.1177/1066896906292274`.

[128] R. Zwönitzer, H. Hofmann, A. Roessner, and T. Kalinski, "Virtual 3D microscopy in pathology education", *Human Pathology*, vol. 41, no. 3, pp. 457–458, 2010. DOI: `10.1016/j.humpath.2009.10.012`.

[129] V. J. Tuominen and J. Isola, "The Application of JPEG2000 in Virtual Microscopy", *Journal of Digital Imaging*, vol. 22, no. 3, pp. 250–258, 1, 2009. DOI: `10.1007/s10278-007-9090-z`.

[130] A. Goode, B. Gilbert, J. Harkes, D. Jukic, and M. Satyanarayanan, "OpenSlide: A vendor-neutral software foundation for digital pathology", *Journal of Pathology Informatics*, vol. 4, no. 1, pp. 27–27, 2013.

[131] R. Schuler, D. E. Smith, G. Kumaraguruparan, A. Chervenak, A. D. Lewis, *et al.*, "A Flexible, Open, Decentralized System for Digital Pathology Networks", *Studies in health technology and informatics*, vol. 175, pp. 29–38, 2012.

[132] V. J. Tuominen and J. Isola, "Linking whole-slide microscope images with DICOM by using JPEG2000 interactive protocol", *Journal of Digital Imaging*, vol. 23, no. 4, pp. 454–462, 2010. DOI: `10.1007/s10278-009-9200-1`.

[133] E. J. C. Arguinarena, J. E. Macchi, P. P. Escobar, M. Del Fresno, J. M. Massa, *et al.*, "Dcm-ar: A fast flash-based Web-PACS viewer for displaying large DICOM images", *Conference proceedings: ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference*, vol. 2010, pp. 3463–3466, 2010. DOI: `10.1109/IEMBS.2010.5627827`.

[134] E. J. M. Monteiro, C. Costa, and J. L. Oliveira, "A DICOM viewer based on web technology", in *2013 IEEE 15th International Conference on E-Health Networking, Applications and Services (Healthcom 2013)*, 2013, pp. 167–171. DOI: `10.1109/HealthCom.2013.6720660`.

[135] M. G. Rojo and G. Bueno, "Analysis of the impact of high-resolution monitors in digital pathology", *Journal of Pathology Informatics*, vol. 6, p. 57, 2015.

[136] B. Hamilton, "Big Data is the Future of Healthcare", Cognizant, 2012. [Online]. Available: `https://www.cognizant.com/industries-resources/life_sciences/Big-Data-is-the-Future-of-Healthcare.pdf` (visited on 11/07/2017).

[137] M. Santos, L. Bastião, C. Costa, A. Silva, and N. Rocha, "Clinical Data Mining in Small Hospital PACS: Contributions for Radiology Department Improvement", in *Information Systems and Technologies for Enhancing Health and Social Care*, IGI Global, 2013, pp. 236–251.

[138] M. Hu, W. Pavlicek, P. T. Liu, M. Zhang, S. G. Langer, *et al.*, "Informatics in Radiology: Efficiency Metrics for Imaging Device Productivity", *RadioGraphics*, vol. 31, no. 2, pp. 603–616, 2011. DOI: 10.1148/rg.312105714.

[139] S. Ondategui-Parra, S. M. Erturk, and P. R. Ros, "Survey of the Use of Quality Indicators in Academic Radiology Departments", *American Journal of Roentgenology*, vol. 187, no. 5, W451–W455, 1, 2006. DOI: 10.2214/AJR.05.1064.

[140] W. Raghupathi and V. Raghupathi, "Big data analytics in healthcare: Promise and potential", *Health Information Science and Systems*, vol. 2, no. 1, p. 3, 1, 2014. DOI: 10.1186/2047-2501-2-3.

[141] P. G. Nagy, M. J. Warnock, M. Daly, C. Toland, C. D. Meenan, *et al.*, "Informatics in Radiology: Automated Web-based Graphical Dashboard for Radiology Operational Business Intelligence", *RadioGraphics*, vol. 29, no. 7, pp. 1897–1906, 1, 2009. DOI: 10.1148/rg.297095701.

[142] S. G. Langer, "Challenges for Data Storage in Medical Imaging Research", *Journal of Digital Imaging*, vol. 24, no. 2, pp. 203–207, 2011. DOI: 10.1007/s10278-010-9311-8.

[143] H. J. Watson and B. H. Wixom, "The Current State of Business Intelligence", *Computer*, vol. 40, no. 9, pp. 96–99, 2007. DOI: 10.1109/MC.2007.331.

[144] H. Chen, R. H. L. Chiang, and V. C. Storey, "Business Intelligence and Analytics: From Big Data to Big Impact", *MIS Q.*, vol. 36, no. 4, pp. 1165–1188, 2012.

[145] J. T. L. Wang, M. J. Zaki, H. T. T. Toivonen, and D. Shasha, "Introduction to Data Mining in Bioinformatics", in *Data Mining in Bioinformatics*, Springer, London, 2005, pp. 3–8. DOI: 10.1007/1-84628-059-1_1.

[146] H.-E. Källman, E. Halsius, M. Olsson, and M. Stenström, "DICOM Metadata repository for technical information in digital medical images", *Acta Oncologica (Stockholm, Sweden)*, vol. 48, no. 2, pp. 285–288, 2009. DOI: 10.1080/02841860802258786.

Annex A

Intelligent Generator of Big Data Medical Imaging Repositories

A.1 Abstract

The production of medical imaging data has grown tremendously in the last decades. Nowadays, even small institutions produce a considerable amount of studies. Furthermore, the general trend in new imaging modalities is to produce more data per examination. As a result, the design and implementation of tomorrow's storage and communication systems must deal with Big Data issues. The research on technologies to cope with Big Data issues in large scale medical imaging environments is still in its early stages. This is mostly due to the difficulty of implementing and validating new technological approaches in real environments, without interfering with clinical practice. Therefore, it is crucial to create test bed environments for research purposes. This article proposes a methodology for creating simulated medical imaging repositories, based on the indexing of model datasets, extraction of patterns and modelling of study production. The system creates a model from a real-world repository's representative time window and expands it according to ongoing research needs. In addition, the solution provides distinct approaches to reducing the size of the generated datasets. The proposed system has already been used by other research projects in validation processes that aim to assess the performance and scalability of developed systems.

A.2 Introduction

In the last decade, medical imaging studies have become one of the most important means of diagnosis [1]. The introduction of digital modalities has lowered the exploitation costs and increased the quality and usefulness of medical images [2]. For these reasons, nowadays, even small medical institutions are capable of producing large quantities of medical imaging studies [3].

PACS is the common designation of information systems that manage medical imaging data and associated workflows [4]. They encompass not only storage infrastructures that allow the storing of images for later use but also communication infrastructures, which support sharing and remote access to data across distinct institutions and applications. A typical PACS environment includes three major groups of applications: Image Repositories, Acquisition devices, and Viewer applications [4]. In order to integrate these heterogeneous components, PACS rely on the DICOM Standard [5], which defines the format for storing medical imaging data and the network communication protocol.

Currently, the community is particularly interested in medical imaging content discovery for supporting clinical and technological research activities, including features extracted from pixel data (image) and metadata [6]. In fact, DICOM objects are composed of a metadata header along with the actual pixel data. The metadata section includes information related to imaging procedures such as patient identification, equipment acquisition parameters, diagnosis report or radiation dose exposure [6]. However, the heterogeneous nature of the DICOM metadata overburdens general purpose technologies. In DICOM, different modalities and clinical protocols produce different sets of metadata, making it difficult to use strict data models such as relational databases [7]. Furthermore, the massive volume of data generated by medical imaging procedures is itself a huge problem [8], which raises several Big Data related issues in their exploitation [9]. At the same time, there is also a considerable research effort in healthcare services federation [10] and Cloud outsourcing of medical imaging repositories. Managing communication between multiple geo-distributed locations is still challenging due to the huge volume of data that must be readily available for medical practice.

Most of the traditional technologies are not prepared to handle the requirements raised by those Big Data environments. Recent research efforts have been made with the adoption of non-conventional database technologies, such as document stores and free text indexes, in PACS [7, 11–13]. Concerning networks, several distributed architectures have been proposed, contemplating optimized communication processes, cache, and pre-fetch mechanisms to support those specific scenarios [3, 14].

These technologies promise to increase search performance and the flexibility of repository databases and reduce data access latency. However, they generally have poor validation processes, due to the lack of real-world data sets and workflows able to reproduce Big Data environments. Some of the reasons for this are: Firstly, privacy reasons, which make it more difficult to perform tests using real-world data. In addition, tests could be hazardous since there is a considerable probability of jeopardizing the regular operation of the healthcare institution. Another reason is that a particular institution dataset may not cover all the specificity required for validation purposes. Finally, it is crucial to analyse how new contributions will perform in the long-term.

This paper proposes and describes a new system able to generate big datasets using only representative portions of data extracted from real environments, causing no impact on production processes. The generated repositories are excellent elements for supporting research, development and validation of PACS technologies.

A.3 Background

The DICOM Standard [15] was introduced in 1993 by NEMA, with the main purpose of normalizing data structures and communications to promote interoperability between different PACS components. Before that, every manufacturer had its own protocol, which promoted vendor locking situations and poor exploitation of resources. Nowadays, it is a de facto standard, universally adopted by distinct medical imaging modalities and manufacturers. This standard normalizes the content of medical imaging objects, how they should be formatted and broadcast among the different PACS applications [16]. For simplicity purposes, this article will not deal with the communication part of the standard, since the proposed methods only demand knowledge of DICOM data structures.

A DICOM object aggregates metadata with images' pixel data [16]. The header contains meaningful information associated with the examination such as patient identification, equipment details, and radiation exposure information. These elements have a Unique Identifier [5] denominated as tag. For instance, the PatientName attribute is identified by the tag (0010,0010). These attributes are organized in semantic groups following the DIM, a hierarchical database that captures real-world organisation: Patient–Study–Series–Images. A patient has multiple studies that may have multiple series that include one or more images. DICOM supports 27 data types; the attributes are matched to their data type using a dictionary [5], alternately, each attribute may declare its data type explicitly. Although the DICOM standard specifies a wide range of attributes, new ones can be declared via private dictionaries. Therefore, the standard is extensible and manufacturers can include their latest equipment metadata. Finally, a file object may contain one or more images (i.e. cine-loop) [17]. Figure A.1 provides an example of a DICOM file's metadata and its hierarchical organization. It is perceivable that the root object holds multiple data elements. These elements can be composite elements holding further members in their hierarchy.

Regarding the problem stated in this article, the literature reports several studies that use state-of-the-art technologies for storage, search and retrieval of medical imaging data, but where the lack of representative datasets is evident in the validation process. For instance, Savaris et al. propose a decomposed storage model [7], an approach similar to document-oriented databases, and present its comparison with the relational model. However, the dataset used in the trials has only 67 studies with a combined size of around 4GB. In fact, this dataset is very small when compared to real medical imaging repositories, and does not validate the system's long-term performance. In [14], a regional PACS archive is proposed and

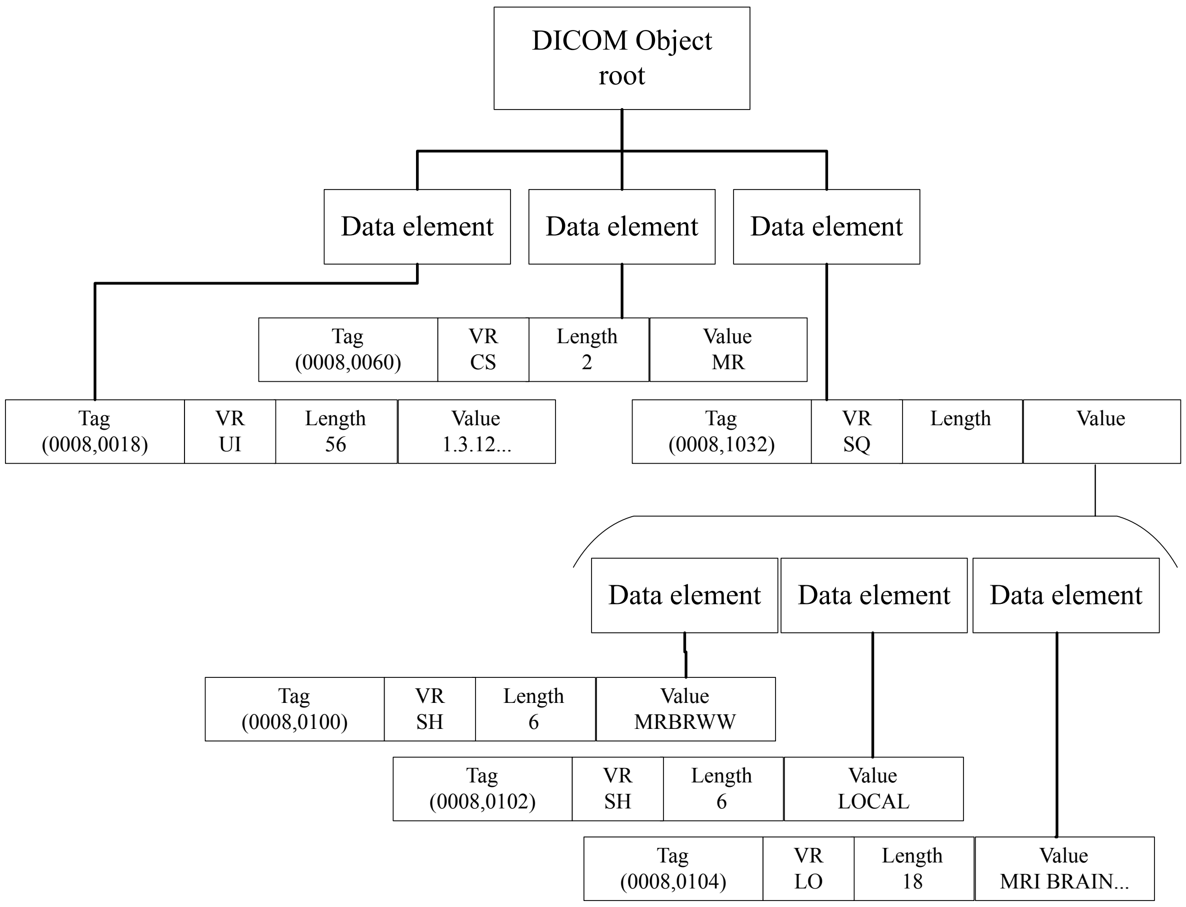

DICOM Object root
Data element
Data element
Data element
Tag (0008,0060) | VR CS | Length 2 | Value MR
Tag (0008,0018) | VR UI | Length 56 | Value 1.3.12...
Tag (0008,1032) | VR SQ | Length | Value
Data element
Data element
Data element
Tag (0008,0100) | VR SH | Length 6 | Value MRBRWW
Tag (0008,0102) | VR SH | Length 6 | Value LOCAL
Tag (0008,0104) | VR LO | Length 18 | Value MRI BRAIN...

Figure A.1: Illustration of nested objects in DICOM.

the validation scenario includes two small diagnostic centres that handle an average of 3 000 monthly examinations, with a combined volume of around 60 GBytes. In this scenario, a trial with a 4GB dataset is not sufficient to analyse even a week of the PACS operation, nor the long-term scalability of the proposed system.

Multiple document-oriented databases for PACS archives have been put through a comparative laboratory trial [13]. The authors considered the performance of the databases in the short, medium and long-term by using three different datasets. However, their largest dataset might not be representative enough of the current scenario, since it contains only around 50 thousand images, approximately 4 times the daily CT production of our dataset C (see section IV). More evidence of this issue is found in [11, 18–20].

A.4 Method

This section presents our methodology for generating Big Data medical imaging repositories to be used in controlled laboratory trials. This methodology was instantiated as a software system which orchestrates multiple components in order to (1) capture the production statistics of an existing repository over a period of time; (2) create a representative regression model of the collected productivity statistics; (3) and finally, generate datasets of DICOM files with similar productivity to the model, but possibly encompassing larger time periods. The main challenge is to capture as accurately as possible how productivity has evolved over time in the model repository. In this regard, we have also included a results section intended to assess this accuracy, and validate our method.

The Dicoogle PACS arises as a crucial component in our method for generating medical imaging datasets. It is an open-source PACS archive that offers a plugin-based architecture and a SDK [12, 21]. It is intended to support the research and development of new PACS components and applications as plugins. These plugins may leverage the existing capabilities of Dicoogle, such as storage, indexing of DICOM metadata or even other functionalities. Therefore, it is possible to orchestrate services, creating complex workflows between multiple plugins, as demonstrated further. The components of the proposed system were developed as Dicoogle plugins, namely the DICOM Statistics Analyser and the DICOM Exporter.

Our method can be broken down into three stages: indexing, statistical analysis and exportation. Next, we provide a description of each stage, as well as the roles of the components presented above, as illustrated in Figure A.2.

In order to generate an accurate dataset, our method starts by indexing a real-world repository that will be used as a model. The study patterns discovered in this model repository will be translated into the synthesized datasets. This indexing stage is a crucial task and requires full access to DICOM metadata of the model repository. Dicoogle's data interfaces and indexation capabilities support this task. It can import images from the institutional PACS repository using the DICOM Query/Retrieve services. Alternatively, Dicoogle can detect and automatically index DICOM files stored in the local file systems or network folders. This can be used to speed-up the indexation process, but it has more limited applicability. Once Dicoogle gains access to a DICOM image, it parses the metadata within the object and stores it in an indexing plugin. Although Dicoogle supports multiple indexing plugins, we have been using the Lucene plugin supplied with its distribution.

In this stage, Dicoogle also captures sensitive information related to patients and medical professionals such as names, identification numbers and birthdates. This information is

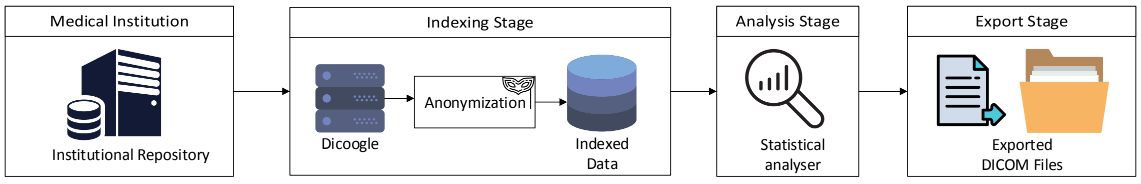


Figure A.2: Architecture and information flow of the dataset generator.

not required by our method, and it may even constitute a security hazard, hindering the generalized usage of our method. As a result, this stage also includes an anonymization mechanism that removes those elements to ensure data privacy. It involves the obfuscation of several fields that allow identification of the different subjects in the study, including PatientName, InstituionName, PhysicianNames etc. The fields that require anonymization are configurable, allowing different levels of privacy. The algorithm is based on term replacement, where attributes' values are replaced by an anonymous counterpart. We used a one-to-one relation keeping the coherence of the indexed data. For instance, a given patient name will produce the same anonymized value, keeping the relations between the different image objects intact.

Our statistical analyser is responsible for generating a statistical model out of the model dataset. This model will enable the generation of more imaging studies with similar production characteristics. A model is generated for each medical imaging modality separately. Preliminary experiments show that this option produces more accurate results. Moreover, it also allows us to customize the exported datasets since other modalities can be included or excluded according to user preferences. For each modality, the study production is modelled using an Autoregressive Moving Average Model (ARMA) regression [22]. This regression analysis tool is well suited to time series data because it models the series' general underlying trend, as well as some extrinsic variation factor. In our particular case, this configuration allows the generation of a dataset that better represents the long-term production evolution. In fact, the inclusion of the ARMA regression model was a fundamental improvement in our method. Previously, our model was composed of simple statistical functions, which were unable to capture the trend in the study production series or other fluctuations. The quality of the model produced by the ARMA regression is heavily influenced by the timespan of the index collected from the real-world repository. Longer timespans allow for a more accurate model, which will perform better at generating datasets which extend the collected timespan. A more in-depth description of the ARMA regression is outside the scope of this work and can be found in [22].

As soon as statistics collection finishes, the DICOM Exporter module is able to start the process of generating new datasets. The synthesized dataset replicates the patterns discovered in the model repository but many options can be configured according to researchers' requirements. The volume of the generated dataset can be controlled by either directly specifying the dataset's time window in number of weeks, or indirectly by defining the maximum dataset size. The number of exported modalities can also be configured. For instance, it is possible to generate only CT studies even if the model dataset contains other modalities.

As expressed, the generated DICOM images have a similar content to the model dataset images. This is accomplished by using several models for each modality. The DICOM Export

Plugin starts by selecting a model that matches the modality of the exported study, then replaces the content of several fields to create a unique image instance. Finally, the actual DICOM image file is written to the output storage medium.

The DICOM export plugin can operate in two modes regarding its behaviour with pixel data. It can use real images, collected from the model dataset. Despite producing genuine data, this mode may raise some privacy considerations related to the use of real images. Alternatively, the plugin can generate pixel data with "noise", i.e., random byte streams or black images, according to users' requirements.

The generation of a dataset of several years' radiology practice will require several terabytes of storage space to accommodate it. This can be a serious problem in many experimental environments. Despite seeming a major drawback, pixel data is rarely needed in many technology trials. Moreover, laboratory trials requiring access to image pixel data also usually need to have datasets that are manually curated and controlled. Taking this into consideration, the proposed system also allows the reduction of pixel data size, making the generation of huge datasets possible. The strategy to deal with this issue involves stripping the pixel data from the generated images. In experiments performed in the Dicoogle environment, these DICOM objects can be completed with pixel data when requested because the developed system includes a third module (plugin) that generates it on the fly.

A.5 Results

This section demonstrates the feasibility of the proposed system in generating big data repositories to support medical imaging informatics research. In the scope of other research projects, involving clinical partners, Dicoogle was used to collect real data from distinct types of healthcare institutions. One of these datasets, from a medium-sized facility, covers roughly four years of medical imaging practice, enclosing 300 000 studies from four modalities (CT, CR, DX, US). It is a comprehensive dataset that shows the enormous amount of data produced in medical imaging laboratories nowadays.

The distribution of the ultrasound weekly study production is shown in Figure A.3 (blue), as captured by Dicoogle and our statistical analyser. Over the years, roughly 31 000 ultrasound studies were produced. We have chosen this modality because early in 2012 it had a significant increase in the number of studies produced, due to the inclusion of much equipment in the institutional PACS. This allows us to better demonstrate the pertinence of the proposed method.

The system was used to generate an ultrasound modality dataset two years longer than the model dataset. The production distribution of the generated dataset is shown in Figure A.3 (orange). It can be observed that the study production closely follows the model dataset. Moreover, comparing with the previous version of our method Figure A.3 (green), which was

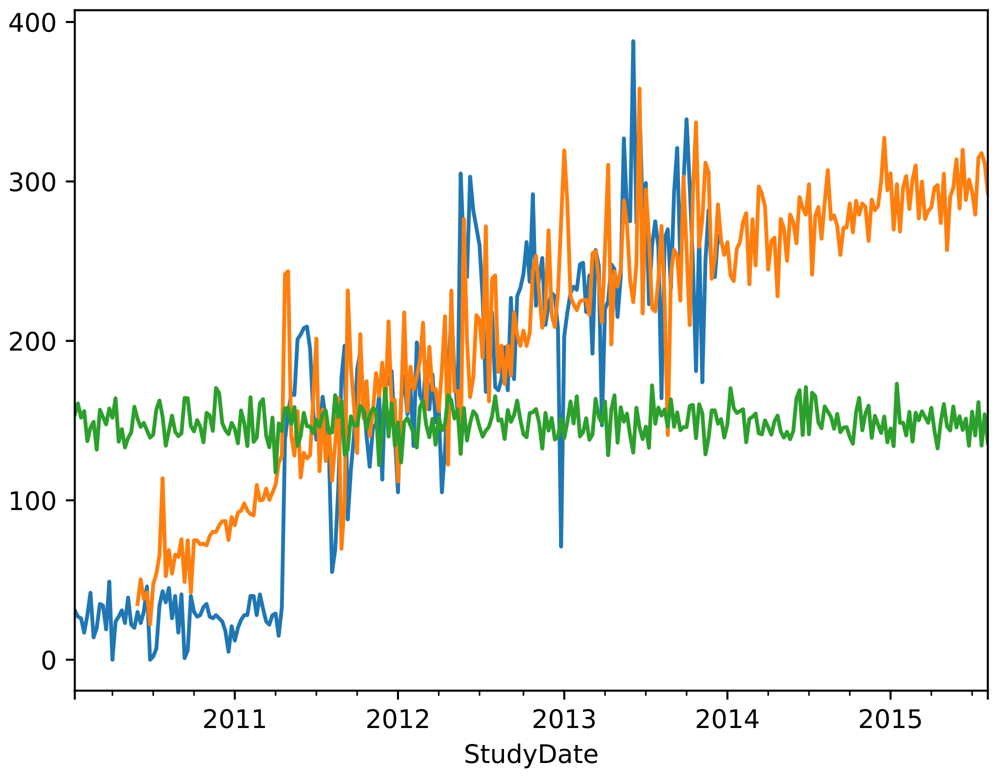


Figure A.3: Distribution of weekly ultrasound studies over time on the different datasets. The Model dataset in (blue), Generated dataset with the presented method (orange); A generated dataset with an alternative statistical model (green).

uniquely based on descriptive statistics rather than on stochastic processes, it models the trend in the production distribution more effectively. This is specially felt when the model is used in forecasting mode, i.e., used to generate a dataset larger than the model.

The same pattern is observed when analysing the statistical markers of both datasets over time, as shown in Table A.1. The Table A.1 shows the cumulative mean (μ), standard deviation (σ), and total number of studies (Σ) in the first N$^{\text{th}}$ year period. It is perceivable that in the first year the model dataset has a relatively small weekly study production, and that it increases steadily over time. This leads to overestimation of the weekly study production by the old model (Generated C). This effect is severely limited by the capacity of the ARMA regression to capture the rising trend of study production (Generated B).

In relation to the volume of data produced, the proposed system is able to provide a clear reduction of storage space for a dataset generated without pixel data. For a dataset consisting of 4 295 069 images, which required 870 955.6 MB on the storage medium, the exported dataset ended up requiring less than 2%, 17.161 MB, of the dataset stored along with the pixel data.

Table A.1: Productivity statistics across the different datasets.

Datasets		1st Year	2nd Year	3rd Year	4th Year	5th Year	Overall
	μ	25.9	70.8	114.9	149.1		149.1
Model	σ	12.2	65.3	86.5	98.6		98.6
	Σ	1 323	7 292	17 923	30 861		30 861
	μ	65.9	112.3	145.8	175.5	196.7	208.1
Generated B	σ	19.4	50.2	61.9	74.1	77.5	79.4
	Σ	2 043	9 324	19 826	32 988	47 214	56 599
	μ	150	150.3	150	149.6	149.6	149.3
Generated C	σ	10.4	9.9	9.6	10	9.7	9.7
	Σ	7 650	15 476	23 400	31 112	38 902	43 582

A.6 Discussion

The proposed methods are well suited to any scenario that requires simulated DICOM metadata. The results in the previous section reinforce that the quality of resulting datasets makes possible their usage to test multiple PACS services in load environments, such as storage, query and retrieve. A major aspect of the proposed system is its immediate utility and the Dicoogle project is a good example of this statement. Dicoogle has been offering search and retrieval services over all metadata contained in the DICOM images of its repository. The development of its indexing engine was subject to a continuous validation process to assess its response to increasing load, and how the various patches developed influenced the system behaviour. This process required strong datasets to validate the solution. For instance, the dataset used to address the reduction in storage space only produced an index with 50 GB. Despite seeming a considerable size, it is not representative of several years of data produced in a medium-sized institution. Using datasets that are not representative of actual real-world institutions, we can never be sure about the long-term performance of new technological approaches until they are actually deployed in a real production environment.

Currently, there are two other projects using datasets generated by our system, aiming to test database technologies in medical imaging Big Data scenarios, namely architectures based on relational and document-oriented models [23].

Finally, our tool falls short in scenarios that require the actual pixel data. In order to overcome this limitation, the system has the option of inserting the pixel data gathered from actual images. This trick is useful in situations such as testing the storage services of the PACS. Nonetheless, in scenarios such as CBIR, this tool is not applicable as it cannot generate meaningful pixel data.

A.7 Conclusions

Nowadays, researchers on medical imaging networks and storage systems face a major issue when validating their contributions. The lack of big datasets that perfectly mimic real institutional scenarios is a serious constraint regarding the quality of their contributions. Many of those proposals have been validated with datasets that are simply not large enough to stress them, i.e. push the system towards its operational limit to determine its robustness, availability, and error handling under heavy load as real laboratories would. Drawing conclusions based on poorly designed trials may induce errors and hinder future research efforts. Moreover, most developers do not have a clear picture of how their solutions will perform in the long-term. Due to this lack of knowledge, unpredicted episodes may take place during the PACS lifetime, compromising the healthcare service provision. The proposed methods address these issues by enabling the creation of artificial datasets with the same characteristics as a model dataset, for instance, a real-world institutional repository. Moreover, these methods enable the creation of larger datasets to be used in long-term and Big Data exploration scenarios. For this purpose, the approach used to reduce the size of the generated datasets seems vital.

A.8 References

[1] Carlos Viana-Ferreira and Carlos Costa, "Challenges of Using Cloud Computing in Medical Imaging", in *Advances in Cloud Computing Research.* Nova Science Publishers, Inc., 2014.

[2] P. Tavakol, F. Labruto, L. Bergstrand, and L. Blomqvist, "Effects of outsourcing magnetic resonance examinations from a public university hospital to a private agent", *Acta Radiologica*, vol. 52, no. 1, pp. 81–85, 1, 2011. DOI: 10.1258/ar.2010.090320.

[3] L. A. B. Silva, C. Costa, and J. L. Oliveira, "A PACS archive architecture supported on cloud services", *International Journal of Computer Assisted Radiology and Surgery*, vol. 7, no. 3, pp. 349–358, 2012. DOI: 10.1007/s11548-011-0625-x.

[4] H. K. Huang, *PACS and Imaging Informatics: Basic Principles and Applications*, 2nd ed. Wiley-Blackwell, 2010, 978 pp.

[5] O. S. Pianykh, *Digital Imaging and Communications in Medicine (DICOM): A Practical Introduction and Survival Guide.* Springer Science & Business Media, 26, 2009, 424 pp.

[6] M. Santos, S. de Francesco, L. A. B. Silva, A. Silva, C. Costa, *et al.*, "Multi vendor DICOM metadata access a multi site hospital approach using Dicoogle", in *2013 8th Iberian Conference on Information Systems and Technologies (CISTI)*, IEEE, 2013, pp. 1–7.

[7] A. Savaris, T. Harder, and A. von Wangenheim, "DCMDSM: A DICOM decomposed storage model", *Journal of the American Medical Informatics Association*, vol. 21, no. 5, pp. 917–924, 1, 2014. DOI: `10.1136/amiajnl-2013-002337`.

[8] Z. Jin and Y. Chen, "Telemedicine in the Cloud Era: Prospects and Challenges", *IEEE Pervasive Computing*, vol. 14, no. 1, pp. 54–61, 2015.

[9] H. Yang, E. Kundakcioglu, J. Li, T. Wu, J. R. Mitchell, *et al.*, "Healthcare Intelligence: Turning Data into Knowledge", *IEEE Intelligent Systems*, vol. 29, no. 3, pp. 54–68, 2014. DOI: `10.1109/MIS.2014.45`.

[10] P. Sernadela, P. Lopes, and J. L. Oliveira, "A knowledge federation architecture for rare disease patient registries and biobanks", *Journal of Information Systems Engineering & Management*, vol. 1, no. 2, pp. 83–90, 22, 2016. DOI: `10.20897/lectito.201615`.

[11] T. C. Prado, D. D. J. de Macedo, M. A. R. Dantas, and A. von Wangenheim, "Optimization of PACS Data Persistency Using Indexed Hierarchical Data", *Journal of Digital Imaging*, 9, 2014. DOI: `10.1007/s10278-013-9665-9`.

[12] C. Costa, C. Ferreira, L. Bastião, L. Ribeiro, A. Silva, *et al.*, "Dicoogle - an Open Source Peer-to-Peer PACS", *Journal of Digital Imaging*, vol. 24, no. 5, pp. 848–856, 2011. DOI: `10.1007/s10278-010-9347-9`.

[13] L. A. B. Silva, L. Beroud, C. Costa, and J. L. Oliveira, "Medical imaging archiving: A comparison between several NoSQL solutions", in *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, IEEE, 2014, pp. 65–68.

[14] T. M. Godinho, C. Viana-Ferreira, L. A. Bastiao Silva, and C. Costa, "A Routing Mechanism for Cloud Outsourcing of Medical Imaging Repositories", *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 1, pp. 367–375, 2016. DOI: `10.1109/JBHI.2014.2361633`.

[15] "Digital Imaging and Communications in Medicine (DICOM)", NEMA, Standard, 2017.

[16] M. Larobina and L. Murino, "Medical Image File Formats", *Journal of Digital Imaging*, vol. 27, no. 2, pp. 200–206, 2014. DOI: `10.1007/s10278-013-9657-9`.

[17] M. Ismail and J. Philbin, "Multi-series DICOM: An Extension of DICOM That Stores a Whole Study in a Single Object", *Journal of Digital Imaging*, vol. 26, no. 4, pp. 691–697, 2013. DOI: `10.1007/s10278-013-9577-8`.

[18] A. Savaris, T. Härder, and A. v Wangenheim, "Evaluating a Row-Store Data Model for Full-Content DICOM Management", in *2014 IEEE 27th International Symposium on Computer-Based Medical Systems*, 2014, pp. 193–198. DOI: `10.1109/CBMS.2014.61`.

[19] S. J. Rascovsky, J. A. Delgado, A. Sanz, V. D. Calvo, and G. Castrillón, "Informatics in Radiology: Use of CouchDB for Document-based Storage of DICOM Objects", *RadioGraphics*, vol. 32, no. 3, pp. 913–927, 1, 2012. DOI: `10.1148/rg.323115049`.

[20] D. D. De Macedo, A. Von Wangenheim, M. Dantas, and H. G. Perantunes, "An architecture for DICOM medical images storage and retrieval adopting distributed file systems", *International Journal of High Performance Systems Architecture*, vol. 2, no. 2, pp. 99–106, 1, 2009. DOI: 10.1504/IJHPSA.2009.032027.

[21] F. Valente, L. A. B. Silva, T. M. Godinho, and C. Costa, "Anatomy of an Extensible Open Source PACS", *Journal of Digital Imaging*, vol. 29, no. 3, pp. 284–296, 2016. DOI: 10.1007/s10278-015-9834-0.

[22] P. J. Brockwell and R. A. Davis, *Time Series: Theory and Methods.* Springer-Verlag, 1987.

[23] A. P. Alves, T. M. Godinho, and C. Costa, "Assessing the relational database model for optimization of content discovery services in medical imaging repositories", IEEE, 2016, pp. 1–6. DOI: 10.1109/HealthCom.2016.7749484.