WEB.INF.UFPR.BR/SBCAS2021

SBCAS 2021

DE 15 A 18 DE JUNHO DE 2021

LIVRO DE MINICURSOS

REALIZAÇÃO

SBC
Sociedade Brasileira de Computação

ORGANIZAÇÃO

UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

PUCPR
GRUPO MARISTA

APOIO

sbis
sociedade brasileira de informática em saúde

Unimed
Paraná

Organizadores:
ANDREY RICARDO PIMENTEL
LUCAS FERRARI DE OLIVEIRA

MINICURSOS DO XXI SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO APLICADA À SAÚDE

CURITIBA, 15 A 18 DE JUNHO DE 2021.

Porto Alegre
Sociedade Brasileira de Computação – SBC
2021

Dados Internacionais de Catalogação na Publicação (CIP)

S612 Simpósio Brasileiro de Computação Aplicada à Saúde (21. : 2021 : Curitiba, PR)
Minicursos do XXI Simpósio Brasileiro de Computação Aplicada à Saúde [recurso eletrônico] – organizadores: Andrey Ricardo Pimentel, Lucas Ferrari de Oliveira – Porto Alegre : SBC, 2021.

ISBN 978-65-87003-70-2

1. Computação. 2. Saúde. I. Título.

CDU 004:61

Catalogação elaborada por Francine Conde Cabral
CRB-10/2606

APRESENTAÇÃO

O Simpósio Brasileiro de Computação Aplicada à Saúde (SBCAS) é um dos principais fóruns de divulgação científica e de encontro de pesquisadores das áreas de computação e saúde. O SBCAS é o principal evento nacional organizado pela Comissão Especial de Computação Aplicada à Saúde (CE-CAS) da Sociedade Brasileira de Computação (SBC). O SBCAS teve sua origem no Workshop de Informática Médica (WIM), evento consolidado e reconhecido nacionalmente, com 17 edições realizadas regularmente entre 2001 e 2017. A partir de 2018, WIM passou a se chamar SBCAS. A mudança de nome e de "status" do evento teve como principal objetivo o estabelecimento de uma marca atualizada com o campo de pesquisa cada vez mais interdisciplinar, bem como intensificar a presença da SBC na comunidade atuante na área, proporcionando maior impacto e visibilidade. O SBCAS 2019 foi realizado pela primeira vez como um evento independente, no Instituto de Computação da Universidade Federal Fluminense, na cidade de Niterói, Rio de Janeiro, em junho de 2019. A 21ª edição do evento, SBCAS 2021, foi realizada no período de 15 a 18 de junho de 2021 na cidade de Curitiba, Paraná, com uma programação diversificada e com marcante caráter interdisciplinar, com duração de quatro dias com a apresentação de minicursos, sessões técnicas com apresentações de artigos aceitos no evento, concurso de teses e dissertações, palestras e uma sessão sobre avanços tecnológicos.

MENSAGEM DA COORDENAÇÃO DOS MINICURSOS

O Livro de Minicursos do 21º Simpósio Brasileiro de Computação Aplicada à Saúde (SBCAS 2021) traz os textos dos minicursos selecionados e apresentados nesta edição do evento, incluindo quatro minicursos aceitos. Os minicursos são uma oportunidade de atualizar os conhecimentos da comunidade com novos temas relacionados à saúde e à computação, de uma forma didática e de amplo acesso ao público. Os minicursos foram selecionados por meio de um processo de revisão por pares do tipo "blind", de um total de 6 propostas de minicursos submetidas, o que implicou uma taxa de aceitação de 66%. O comitê de avaliação contou com 12 pesquisadores da área, que realizaram uma criteriosa seleção. Agradecemos a todos os membros do comitê, por suas valiosas contribuições para os trabalhos e dedicação no processo de revisão. Dos quatro trabalhos selecionados, um declinou da publicação, portanto temos somente 3 capítulos publicados. Os três trabalhos apresentam alta qualidade, resultado do esforço dos autores, que dedicaram muitas horas para a produção do conteúdo escrito e da apresentação. Somos gratos a eles também pelo esforço aplicado e pelo sucesso do trabalho. Agradecemos também ao Comitê de Organização do SBCAS 2021, em especial aos Coordenadores Gerais, Prof. Lucas Ferrari de Oliveira (UFPR) e Prof. Rodrigo Veras (UFPI), por todo o suporte durante a seleção e elaboração deste livro. Finalmente, desejamos que todos os participantes dos minicursos do SBCAS 2021 aproveitem as apresentações, que este ano serão realizadas remotamente!

Andrey Ricardo Pimentel (UFPR)

Lucas Ferrari de Oliveira (UFPR)

Coordenador e Editor do livro de Minicursos do SBCAS 2020

Organização do SBCAS 2021

Coordenadores Gerais
Lucas Ferrari de Oliveira (UFPR) – Coordenador Geral
Rodrigo de Melo Souza Veras (UFPI) – Coordenador de Programa

Coordenador da Trilha de Minicursos
Andrey Ricardo Pimentel (UFPR)

CE-CAS - Comissão Especial de Computação Aplicada à Saúde

Lucas Ferrari de Oliveira, UFPR (Coordenador)
Paulo E. Ambrósio, UESC (Vice-coordenador)
Artur Ziviani, LNCC (*in memoriam)*
Cristiano André da Costa, UNISINOS
Débora Christina Muchaluat Saade, UFF
José Raphael Bokehi, UFF/ SBEB
Marcia Ito, FATEC-SP/SBIS
Marcos Ennes Barreto, UFBA
Natalia Castro Fernandes, UFF
Rodrigo de Melo Souza Veras, UFPI
Rodrigo Rafael Villarreal Goulart, FEEVALE

Sumário

Capítulo 1

Precision Radiomic Biomarkers: a brief introduction, some technical development, and several clinical applications

José Raniery Ferreira Junior

Abstract

Computer tools have been part of the clinical routine on a daily basis for radiological image interpretation. However, they are limited to quantifying basic information about the lesions that a medical examination may present, like a nodule size or mass volume. Radiomic biomarkers emerged in this context to address this problem by quantifying the images massively and characterizing them comprehensively to allow the precision phenotyping needed in the current personalized medicine era. Thus, this book chapter introduces some robust radiomic biomarkers identified in the past few years for different pathological imaging patterns of diseases from two critical human systems, i.e., respiratory and musculoskeletal. The text initiates with the primary motivation for radiomic biomarker development, discovery, and validation. The following sections present a quick background with the basic theory for the remainder of the manuscript. Finally, the chapter approaches the state-of-the-art radiomic precision biomarkers for three different diseases and modalities of medical images: covid-19 in chest radiography, lung neoplasms in computed tomography, and spondyloarthritis in magnetic resonance imaging.

1.1. Introduction

Medical imaging is an essential component to evaluate patients with a suspicion of several diseases. Beyond the importance, radiological images are still mostly dependent on the diagnostic level of specialists, and rapid imaging interpretation is not always possible as it relies on the availability of expert physicians [Liang and Zheng 2019, Azevedo-Marques and Ferreira Junior 2021]. Furthermore, the imaging appearance of different diseases can

overlap (Figure 1.1) and mimic other body complications due to the low contrast that the examination may have [Santos et al. 2019, Ferreira Junior et al. 2020a].

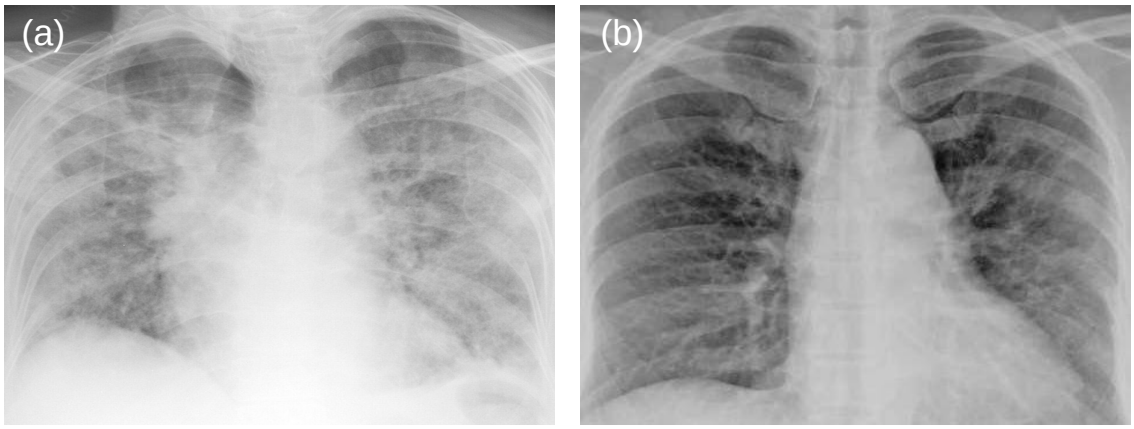

Figura 1.1. Overlapping radiographic characteristics of SARS (a) and covid-19 (b). Both diseases commonly present bilateral ground-glass opacities, as shown in the figures. Source: Author.
SARS, severe acute respiratory syndrome.
covid-19, coronavirus disease 2019.

In the face of these challenges, it is vital to include computer-based tools to aid specialists in evaluating diseases early, as they can improve the accuracy and consistency of medical image interpretation through computational support used as reference [Ferreira Junior et al. 2017]. Radiomics has grown in this context as a quantitative imaging approach that associates computer-extracted image data with clinical endpoints (Figure 1.2). This radiomic association allows a more comprehensive characterization of underlying phenotypes, ultimately increasing the power of decision support models for precision medicine [Tomaszewski and Gillies 2021]. In light of recent advances in target therapies, the need for an inexpensive and easily obtainable imaging approach for phenotyping diseases has become imperative, and radiomics can provide it as is a noninvasive, reproducible tool [Sacconi et al. 2017].

Radiomics emerged as a quantitative approach for personalized medicine to develop medical imaging biomarkers and predictive models for clinical decision support [Santos et al. 2019]. At first, it was based on the extraction of hand-designed features from previously segmented images with the potential to be associated with clinical outcomes (Figure 1.3). During this initial moment, traditional statistical methods, such as test-retest, stability, univariate inference tests, and multivariate regression models, assessed the radiomics associations [Aerts et al. 2014, Sacconi et al. 2017, Ferreira Junior et al. 2020b]. Then, radiomics studies with multivariate analyses started focusing on artificial intelligence and machine learning methods to improve the predictive power of the imaging biomarkers. Some of the machine-learning models studied were standard Bayesian methods, artificial neural networks, and decision trees [Leger et al. 2017, Dawes et al. 2017, Kickingereder et al. 2016].

The main applications of radiomics during those early phases were on the oncological domain for predicting genomic mutations and tumor staging, for instance, [Kickingereder et al. 2016, Aerts et al. 2014, Sacconi et al. 2017, Ferreira Junior et al. 2021d].

Figura 1.2. Possibilities of clinical endpoints to be investigated through radiomics and medical imaging quantification. Source: Author.

However, literature has shown radiomics could be expanded to analyze other diseases from a wide range of medical specialties. Such as cardiology to predict patient survival, myocardial tissue alterations, and right ventricular failure; and neurology for the characterization of attention deficit hyperactivity disorder [Dawes et al. 2017, Baeßler et al. 2018, Sun et al. 2017].

More recently, research groups have started using deep-learning networks to potentially improve the associative performance [Kermany et al. 2018, Liang and Zheng 2019, Lu et al. 2019]. Those networks are mainly based on deep convolutional neural networks (CNNs) as they are capable of processing high dimensional arrays, like medical images [LeCun et al. 2015, Santos et al. 2019]. However, as those methods characterize the images implicitly (creating the so-called black-box), they will not be further described in this book chapter.

In this manuscript, some theory and practice on robust quantitative biomarker discovery will be introduced to highlight the valuable role that radiomics has in precision

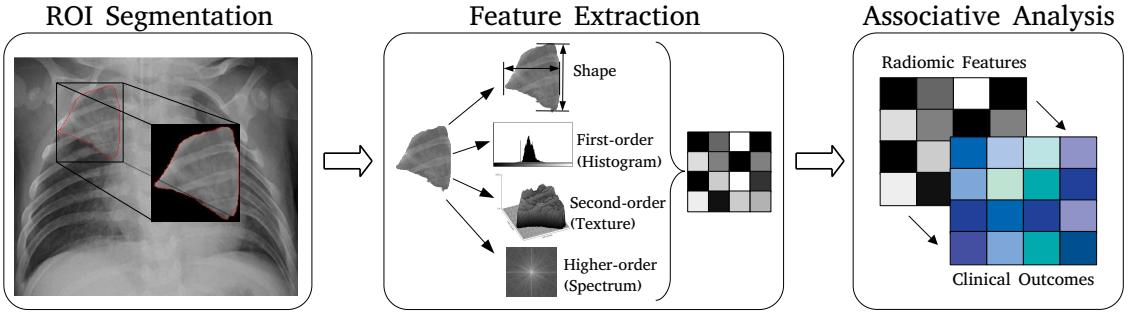

Figura 1.3. Standard investigative pipeline for radiomics. Source: Author. ROI, region of interest.

medicine. To do that, the state-of-the-art of quantitative precision biomarkers in three different pathological imaging domains will be approached: (I) covid-19 in chest radiography (XR) images, (II) lung neoplasms in computed tomography (CT) scans, and (III) spondyloarthritis in magnetic resonance imaging (MRI).

1.2. Background

1.2.1. Digital radiology images

A standard medical image from the radiology workflow is digitally an n-dimensional discretized matrix where each index identifies a point of interest in the human body. Radiography, for instance, is a two-dimensional matrix with indexes of rows and columns, and the value of each image point (*i.e.,* pixel) corresponds to a gray intensity (Figure 1.4). For example, the grayscale varies 256 levels considering an 8-bit image, from 0 indicating black, 1 to 254 indicating different shades of gray (from dark to light tones), until 255 indicating complete white.

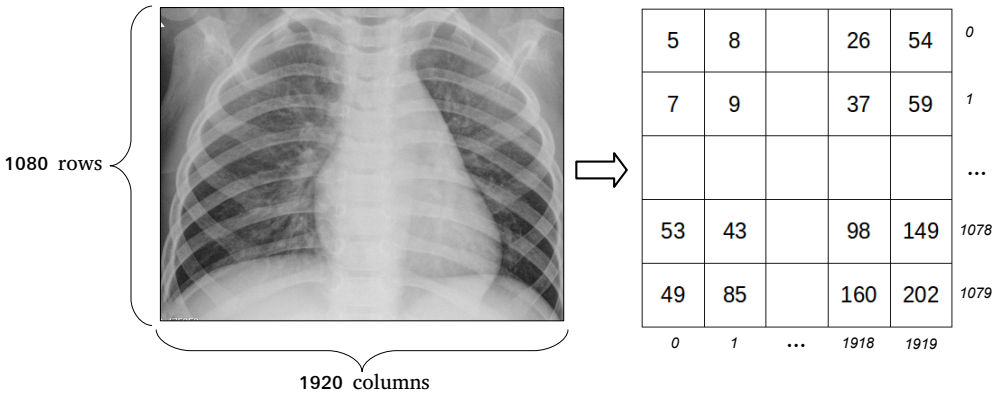

Figura 1.4. Digital XR image representation. Source: Author.

Radiological images as acquired have uniform spacing between pixels and are commonly obtained in the form of a volume of parallel scans, as in the example of CT and MRI. The resulted acquired image is projected in an anatomical plane: coronal, axial, or sagittal (Figure 1.5). The physical distance between the images from a volume is called slice thickness, and each point of the image volume becomes a voxel.

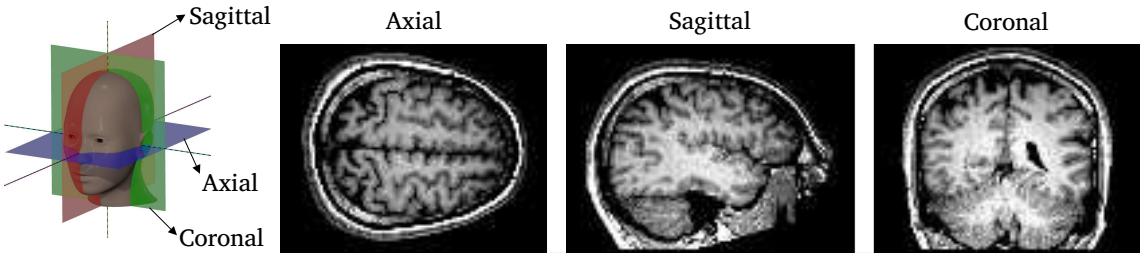

Figura 1.5. Anatomical planes for medical image projection after acquisition. Source: Author licensed under CC BY-SA 4.0. Adapted from Brainscandude / CC BY-SA 3.0 and Slashme / CC BY-SA 4.0.

1.2.2. Image segmentation

Segmentation is a key process for medical image processing, leading to major complications in pattern recognition when presenting low efficiency. Image segmentation aims to identify areas correlated with each other in order to separate a volume of interest (VOI) [Ferreira Junior et al. 2021b].

Methods of medical image segmentation generally use basic properties of gray level discontinuity or similarity [Gonzalez and Woods 2007]. The former approach separates regions according to rough changes on gray intensities, like in the boundaries. The latter strategy is based on comparing gray levels, such as thresholding, the most fundamental technique for segmentation [Ferreira Junior 2019].

Image thresholding is defined as an operation in which an input image is transformed to an output image, as follows:

$$g(i,j) = \begin{cases} f(i,j) & \text{if} \quad T_1 \leq f(i,j) \leq T_2, \\ 0 \quad \text{if} & f(i,j) < T_1 \quad \text{or} \quad f(i,j) > T_2, \end{cases} \tag{1}$$

where f is the input image, g is the output image, i, j are points of the image matrix, and T_n are threshold values. The threshold is the gray level determined as reference for the comparison between the voxels' intensities (Figure 1.6). In the case of Equation 1.1, the values T_1 and T_2 correspond to a threshold interval, where voxels with intensity out of that gray level range correspond to the image background, and hence, are excluded; while voxels with intensity within the range correspond to the VOI, thus they remained in the segmented output image.

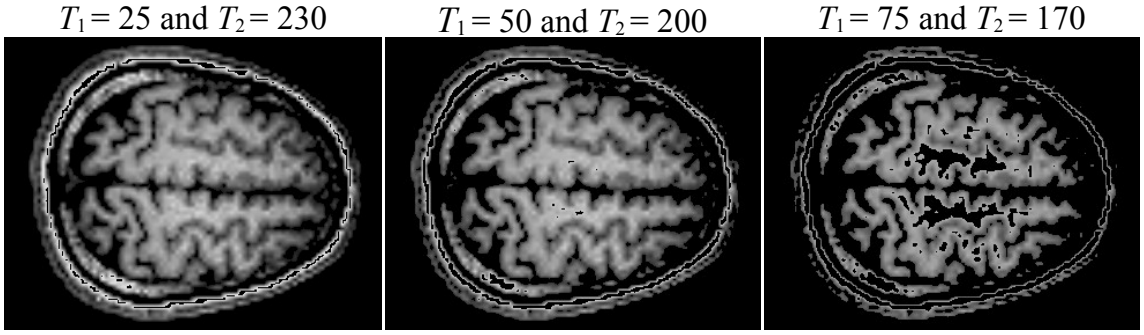

Figura 1.6. Examples of output generated after applying different gray level thresholds for brain image segmentation. Source: Author.

Thresholding is appropriate for very distinct gray intensities between the VOI and background. However, it fails when there is a low contrast in the image, resulting in similar gray levels within the VOI [Ferreira Junior et al. 2020a]. In these particular cases, it is necessary to address this problem using region-oriented approaches, such as region growing methods and convolutional neural networks [Ferreira Junior et al. 2021d, Ferreira Junior et al. 2021c]. But those techniques will not be further described in this book chapter due to space constraints.

1.2.3. Radiomic feature extraction

The process to extract radiomic features is basically the calculation of quantitative measures in a segmented image to represent the visual content. Feature extraction algorithms perform mathematical procedures to characterize the VOIs differently, such as histograms, matrices, transforms, geometrical elements, among others [Tomaszewski and Gillies 2021]. Traditionally, these hand-engineered features describe four primary imaging levels: (I) first-order, characterizing the distribution of intensities in a gray-level histogram; (II) second-order, describing voxel spatial relationships in a gray-level matrix; (III) higher-order, depicting the image spectrum in the frequency domain; and (IV) shape, characterizing geometric and size-related components of a VOI [Ferreira Junior et al. 2020b].

Features from the first imaging order

First-order features are calculated from a gray-level histogram representing a particular VOI (Figure 1.7). These intensity features describe the distribution of voxels' values from the VOI individually without concern for spatial relationships [Santos et al. 2019].

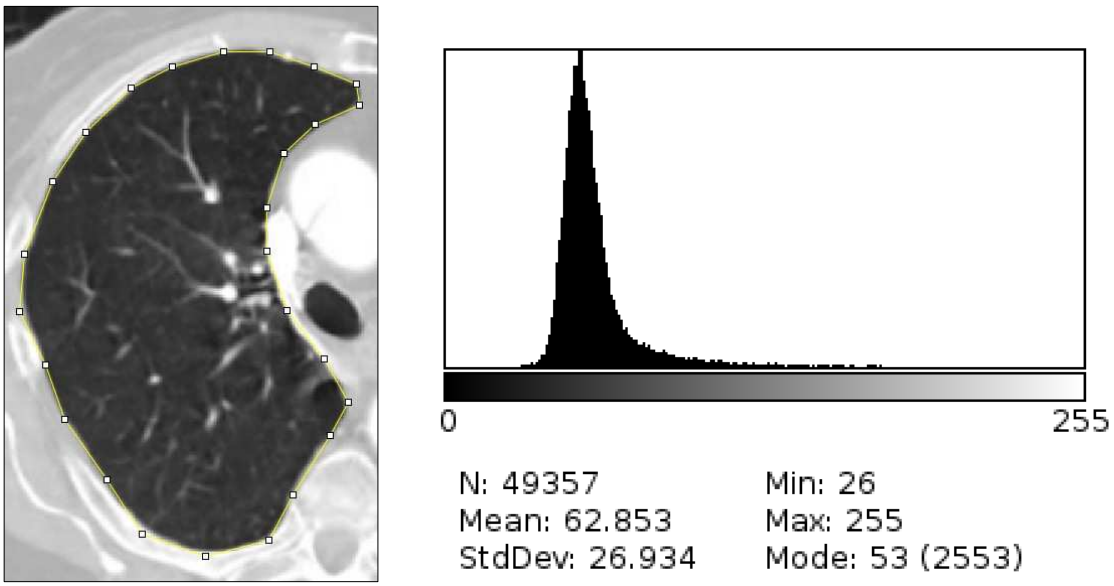

Figura 1.7. Example of an intensity histogram from a cropped CT image of the lung with 256 gray levels. Source: Author.

The first-order features are regular histogram-computed statistical measures, like *mean*, *variance*, *standard deviation*, *coefficient of variation*, *energy*, *entropy*, *mean absolute deviation*, *root mean squared*, *skewness*, and *kurtosis*. Equations 1.2-1.11 list some first-order measures:

$$mean\ (\mu) = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{2}$$

$$variance\ (v) = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)^2, \tag{3}$$

$$\textit{standard deviation } (\sigma) = \sqrt{v}, \tag{4}$$

$$\textit{coefficient of variation} = \frac{\sigma}{\mu}, \tag{5}$$

$$\textit{energy (or uniformity)} = \sum_{i=1}^{n} x_i^2, \tag{6}$$

$$\textit{entropy} = -\sum_{i=1}^{n} x_i \log_2 x_i, \tag{7}$$

$$\textit{mean absolute deviation} = \frac{1}{n}\sum_{i=1}^{n} |x_i - \mu|, \tag{8}$$

$$\textit{root mean squared} = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}, \tag{9}$$

$$\textit{skewness} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^3}{\sigma^3}, \tag{10}$$

$$\textit{kurtosis} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^4}{\sigma^4}, \tag{11}$$

where x is a histogram of n gray levels.

Those statistical measures are important because they can quantify the image density, like the *mean*. The *standard deviation*, *variance*, *coefficient of variation*, and *mean absolute deviation* are dispersion measures and they describe how much the gray levels differ from the mean intensity. While the *energy* and the *root mean squared* are measures of magnitude, the *entropy* can characterize the randomness and variations present in the image, measuring the average amount of information required to encode the image values. *Skewness* and *kurtosis* are the histogram central moments and they mainly quantify the asymmetry and sharpness degrees, respectively, around the mean [Zwanenburg et al. 2020, Aerts et al. 2014].

Features from the second imaging order

Histogram-based features normally are not enough to fully describe the VOI. As previously stated, those first-order measures do not consider spatial relationships, which are crucial for distinguishing similar textural images, such as the ones in Figure 1.1.

On the other hand, second-order features can describe intrinsic characteristics of the image texture using advanced statistical mechanisms. Although there is no consensus about the formal definition of image texture, it can be defined as the repetition of patterns of even minor variations in a VOI [Ferreira Junior 2019]. In medical imaging, that is essential due to the tiniest details at a molecular level that the image may portray.

Several approaches can analyze the images and recognize different texture patterns. Probably the most known in literature is the so-called gray-level co-occurrence matrix (GLCM) [Haralick et al. 1973]. The co-occurrence matrices obtain from the VOI the occurrence probability of pairs of voxel intensities i, j given a distance d and an orientation θ for the x, y dimensions and an orientation ϕ for the z dimension. The GLCM computes in two-dimensional arrays the occurrence of individual intensity pairs in slice by slice manner. The final matrix is the summation of appearances in all image slices. The GLCM tridimensional version directly computes the occurrence of individual intensity pairs in the entire VOI [Ferreira et al. 2017].

The texture features provided by the GLCM are computed by calculating the following measures (Equations 1.12-1.34) on the produced matrix:

$$autocorrelation = \sum_{i,j=1}^{n} ijM_{ij}, \tag{12}$$

$$cluster\ prominence = \sum_{i,j=1}^{n} (i+j-\mu_x-\mu_y)^4 M_{ij}, \tag{13}$$

$$cluster\ shade = \sum_{i,j=1}^{n} (i+j-\mu_x-\mu_y)^3 M_{ij}, \tag{14}$$

$$contrast = \sum_{i,j=1}^{n} (i-j)^2 M_{ij}, \tag{15}$$

$$correlation = \frac{1}{\sigma_x \sigma_y} \sum_{i,j=1}^{n} (i-\mu_x)(j-\mu_y) M_{ij}, \tag{16}$$

$$difference\ average\ (or\ dissimilarity) = \sum_{k=0}^{n-1} kM_{i-j,k}, \tag{17}$$

$$difference\ entropy = -\sum_{k=0}^{n-1} M_{i-j,k} log_2 M_{i-j,k}, \tag{18}$$

$$difference\ variance = \sum_{k=0}^{n-1} (k-dissimilarity)^2 M_{i-j,k}, \tag{19}$$

$$energy\ (or\ uniformity\ or\ angular\ second\ moment) = \sum_{i,j=1}^{n} M_{ij}^2, \tag{20}$$

$$IMC_1 = \frac{-\sum_{i,j=1}^{n} M_{ij} log_2 M_{ij} + \sum_{i,j=1}^{n} M_{ij} log_2 (M_i M_j)}{-\sum_{i=1}^{n} M_i log_2 M_i}, \tag{21}$$

$$IMC_2 = \sqrt{1 - exp(-2(-\sum_{i,j=1}^{n} M_i M_j log_2 M_i M_j + \sum_{i,j=1}^{n} M_{ij} log_2 M_{ij}))}, \tag{22}$$

$$inverse\ difference = \sum_{i,j=1}^{n} \frac{M_{ij}}{1+|i-j|}, \tag{23}$$

$$inverse\ difference\ normalized = \sum_{i,j=1}^{n} \frac{M_{ij}}{1+\frac{|i-j|}{n}}, \tag{24}$$

$$inverse\ difference\ moment = \sum_{i,j=1}^{n} \frac{M_{ij}}{1+(i-j)^2}, \tag{25}$$

$$inverse\ difference\ moment\ normalized = \sum_{i,j=1}^{n} \frac{M_{ij}}{1+\frac{(i-j)^2}{n^2}}, \tag{26}$$

$$inverse\ variance = 2\sum_{i,j=1}^{n} \frac{M_{ij}}{(i-j)^2}, i \neq j, \tag{27}$$

$$joint\ average = \sum_{i,j=1}^{n} iM_{ij}, \tag{28}$$

$$(joint\ or\ sum)\ entropy = -\sum_{i,j=1}^{n} M_{ij} log_2 M_{ij}, \tag{29}$$

$$joint\ maximum\ (or\ max\ probability) = max(M_{ij}), \tag{30}$$

$$(joint)\ variance\ (or\ sum\ of\ squares) = \sum_{i,j=1}^{n} (i - joint\ average)^2 M_{ij}, \tag{31}$$

$$sum\ average = 2 \times joint\ average, \tag{32}$$

$$sum\ entropy = -\sum_{k=2}^{2n} M_{i+j,k} log_2 M_{i+j,k}, \tag{33}$$

$$sum\ variance\ (or\ cluster\ tendency) = \sum_{k=2}^{2n} (k - sum\ average)^2 M_{i+j,k}, \tag{34}$$

where i, j is the gray-level pair, M_{ij} is an element of the GLCM (here computed from a two-dimensional array), n is the number of different gray levels, μ_x, μ_y are mean values in the x, y directions, σ_x, σ_y are standard deviation values in the x, y directions, and IMC is the *informational measure of correlation*.

Inverse difference and *inverse difference moment* were also referred simply as *homogeneity*, but this nomenclature is deprecated. Those two features and their normalized versions measure the local homogeneity of the image. *Energy* analogously is a measure of uniform imaging patterns, in which a greater value implies that there are more intensity pairs in the image that neighbor each other at higher frequencies. On the other hand, features of *entropy* are measures of the randomness/variability in neighborhood intensity values. The *difference variance* measures heterogeneous patterns that place higher weights on differing intensity pairs that deviate more from the mean. *Contrast* is also a measure of the local intensity variation, where a larger value correlates with a greater disparity in intensity values among neighboring voxels. The IMC metrics quantify the texture complexity evaluating the correlation between the probability distributions through mutual information. *Autocorrelation* is a measure of the magnitude of the texture

fineness and coarseness. The features of *cluster prominence*, *shade*, and *tendency* emulate the human perception, and they measure the skewness of the GLCM, where higher values imply greater asymmetry about the mean. The *correlation* shows the linear dependency of gray levels to their respective voxels in a region. The *difference average* and *sum average* measure the relationship between occurrences of pairs. But the former relates similar and differing intensities, and the latter relates lower with higher intensity values [Zwanenburg et al. 2020, Van Griethuysen et al. 2017, Yip et al. 2017, Phillips et al. 2017].

A second alternative for texture analysis is based on the gray-level run-length matrix (GLRLM) [Tang 1998, Galloway 1975]. The run-length matrix tracks the frequencies of sequencies with different lengths of the same gray level at a predetermined orientation. The GLRLM is interesting for image characterization in that fine textures have more short sequencies with similar gray levels, and rough textures have more long sequencies with different intensities [Davnall et al. 2012]. Analogously to the GLCM, the following attributes (Equations 1.35-1.50) can be calculated to form the GLRLM texture features:

$$\textit{short run emphasis} = \frac{\sum_{i}^{n}\sum_{j}^{l}\frac{p(i,j|\theta)}{j^2}}{\sum_{i}^{n}\sum_{j}^{l}p(i,j|\theta)}, \tag{35}$$

$$\textit{long run emphasis} = \frac{\sum_{i}^{n}\sum_{j}^{l}j^2 p(i,j|\theta)}{\sum_{i}^{n}\sum_{j}^{l}p(i,j|\theta)}, \tag{36}$$

$$\textit{gray level non-uniformity} = \frac{\sum_{i}^{n}(\sum_{j}^{l}p(i,j|\theta))^2}{\sum_{i}^{n}\sum_{j}^{l}p(i,j|\theta)}, \tag{37}$$

$$\textit{gray level non-uniformity normalized} = \frac{\sum_{i}^{n}(\sum_{j}^{l}p(i,j|\theta))^2}{(\sum_{i}^{n}\sum_{j}^{l}p(i,j|\theta))^2}, \tag{38}$$

$$\textit{run length non-uniformity} = \frac{\sum_{j}^{l}(\sum_{i}^{n}p(i,j|\theta))^2}{\sum_{i}^{n}\sum_{j}^{l}p(i,j|\theta)}, \tag{39}$$

$$\textit{run length non-uniformity normalized} = \frac{\sum_{j}^{l}(\sum_{i}^{n}p(i,j|\theta))^2}{(\sum_{i}^{n}\sum_{j}^{l}p(i,j|\theta))^2}, \tag{40}$$

$$run\ percentage = \frac{\sum_{i}^{n}\sum_{j}^{l} p(i,j|\theta)}{v}, \tag{41}$$

$$gray\ level\ variance = \sum_{i}^{n}\sum_{j}^{l} q(i,j|\theta)(i - \sum_{i}^{n}\sum_{j}^{l} q(i,j|\theta)i)^2, \tag{42}$$

$$run\ variance = \sum_{i}^{n}\sum_{j}^{l} q(i,j|\theta)(j - \sum_{i}^{n}\sum_{j}^{l} q(i,j|\theta)j)^2, \tag{43}$$

$$run\ entropy = -\sum_{i}^{n}\sum_{j}^{l} q(i,j|\theta) log_2(q(i,j|\theta)), \tag{44}$$

$$low\ gray\ level\ run\ emphasis = \frac{\sum_{i}^{n}\sum_{j}^{l} \frac{p(i,j|\theta)}{i^2}}{\sum_{i}^{n}\sum_{j}^{l} p(i,j|\theta)}, \tag{45}$$

$$high\ gray\ level\ run\ emphasis = \frac{\sum_{i}^{n}\sum_{j}^{l} i^2 p(i,j|\theta)}{\sum_{i}^{n}\sum_{j}^{l} p(i,j|\theta)}, \tag{46}$$

$$short\ run\ low\ gray\ level\ emphasis = \frac{\sum_{i}^{n}\sum_{j}^{l} \frac{p(i,j|\theta)}{i^2 j^2}}{\sum_{i}^{n}\sum_{j}^{l} p(i,j|\theta)}, \tag{47}$$

$$short\ run\ high\ gray\ level\ emphasis = \frac{\sum_{i}^{n}\sum_{j}^{l} \frac{p(i,j|\theta) i^2}{j^2}}{\sum_{i}^{n}\sum_{j}^{l} p(i,j|\theta)}, \tag{48}$$

$$long\ run\ low\ gray\ level\ emphasis = \frac{\sum_{i}^{n}\sum_{j}^{l} \frac{p(i,j|\theta) j^2}{i^2}}{\sum_{i}^{n}\sum_{j}^{l} p(i,j|\theta)}, \tag{49}$$

$$long\ run\ high\ gray\ level\ emphasis = \frac{\sum_{i}^{n}\sum_{j}^{l} p(i,j|\theta) i^2 j^2}{\sum_{i}^{n}\sum_{j}^{l} p(i,j|\theta)}, \tag{50}$$

where $p(i,j|\theta)$ is an element of the GLRLM, $q(i,j|\theta)$ is an element of the normalized GLRLM, i is a gray intensity, j is the frequency of i, θ is the orientation, and n, l, v are the number of gray levels, run lengths, and voxels, respectively, in the image.

Short and *long run emphasis* measure run distributions in the image. The *short run emphasis* is a measure in which a greater value is indicative of shorter run lengths and more fine textural patterns. The *long run emphasis* is a measure in which a greater value is indicative of extended run lengths and more coarse structural textures. *Gray level non-uniformity* and its normalized version measure the similarity of gray-level intensities in the image, where a lower value correlates with a greater similarity in intensities. The *run length non-uniformity* and its normalized version measure the similarity of run lengths throughout the image, with a lower value indicating more homogeneity among run lengths in the image. *Run percentage* measures the coarseness of the texture, and its higher values indicate a larger portion of the ROI consists of short runs and a more fine texture. The features of *gray level* and *run variances* measure the variance in gray level intensity for the runs and the variance in runs for the run lengths, respectively. *Run entropy* is a measure of uncertainty/randomness of run lengths and gray levels, in which a higher value indicates more heterogeneity in the texture patterns. The features of *low* and *high gray level run emphasis* measure the distribution of gray levels with a higher value indicating a greater concentration of low and high gray-level values, respectively, in the image. Finally, the features of *short/long run low/high gray level emphasis* measure the joint distribution of run lengths with the gray-level values [Van Griethuysen et al. 2017, Phillips et al. 2017, Davnall et al. 2012].

A more recent approach for texture characterization, in comparison to the ones in [Haralick et al. 1973, Galloway 1975, Tamura et al. 1978], is the gray level size zone matrix (GLSZM) [Thibault et al. 2013]. A gray level zone is defined as the number of connected voxels (with distance 1 with each other) that share the same gray level intensity. Contrary to GLCM and GLRLM, the GLSZM is rotation independent, with only one matrix calculated for all directions in the ROI [Van Griethuysen et al. 2017]. The following attributes (Equations 1.51-1.66) can be calculated in the GLSZM to form the features:

$$\textit{small area emphasis} = \frac{\sum_{i}^{n}\sum_{j}^{l}\frac{p(i,j)}{j^2}}{\sum_{i}^{n}\sum_{j}^{l}p(i,j)}, \tag{51}$$

$$\textit{large area emphasis} = \frac{\sum_{i}^{n}\sum_{j}^{l}j^2 p(i,j)}{\sum_{i}^{n}\sum_{j}^{l}p(i,j)}, \tag{52}$$

$$\textit{gray level non-uniformity} = \frac{\sum_{i}^{n}(\sum_{j}^{l}p(i,j))^2}{\sum_{i}^{n}\sum_{j}^{l}p(i,j)}, \tag{53}$$

$$\textit{gray level non-uniformity normalized} = \frac{\sum_{i}^{n}(\sum_{j}^{l}p(i,j))^2}{(\sum_{i}^{n}\sum_{j}^{l}p(i,j))^2}, \tag{54}$$

$$\textit{size-zone non-uniformity} = \frac{\sum_{j}^{l}(\sum_{i}^{n} p(i,j))^2}{\sum_{i}^{n}\sum_{j}^{l} p(i,j)}, \tag{55}$$

$$\textit{size-zone non-uniformity normalized} = \frac{\sum_{j}^{l}(\sum_{i}^{n} p(i,j))^2}{(\sum_{i}^{n}\sum_{j}^{l} p(i,j))^2}, \tag{56}$$

$$\textit{zone percentage} = \frac{\sum_{i}^{n}\sum_{j}^{l} p(i,j)}{v}, \tag{57}$$

$$\textit{gray level variance} = \sum_{i}^{n}\sum_{j}^{l} q(i,j)(i - \sum_{i}^{n}\sum_{j}^{l} q(i,j)i)^2, \tag{58}$$

$$\textit{zone variance} = \sum_{i}^{n}\sum_{j}^{l} q(i,j)(j - \sum_{i}^{n}\sum_{j}^{l} q(i,j)j)^2, \tag{59}$$

$$\textit{zone entropy} = -\sum_{i}^{n}\sum_{j}^{l} q(i,j) log_2(q(i,j)), \tag{60}$$

$$\textit{low gray level zone emphasis} = \frac{\sum_{i}^{n}\sum_{j}^{l} \frac{p(i,j)}{i^2}}{\sum_{i}^{n}\sum_{j}^{l} p(i,j)}, \tag{61}$$

$$\textit{high gray level zone emphasis} = \frac{\sum_{i}^{n}\sum_{j}^{l} i^2 p(i,j)}{\sum_{i}^{n}\sum_{j}^{l} p(i,j)}, \tag{62}$$

$$\textit{small area low gray level emphasis} = \frac{\sum_{i}^{n}\sum_{j}^{l} \frac{p(i,j)}{i^2 j^2}}{\sum_{i}^{n}\sum_{j}^{l} p(i,j)}, \tag{63}$$

$$\textit{small area high gray level emphasis} = \frac{\sum_{i}^{n}\sum_{j}^{l} \frac{p(i,j) i^2}{j^2}}{\sum_{i}^{n}\sum_{j}^{l} p(i,j)}, \tag{64}$$

$$\textit{large area low gray level emphasis} = \frac{\sum_{i}^{n}\sum_{j}^{l} \frac{p(i,j) j^2}{i^2}}{\sum_{i}^{n}\sum_{j}^{l} p(i,j)}, \tag{65}$$

$$large\ area\ high\ gray\ level\ emphasis = \frac{\sum_{i}^{n}\sum_{j}^{l} p(i,j)i^2 j^2}{\sum_{i}^{n}\sum_{j}^{l} p(i,j)}, \tag{66}$$

where $p(i,j)$ is an element of the GLSZM, $q(i,j)$ is an element of the normalized GLSZM, i is a gray intensity, j is the zone size of i, and n,l,v are the number of gray levels, zone sizes, and voxels, respectively, in the image.

Analogously to the GLRLM, the *small area emphasis* measures the distribution of small size zones, with a greater value indicative of more fine textures. *Large area emphasis* is a measure of the distribution of large size zones, with a greater value indicative of more coarse textures. All GLSZM *non-uniformity*, *percentage*, *variance*, and *entropy* features have equivalent definitions to the GLRLM measures but considering gray-level zones, not runs. The features of *low* and *high gray level zone emphasis* measure the distribution of gray levels with a higher value indicating a greater proportion of low and high gray-level values, respectively, and size zones in the image. Moreover, the features of *small/large area low/high gray level emphasis* measure the joint distribution of smaller and larger size zones, respectively, with the gray levels [Van Griethuysen et al. 2017].

Another approach vastly used for texture analysis and image characterization is the neighborhood intensity difference matrix (NIDM) [Amadasun and King 1989]. This matrix is actually a one-column array that tracks the average difference between voxel intensities and their neighbors according to a distance. The main advantage of the NIDM is that it examines spatial relationships between three or more voxels at once, not just pairs like the GLCM [Lubner et al. 2017, Yang et al. 2016]. The NIDM attributes are listed in Equations 1.67-1.71:

$$coarseness = \left[\sum_{i}^{n} P(i)S(i)\right]^{-1}, \tag{67}$$

$$contrast = \left[\frac{1}{N_a(N_a-1)}\sum_{i_1,i_2}^{n} P(i_1)P(i_2)(i_1-i_2)^2\right]\left[\frac{1}{N}\sum_{i}^{n} S(i)\right], P(i_1)\neq 0, P(i_2)\neq 0 \tag{68}$$

$$busyness = \frac{\sum_{i}^{n} P(i)S(i)}{\sum_{i_1,i_2}^{n} i_1 P(i_1) - i_2 P(i_2)}, P(i_1)\neq 0, P(i_2)\neq 0, \tag{69}$$

$$complexity = \frac{1}{N_v}\sum_{i_1,i_2}^{n} |i_1-i_2|\frac{P(i_1)S(i_1)+P(i_2)S(i_2)}{P(i_1)+P(i_2)}, P(i_1)\neq 0, P(i_2)\neq 0, \tag{70}$$

$$strength = \frac{\sum_{i_1,i_2}^{n} (P(i_1)+P(i_2))(i_1-i_2)^2}{\sum_{i}^{n} S(i)}, P(i_1)\neq 0, P(i_2)\neq 0, \tag{71}$$

where $P(i)$ is the occurrence probability of the gray level i, $S(i)$ is the NIDM element, n is the number of discretized gray levels of the image, N, N_a, N_v are the number of voxels, discretized gray levels with $a > 0$, and voxels with at least 1 neighbor.

The *coarseness* feature is a measure of the border density, and it averages the difference between the center voxel and its neighborhood, where a higher value indicates a lower spatial change rate and a locally more uniform texture. *Contrast* is a measure of local spatial intensity change, yielding a high value when both the dynamic range and the spatial change rate are increased. The *busyness* measures the ratio of spatial intensity change in a region, where higher values indicate rapid changes of intensity between voxels and the neighborhood. *Complexity* and *strength* are measures of primitive texture components in the region, in which high values indicate the primitives are easily defined, visible, and the image is non-uniform [Van Griethuysen et al. 2017, Phillips et al. 2017, Davnall et al. 2012].

One final matrix-based approach worth mention is the gray level dependence matrix (GLDM) [Sun and Wee 1983]. It quantifies intensity dependencies in an image and describe the overall texture coarseness, in which the number of connected voxels within a predetermined distance are dependent on the center voxel [Zwanenburg et al. 2020, Van Griethuysen et al. 2017]. Analogously to the GLSZM, the GLDM is rotation invariant and has the following measures (Equations 1.72-1.85) calculated to form the texture features:

$$\textit{small (or low) dependence (or number) emphasis} = \frac{\sum_{i}^{n}\sum_{j}^{l}\frac{p(i,j)}{j^2}}{\sum_{i}^{n}\sum_{j}^{l}p(i,j)}, \tag{72}$$

$$\textit{large (or high) dependence (or number) emphasis} = \frac{\sum_{i}^{n}\sum_{j}^{l}j^2 p(i,j)}{\sum_{i}^{n}\sum_{j}^{l}p(i,j)}, \tag{73}$$

$$\textit{gray level non-uniformity} = \frac{\sum_{i}^{n}(\sum_{j}^{l}p(i,j))^2}{\sum_{i}^{n}\sum_{j}^{l}p(i,j)}, \tag{74}$$

$$\textit{dependence non-uniformity} = \frac{\sum_{j}^{l}(\sum_{i}^{n}p(i,j))^2}{\sum_{i}^{n}\sum_{j}^{l}p(i,j)}, \tag{75}$$

$$\textit{dependence non-uniformity normalized} = \frac{\sum_{j}^{l}(\sum_{i}^{n}p(i,j))^2}{(\sum_{i}^{n}\sum_{j}^{l}p(i,j))^2}, \tag{76}$$

$$\textit{gray level variance} = \sum_{i}^{n}\sum_{j}^{l}q(i,j)(i-\sum_{i}^{n}\sum_{j}^{l}q(i,j)i)^2, \tag{77}$$

$$dependence\ variance = \sum_{i}^{n}\sum_{j}^{l} q(i,j)(j - \sum_{i}^{n}\sum_{j}^{l} q(i,j)j)^2, \tag{78}$$

$$dependence\ entropy = -\sum_{i}^{n}\sum_{j}^{l} q(i,j) log_2(q(i,j)), \tag{79}$$

$$low\ gray\ level\ (count)\ emphasis = \frac{\sum_{i}^{n}\sum_{j}^{l} \frac{p(i,j)}{i^2}}{\sum_{i}^{n}\sum_{j}^{l} p(i,j)}, \tag{80}$$

$$high\ gray\ level\ (count)\ emphasis = \frac{\sum_{i}^{n}\sum_{j}^{l} i^2 p(i,j)}{\sum_{i}^{n}\sum_{j}^{l} p(i,j)}, \tag{81}$$

$$small\ (or\ low)\ dependence\ low\ gray\ level\ emphasis = \frac{\sum_{i}^{n}\sum_{j}^{l} \frac{p(i,j)}{i^2 j^2}}{\sum_{i}^{n}\sum_{j}^{l} p(i,j)}, \tag{82}$$

$$small\ (or\ low)\ dependence\ high\ gray\ level\ emphasis = \frac{\sum_{i}^{n}\sum_{j}^{l} \frac{p(i,j) i^2}{j^2}}{\sum_{i}^{n}\sum_{j}^{l} p(i,j)}, \tag{83}$$

$$large\ (or\ high)\ dependence\ low\ gray\ level\ emphasis = \frac{\sum_{i}^{n}\sum_{j}^{l} \frac{p(i,j) j^2}{i^2}}{\sum_{i}^{n}\sum_{j}^{l} p(i,j)}, \tag{84}$$

$$large\ (or\ high)\ dependence\ high\ gray\ level\ emphasis = \frac{\sum_{i}^{n}\sum_{j}^{l} p(i,j) i^2 j^2}{\sum_{i}^{n}\sum_{j}^{l} p(i,j)}, \tag{85}$$

where $p(i,j)$ is an element of the GLDM, $q(i,j)$ is an element of the normalized GLDM, i is the center voxel, j is a neighbouring voxel of i, and n, l are the number of gray levels and dependency sizes, respectively, in the image.

The *small dependence emphasis* is a measure of the distribution of small dependencies with higher values indicative of less homogeneous textures. Opposite to that, the *large dependence emphasis* is a measure of the distribution of large dependencies with a higher value indicative of more homogeneous textures. The GLDM features of *non-uniformity*, *variance*, and *entropy* have equivalent definitions to the GLRLM and GLSZM measures but considering the gray-level dependency, not runs or zones. *Low gray level emphasis* measures the distribution of low intensities with a higher value indicating a greater concentration of low gray-level values in the image. *High gray level*

emphasis measures the distribution of the high intensities with higher values indicating a greater concentration of high gray-level values. *Small/large dependence low/high gray level emphasis* features measure the joint distribution of intensity dependences with lower and higher gray-level values [Van Griethuysen et al. 2017].

Another textural approach was proposed in [Tamura et al. 1978]. Those attributes theorically correspond to human visual perception features, providing better description of the texture [Faleiros et al. 2020]. Tamura's features are *line-likeness*, *regularity*, *roughness*, *contrast*, *granularity*, and *directionality*, but the last three are known to better describe the image texture. However, *granularity* is the most fundamental feature where higher values indicate greater or less repeted textures. *Directionality* is a global property of the image that considers the textural shape and location, without taking into account the orientation [Tenorio et al. 2020].

A complementary strategy for texture characterization and an alternative to the statistics-based ones previously described uses structural geometry and fractal analysis. This approach implies several regions have a standard statistical pattern of roughness and irregularity in different scales [Faleiros et al. 2020, Ferreira Junior 2019]. The fractal measures represent various aspects of the image and provide essential information about spatial heterogeneity [Kolossvary et al. 2018]. Fractal analyses can result in, for instance, the dimension estimate that quantifies how an object fills spaces, the abundance that measures the volume of space filled, and lacunarity that quantifies structural heterogeneity within an object [Davnall et al. 2012].

Features from the higher imaging order

When first and second-order features are inefficient in image characterization, it is necessary to expand the characteristic spectrum by incorporating descriptors beyond the spatial level. Signal processing methods emerged in this context due to their ability to analyze frequency domain properties of the image [Tomaszewski and Gillies 2021, Ferreira Junior and Cardona Cardenas 2021]. The basics to include higher-order characteristics to the feature multi-dimensional space are applying a transform, filter, or wavelet and then calculating standard mathematical measures on the resulting filtered image.

The most traditional approach to obtain the frequency power spectrum uses the Fourier transform, and polar coordinates of each pixel from the transformed image [Schneider et al. 2012]. The Fourier transform analyzes the frequency content disregarding temporal and spatial locations by converting the image in the spatial domain into a set of sine and cosine components [Davnall et al. 2012]. The identification of frequency peaks, prominence, and location reveals information about the periodicity and directions of the image texture [Ferreira Junior 2019].

However, computing the Fourier transform is a time-demanding process, and thus, several implementations have been proposed in the literature over the decades to reduce the computational cost. The most used strategy is the so-called fast Fourier transform (FFT) [Faleiros et al. 2020, Tenorio et al. 2020]. After applying the FFT in an image, first-order measures could be calculated to compose the Fourier-based features (Figure 1.8). The discrete Fourier transform can be formulated as given by the Equation 1.86 [Parker

2011]:

$$F(w) = \sum_{k=0}^{N-1} f(k) e^{\frac{2\pi jwk}{N}} \tag{86}$$

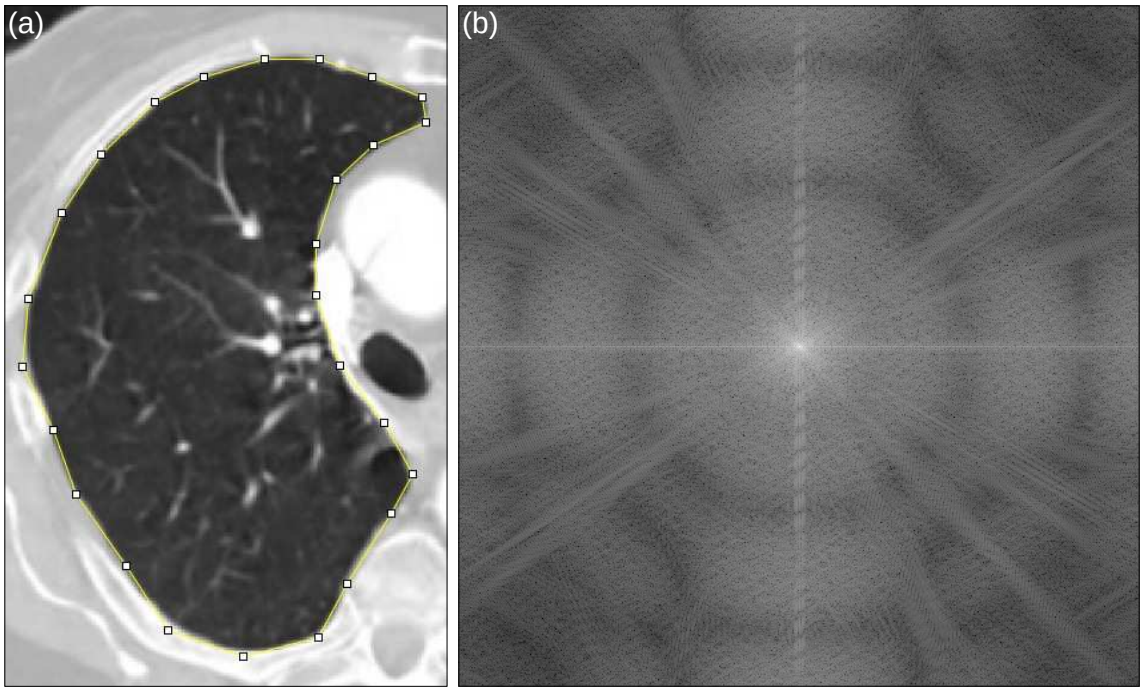

Figura 1.8. Application of the fast Fourier transform in a cropped CT image of the lung: (a) and (b) show the original and resulted images, respectively. Source: Author.

An expanded spectral method for higher-order feature characterization is based on the Gabor transform or filter, which describe textural patterns by sinusoidal functions and allow the spatial, temporal, and frequency representation of the signal [Kolossvary et al. 2018]. A Gabor filter is essentialy a windowed Fourier transform after introducing a Gaussian function, resulting in the acquisition of measures in different time-frequency bands according to a determined scale and orientation [Davnall et al. 2012]. Gabor filter bank is a more powerful approach because it can manipulate local texture parameters of frequency, orientation, excentricity, and symmetry [Ferreira Junior 2019]. The formulation of a Gabor filter in the spatial domain is given by the Equation 1.87 [Bianconi and Fernandez 2007]:

$$\psi(x,y) = \frac{F^2}{\pi\gamma\eta} e^{-F^2[(x'/\gamma)^2+(y'/\eta)^2]} e^{i2\pi F x'}, \tag{87}$$

where x' is $x\cos\theta + y\sin\theta$, y' is $-x\sin\theta + y\cos\theta$, F is the central filter frequency, θ is the angle between the sinusoidal wave direction and the axis x in the spatial domain, γ and η are Gaussian standard deviations. Analogously to the FFT-filtered image, first-order measures could be calculated to compose the Gabor-based features (Figure 1.9).

Although the Gabor filter is an interesting approach, it is limited by the spatial resolution with single window, which is opposite to the wavelet transforms. These methods

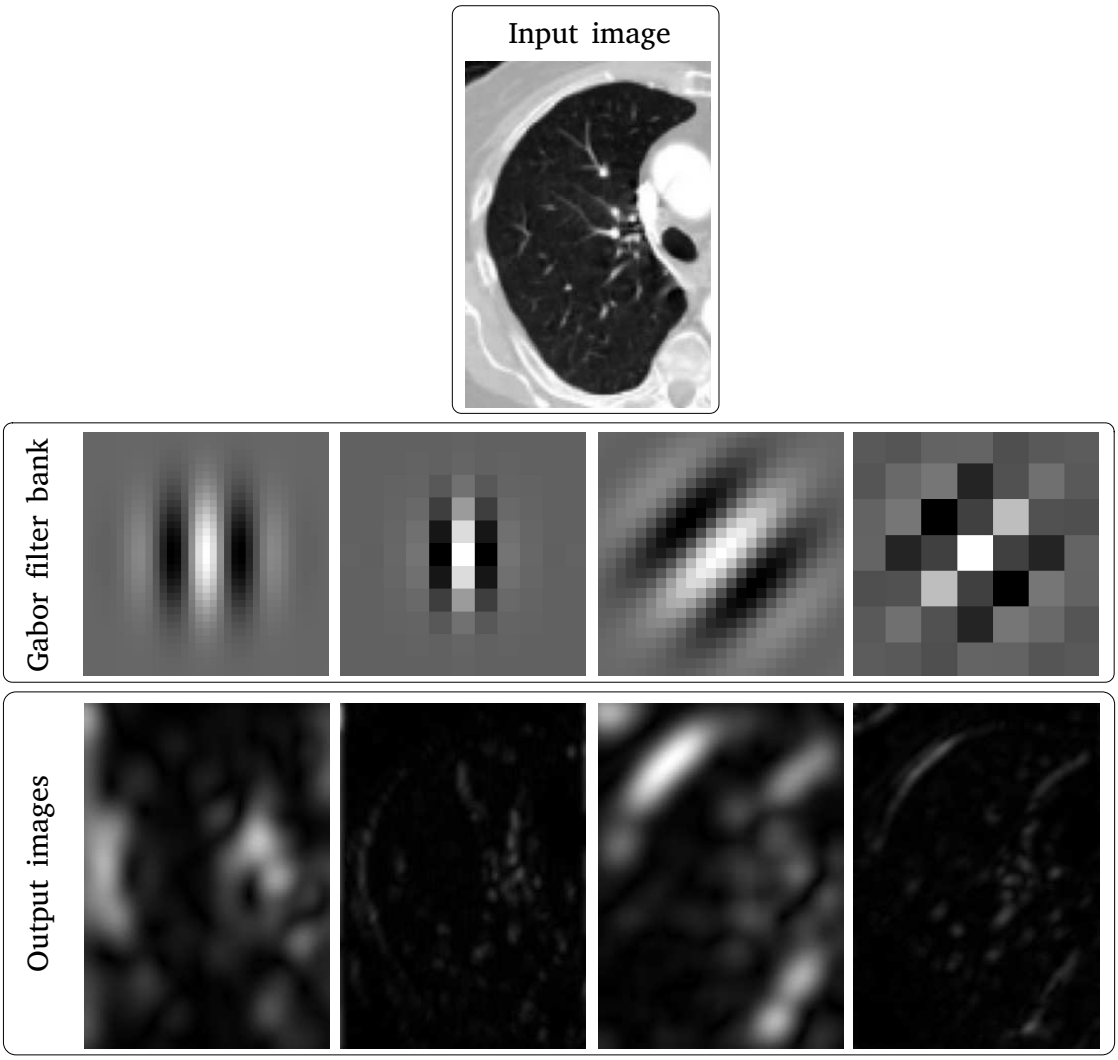

Figura 1.9. Application of a Gabor filter bank in a cropped CT image of the lung. Source: Author.

use multiple scalable and translatable functions in different frequencies, which could represent the texture more comprenhensively than the previous techniques [Kolossvary et al. 2018]. Several wavelets have been proposed in the literature, but they all derived from $\Psi(t)$ in Equation 1.88 [Davnall et al. 2012]:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right), \quad a,b \in R, a \neq 0, \tag{88}$$

where a is a scaling parameter that measures the compression degree and b is the translation parameter that indicates the wavelet time location. Probably the most known wavelet functions are Coiflets, Haar, Daubechies, Symlets, Discrete Meyer, Biorthogonal, and Reverse Biorthogonal (Figure 1.10). They can decompose the image in different frequency domain bands (HH, HL, LH, and LL) and spectrum levels (Figure 1.11) to allow the extensive higher-order feature extraction [Lee et al. 2019].

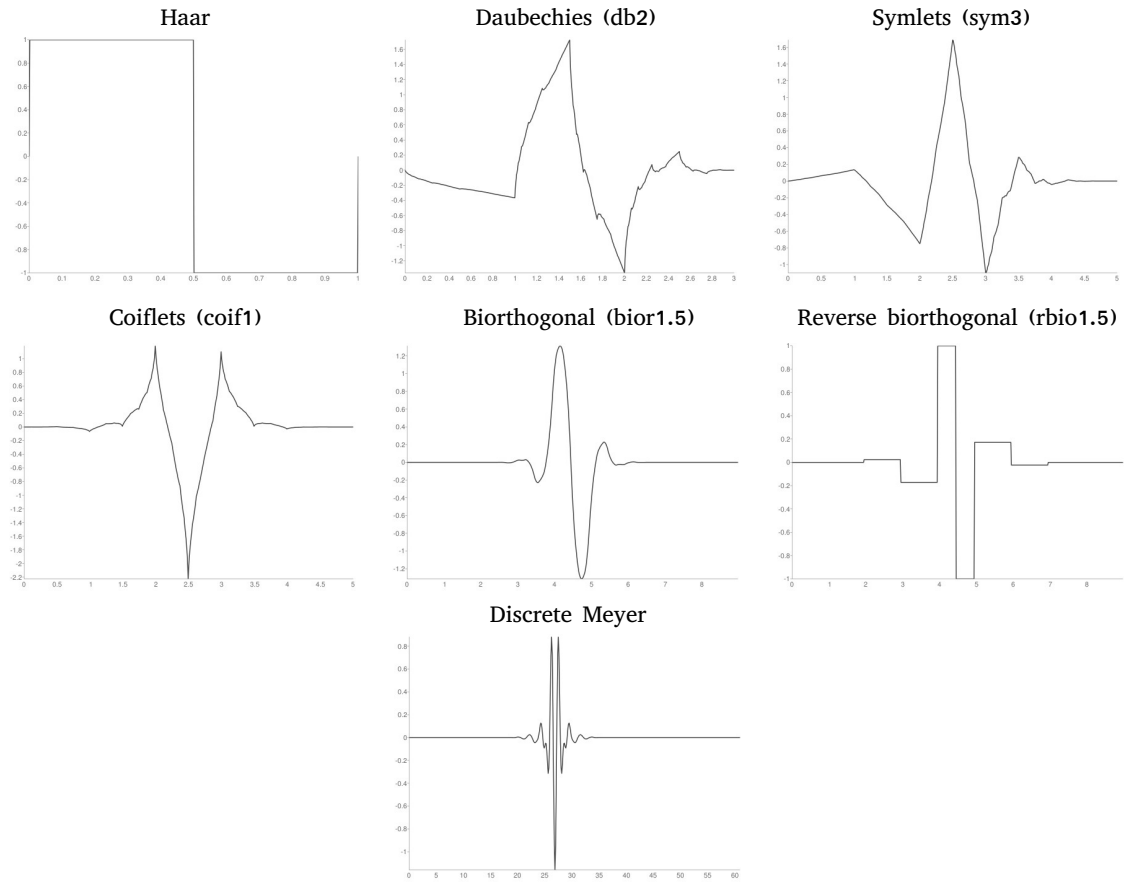

Figura 1.10. Wavelet functions. Source: Adapted from Lee et al. 2019.

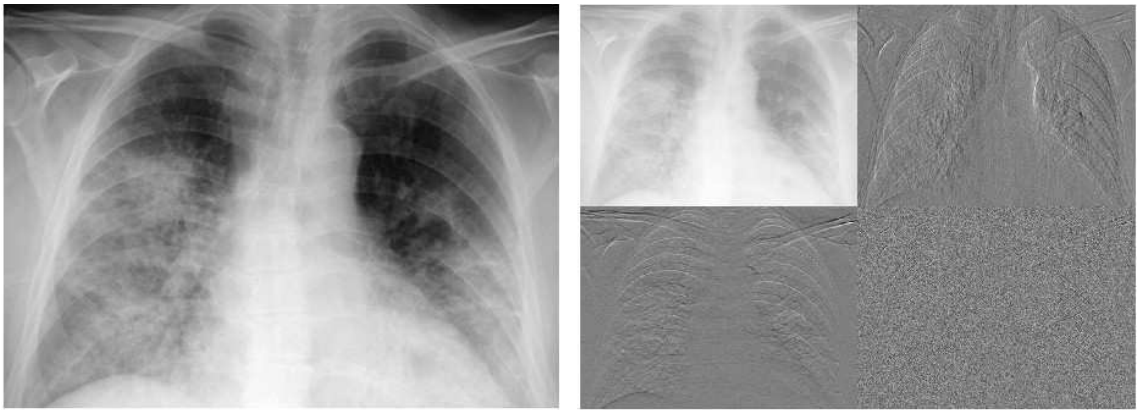

Figura 1.11. Application of a discrete Haar wavelet transform in a chest XR. Source: Author.

Features based on shape

The term *shape* refers to the information inferred from the VOI but could not be represented directly from the intensities, like gray levels and texture. Shape features describe the VOI through geometrical characteristics from the border, contour, curves, among others, which are important in cases where there is a massive difference in the VOI definition of

radiological findings (Figure 1.12) [Echegaray et al. 2015].

Characterizing VOIs quantitatively is a challenging task because it depends directly on the efficiency of image segmentation algorithms [Ferreira Junior et al. 2021b]. Moreover, it is common for image segmentation to have low performances when the VOI has low contrast in the gray levels (Figure 1.13) [Ferreira Junior et al. 2020a], resulting in poor shape-based characterization. Traditionally, shape features are categorized as contour or region-based. The former obtains the features by analyzing the border coordinates from the VOI, and the latter obtains features by considering the region within the VOI. Some of the most used shape features are listed in Equations 1.89-1.95:

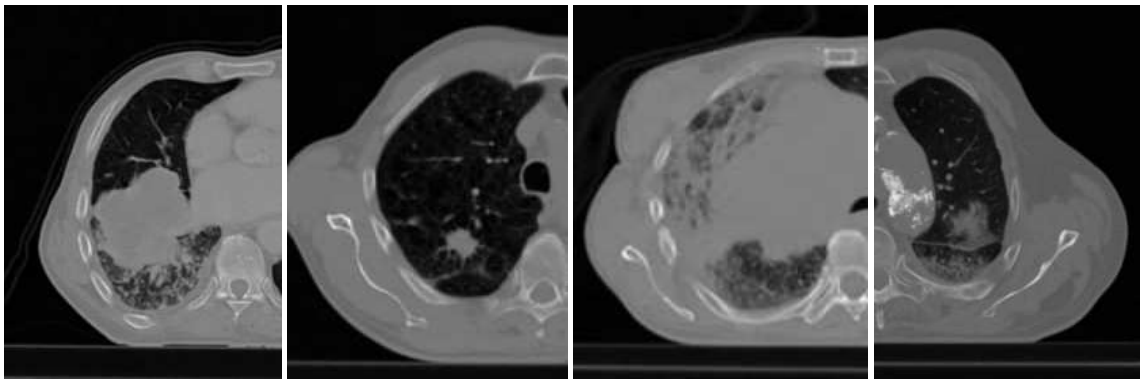

Figura 1.12. Morfological heterogeneity of lung tumors presented in CT images. These are an example in which shape features could be useful in distinguishing radiological abnormalities. Source: Author.

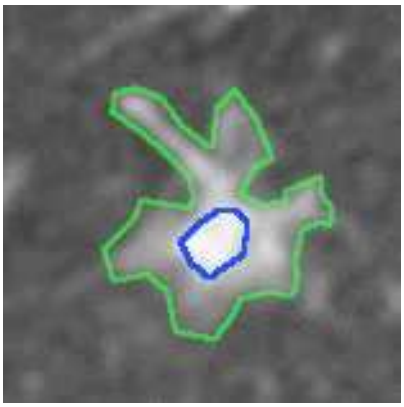

Figura 1.13. Semi-automatic segmentation of solid (blue mark) and subsolid (green mark) components of a lung nodule in CT. This is an example in which image segmentation could hinder shape characterization due to the low contrast of gray levels. Source: Author.

$$surface\ area\ (A) = \sum_{i}^{N} \frac{1}{2} |a_i b_i \times a_i c_i|, \tag{89}$$

$$volume\ (V) = \sum_{k}^{v} V_k, \tag{90}$$

$$density\ (or\ surface\ area\ to\ volume\ ratio) = \frac{A}{V}, \tag{91}$$

$$compactness_1 = \frac{V}{\sqrt{\pi} A^{\frac{2}{3}}}, \tag{92}$$

$$compactness_2 = 36\pi \frac{V^2}{A^3}, \tag{93}$$

$$spherical\ disproportion = \frac{A}{4\pi R^2}, \tag{94}$$

$$sphericity = \frac{\pi^{\frac{1}{3}} (6V)^{\frac{2}{3}}}{A}, \tag{95}$$

where N is the number of connected triangles covering the surface, v is the number of voxels inside the VOI, V_k is the volume of a single voxel, R is the sphere radius defined as $\sqrt[3]{\frac{3V}{4\pi}}$, and a, b, c are the connected triangle vertices.

Features of *maximum diameter*, *volume*, and *surface area* provide size-related information. Measures of *compactness*, *spherical disproportion*, *sphericity*, and *density* quantify how much the VOI is spherical, compact, and rounded [Zhang et al. 2015].

A special case of shape features consider the specific region in the transition from inside the VOI to the outside. The so-called margin sharpness is essential in cases where the disease grows and invade neibouring tissues, such as malignant neoplasms [Levman and Martel 2011]. One of the first margin sharpness features was proposed in [Gilhuijs et al. 1998] for the characterization of breast lesions in MR images. The authors based on the spatial gradient of the boundary, as described in Equations 1.96 and 1.97 [Xu et al. 2012]:

$$average\ of\ margin\ gradient = \max_{i=0,\ \ldots\ ,M-1} \left\{ \frac{\text{mean}_r\ ||\nabla I_m(r)||}{\text{mean}_r I_m(r)} \right\}, \tag{96}$$

$$variance\ of\ margin\ gradient = \frac{\text{var}_r\ ||\nabla I_m(r)||}{[\text{mean}_r I_m(r)]^2}, \tag{97}$$

where $I_m(r)$ is a gray-level intensity and the amplitude of vector r in $I_m(\ \cdot\ )$ is limited to the region surface.

A second approach developed for margin sharpness quantification also used MR images and breast lesions as application [Levman and Martel 2011]. However, the voxels included in that belonged to both the interior and exterior of the VOI. That method resulted in only one feature defined in Equation 1.98 [Xu et al. 2012]:

$$margin\ sharpness = \frac{\overline{I(r_i) - I(N(r_i))}}{d}, \tag{98}$$

where $I(r_i)$ is the gray intensity within the VOI, $N(r_i)$ is the three-dimensional operator that provides a set of voxels that neighbor r_i but are outside the VOI, and d is the normalization term.

A different group of researchers developed a two-part feature of margin sharpness, but opposite to the previous ones, it was tested on CT scans [Xu et al. 2012]. The first feature quantifies the intensity difference between gray levels of the surrounding VOI

and itself through normal line segments across the boundary at fixed intervals around its circumference (Figure 1.14). The second feature quantifies the abruptness of the transition in intensity from the VOI to the surrounding region. The authors affirmed that a sharper border has a more abrupt transition and may have a higher intensity difference outside and inside the lesion. In contrast, a blurred border will have a smoother transition and may have a smaller intensity difference. For each normal line segment I perpendicular to the margin, the problem can be formulated as defined in Equation 1.99:

$$\arg \min_{S,W,x_o,L_o} \sum_{x} \left\lfloor L(x) - L_o - \frac{S}{1+e^{-\frac{x-x_o}{W}}} \right\rfloor, \tag{99}$$

where x is the distance along the normal, x_o is the intersection of the boundary point with the normal, $L(x)$ is the intensity along the normal at x, and L_o is the intensity offset [Xu et al. 2012].

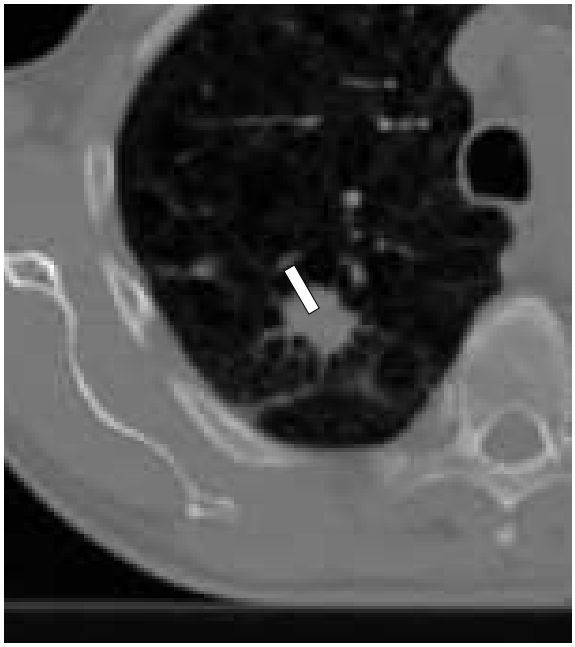

Figura 1.14. Representation of an ortogonal line over the border to serve as reference for margin sharpness characterization. Source: Author.

1.2.4. Performance evaluation

Given the extraction of a set of n features, a performance evaluation is mandatory to assess the efficiency of all radiomic features to become biomarkers. All levels of evidence for biomarkers (Table 1.1) can use the following validation strategies [Lambin 2021].

At least two image cohorts from different sources should comprise the performance evaluation. One is used for discovery, and the other for validation purposes (Figure 1.15(a)). A widely used strategy in medicine to validate predictive models uses three data sets: one for discovery, one for testing, and one for external validation (Figure 1.15(b)). But it is not the most appropriate approach because not all samples are tested, and thus, not all patient heterogeneity evaluated [Keek et al. 2018, Larue et al. 2017]. Cross-validation, which separates the samples in m folds, from which $m-1$ is for

Tabela 1.1. Levels of evidence for biomarkers. Source: Adapted from Lambin 2021.

Level	Study Design	Definition
I	Prospective	Marker as primary objective
II	Prospective	Marker as secondary objective
III	Retrospective	Multivariate analysis with outcomes
IV	Retrospective	Univariate analysis with outcomes
V	Retrospective	Correlation with other marker

discovery and 1 is for testing (Figure 1.15(c)), can mitigate this limitation. The cross-validation is performed until all samples of the initial fold are tested. It aims to improve the chances of generalizing the solution and decreasing the risk of overfitting. When m corresponds to the total number of samples, the process is then called leave-one-out cross-validation.

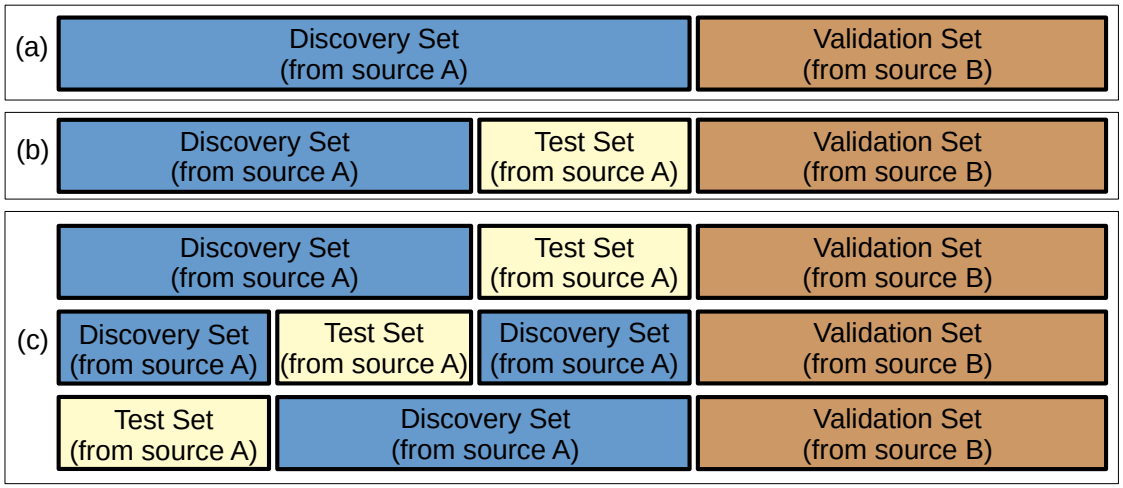

Figura 1.15. Representation of data setting for proper performance evaluation. Source: Author.

The terms *validation* and *testing* are interchangeable, depending on the scientific background. For instance, in medicine it is common to have the *validation* set used independently. But computing and engineering nomenclature considers the *testing* set as the independent one, and the *validation* set is used for fine-tuning.

Two of the most relevant methods used in performance evaluation are the receiver operating characteristic (ROC) curve and the confusion matrix. The ROC curve is defined as a graph of a resulting test where the axis y presents the sensitivity or true positive rate of the test, and the axis x shows the false positive rate defined as 1 - specificity (Figure 1.16). The area under the ROC curve (AUC) measures the final performance from the method. The AUC ranges 0-1 where $0 \leq \text{AUC} \leq 0.50$ indicate bad performance, $0.50 < \text{AUC} \leq 0.70$ indicate low performance, $0.70 < \text{AUC} \leq 0.85$ indicate moderate performance, and $0.85 < \text{AUC} \leq 1$ indicate a high performance [Carvalho et al. 2018, Dou et al. 2018, Yip et al. 2017].

The metrics sensitivity (Equation 1.100) and specificity (Equation 1.101) refer to the proportion of positive and negative cases, respectively, classified correctly as such.

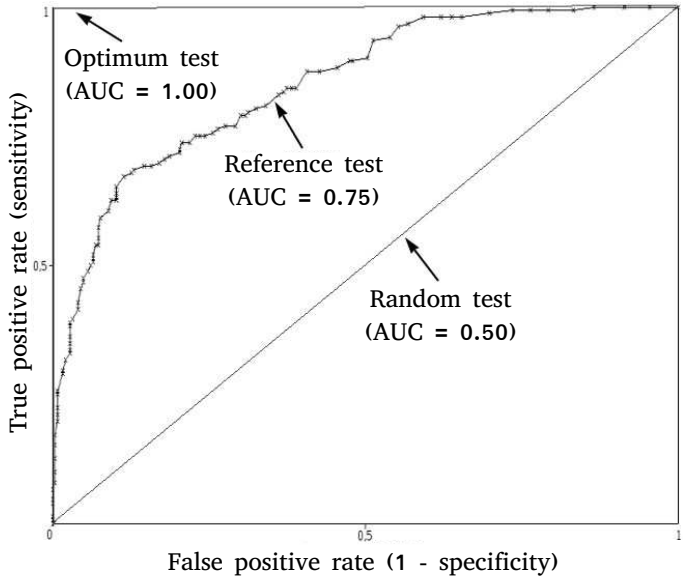

Figura 1.16. ROC curves used for performance evaluation. Source: Author.

Both measures can be calculated from the confusion matrix, which presents the number of samples considered as true-positive (TP), true-negative (TN), false-positive (FP), and false-negative (FN), as in Table 1.2 [Ferreira Junior et al. 2018].

$$sensitivity = \frac{TP}{TP+FN}, \tag{100}$$

$$specificity = \frac{TN}{TN+FP}. \tag{101}$$

Tabela 1.2. Confusion matrix construction. TP, TN, FP, and FN are the number of true positives, true negatives, false positives, and false negatives, respectively. Source: Author.

		Test	
		Positive	Negative
Real	Positive	TP	FN
	Negative	FP	TN

A particular method called survival or time-to-event analysis is used to evaluate the performance of a prediction procedure that constitutes a variable relating to the time between the beginning of a study and the occurrence of an event [Ferreira Junior 2019]. The event is not necessarily attached to death, as the term survival indicates, but to any clinical outcome, such as recurrence, hospitalization, among others [George et al. 2014]. Historically, overall survival is considered the temporal outcome of most importance to the clinical practice due to its objectivity and unambiguity [Cheema and Burkes 2013]. In this particular case, the event of interest is the patient's death by any nature. In survival analyses, the patients who did not reach the event ou had lost follow-up are censored as the exact survival time is unknown [Ferreira Junior et al. 2021d].

The Kaplan-Meier estimation is a statistical method used for time-to-event evaluation [Kaplan and Meier 1958]. Once defined the event of interest and reasons for censorship, it is possible to build probability curves that describe the event occurrence rate along the time, allowing the comparison of different patient groups (Figure 1.17) [Ferreira Junior 2019]. The survival probability $S(t)$ in a time instant t is computationaly given by Equation 1.102:

$$S(t) = S(t-1) \times \frac{N_s}{N_r}, \tag{102}$$

where $S(t-1)$ is the likelihood in a previous instant to t, N_r is the number of patients at risk in the study excluding the censored cases in the instant $t-1$, and N_s is the number of patients that survived until the instant t.

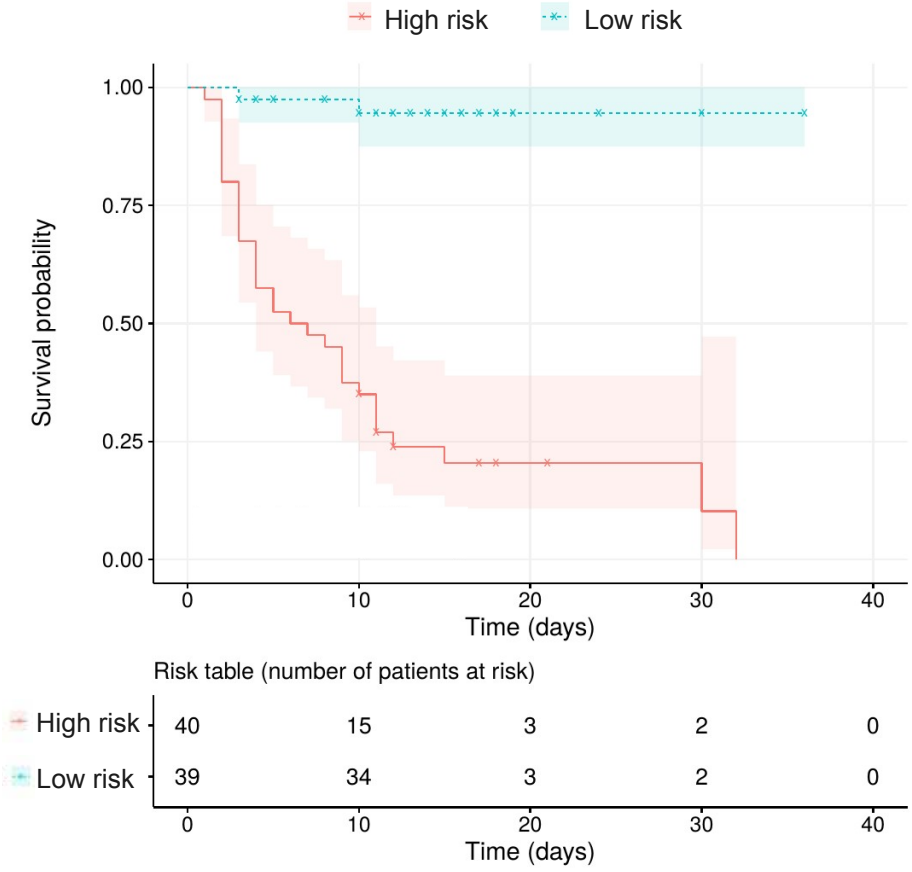

Figura 1.17. Example of Kaplan-Meier curves with confidence intervals and risk table for different patient groups (a high and a low risk). Each marked point in the curves represents a censored patient. Source: Author.

1.3. State-of-the-art

1.3.1. Covid-19 in chest XR

The covid-19 pandemic is the current major public health issue in the world that has caused over 3 million deaths in less than 18 months [European Centre for Disease Prevention and Control 2021]. The disease is mainly characterized by an inflammatory process in

the lungs and airways. The earliest possible diagnosis of covid-19 is imperative for the patient's isolation to prevent virus spread and for rapid treatment decisions to improve the patient's prognosis [Greenhalgh et al. 2020, Osman et al. 2021, Chiu et al. 2020].

A few alternatives exist as a screening tool for appropriate triage of suspected and high-risk patients in several low-incoming healthcare centers with a high demand for suspicious cases. One of those alternatives is chest XR, due to its high availability and portability [Ferreira Junior et al. 2021b, Wehbe et al. 2020, Ferreira Junior et al. 2021a]. However, it is widely known that XR has limited performance in the current clinical environment compared to CT, especially to assess covid-19 pneumonia in early disease stages with very subtle characteristics [Zhang et al. 2021, Wong et al. 2020, Rajaraman et al. 2020]. But radiomics can support the XR assessment of covid-19 and improve the identification of disease-related lesions [Chiu et al. 2020, Zhang et al. 2021, Wehbe et al. 2020, Rajaraman et al. 2020].

Ferreira Junior et al. identified 51 radiomic biomarkers in chest XR; most of them were higher-order features extracted after the Coiflet wavelet, for covid-19 [Ferreira Junior et al. 2021b]. The GLDM feature of *small dependence low gray level emphasis* (Equation 1.82) after the Coiflet transform obtained the highest performance, yielding an AUC of 0.87, sensitivity of 0.85, and specificity of 0.67 ($p < 0.001$). The authors found that higher values of the biomarker correlated with covid-19 patterns, even in cases with XR negative to discrete ground-glass opacities (Figure 1.18).

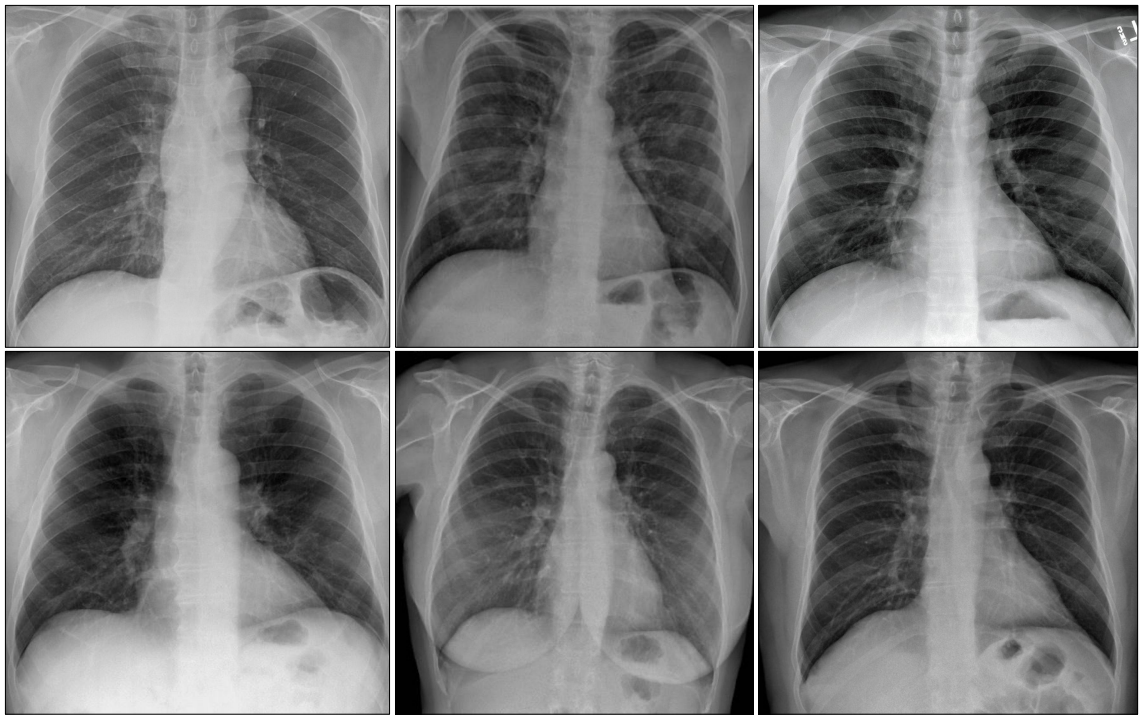

Figura 1.18. Examples of XR images from mild covid-19 patients. Source: Author.

The authors also identified other radiomic biomarkers with prognostic value to predict overall and deterioration-free survival. The first-order feature of *mean absolute deviation* (Equation 1.9) after the Coiflet transform yielded a significant difference in

overall survival rates from the stratified risk groups of covid-19 patients ($p < 0.05$). High values of the biomarker identified low-risk patients with a mean survival time of 25 days. Low values of the feature stratified patients with a higher risk of death, presenting a mean survival time of 13 days.

Furthermore, the biomarker *size zone non-uniformity* (Equation 1.55) of GLSZM after a square filter yielded the highest significant difference in Kaplan-Meier curves to predict short-term deterioration of the patient clinical status ($p < 0.05$). High values of the feature identified lower-risk patients, while low values stratified patients with a higher risk of rapid worsening, presenting a hazard ratio of 3.198 (95% confidence interval – CI: 1.145–8.932). Figure 1.19 presents the survival probabilities of the radiomic biomarkers.

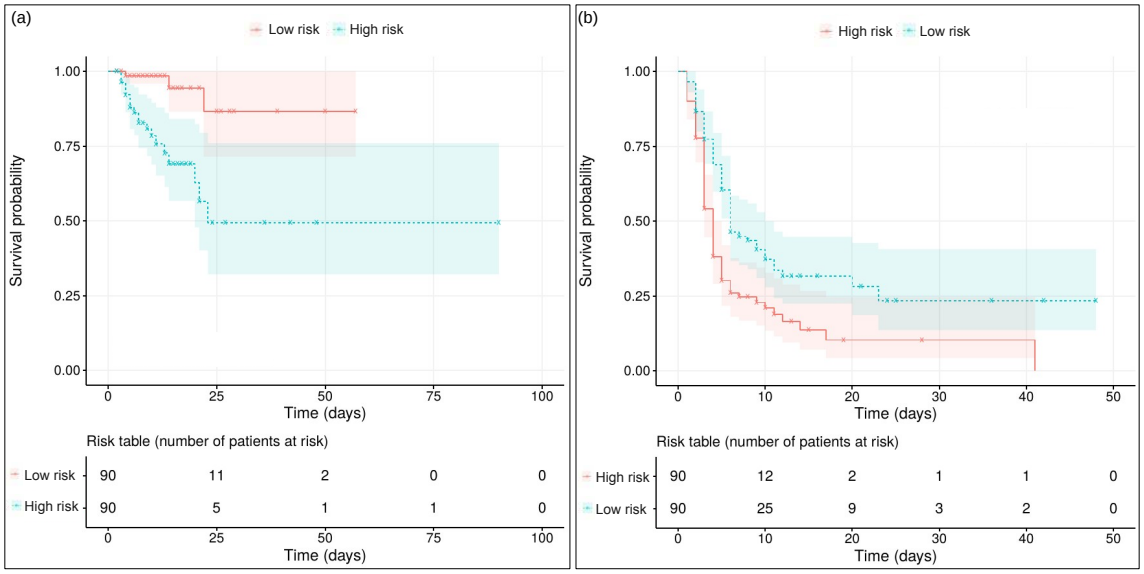

Figura 1.19. Kaplan-Meier curves and risk tables from prognostic radiomic biomarkers to predict death (a) and short-term deterioration (b). Source: Author.

Those XR features significantly associated with covid-19 outcomes could stratify the patient's short-term risk even without comorbidity conditions, at hospitalization, or any early stage of health care. These features could indicate the patient's rapid worsening before the clinical condition deteriorates when intensive therapy is more likely to have greater benefit [Ferreira Junior et al. 2021b]. Moreover, chest XR-based biomarkers may have a significant impact on supporting daily clinical decisions due to the accessibility of radiographic scanners.

The covid-19 biomarkers highlighted the challenge of visually recognizing intricate image characteristics, as they were discovered after algebraic filtering. In the early stages of the disease, small patchy shadows and interstitial changes emerge in the lungs when visible [Li et al. 2020]. But the wavelet transforms enabled to capture higher textural heterogeneity from covid-19 and not from other pneumonia etiologies on radiography [Ferreira Junior et al. 2021b].

Future investigations in this subject include correlating imaging with other clinical outcomes, assessing pathologic aggressiveness, and primarily identifying genetic traits associated with disease progression and therapeutical resistance. Studies have shown that

patients with critical or severe covid-19 have high expression of tyrosine kinase 2 and CC-chemokine receptor 2, respectively [Pairo-Castineira et al. 2021]. Those findings could bring significant clinical benefits with the existing drugs, and radiogenomics could play a key role in decoding covid-19 genotypes [Ferreira Junior and Cardona Cardenas 2021].

1.3.2. Lung neoplasms in computed tomography

Lung cancer is the most lethal malignant neoplasm globally, with a 5-year overall survival rate of about 15%. Moreover, the prognosis of patients with lung neoplasms is still poor and varies markedly according to tumor staging at diagnosis [Ferreira Junior et al. 2020b]. Several clinical aspects may influence therapy decision-making, like staging, histology, genomics, CT imaging, among others. Radiomic biomarkers emerged in this context to increase the clinical applicability of the previous computer-aided detection tools focused on the automated diagnosis of pulmonary nodules [Santos et al. 2019].

Ferreira Junior, Oliveira, and Azevedo-Marques proposed a novel margin sharpness characterization to classify the malignancy likelihood of pulmonary nodules in CT [Ferreira Junior et al. 2018]. The method extracts statistical properties across the nodule boundary in each CT scan (Figure 1.20) [Ferreira Junior and Oliveira 2015]. The authors discovered that the combination of the margin sharpness *amplitude* and the GLCM *inverse difference moment* (Equation 1.25) has potential to identify the malignancy of the lung lesions ($p < 0.05$). To do that, the authors used a standard decision tree to perform the multivariate classification [Ferreira Junior et al. 2018]. Calheiros et al. investigated further margin sharpness features by including perinodular zone characterization [Calheiros et al. 2021]. The developed method increased the performance when integrating parenchyma-originated features of the histogram *skewness* (Equation 1.10), and the GLCM *prominence*, *shade*, *correlation*, *energy*, and *entropy* (Equations 1.13, 1.14, 1.16, 1.20, and 1.29, respectively).

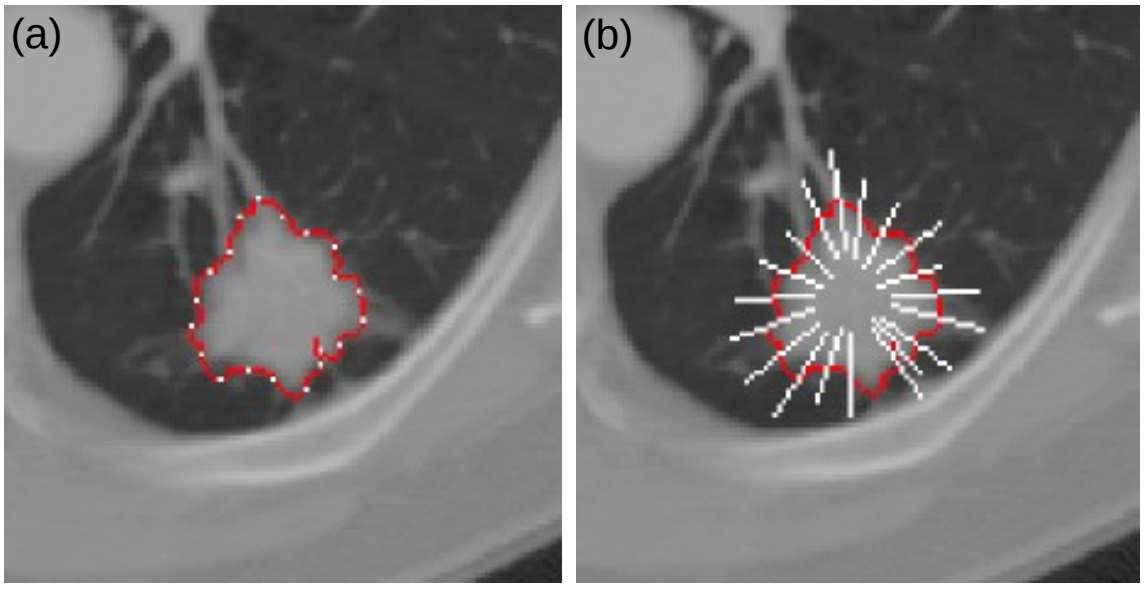

Figura 1.20. Margin sharpness characterization based on boundary control points (a) and normal line segments (b). Source: Author.

In another work, Ferreira Junior et al. found that several features are associated with nodal and distant metastases and could serve as biomarkers for tumor staging ($p < 0.05$) [Ferreira Junior et al. 2020b]. Some of them are the *energy* (Equation 1.6) after the Haar wavelet, GLCM IMCs (Equations 1.21 and 1.22), NIDM *busyness* (Equation 1.69), *directionality* from Tamura, estimation of *fractal dimension*, *surface area* (Equation 1.89), and *volume* (Equation 1.90). Moreover, the shape *diameter* and the GLCM feature of *maximum probability* (Equation 1.30) distinguished types of non-small cell lung cancer (NSCLC) and could serve as biomarkers for histology. The greater are their values, the greater are the chances of tumor to be squamous cell carcinoma, indicating more homogeneous patterns on CT for it and heterogeneous for adenocarcinoma (Figure 1.21). This finding was confirmed in [Zhu et al. 2018, Digumarthy et al. 2019].

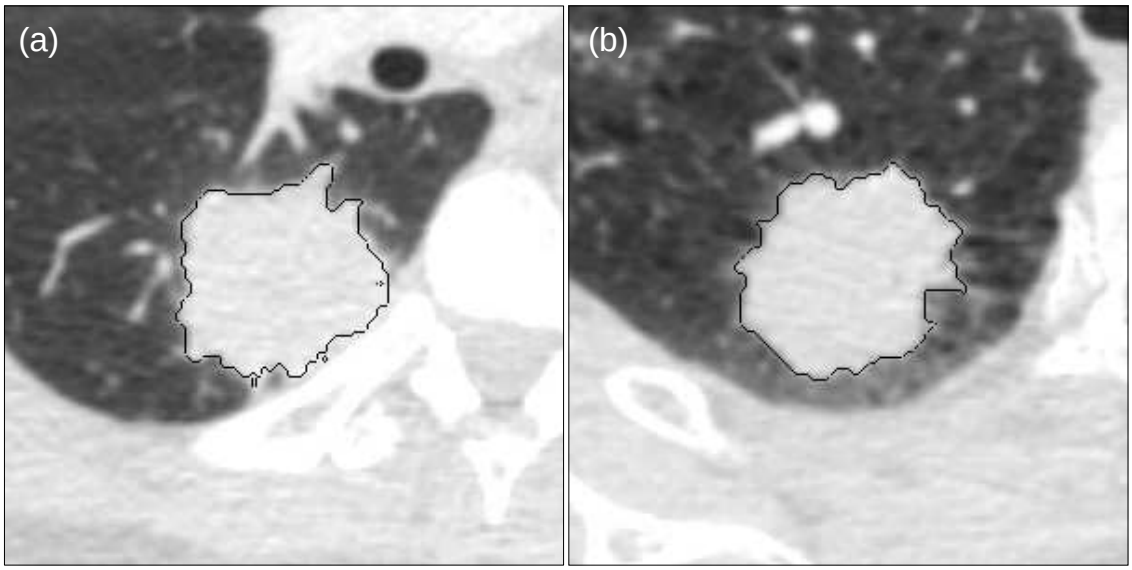

Figura 1.21. Non-small cell lung cancer types of adenocarcinoma (a) and squamous cell carcinoma (b). Source: Author.

van Timmeren et al. discovered three CT features with prognostic value for NSCLC: the mode (most common value) of the image histogram after a Laplacian-of-Gaussian filter, the mean intensity of a VOI centered on the highest gray level, and the GLCM *inverse variance* (Equation 1.27) calculated after a wavelet transform [van Timmeren et al. 2019]. Carvalho et al. discovered the GLRLM feature of *short-run emphasis* (Equation 1.35) on positron emission tomography images combined with CT correlates with the prognosis in patients with lung neoplasms [Carvalho et al. 2018]. Aerts et al. identified a radiomic signature associated with survival in patients with NSCLC composed of four features: first-order *energy* (Equation 1.6), shape *compactness* (Equation 1.93), GLRLM *non-uniformity* (Equation 1.74), and the previous one after a wavelet transform [Aerts et al. 2014]. Ferreira Junior et al. identified another prognostic biomarker for malignant neoplasms of the lung: the *mean* (Equation 1.2) after the Fourier transform ($p < 0.05$) [Ferreira Junior et al. 2021d]. Patients with a high value were identified as being at high risk with a hazard ratio of 2.12 (95% CI: 1.01–4.48). The authors also showed that the lesions from higher-risk patients have greater heterogeneity, and possibly be more aggressive, in comparison to lower-risk lesions, characterized by the presence of

more infiltrating regions (Figure 1.22) [Ferreira Junior et al. 2021c].

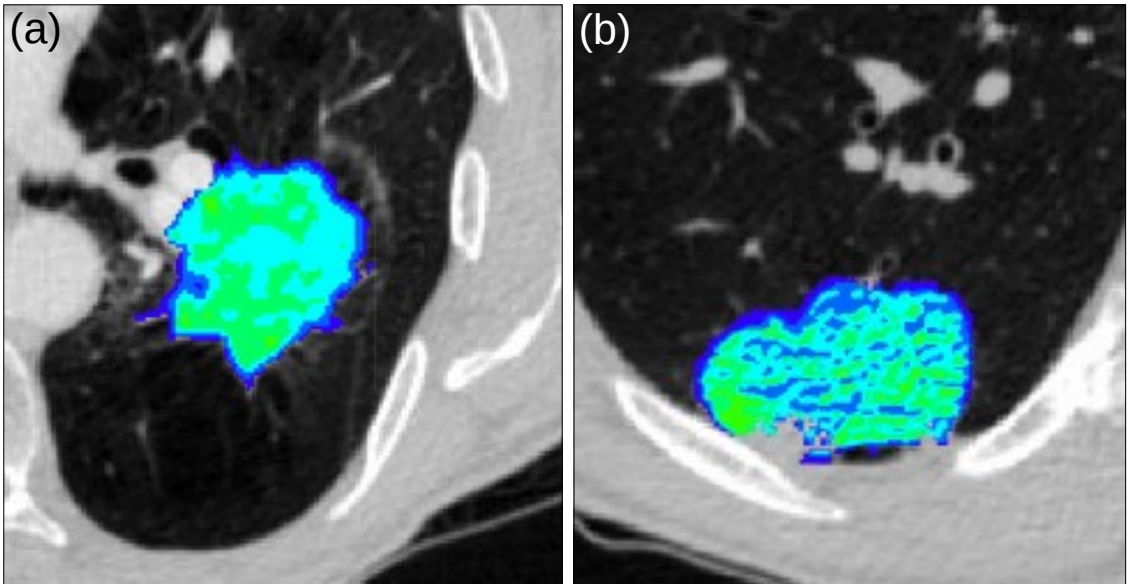

Figura 1.22. Intratumor heterogeneity of lung neoplasms with different staging: (a) stage-I, (b) stage-IV. Source: Author.

These findings highlight the importance of extracting features from all imaging levels. The discovered higher-order biomarkers correlate to smoother or rougher texture variations and thus, could quantify tumor heterogeneity [Ferreira Junior et al. 2021d]. That is important because intratumor heterogeneity can be associated with disease progression and treatment resistance, which ultimately serve as insight to targeted therapies needed for precision medicine [Ferreira Junior et al. 2021c].

Future directions for the area include identifying specific regions of resistance to targeted treatment, which would allow therapies located at a low molecular level.

1.3.3. Spondyloarthritis in magnetic resonance imaging

Spondyloarthritis (SpA) is a set of diseases with common clinical manifestations, such as inflammatory axial pain and peripheral arthritis. The active inflammation in sacroiliac joints, so-called sacroiliitis, is one of the most important criteria to diagnose SpA, and it can be identified in MRI [Rudwaleit et al. 2009]. The major MRI finding of active sacroiliitis is bone marrow edema (Figure 1.23). Therapy decision of SpA consists mainly on the subtype of the disease, *i.e.,* axial or peripheral [Sieper et al. 2009]. In this sense, MRI-based radiomics could play a key role in early diagnosis and therapy decision-making to indicate the presence of SpA and subtyping.

Tenorio et al. identified 63 MRI radiomic biomarkers specific for sacroiliitis, from which most of them were derived from a Gabor bank [Tenorio et al. 2020]. The histogram *skewness* (Equation 1.10) yielded an AUC of 0.86 ($p < 0.001$). High values characterized the active inflammation in MRI, as expected, due to the brightness patterns of the lesion (Figure 1.24). Faleiros et al. confirmed these findings by wrapping six radiomic features into an artificial neural network, yielding an AUC of 0.96 [Faleiros et al. 2020]. Those MRI biomarkers are the mean and standard deviation of the Tamura's *directionality*, *sum*

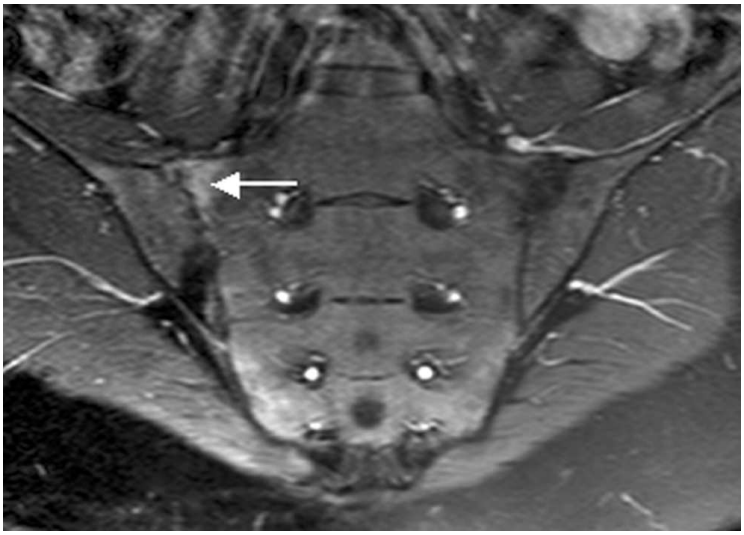

Figura 1.23. MRI of the sacroiliac joints with active inflammation (arrow). Source: Fiona McQueen, Marissa Lassere and Mikkel Ostergaard licensed under CC BY 2.0.

variance of the gray levels, maximum intensity, *mean* after a Gabor filter, and *energy* after the Haar wavelet (LH band on level 2).

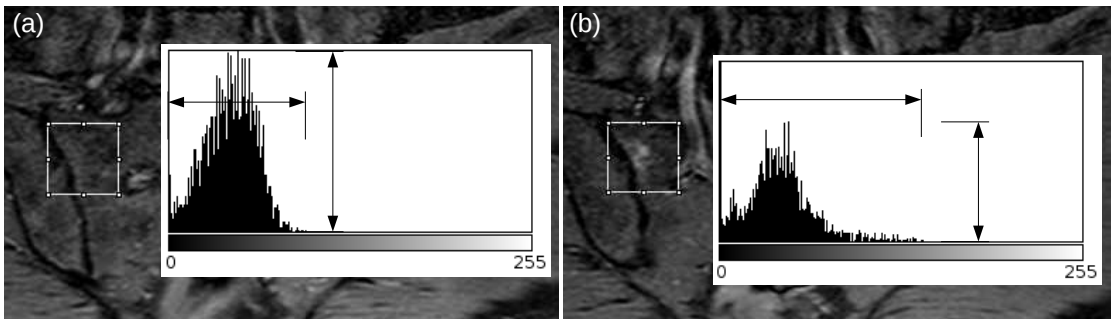

Figura 1.24. Sacroiliitis characterization using histogram and the skewness feature: (a) sacroiliac joint without active inflammation (histogram symmetrical), (b) sacroiliac joint with active inflammation (histogram skewed right). Source: Author.

Tenorio et al. also discovered 27 biomarkers for SpA [Tenorio et al. 2020]. Tamura features were predominant, and the *directionality* yielded the highest performance in identifying SpA with an AUC of 0.80 and distinguishing axial and peripheral with an AUC of 0.97 ($p < 0.001$). High values of the *directionality* standard deviation characterized axial SpA, distinguishing it from the peripheral form and other diseases, like arthrosis, fibromyalgia, and bone injury.

Although this clinical problem does not consider shape features, we highlight the use of all imaging levels for feature extraction. The studies showed that even though the first-order histogram is a simple strategy for characterization, it can quantify the images comprehensively enough to recognize complex clinical patterns.

Future perspectives to improve radiomic biomarker discovery include confirming all findings by testing the features prospectively. It is also advisable the clinical validation of the biomarkers and hence the assessment of whether the biomarkers can impact clinical routine.

1.4. Conclusion

This book chapter introduced theoretical and practical quantitative biomarker development and discovery. We highlighted the valuable role that radiomics can have in precision medicine. Therefore, radiomic biomarkers disclose a vast potential to improve clinical practice.

Referências

[Aerts et al. 2014] Aerts, H., Velazquez, E., Leijenaar, R., Parmar, C., Grossmann, P., Carvalho, S., Bussink, J., Monshouwer, R., Kains, B., Rietveld, D., Hoebers, F., Rietbergen, M., Leemans, C., Dekker, A., Quackenbush, J., Gillies, R., and Lambin, P. (2014). Decoding tumour phenotype by noninvasive imaging using a quantitative radiomics approach. *Nature Communications*, 5(4006).

[Amadasun and King 1989] Amadasun, M. and King, R. (1989). Textural features corresponding to textural properties. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1264–1274.

[Azevedo-Marques and Ferreira Junior 2021] Azevedo-Marques, P. M. and Ferreira Junior, J. R. (2021). Medical image analyst: A radiology career focused on comprehensive quantitative imaging analytics to improve healthcare. *Academic Radiology*. DOI:10.1016/j.acra.2020.11.028.

[Baeßler et al. 2018] Baeßler, B., Mannil, M., Maintz, D., Alkadhi, H., and Manka, R. (2018). Texture analysis and machine learning of non-contrast T1-weighted MR images in patients with hypertrophic cardiomyopathy—preliminary results. *European Journal of Radiology*, 102:61–67.

[Bianconi and Fernandez 2007] Bianconi, F. and Fernandez, A. (2007). Evaluation of the effects of Gabor filter parameters on texture classification. *Pattern Recognition*, 40(12):3325–3335.

[Calheiros et al. 2021] Calheiros, J. L. L., Amorim, L. B. V., Lima, L. L., Lima Filho, A. F., Ferreira Junior, J. R., and Oliveira, M. C. (2021). The effects of perinodular features on solid lung nodule classification. *Journal of Digital Imaging*. DOI:10.1007/s10278-021-00453-2.

[Carvalho et al. 2018] Carvalho, S., Leijenaar, R. T., Troost, E. G., van Timmeren, J. E., Oberije, C., van Elmpt, W., de Geus-Oei, L.-F., Bussink, J., and Lambin, P. (2018). 18F-fluorodeoxyglucose positron-emission tomography (FDG-PET)-Radiomics of metastatic lymph nodes and primary tumor in non-small cell lung cancer (NSCLC)–A prospective externally validated study. *PloS One*, 13(3):e0192859.

[Cheema and Burkes 2013] Cheema, P. and Burkes, R. (2013). Overall survival should be the primary endpoint in clinical trials for advanced non-small-cell lung cancer. *Current Oncology*, 20(2):e150.

[Chiu et al. 2020] Chiu, W. H. K., Vardhanabhuti, V., Poplavskiy, D., Yu, P. L. H., Du, R., Yap, A. Y. H., Zhang, S., Fong, A. H.-T., Chin, T. W.-Y., Lee, J. C. Y., et al. (2020). Detection of COVID-19 using deep learning algorithms on chest radiographs. *Journal of Thoracic Imaging*, 35(6):369–376.

[Davnall et al. 2012] Davnall, F., Yip, C. S., Ljungqvist, G., Selmi, M., Ng, F., Sanghera, B., Ganeshan, B., Miles, K. A., Cook, G. J., and Goh, V. (2012). Assessment of tumor heterogeneity: an emerging imaging tool for clinical practice? *Insights Into Imaging*, 3(6):573–589.

[Dawes et al. 2017] Dawes, T. J., de Marvao, A., Shi, W., Fletcher, T., Watson, G. M., Wharton, J., Rhodes, C. J., Howard, L. S., Gibbs, J. S. R., Rueckert, D., et al. (2017). Machine learning of three-dimensional right ventricular motion enables outcome prediction in pulmonary hypertension: a cardiac MR imaging study. *Radiology*, 283(2):381–390.

[Digumarthy et al. 2019] Digumarthy, S. R., Padole, A. M., Gullo, R. L., Sequist, L. V., and Kalra, M. K. (2019). Can CT radiomic analysis in NSCLC predict histology and EGFR mutation status? *Medicine*, 98(1):e13963.

[Dou et al. 2018] Dou, T. H., Coroller, T. P., van Griethuysen, J. J., Mak, R. H., and Aerts, H. J. (2018). Peritumoral radiomics features predict distant metastasis in locally advanced NSCLC. *PloS One*, 13(11):e0206108.

[Echegaray et al. 2015] Echegaray, S., Gevaert, O., Shah, R., Kamaya, A., Louie, J., Kothary, N., and Napel, S. (2015). Core samples for radiomics features that are insensitive to tumor segmentation: method and pilot study using CT images of hepatocellular carcinoma. *Journal of Medical Imaging*, 2(4):041011.

[European Centre for Disease Prevention and Control 2021] European Centre for Disease Prevention and Control (2021). COVID-19 situation update worldwide. Online (last update on Jun 3, 2021); last access on Jun 4, 2021. Available at www.ecdc.europa.eu/en/geographical-distribution-2019-ncov-cases.

[Faleiros et al. 2020] Faleiros, M. C., Nogueira-Barbosa, M. H., Dalto, V. F., Ferreira Junior, J. R., Tenorio, A. P. M., Luppino-Assad, R., Louzada-Junior, P., Rangayyan, R. M., and Azevedo-Marques, P. M. (2020). Machine learning techniques for computer-aided classification of active inflammatory sacroiliitis in magnetic resonance imaging. *Advances in Rheumatology*, 60:1–10. DOI:10.1186/s42358-020-00126-8.

[Ferreira et al. 2017] Ferreira, J. R., Azevedo-Marques, P. M., and Oliveira, M. C. (2017). Selecting relevant 3D image features of margin sharpness and texture for lung nodule retrieval. *International Journal of Computer Assisted Radiology and Surgery*, 12(3):509–517.

[Ferreira Junior 2019] Ferreira Junior, J. R. (2019). *Framework for Classification, Content-Based Retrieval, and Radiomics of Medical Images: an investigation of quantitative biomarkers for lung cancer*. PhD thesis, Sao Carlos School of Engineering, University of Sao Paulo. DOI:10.11606/T.82.2020.tde-27022020-113956.

[Ferreira Junior et al. 2021a] Ferreira Junior, J. R., Cardenas, D. A. C., Moreno, R. A., Rebelo, M. d. F. d. S., Krieger, J. E., and Gutierrez, M. A. (2021a). A general fully automated deep-learning method to detect cardiomegaly in chest x-rays. In *SPIE Medical Imaging 2021: Computer-Aided Diagnosis*, volume 11597, page 115972B. DOI:10.1117/12.2581980.

[Ferreira Junior et al. 2020a] Ferreira Junior, J. R., Cardenas, D. A. C., Moreno, R. A., Rebelo, M. F. S., Krieger, J. E., and Gutierrez, M. A. (2020a). Multi-view ensemble convolutional neural network to improve classification of pneumonia in low contrast chest x-ray images. In *42nd Annual International Conferences of the IEEE Engineering in Medicine and Biology Society*, pages 1238–1241. DOI:10.1109/EMBC44109.2020.9176517.

[Ferreira Junior et al. 2021b] Ferreira Junior, J. R., Cardenas, D. A. C., Moreno, R. A., Rebelo, M. F. S., Krieger, J. E., and Gutierrez, M. A. (2021b). Novel chest radiographic biomarkers for COVID-19 using radiomic features associated with diagnostics and outcomes. *Journal of Digital Imaging*. DOI:10.1007/s10278-021-00421-w.

[Ferreira Junior and Cardona Cardenas 2021] Ferreira Junior, J. R. and Cardona Cardenas, D. A. (2021). The potential role of radiogenomics in precision medicine for covid-19. *Journal of Thoracic Imaging*, 36(3):W34.

[Ferreira Junior et al. 2021c] Ferreira Junior, J. R., Koenigkam-Santos, M., de Vita Graves, C., Correia, N. S. C., Cipriano, F. E. G., Fabro, A. T., and Azevedo-Marques, P. M. (2021c). Quantifying intratumor heterogeneity of lung neoplasms with radiomics. *Clinical Imaging*, 74:27–30.

[Ferreira Junior et al. 2021d] Ferreira Junior, J. R., Koenigkam-Santos, M., Machado, C. V. B., Faleiros, M. C., Correia, N. S. C., Cipriano, F. E. G., Fabro, A. T., and Azevedo-Marques, P. M. (2021d). Radiomic analysis of lung cancer for the assessment of patient prognosis and intratumor heterogeneity. *Radiologia Brasileira*, 54(2):87–93.

[Ferreira Junior and Oliveira 2015] Ferreira Junior, J. R. and Oliveira, M. C. (2015). Evaluating margin sharpness analysis on similar pulmonary nodule retrieval. In *Computer-Based Medical Systems, IEEE 28th International Symposium on*, pages 60–65. DOI:10.1109/CBMS.2015.16.

[Ferreira Junior et al. 2018] Ferreira Junior, J. R., Oliveira, M. C., and Azevedo-Marques, P. M. (2018). Characterization of pulmonary nodules based on features of margin sharpness and texture. *Journal of Digital Imaging*, 31(4):451–463.

[Ferreira Junior et al. 2017] Ferreira Junior, J. R., Oliveira, M. C., and de Azevedo-Marques, P. M. (2017). Integrating 3D image descriptors of margin sharpness and texture on a GPU-optimized similar pulmonary nodule retrieval engine. *The Journal of Supercomputing*, 73(8):3451–3467.

[Ferreira Junior et al. 2020b] Ferreira Junior, J. R., Santos, M. K., Tenorio, A. P. M., Faleiros, M. C., Cipriano, F. E. G., Fabro, A. T., Nappi, J., Yoshida, H., and Azevedo Marques, P. M. (2020b). CT-based radiomics for prediction of histologic subtype and metastatic disease in primary malignant lung neoplasms. *International Journal of Computer Assisted Radiology and Surgery*, 15:163–172.

[Galloway 1975] Galloway, M. (1975). Texture analysis using gray level run lengths. *Computer Graphics and Image Processing*, 4(2):172–179.

[George et al. 2014] George, B., Seals, S., and Aban, I. (2014). Survival analysis and regression models. *Journal of Nuclear Cardiology*, 21(4):686–694.

[Gilhuijs et al. 1998] Gilhuijs, K. G., Giger, M. L., and Bick, U. (1998). Computerized analysis of breast lesions in three dimensions using dynamic magnetic-resonance imaging. *Medical Physics*, 25(9):1647–1654.

[Gonzalez and Woods 2007] Gonzalez, R. C. and Woods, R. E. (2007). *Digital Image Processing*. Prentice Hall, New Jersey, USA.

[Greenhalgh et al. 2020] Greenhalgh, T. et al. (2020). Management of post-acute covid-19 in primary care. *BMJ*, 370:m3026.

[Haralick et al. 1973] Haralick, R., Shanmugam, K., and Dinstein, I. (1973). Textural features for image classification. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-3(6):610–621.

[Kaplan and Meier 1958] Kaplan, E. and Meier, P. (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53(282):457–481.

[Keek et al. 2018] Keek, S. A., Leijenaar, R. T., Jochems, A., and Woodruff, H. C. (2018). A review on radiomics and the future of theragnostics for patient selection in precision medicine. *The British Journal of Radiology*, 91(1091):20170926.

[Kermany et al. 2018] Kermany, D. S., Goldbaum, M., Cai, W., Valentim, C. C., Liang, H., Baxter, S. L., McKeown, A., Yang, G., Wu, X., Yan, F., et al. (2018). Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131.

[Kickingereder et al. 2016] Kickingereder, P., Bonekamp, D., Nowosielski, M., Kratz, A., Sill, M., Burth, S., Wick, A., Eidel, O., Schlemmer, H.-P., Radbruch, A., et al. (2016). Radiogenomics of glioblastoma: machine learning–based classification of molecular characteristics by using multiparametric and multiregional MR imaging features. *Radiology*, 281(3):907–918.

[Kolossvary et al. 2018] Kolossvary, M., Kellermayer, M., Merkely, B., and Maurovich-Horvat, P. (2018). Cardiac computed tomography radiomics: A comprehensive review on radiomic techniques. *Journal of Thoracic Imaging*, 33(1):26–34.

[Lambin 2021] Lambin, P. (2021). Radiomics: transforming standard imaging into mineable data for diagnostic and theragnostic applications. In *SPIE Medical Imaging 2021: Physics of Medical Imaging*, volume 11595, page 1159502. DOI:10.1117/12.2585711.

[Larue et al. 2017] Larue, R. T., Defraene, G., De Ruysscher, D., Lambin, P., and Van Elmpt, W. (2017). Quantitative radiomics studies for tissue characterization: a review of technology and methodological procedures. *The British Journal of Radiology*, 90(1070):20160665.

[LeCun et al. 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[Lee et al. 2019] Lee, G. R., Gommers, R., Waselewski, F., Wohlfahrt, K., and O'Leary, A. (2019). Pywavelets: A python package for wavelet analysis. *Journal of Open Source Software*, 4(36):1237.

[Leger et al. 2017] Leger, S., Zwanenburg, A., Pilz, K., Lohaus, F., Linge, A., Zophel, K., Kotzerke, J., Schreiber, A., Tinhofer, I., Budach, V., et al. (2017). A comparative study of machine learning methods for time-to-event survival data for radiomics risk modelling. *Scientific Reports*, 7(1):13206.

[Levman and Martel 2011] Levman, J. E. and Martel, A. L. (2011). A margin sharpness measurement for the diagnosis of breast cancer from magnetic resonance imaging examinations. *Academic Radiology*, 18(12):1577–1581.

[Li et al. 2020] Li, M., Lei, P., Zeng, B., Li, Z., Yu, P., Fan, B., Wang, C., Li, Z., Zhou, J., Hu, S., et al. (2020). Coronavirus disease (COVID-19): spectrum of CT findings and temporal progression of the disease. *Academic Radiology*, 27(5):603–608.

[Liang and Zheng 2019] Liang, G. and Zheng, L. (2019). A transfer learning method with deep residual network for pediatric pneumonia diagnosis. *Computer Methods and Programs in Biomedicine*. Ahead of print. DOI:10.1016/j.cmpb.2019.06.023.

[Lu et al. 2019] Lu, M. T., Ivanov, A., Mayrhofer, T., Hosny, A., Aerts, H. J., and Hoffmann, U. (2019). Deep learning to assess long-term mortality from chest radiographs. *JAMA Network Open*, 2(7):e197416–e197416.

[Lubner et al. 2017] Lubner, M. G., Smith, A. D., Sandrasegaran, K., Sahani, D. V., and Pickhardt, P. J. (2017). CT texture analysis: definitions, applications, biologic correlates, and challenges. *Radiographics*, 37(5):1483–1503.

[Osman et al. 2021] Osman, A. H., Aljahdali, H. M., Altarrazi, S. M., and Ahmed, A. (2021). SOM-LWL method for identification of COVID-19 on chest x-rays. *PloS One*. DOI:10.1371/journal.pone.0247176.

[Pairo-Castineira et al. 2021] Pairo-Castineira, E., Clohisey, S., Klaric, L., Bretherick, A. D., Rawlik, K., Pasko, D., Walker, S., Parkinson, N., Fourman, M. H., Russell, C. D., et al. (2021). Genetic mechanisms of critical illness in covid-19. *Nature*, 591(7848):92–98.

[Parker 2011] Parker, J. R. (2011). *Algorithms for Image Processing and Computer Vision*. Wiley Publishing, Indianapolis, USA.

[Phillips et al. 2017] Phillips, I., Ajaz, M., Ezhil, V., Prakash, V., Alobaidli, S., McQuaid, S. J., South, C., Scuffham, J., Nisbet, A., and Evans, P. (2017). Clinical applications of textural analysis in non-small cell lung cancer. *The British Journal of Radiology*, 91(1081):20170267.

[Rajaraman et al. 2020] Rajaraman, S., Sornapudi, S., Alderson, P. O., Folio, L. R., and Antani, S. K. (2020). Analyzing inter-reader variability affecting deep ensemble learning for COVID-19 detection in chest radiographs. *PloS One*, 15(11):e0242301.

[Rudwaleit et al. 2009] Rudwaleit, M., Jurik, A.-G., Hermann, K. A., Landewé, R., van der Heijde, D., Baraliakos, X., Marzo-Ortega, H., Østergaard, M., Braun, J., and Sieper, J. (2009). Defining active sacroiliitis on magnetic resonance imaging (MRI) for classification of axial spondyloarthritis: a consensual approach by the ASAS/OMERACT MRI group. *Annals of the Rheumatic Diseases*, 68(10):1520–1527.

[Sacconi et al. 2017] Sacconi, B., Anzidei, M., Leonardi, A., Boni, F., Saba, L., Scipione, R., Anile, M., Rengo, M., Longo, F., Bezzi, M., et al. (2017). Analysis of ct features and quantitative texture analysis in patients with lung adenocarcinoma: a correlation with egfr mutations and survival rates. *Clinical Radiology*, 72(6):443–450.

[Santos et al. 2019] Santos, M. K., Ferreira Junior, J. R., Wada, D. T., Tenorio, A. P. M., Barbosa, M. H. N., and Azevedo Marques, P. M. (2019). Artificial intelligence, machine learning, computer-aided diagnosis, and radiomics: advances in imaging towards to precision medicine. *Radiologia Brasileira*, 52(6):387–396.

[Schneider et al. 2012] Schneider, C., Rasband, W., and Eliceiri, K. (2012). NIH Image to ImageJ: 25 years of image analysis. *Nature Methods*, 9(7):671–675.

[Sieper et al. 2009] Sieper, J., Rudwaleit, M., Baraliakos, X., Brandt, J., Braun, J., Burgos-Vargas, R., Dougados, M., Hermann, K., Landewe, R., Maksymowych, W., et al. (2009). The Assessment of SpondyloArthritis international Society (ASAS) handbook: a guide to assess spondyloarthritis. *Annals of the Rheumatic Diseases*, 68(Suppl 2):ii1–ii44.

[Sun and Wee 1983] Sun, C. and Wee, W. G. (1983). Neighboring gray level dependence matrix for texture classification. *Computer Vision, Graphics, and Image Processing*, 23(3):341–352.

[Sun et al. 2017] Sun, H., Chen, Y., Huang, Q., Lui, S., Huang, X., Shi, Y., Xu, X., Sweeney, J. A., and Gong, Q. (2017). Psychoradiologic utility of MR imaging for diagnosis of attention deficit hyperactivity disorder: a radiomics analysis. *Radiology*, 287(2):620–630.

[Tamura et al. 1978] Tamura, H., Mori, S., and Yamawaki, T. (1978). Textural features corresponding to visual perception. *IEEE Transactions on Systems, Man and Cybernetics*, 8(6):460–473.

[Tang 1998] Tang, X. (1998). Texture information in run-length matrices. *IEEE Transactions on Image Processing*, 7(11):1602–1609.

[Tenorio et al. 2020] Tenorio, A. P. M., Faleiros, M. C., Junior, J. R. F., Dalto, V. F., Assad, R. L., Louzada-Junior, P., Yoshida, H., Nogueira-Barbosa, M. H., and Azevedo-Marques, P. M. (2020). A study of MRI-based radiomics biomarkers for sacroiliitis and spondyloarthritis. *International Journal of Computer Assisted Radiology and Surgery*, 15(10):1737–1748.

[Thibault et al. 2013] Thibault, G., Angulo, J., and Meyer, F. (2013). Advanced statistical matrices for texture characterization: application to cell classification. *IEEE Transactions on Biomedical Engineering*, 61(3):630–637.

[Tomaszewski and Gillies 2021] Tomaszewski, M. R. and Gillies, R. J. (2021). The biological meaning of radiomic features. *Radiology*, 298(3):505–516.

[Van Griethuysen et al. 2017] Van Griethuysen, J. J., Fedorov, A., Parmar, C., Hosny, A., Aucoin, N., Narayan, V., Beets-Tan, R. G., Fillion-Robin, J.-C., Pieper, S., and Aerts, H. J. (2017). Computational radiomics system to decode the radiographic phenotype. *Cancer Research*, 77(21):e104–e107.

[van Timmeren et al. 2019] van Timmeren, J. E., van Elmpt, W., Leijenaar, R. T., Reymen, B., Monshouwer, R., Bussink, J., Paelinck, L., Bogaert, E., De Wagter, C., Elhaseen, E., et al. (2019). Longitudinal radiomics of cone-beam ct images from non-small cell lung cancer patients: evaluation of the added prognostic value for overall survival and locoregional recurrence. *Radiotherapy and Oncology*, 136:78–85.

[Wehbe et al. 2020] Wehbe, R. M., Sheng, J., Dutta, S., Chai, S., Dravid, A., Barutcu, S., Wu, Y., Cantrell, D. R., Xiao, N., Allen, B. D., et al. (2020). DeepCOVID-XR: An artificial intelligence algorithm to detect COVID-19 on chest radiographs trained and tested on a large US clinical dataset. *Radiology*. DOI:10.1148/radiol.2020203511.

[Wong et al. 2020] Wong, H. Y. F., Lam, H. Y. S., Fong, A. H.-T., Leung, S. T., Chin, T. W.-Y., Lo, C. S. Y., Lui, M. M.-S., Lee, J. C. Y., Chiu, K. W.-H., Chung, T. W.-H., et al. (2020). Frequency and distribution of chest radiographic findings in patients positive for COVID-19. *Radiology*, 296(2):E72–E78.

[Xu et al. 2012] Xu, J., Napel, S., Greenspan, H., Beaulieu, C. F., Agrawal, N., and Rubin, D. (2012). Quantifying the margin sharpness of lesions on radiological images for content-based image retrieval. *Medical Physics*, 39:5405–5418.

[Yang et al. 2016] Yang, J., Zhang, L., Fave, X., Fried, D., Stingo, F., Ng, C., and Court, L. (2016). Uncertainty analysis of quantitative imaging features extracted from contrast-enhanced CT in lung tumors. *Computerized Medical Imaging and Graphics*, 48:1–8.

[Yip et al. 2017] Yip, S. S., Liu, Y., Parmar, C., Li, Q., Liu, S., Qu, F., Ye, Z., Gillies, R. J., and Aerts, H. J. (2017). Associations between radiologist-defined semantic and automatically computed radiomic features in non-small cell lung cancer. *Scientific Reports*, 7(1):3519.

[Zhang et al. 2015] Zhang, L., Fried, D., Fave, X., Hunter, L., Yang, J., and Court, L. (2015). IBEX: an open infrastructure software platform to facilitate collaborative work in radiomics. *Medical Physics*, 42(3):1341–1353.

[Zhang et al. 2021] Zhang, R., Tie, X., Qi, Z., Bevins, N. B., Zhang, C., Griner, D., Song, T. K., Nadig, J. D., Schiebler, M. L., Garrett, J. W., et al. (2021). Diagnosis of coronavirus disease 2019 pneumonia by using chest radiography: Value of artificial intelligence. *Radiology*, 298(2):E88–E97.

[Zhu et al. 2018] Zhu, X., Dong, D., Chen, Z., Fang, M., Zhang, L., Song, J., Yu, D., Zang, Y., Liu, Z., Shi, J., et al. (2018). Radiomic signature as a diagnostic factor for histologic subtype classification of non-small cell lung cancer. *European radiology*, 28(7):2772–2778.

[Zwanenburg et al. 2020] Zwanenburg, A., Vallieres, M., Abdalah, M. A., Aerts, H. J., Andrearczyk, V., Apte, A., Ashrafinia, S., Bakas, S., Beukinga, R. J., Boellaard, R., et al. (2020). The image biomarker standardization initiative: standardized quantitative radiomics for high-throughput image-based phenotyping. *Radiology*, 295(2):328–338.

Capítulo 2

Internet das coisas, blockchain e contratos inteligentes aplicados à saúde

Jauberth Abijaude, Henrique Serra,
Rita Barretto, Aprígio Bezerra,
Péricles Sobreira, Fabíola Greve

Abstract

The Internet of Things aggregates devices able to capture information and interfere in the environment, acting in systems of different domains of application, such as healthcare. These systems need a layer of security to guarantee, among other characteristics, the irrefutability, anonymity, and integrity of the manipulated data. In this sense, an integration with a blockchain, through smart contracts, would meet this need. This chapter, therefore, presents current research using IoT, blockchain, and smart contracts in healthcare. The details for using these technologies in healthcare, the technical challenges and the consensus protocols involved in the main applications will be discussed. This chapter presents a practice that applies knowledge in the health supply chain, building a decentralized application (DApp) that monitors the temperature of vaccines during their storage. In the end, it offers an informative guide that allows participants to design training in this area, including practical exercises.

Resumo

A Internet das Coisas (Internet of Things (IoT)) agrega dispositivos capazes de capturar informações e interferir no ambiente, atuando em sistemas de domínios de aplicações diferentes, como por exemplo o da saúde. Estes sistemas precisam de uma camada de segurança para garantir, dentre outras características, a irrefutabilidade, o anonimato e a integridade dos dados manipulados. Neste sentido, a integração com a blockchain, através dos contratos inteligentes, atenderia a esta necessidade. Este capítulo apresenta, portanto, pesquisas recentes que utilizam IoT, blockchain e contratos inteligentes na área da saúde. Serão apresentados os detalhes para se empregar estas tecnologias na área da saúde, os desafios técnicos e os protocolos de consenso envolvidos nas principais aplicações. Na sequência, apresenta-se uma prática que aplica os conhecimentos abordados na

cadeia de suprimentos para a saúde, construindo uma aplicação descentralizada (DApp) que monitora a temperatura de vacinas durante o seu armazenamento. Ao final, fornece um guia de informações que permite aos interessados a concepção de treinamentos nesta área, contemplando, inclusive, a realização de exercícios práticos.

2.1. Introdução

A Internet das Coisas (IoT - do inglês *Internet of Things*) é capaz de impulsionar várias aplicações médicas, como monitoramento remoto de saúde, programas de condicionamento físico, reabilitação, doenças crônicas e atendimento a idosos [Adibi 2015].

A conformidade com o monitoramento remoto para tratamento e medicação em casa é um potencial importante para a aplicação da telemedicina. Portanto, vários dispositivos médicos, sensores e dispositivos de imagem são essenciais como dispositivos inteligentes ou objetos que constituem uma parte central da IoT para a arquitetura da telemedicina (Kortuem et al., 2010). Assim, o futuro setor de saúde em todo o mundo deve estar preparado para o monitoramento remoto extenso de saúde por meio de IoT e telemedicina (Talalet al., 2019).

Diante dos desafios impostos pela IoT, como segurança e privacidade, a blockchain tem contribuído com a Internet das Coisas Médicas (IoMT) no sentido de melhorar a segurança de dados médicos compartilhados em termos de autenticação de usuário, controle de acesso e privacidade de dados. Também, a mencionada tecnologia tem o potencial de mudar, para descentralizada, a topologia de uma rede de saúde. A blockchain possibilita que os pacientes estejam em um ambiente de ecossistema enquanto aumenta a segurança, a confidencialidade e a interoperabilidade dos dados [Hussien e outros 2021].

Esta seção, de caráter introdutório, tem como objetivo nivelar os conhecimentos básicos nos temas principais a serem abordados nesse capítulo. Ela está dividida nas seguintes subseções: 2.1.1. Internet das Coisas, 2.1.2. Blockchain, 2.1.3. Contratos Inteligentes, 2.1.4. Blockchain e IoT e 2.1.5. Aplicações Distribuídas (DApps).

2.1.1. Internet das Coisas

A IoT é composta por dispositivos físicos com funções de rede, componentes microcomputadorizados e itens incorporados com funções de conectividade [Lao e outros 2020], criando uma classe de objetos inteligentes. Com ela é possível conectar coisas e pessoas a qualquer hora e em qualquer lugar para qualquer serviço. Com o surgimento da IoT, a área da saúde requer uma assistência de outras áreas, principalmente a de Ciência da Computação. Através desta nova classe de objetos inteligentes é possível a aquisição e gerenciamento de registros eletrônicos de saúde para ferramentas que podem auxiliar no diagnóstico e predição de doenças.

A IoT pode ser vista como a combinação de diversas tecnologias, as quais são complementares no sentido de viabilizar a integração dos objetos no ambiente físico ao mundo virtual, ilustrados na Figura 2.1 [Santos e outros 2016].

O bloco de Identificação é um dos componentes mais importantes, pois é fundamental garantir aos objetos inteligentes um meio único de ser reconhecido na internet, como por exemplo um endereço IP. O bloco de Comunicação representa os mecanismos

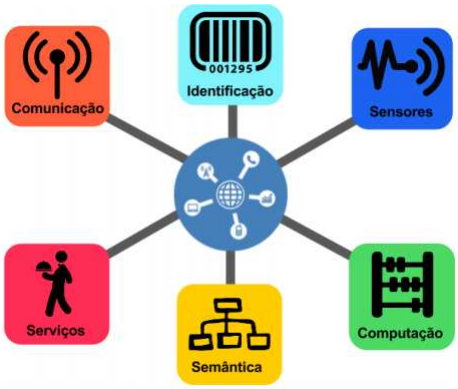

Figura 2.1. Componentes e tecnologias básicas da IoT. *Fonte* [Santos e outros 2016]

para que os dispositivos de IoT possam se comunicar, consideradas as restrições inerentes de potência, processamento e armazenamento. Alguns protocolos usados são o LoraWan, Wi-fi, Bluetooth e Zigbee.

O bloco de sensores representam os sensores e atuadores que coletam informações ou interferem no meio em que se encontram. Os dados capturados são enviados para um middleware, aplicações ou para a blockchain. A computação significa as unidades de processamento capazes de processar algoritmos e disparar as ações de captura de dados ou de acionamento dos atuadores.

Os blocos de serviços e semântica representam, respectivamente, atividades agrupadas em classes (Identificação, Agregação de dados, Colaboração, Inteligência e Ubiquidade) e a habilidade de extrair conhecimentos dos dispositivos de IoT.

De modo geral, a implementação de soluções que abrangem IoT empregam serviços de middleware para abstrair as dificuldades de acesso aos dispositivos devido a heterogeneidade de protocolos e interfaces. A Figura 2.2 ilustra uma organização geral de elementos para Internet das Coisas [Sztajnberg e outros 2018]. Os sensores e atuadores são agrupados em dispositivos e serviços, que por sua vez precisam das interfaces de comunicação para enviar/receber dados das camadas superiores.

Figura 2.2. Organização de elementos na IoT. Fonte [Sztajnberg e outros 2018]

Existe uma variedade de protocolos e padrões de comunicação que podem ser

empregados, entre eles, destacam-se o MQTT, CoAP, AMQP, XMPP, WebSocket, REST, Lorawan, etc.

As práticas deste capítulo utilizam o protocolo REST. Ele permite o uso da infraestrutura do HTTP para acionar ou obter recursos, apenas dando uma interpretação diferente para os métodos GET, PUT, POST e DELETE, e valendo-se da possibilidade de enviar informações adicionais numa mensagem HTTP [Sztajnberg e outros 2018].

2.1.2. Blockchain

A blockchain é uma tecnologia emergente que oferece suporte distribuído confiável para realização de transações entre participantes que não necessariamente têm confiança entre si e que se encontram dispersos numa rede P2P. É considerada uma tecnologia disruptiva e com potencial de substituir entidades certificadoras e centralizadoras das transações de negócios, tais como bancos, governos, cartórios, etc. [Greve e outros 2018].

A primeira rede blockchain, apresentada em 2007, detalhava ao mundo um sistema econômico alternativo com uma moeda digital (Bitcoin) [Nakamoto 2008]. Esta rede permitia transacionar valores digitais através de uma estrutura computacional distribuída. Em 2009, tal rede entra em operação utilizando uma máquina de estado simplificada, com um arranjo até então inédito, que proporcionava eliminar a terceira parte de confiança, necessária para as transações financeiras tradicionais.

Para sustentar esta tecnologia, diversos elementos foram combinados de forma engenhosa e harmoniosa, de forma a pavimentar o caminho para as aplicações descentralizadas. São eles:

- Criptografia: Satisfaz os requisitos de segurança do sistema e das aplicações. Dentre os recursos mais utilizados, destacam-se os resumos criptográficos (funções *hash*) e as assinaturas digitais;
- Consenso distribuído: Permite com que participantes distribuídos coordenem as suas ações, de forma a alcançar decisões comuns, e assim garantir a manutenção da consistência dos seus estados (*safety*) e o progresso do sistema (*liveness*), apesar da existência de falhas [Greve 2005];
- Livro razão distribuído: O livro-razão (*ledger*) é uma estrutura de dados imutável, em que transações são registradas e o estado global do sistema é mantido replicado em todos os nós da rede P2P.

A consequência desta composição garante algumas propriedades que contribuem de forma inovadora para o desenvolvimento de novas soluções tecnológicas, entre elas as aplicações descentralizadas (DApps) como por exemplo [Greve e outros 2018]:

- Descentralização: Sistemas e aplicações que usam a BC não precisam de uma entidade central para coordenar as ações, as tarefas são executas de forma distribuída;
- Disponibilidade e integridade: Os dados e as transações são replicados para todos os participantes da BC, mantendo o sistema seguro e consistente;

- Transparência e auditabilidade: A cadeia de blocos que registra as transações é pública e pode ser auditada e verificada;
- Imutabilidade e Irrefutabilidade: os registros são imutáveis e a correção só pode ser feita a partir de novos registros. O uso de recursos criptográficos garante que os lançamentos não podem ser refutados;
- Privacidade e Anonimidade: As transações são anônimas, com base nos endereços dos usuários. Os servidores armazenam apenas fragmentos criptografados dos dados do usuário;
- Desintermediação: A BC consegue eliminar terceiros em suas transações, atuando como um conector de sistemas de forma confiável e segura;
- Cooperação e incentivos: Uso do modelo de teoria dos jogos como forma de incentivo.

Em 2013, surge a plataforma *Ethereum*, que evolui para além das transações de uma criptmoeda. Implementada sob um modelo de máquina de *turing* completa, com uma nova criptomoeda e ancorada sob alguns conceitos de seu antecessor, esta nova plataforma inova ao permitir que programas de computador possam ser armazenados e executados nas cadeias de blocos. Tais programas, conhecidos como contratos inteligentes.

A blockchain, segundo [Wu e outros 2019], tem uma arquitetura divida em quatro camadas: (i) Dados, onde se encontram os blocos, o armazenamento de dados e a estrutura de árvore utilizada; (ii) Rede, onde se encontra a rede P2P e os mecanismos de comunicação; (iii) Consenso, onde naturalmente estão os protocolos de consenso; e, (iv) Aplicação, onde estão os contratos inteligentes, as criptomoedas e as *sidechains*. A Figura 2.3 ilustra esta divisão.

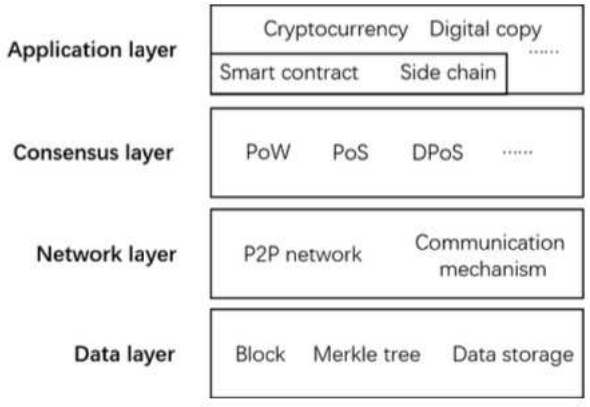

Figura 2.3. Arquitetura blockchain em quatro camadas. *Fonte* [Wu e outros 2019]

Na camada de dados estão a estrutura, organização e armazenamento de dados. Fatores como desempenho e acesso são cruciais para a rede Blockchain. Cada uma destas redes utiliza estruturas diferentes. De um modo geral, o banco de dados para armazenamento é o Google LevelDb. A rede Bitcoin usa a árvore de *Merkle* como forma de organizar e armazenar as informações, enquanto a *Ethereum* usa a *Merkle Patricia Tree*.

A Camada de rede da blockchain é autonomamente mantida e gerenciada por uma rede P2P composta por mineradores e usuários. É uma estrutura descentralizada e sem a necessidade de controle de entrada e saída, com tolerância a falhas A comunicação e a autenticação devem ser protegidas em caso de ataques.

A camada de consenso é fundamental em uma rede blockchain. Chegar a um consenso não é uma tarefa não trivial e muitos algoritmos de consenso foram propostos para atingir esse objetivo[Greve 2005]. Esses algoritmos ou mecanismos podem ser classificados em: PoW (do inglês *Proof-of-Work*), PoS (do inglês *Proof-os-Stake*) e suas variantes; BFT (do inglês *Bizantine Fault Tolerance*) e suas variantes.

Por fim, a camada de Aplicação estende a capacidade do blockchain e torna mais fácil para os desenvolvedores construir aplicativos blockchain através dos contratos inteligentes, explicados na sequência; das *sidechains* e do emprego de BaaS (do inglês *Blockchain as a Service*) [Samaniego e outros 2016], por exemplo. Nesta camada também estão as criptomoedas e uma série de aplicações incluindo *Fintechs*, seguros, pagamentos, governo, etc. Existe inclusive a possibilidade de ser combinada com inteligência artificial, big data, computação quântica, IoT etc.

2.1.3. Contratos Inteligentes

Os Contratos Inteligentes (CIs), definidos pela primeira vez por Nick Szabo [Szabo 1997], representam "um conjunto de promessas, especificado em formato digital, incluindo protocolos nos quais as partes cumprem estas promessas". Este conceito evoluiu, especialmente após a introdução de plataformas blockchain descentralizadas.

Com o surgimento da *Ethereum* [Buterin e outros. 2014], tivemos o arcabouço tecnológico capaz de implementar a definição de Szabo através de programas de computador imutáveis, que são executados de forma determinística, no contexto de uma Máquina Virtual Ethereum (EVM, do inglês *Ethereum Virtual Machine*), como parte do protocolo de rede *Ethereum*.

Tais contratos, então, são sistemas que movem ativos digitais automaticamente de acordo com regras pré-especificadas. A palavra **contrato** não tem significado legal neste contexto. Com a implantação generalizada da blockchain, o contrato inteligente recebeu grande atenção das empresas e academia.

Os CIs são imutáveis, por que uma vez implementado em uma rede *Ethereum*, o código não pode ser alterado e nem substituído. A única forma de se modificar o seu conteúdo é implementando um novo contrato, o qual terá um novo endereço.

Assim como os softwares, os contratos são determinísticos, pois o resultado de sua execução é sempre o mesmo para todos os que o executam, conservando-se o contexto no momento da execução. Os CIs estão em constante evolução e operam com um contexto muito limitado, por enquanto. No caso dos CIs para a rede *Ethereum*, existem diversas versões do compilador (*solc*), com mudanças significativas entre elas. De modo geral, os CIs acessam seu próprio estado, o contexto da transação que os chamou e algumas informações sobre os blocos mais recentes.

2.1.4. Blockchain e IoT

Blockchains para IoT são sistemas de blockchain personalizados e otimizados para aplicações de IoT. Estas aplicações são desenvolvidas em muitos campos. No entanto, a maioria desses aplicativos possui problemas como vazamento e confiabilidade de dados. Para mitigar esses efeitos problemáticos, a blockchain pode ser usada para fornecer maior segurança e estabilidade aos aplicativos IoT tradicionais.

A tecnologia Blockchain envolve muitos elementos além de simplesmente conectar blocos em uma cadeia. Ao adicionar elementos de IoT, esta complexidade ganha novos contornos que precisam ser desvendados. A arquitetura de aplicações para blockchain-IoT é composta de 5 camadas: Física, Rede, Blockchain, Middleware e Aplicação [Lao e outros 2020]. A Figura 2.4 ilustra estas camadas.

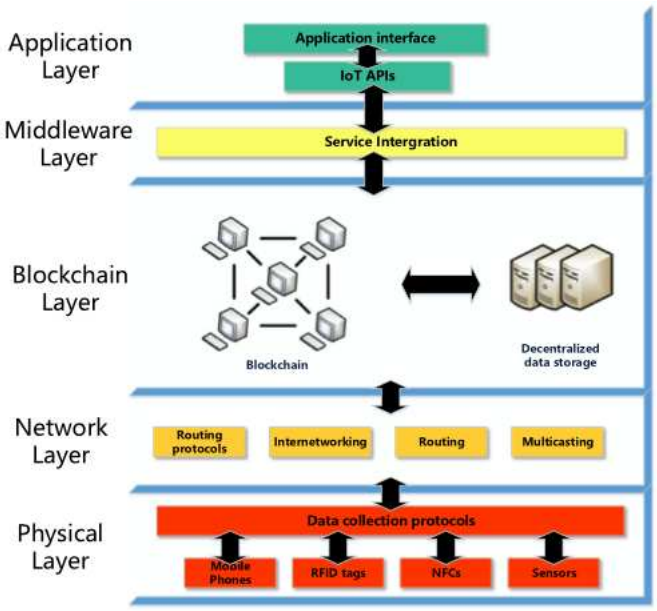

Figura 2.4. Arquitetura Blockchain-IoT em 5 camadas. Fonte [Lao e outros 2020]

A camada física da arquitetura blockchain-IoT é a mesma que a camada física da IoT [Lee e Lee 2015]. Inclui os sensores, atuadores, dispositivos inteligentes, etiquetas RFID, telefones celulares, câmeras de monitoramento e quaisquer outros dispositivos de IoT relacionados à aplicações dentro deste perfil. Os protocolos de coleta de dados empregados também são os mesmos usados em IoT.

A camada de rede é responsável por funções de roteamento, interconexão de redes e *multicasting* [Jiang e outros 2016]. Esta camada é muito similar à camada de rede tradicional da blockchain. A camada de blockchain é composta por funções de consenso, armazenamento de dados e compartilhamento de dados. Pode ser uma plataforma de blockchain específica ou pública[Nakamoto 2008, Ethereum 2014].

As ultimas duas camadas, middleware e aplicação, são responsáveis, respectivamente, por gerenciar a integração de IoT com blockchain, fornecendo serviços de segurança adicionais [Alphand e outros 2018] e fornecer abstrações por intermédio de API's e aplicações, de forma semelhante ao sistema IoT e às arquiteturas blockchain

tradicionais[Dorri e outros 2017].

2.1.5. Aplicações Distribuídas - DApps

Um aplicativo convencional, basicamente é composto de *front-end* e *back-end*. O primeiro elemento representa o uso de linguagens como HTML, CSS e JavaScript, dentre outras, para desenvolver uma interface gráfica a ser apresentada ao usuário. O segundo, refere-se ao desenvolvimento do lado do servidor. Ele é composto de bancos de dados, scripts, arquitetura de sites, lógica do negócio, etc. Esta parte da aplicação contém atividades ocorrem durante a execução de qualquer ação no *front-end*.

A Figura 2.5 ilustra uma possível arquitetura de um sistema convencional. Temos um fomulário apresentado ao usuário, que interage com o *Web Server*, e este, eventualmente com com outro servidor.

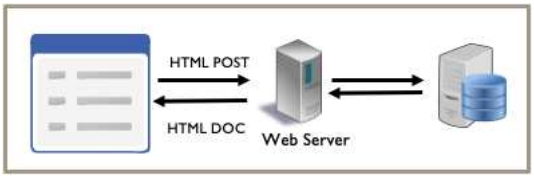

Figura 2.5. Arquitetura web.

Uma DApp é um aplicativo que é majoritariamente ou totalmente descentralizado. Em uma DApp, os contratos inteligentes são usados para armazenar a lógica de negócios (código do programa) e o estado relacionado de seu aplicativo.

A Figura 2.6 exemplifica uma transação em uma DApp que envolve dispositivos de IoT. Em (1), dados provenientes de dispositivos de IoT são enviados para um middleware. O middleware encaminha estes dados para um contrato inteligente hospedado na plataforma *Ethereum* (2), que após o processo de mineração, envia uma confirmação(3) de volta ao middleware. O usuário, tempos depois, através do *front-end* envia um pedido de informações sobre o sensor ao servidor web (4). Para atender a esta requisição, o servidor interage com a blockchain, solicitando ao respectivo contrato, através de uma função específica, o envio dos dados pedidos pelo cliente(5). Ao receber os dados, o servidor os envia para o *front-end* (6).

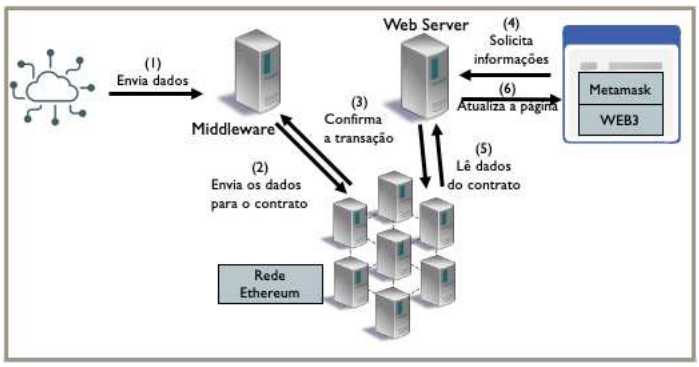

Figura 2.6. Arquitetura que exemplifica uma das formas de uma DApp trocar informações com a IoT.

Existem muitas vantagens na criação de um DApp que uma arquitetura centralizada típica não pode fornecer, entre elas a resiliência, a transparência e a resistência [Antonopoulos 2017].

Uma DApp é resiliente e não perde a sua conexão com a rede e seus clientes. Ela continua disponível enquanto a plataforma blockchain permanecer operando. Ao contrário de um aplicativo implantado em um servidor centralizado.

Como a cadeia de blocos é pública e imutável, as operações e o código da DApp podem ser inspecionados. Com isto temos um grau de confiabilidade sobre sua função e garante-se a transparência.

A rede blockchain dispensa um elemento certificador ou um nó centralizador, portanto, se um usuário consegue acesso a um nó da rede, ele acessa também a DApp, demonstrando a resitência.

Por fim, os dados do sensor ou informações sobre a lógica do negócio estão disponíveis na blockchain, independente do *Web Server*. Tais dados, herdam por conseguinte, as propriedades da blockchain descritas em 2.1.2.

2.2. Plataforma Ethereum e Contratos Inteligentes

Esta seção é divida em duas partes, A primeira descreve a plataforma *Ethereum* e a segunda os contratos inteligentes, preparando o leitor para a prática deste capítulo.

Serão abordados os componentes da *Ethereum*, como a EVM, os conceitos de *ether* e *gas*, as transações, os clientes *Ethereum* , as carteiras eletrônicas, as redes principal e de testes e outros detalhes.

Na segunda parte, apresenta-se os CIs e suas características, o ciclo de vida de um contrato, a linguagem Solidity, o processo de compilação e implementação de um contrato na plataforma *Ethereum* e um contrato inteligente com fins didáticos, exemplificando algumas particularidades inerentes à plataforma *Ethereum*. Estes conceitos são abordados conforme [Buterin e outros. 2014]

2.2.1. Plataforma Ethereum

O *Ethereum* é uma máquina de estado determinística acessível globalmente, composta por uma máquina virtual que aplica alterações a esse estado. Esta plataforma de blockchain transaciona ativos financeiros e é capaz de armazenar e executar códigos de computador em sua estrutura.

A plataforma *Ethereum* é composta por:

- Rede P2P: uma rede *peer-to-peer* capaz de suportar a rede *ethereum* principal e redes de teste. A rede principal é endereçada na porta TCP 30303 e executa o protocolo DEVp2p.
- Regras de Consenso: As regras para o consenso estão definidas no *yellow paper* [Buterin e outros. 2014].
- Transações: são mensagens capazes de transacionar ativos digitais, programas de

computador e informações. Estas transações possuem campos como remetente, destino, valor e área de dados.

- Máquina de estados:A EVM é uma máquina de estados executa os contratos inteligentes escritos em linguagens de alto nível, como o solidity. Os contratos são submetidos a um processo de compilação, resultando nos bytecodes, os quais são enviados para a rede e executados pela EVM.

- Estrutura de dados: os dados são armazenados em uma estrutura binária chamada *Merkle Patrícia Tree*. O banco de dados utilizado para isto é geralmente o LevelDB.

- Algoritmos de Consenso: A plataforma *Ethereum* utiliza o mesmo tipo de protocolo de consenso do Bitcoin. O protocolo PoW implementado na rede *Ethereum* é o *Etash*.A próxima geração, conhecida como Ethereum 2.0 utilizará o PoS, uma vez que o uso de PoW tem-se mostrado ineficiente sob o ponto de vista energético.

- Clientes: São implementações de software, interoperáveis. Estes clientes podem ser completos ou remotos.

Sob uma perspectiva mais prática, a *Ethereum* é uma infraestrutura de computação globalmente descentralizada e de código aberto que executa programas chamados contratos inteligentes. Ela usa a blockchain para sincronizar e armazenar as mudanças de estado do sistema, e incorpora a criptomoeda *ether* para medir e restringir os custos dos recursos de execução [Antonopoulos 2017].

2.2.2. EVM

A Máquina Virtual *Ethereum* é um dos principais componentes da desta plataforma uma vez que ela executa os contratos inteligentes e auxilia na manutenção do estado global da rede.

O termo máquina virtual é bastante conhecido na computação. Ele pode ser empregado para designar tecnologias que emulam um computador virtual através de reserva de recurso em uma máquina real. Também é conhecido como uma abstração que permite executar *bytecodes* gerados após o processo de compilação nas linguagens de alto nível como .Net e Java.

No âmbito da plataforma Ethereum, a EVM opera em um ambiente limitado, fornecendo um mecanismo computacional e executando seus próprios *bytecodes*. Pode-se afirmar ela implementa uma máquina de turing quase completa [Antonopoulos 2017].

A máquina de turing prevê que programas de computador podem ser executados em *loops* infinitos. Isto muitas vezes é útil e simples de ser implementado. No entanto, por descuido, ao executar pequisas ou correlações complexas, ou então de forma intencional ao implementando um ataque de Negação de Serviço (DoS, do inglês *Denial Of Service*), por exemplo, estes *loops* infinitos podem acontecer.

Isto, particularmente, na rede *Ethereum*, é um problemas de proporções sérias. Imagine um contrato inteligente, que ao ser executado em um nó, entre em *loop* infinito, consumindo recursos, energia, poder de processamento e monopolizando o nó durante um

longo tempo. A EVM não pode prever quando um programa vai ser encerrado, portanto, *a priori*, não é capaz de identificar tal situação. Este desperdício de recursos tem impacto global, uma vez que a blockchain possui o mesmo alcance.

Para evitar situações como esta, a plataforma *Ethereum* possui uma abstração monetária - o *gas*. À medida que o EVM executa um contrato inteligente, um algoritomo contabiliza as instruções (computação, acesso a dados, etc.). Cada instrução tem um custo predeterminado em unidades de *gas*.

Quando uma transação aciona a execução de um contrato inteligente, ela possui uma quantidade de *gas* que define o limite superior do que pode ser consumido ao ser executado o contrato inteligente. A EVM encerrará a execução se a quantidade de *gas* consumido pelo cálculo exceder o *gas* disponível na transação. O *gas* é o mecanismo que a *Ethereum* usa para permitir a implementação da máquina completa de *Turing*, enquanto limita os recursos que qualquer programa pode consumir.

2.2.3. Conceitos de *ether* e *gas*

O *ether* é a criptomoeda do *Ethereum*. Para adquirir *ethers* é necessário comprá-los com dólares em uma casas de câmbio virtual existentes. Os *ethers* possuem frações conforme ilustrado na Tabela 2.1.

Tabela 2.1. Divisão de unidades do ether

Valor (em wei)	Potencia	Nome
1	1	wei
1.000	10^3	babage ou Kwei
1.000.000	10^6	lovelace ou Mwei
1.000.000.000	10^9	shanon ou Gwei
1.000.000.000.000	10^{12}	szabo ou Microether
1.000.000.000.000.000	10^{15}	finney ou Miliether
1.000.000.000.000.000.000	10^{18}	ether
1.000.000.000.000.000.000.000	10^{21}	grand ou Kiloether

O *gas* é como se fosse o combustível do *Ethereum*. O *gas* não é *ether*, é uma moeda virtual separada com sua própria taxa de câmbio em relação ao *ether*. O *Ethereum* utiliza o *gas* separado do *ether* como forma de isolar a cotação da moeda *ether* no mundo real do valor das transações na rede pelos quais o *gas* paga (computação, memória e armazenamento).

Existem dois conceitos importantes relativos ao *gas*: O *gasLimit* e o *gasPrice*.

O *gasLimit* indica qual o limite de *gas* a ser consumido pela transação, ou seja, o número máximo de unidades de *gas* que o emissor da transação está disposto a comprar para concluir a transação. O *gasPrice*, em uma transação, permite que o emissor da transação defina o preço que está disposto a pagar para adquirir o *gas*. O preço é medido em *wei* por unidades de *gas*.

Por exemplo, uma transação simples, de transferência de *ether* de uma conta para outra, custa 21.000 unidades de *gas*. O valor a ser pago em *ether* é encontrado multiplicando-se 21.000 x *gasPrice*. As carteiras geralmente possuem um valor médio cobrado na rede para que uma transação seja confirmada dentro de um tempo aceitável

(no momento da escrita deste texto estava em torno de 12,3 *gwei*). Neste caso, uma transferência de *ether* custaria 21.000 x 12,3 = 266.700 *gwei*,o que equivale a 0.0002667 *ether*.

As carteiras podem ajustar o *gasPrice* nas transações originadas para obter uma confirmação mais rápida das transações. Quanto maior o *gasPrice*, mais rápido a transação provavelmente será confirmada. Por outro lado, as transações de baixa prioridade podem ter um preço reduzido, resultando em uma confirmação mais lenta. O valor mínimo que *gasPrice* pode ser definido é zero, o que significa uma transação sem taxas. Durante os períodos de demanda por espaço em um bloco, essas transações podem ser preteridas.

2.2.4. Clientes *Ethereum*

O acesso a plataforma *Ethereum* pode ser classificado sob dois aspectos: (a) Acesso para os desenvolvedores e (b) Acesso para os usuários. Em cada um deles há ferramentas e métodos diferentes.

Em (a), os desenvolvedores, comumente, usam a `web3`. Ela é uma coleção de bibliotecas que permitem a interação com um nó *Ethereum*, local ou remoto, usando HTTP, IPC ou *WebSocket* [Web3js 2016], com APIs(*Application Programming Interface*) disponíveis para Java, Javascript, .Net e outras linguagens de programação.

Em (b), os usuários conectam-se à rede *Ethereum* usando um cliente remoto (um aplicativo de software que implementa a especificação *Ethereum* e se comunica pela rede ponto a ponto com outros clientes *Ethereum*). Estes clientes remotos oferecem um subconjunto da funcionalidade de um cliente completo. Eles não armazenam a blockchain *Ethereum* completa, são mais rápidos de configurar e requerem menos armazenamento de dados.

Geralmente, os clientes remotos permitem: (1) Gerenciar chaves privadas e endereços *Ethereum* em uma carteira; (2) Criar, assinar e transmitir transações; (3) Interagir com contratos inteligentes, usando a carga útil de dados; (4) Navegar e interagir com DApps; (5) Oferecer links para serviços externos, como exploradores de blocos; (6) Converter unidades de *ether* e recuperar taxas de câmbio de fontes externas; (7) Injetar uma instância `web3` no navegador web como um objeto JavaScript; (8) Usar uma instância `web3` fornecida/injetada no navegador por outro cliente; e/ou (9) Acessar os serviços RPC em um nó *Ethereum* local ou remoto.

As carteiras móveis (*wallets*) são clientes remotos, já que os smartphones não têm recursos adequados para executar um cliente *Ethereum* completo. Os mais populares são *Jaxx* [Jaxx 2018], *Status* [Status 2019] , *Trust Wallet* [Trust 2019] e *Coinbase* [Coinbase 2018].

Os usuários podem também usar navegadores, onde as carteiras estão disponíveis como plugins ou extensões dos principais navegadores, como Chrome ou Firefox, por exemplo. Estes clientes remotos são executados no navegador. Os mais populares são *Metamask* [Metamask 2018], *Jaxx*, *MyEtherWallet* [Myetherwallet 2019], *Nifty* e MyCrypto [MyCrypto 2019].

O *Metamask* é um gateway para aplicações da plataforma *Ethereum*. Ele fornece acesso a todas as redes da plataforma com uma única conta, permitindo que se gerencie

as carteiras de todas as redes *Ethereum*. O *Metamask* pode ser instalado nos principais navegadores sob forma de extensão. Ao instalar, você receberá 12 palavras mnemônicas, e deve guardá-las sob o maior sigilo, pois são a única forma de recuperar a sua conta ou fazer transações usando a `web3`. Estas palavras devem ser informadas na ordem em que são apresentadas, na criação da conta, quando for necessário. O procedimento para instalação do *Metamask* pode ser acessado na página do curso [Abijaude e outros 2021].

O *Metamask* pode criar outras contas de acesso às redes *Ethereum*. Isto quer dizer que com as mesmas palavras mnemônicas e com a mesma instância instalada no navegador, o usuário pode ter vários endereços de contas. Isto é muito importante, principalmente quando formos usar o CI e a DApp, descritas no decorrer do texto.

2.2.5. Redes *Ethereum*

A plataforma Ethereum possui mais de uma rede disponível para os usuários. A Figura 2.7 ilustra isto. Na rede principal acontecem as transações reais, com impactos financeiros. Os *ethers* aqui precisam ser comprados com dólares.

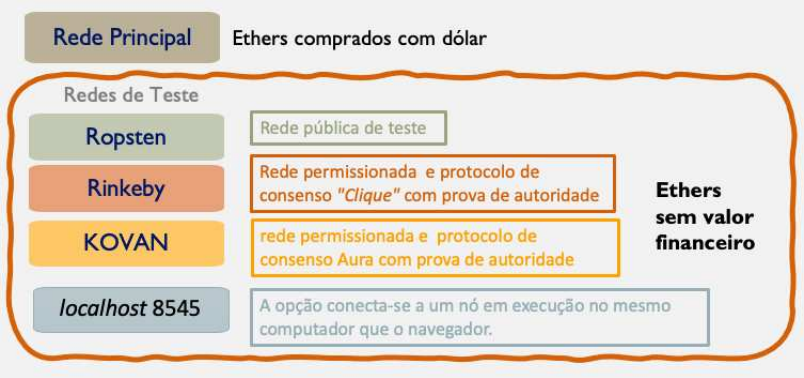

Figura 2.7. Exemplos de redes que compõem a plataforma Ethereum.

As redes de teste (Ropsten, Rinkeby, Kovan) trabalham com *ethers* que não possuem valor real e que podem ser adquiridos em geradores de *ethers* na Internet, sem custos financeiros. A opção *localhost* 8545 conecta-se a um nó em execução no mesmo computador que o navegador.

Além destas redes, há a opção via RPC que permite conexão a qualquer nó com uma interface de Chamada de Procedimento Remoto (RPC) compatível com *Geth*.

Este capítulo usa a rede de testes *Rinkeby* e o plugin *Metamask*. Para abastecer a carteira do *Metamask* com *ethers* sem valor comercial na rede *Rinkeby*, usa-se, por exemplo, o site `https://faucet.rinkeby.io/`. Ao criar uma conta no *Metamask*, automaticamente temos acesso a todas as redes *Ethereum*.

2.2.6. Transações na Blockchain Ethereum

As transações na rede *Ethereum* são originadas por um proprietário de uma conta e enviadas para execução pela EVM. A rede *Ethereum* não é autônoma, os contratos não podem sozinhos dispararem uma ação, eles precisam ser provocados por uma conta para que possam realizar uma tarefa. A Tabela 2.2 lista os campos que compõem as transações.

Basicamente há dois tipos de transação: A transação de criação de contrato e a transação convencional. A transação de criação de contrato, como o próprio nome sugere, adiciona um novo contrato na blockchain *Ethereum*. Isto é feito enviando a transação para um endereço especial conhecido como *zero address* `0x0`. Este endereço não representa uma conta e nem um contrato, ele é exclusivo a criação de novos contratos.

A transação convencional é aquela utilizada para transferência de *ethers* ou informações entre duas contas *Ethereum*, entre dois contratos inteligentes ou entre contas e contratos inteligentes. Estas transações não apontam para o endereço `0x0`. Elas apontam para endereços válidos que representam suas respectivas entidades.

A Tabela 2.2 lista os campos que compõem uma transação.

Tabela 2.2. Campos disponíveis na transação enviadas em uma rede *Ethereum*.

Campo	Descrição
nonce	Número de sequência da transação
gasPrice	Preço do *gas* em *wei* que o usuário está disposto a pagar
gasLimit	Limite de unidades de *gas* que o usuário aceita pagar pela transação
recipient	Endereço da conta *Ethereum* do cliente
value	Quantidade de *ether* que será enviada na transação
data	Campo que contém informações como chamada para funções do contrato ou *bytecodes*
v,r,s	Elementos criptográficos para a assinatura da transação

Entre estes, três campos merecem destaque: *nonce*, *data* e *value*. O primeiro é um número escalar que representa a quantidade de transações realizadas e confirmadas por uma conta. É um método que garante o ordenamento cronológico das transações e evita gastos duplicados.

Se uma fonte de dados encaminhar uma sequência de transações, elas serão processadas em ordem crescente do *nonce*. Uma situação hipotética que ilustra isto é um dispositivo de IoT que encaminha, por exemplo, 10 leituras de batimentos cardíacos para um contrato inteligente. Estas informações, ao serem recepcionadas pela rede *Ethereum* são direcionadas para uma piscina de transações e ao serem escolhidas para um bloco, a rede verifica o campo *nonce*. Se a última transação tiver o valor de *nonce* 6 e dois nós distintos da blockchain elegem transações com *nonce* 7 e 8 na mesma rodada, a transação de *nonce* 8 retornará para as piscina até que a transação de *nonce* 7 seja validada.

Por outro lado, se o *nonce* não existisse um sensor de batimentos cardíacos coletaria um valor e enviaria para um determinado contrato inteligente na blockchain. Minutos depois, na próxima coleta, o mesmo sensor poderia coletar e enviar o mesmo valor, já em outra transação. A rede teria então de processar duas transações idênticas. Caso isto fosse possível, seria possível então que qualquer sensor ou usuário criasse outras transações, com estes valores, enviando-as para o contrato inteligente com intuito malicioso. Com o *nonce* incremental implementado, fica impossível duplicar as transações.

Os campos *data* e *value* podem ser preenchidos ou enviados em branco alternadamente. Cada uma destas combinações tem consequências diferentes, conforme exibido na Tabela 2.3:

Quando envia-se para a blockchain *Ethereum* uma transação com os campos *value*

Tabela 2.3. Significado das transações de acordo como o preenchimento dos campos *data* e *value*.

value	*data*	**Significado**
Vazio	Vazio	Não faz ação nenhuma
Preenchido	Vazio	Operação de pagamento
Vazio	Preenchido	Chamado de função
Preenchido	Preenchido	Pagamento e chamado de função

e *data* vazios, como resultado tem-se apenas o gasto de *gas* e consequentemente de *ethers*. Se a transação contiver apenas o campo *value* preenchido, isto representa um pagamento, portanto, a transferência de *ethers* entre as contas envolvidas. Já uma transação que tenha apenas o campo *data* preenchido representa um chamado, uma invocação a uma função de um contrato, por exemplo. Por fim, caso os dois campos estejam preenchidos, tem-se então um pagamento e uma chamada de função em um contrato inteligente.

2.2.7. Contratos Inteligentes

Conforme definido na Seção 2.1.3, os contratos são programas de computador imutáveis e hospedados na blockchain. As linguagens de programação para a escrita de tais contratos são de alto nível, como *LLL*, *Serpent*, *Vyper*, *Bambu*, *Python* e *Solidity*. Esta última é a mais popular, suportada pela plataforma *Ethereum* e utilizada neste capítulo.

Evidentemente que este não é um capítulo de *Solidity*, pois esta é uma linguagem poderosa e em constante evolução. No entanto, serão apresentados pontos da linguagem, como tipos de variáveis, métodos e funções com o objetivo de fornecer uma base suficiente para que os alunos possam explorar sozinhos novos conhecimentos, entender os contratos apresentados e realizar a prática ao proposta. A Tabela 2.4 lista os principais tipos de dados da linguagem.

O *Solidity* também oferece alguns recursos adicionais que facilitam a construção dos contratos. Quando uma transação é criada e enviada para a rede, algumas informações são encapsuladas na mensagem de forma automática e estão prontas para auxiliar o desenvolvedor. Estas facilidades são agrupadas em 3 categorias: `msg, block e tx`.

`msg` - O objeto `msg` é uma chamada de transação originada de um cliente *Ethereum* ou de um contrato. Ela contém uma série de atributos úteis:

- `msg.sender`: Representa o endereço que iniciou a chamada de contrato
- `msg.value`: O valor de *ether* enviado com esta chamada (em *wei*).
- `msg.gas`: A quantidade de *gas* restante no suprimento de *gas* desse ambiente de execução. Isso foi descontinuado no *Solidity* v0.4.21 e substituído pela função `gasleft()`.
- `msg.data`: A carga útil de dados desta chamada no contrato.
- `msg.sig`: Os primeiros quatro bytes da carga de dados, que é o seletor de função.

`block` - O objeto de bloco contém informações sobre o bloco atual:

Tabela 2.4. Tabela com os tipos de dados disponíveis no *Solidity*

Tipo	Descrição
`int`	Inteiros positivos ou negativos (`int`) declarados em incrementos de 8 bits (`int8`, `int16`, ... `int256`).
`uint`	Inteiros positivos declarados em incrementos de 8 bits (`uint8`, `uint16`... `uint256`).
`bool`	Valor lógico, verdadeiro ou falso, com operadores lógicos ! (não), && (e), || (ou), == (igual) e! = (diferente).
`fixed/ ufixed`	Números de ponto fixo, declarados com `(u) fixedMxN` em que M é o tamanho em bits (incrementos de 8 até 256) e N é o número de decimais após o ponto (até 18); por exemplo, `ufixed32x2`.
`address`	Usado para armazenar endereços *Ethereum* de 20 bytes. O objeto de endereço tem muitas funções membro úteis, como `balance` (retorna o saldo da conta) e `transfer` (transfere *ether* para uma conta).
`byte array (fixed)`	Matrizes de bytes de tamanho fixo, declaradas com bytes.
`byte array (dynamic)`	Matrizes de bytes de tamanho variável, declaradas com bytes ou `string`.
`enum`	Tipo definido pelo usuário para enumerar valores discretos `enum name {rotulo1, rotulo2...}`.
`array`	Um `array` de qualquer tipo, fixo ou dinâmico.
`struc`	`Conteiners` de dados definidos pelo usuário para agrupar variáveis `struct Car { String year; int color;}`.
`Mapping`	Tabelas de pesquisa de *hash* para pares chave => `mapping (key_type=>value_type)`.

- `block.blockhash(blockNumber)`: O *hash* de um bloco específico. Em desuso e substituído pela função `blockhash()` no *Solidity* v0.4.22.
- `block.coinbase`: O endereço do destinatário das taxas do bloco atual e da recompensa do bloco.
- `block.difficulty`: A dificuldade (prova de trabalho) do bloco atual.
- `block.gaslimit`: A quantidade máxima de *gas* que pode ser gasta em todas as transações incluídas no bloco atual.
- `block.number`: O número do bloco atual.
- `block.timestamp`: O carimbo de data/hora colocado no bloco atual pelo minerador.

`tx` - O objeto *tx* fornece um meio de acessar informações relacionadas à transação:

- `tx.gasprice`: o preço do *gas* na transação de chamada.
- `tx.origin`: O endereço da conta *Ethereum* de origem para esta transação. Esta é uma operação considerada insegura!

A manipulação de dados relativos aos endereços de contas *Ethereum* passados como entrada possuem também alguns métodos e atributos que auxiliam a escrita dos contratos. Os principais estão listados abaixo:

- `address.balance`: O saldo do endereço, em *wei*. Por exemplo, o saldo do contrato atual é `address(this).balance`.
- `address.transfer(quantidade)`: Transfere o valor (em *wei*) para este endereço, lançando uma exceção para qualquer erro.
- `address.send(quantidade)`: Semelhante ao `transfer`. Ao invés de lançar uma exceção, ele retorna falso em caso de erro.
- `address.call(payload)`: pode construir uma chamada de mensagem arbitrária com uma carga de dados. Retorna falso em caso de erro. Mas o destinatário pode (acidentalmente ou maliciosamente) esgotar todo o seu *gas*, fazendo com que seu contrato seja interrompido com uma exceção.

Além disto, os usuários podem criar suas próprias funções na escrita dos contratos. Elas podem ser chamadas por uma transação originada em uma carteira *Ethereum* ou em outro contrato. A sintaxe usada para declarar estas funções é a seguinte:

`function FunctionName ([parâmetros]) public|private|internal|external [pure|constant|view|payable] [modifier] [return (tipos de retorno)]`, onde:

`FunctionName` é o nome usado para chamar a função em uma transação de uma carteira *Ethereum*, de outro contrato ou de dentro do mesmo contrato. Uma função pode ser definida sem um nome. Neste caso, é a função de *fallback*, que é chamada quando nenhuma outra função é nomeada. A função de *fallback* não pode ter argumentos ou retornos.

Os parâmetros vêm após o nome, especificado os argumentos que devem ser passados para a função, com seus nomes e tipos.

O próximo atributo especifica a visibilidade da função. O padrão são funções públicas que podem ser chamadas por outros contratos, transações de carteiras *Ethereum*, ou de dentro do contrato. As funções com atributo `external` são como funções públicas, exceto que não podem ser chamadas de dentro do contrato, a menos que explicitamente prefixadas com a palavra-chave this.

As funções com atributo `internal` são acessíveis apenas de dentro do contrato ou por contratos derivados de outro contrato. Elas não podem ser chamadas por outro contrato ou transações de carteiras *Ethereum*. As funções com o atributo `private` são como funções `internal`, mas não podem ser chamadas por contratos derivados.

Lembre-se de que os termos `internal` e `private` são um tanto enganosos. Qualquer função ou dado dentro de um contrato está sempre visível na blockchain pública, o que significa que qualquer pessoa pode ver o código ou os dados. As palavras-chave descritas aqui afetam apenas como e quando uma função pode ser chamada.

O segundo conjunto de palavras-chave (`pure, constant, view, payable`) afetam o comportamento da função:

Uma função `constant` ou `view` promete não modificar nenhum estado. Os termos possuem o mesmo objetivo e o primeiro será descontinuado em uma versão futura.

Uma função `pure` é aquela que não lê nem grava nenhuma variável no armazenamento. Ele só pode operar em argumentos e retornar dados, sem referência a nenhum dado armazenado.

Uma função `payable` é aquela que pode aceitar pagamentos recebidos. Funções não declaradas como `payable` rejeitarão pagamentos recebidos.

Existem 3 tipos especiais de funções que deve-se ficar atento: construtoras, auto-destruição e modificadoras.

As funções construtoras são executadas apenas uma vez, durante a criação do contrato e possuem a palavra-chave `constructor()`. As funções de auto-destruição possuem a palavra-chave `destroy()` e são utilizadas, como o próprio nome diz, para destruir o contrato implementado. As funções modificadoras são aplicadas adicionando-se o nome do `modifier` na declaração da função. São usados para criar condições que se aplicam a muitas situações em um contrato, e para isto, basta acrescentar o seu nome na declaração de uma função.

Após escritos, os contratos precisam ser compilados para depois serem implementados em uma rede *Ethereum*. Como resultado, o processo de compilação cria os *bytecodes* e a *Application Binary Interface* (ABI), conforme ilustrado na Figura 2.8.

Figura 2.8. O contrato, após compilado, gera duas saídas - Os *bytecodes* e a ABI.

Os *bytecodes*, de baixo nível, são implementados na plataforma *Ethereum* usando uma transação de criação de contrato enviada para um endereço especial de criação de contratos. Cada contrato, portanto, possui um endereço *Ethereum*, que é derivado da transação de criação do contrato em função da conta e do *nonce* de origem. Este endereço pode ser usado, em uma transação, para receber *ethers*, por exemplo, de uma outra conta contrato ou de uma conta *Ethereum* cliente.

A ABI possui informações de como acessar as funções do contrato. Somente através dela é que as DApps podem enviar *ethers* ou dados para o contrato. Esta interface, obrigatoriamente, precisa ser importada pela DApp.

É importante acrescentar que os contratos somente executam funções se forem chamados por uma transação. Os contratos nunca podem chamar a si próprios ou atuarem em *background*, mas podem chamar outros contratos em cadeia. As transações são atômicas, e caso a execução ocorra sem erros até o final, toda a transação é registrada.

As contas que representam CI possuem diferenças em relação às contas que representam apenas uma carteira eletrônica. Enquanto nestas, uma única conta pode acessar as

diversas redes Ethereum, nas contas de CI isto não é possível. Um conta que representa um CI só pode acessar a rede na qual ela foi implementada. Para que este contrato possa ser implementado em outra rede *Ethereum* é necessário implementar uma nova instância do contrato nesta nova rede, com outro pagamento das taxas da transação.

Os CIs podem ser gerados basicamente de duas formas. Usando editores online como *Studio Ethereum* [StudioEthereum 2019], *Ethfidlle* [Ethfiddle 2017] e o *Remix* [Remix 2015], ou através de qualquer editor de texto, após configurar adequadamente um ambiente local para desenvolvimento, o qual será discutido na Seção 2.5.

O *Remix* é um ambiente online configurado para programar, compilar e implementar CIs. Além de um editor de texto integrado, ele possui diversas versões de compiladores prontos para usar. Nele, existem 3 modos de se implementar os contratos: (a) Através de uma máquina virtual JavaScript, implementada no navegador; (b) usando a `web3` injetada pelo *Metamask*; ou (c) fornecendo um endereço para conexão de um provedor `web3`.

A Figura 2.9 ilustra o ambiente de desenvolvimento do *Remix*. O contrato em tela é um exemplo didático e pode ser encontrado na página do capítulo. Este contrato é compilado na versão 7.4 do `solc` (compilador do *solidity*). Na linha 1 é informado o tipo de licença para o contrato. Em seguida, na linha 2 informa-se qual a versão do `solc` será usada para compilar o contrato.

Figura 2.9. Exemplo didático de CI utilizando o editor on-line Remix.

Entre as linhas 4 e 18 está o CI propriamente dito. A linha 4 define o nome do contrato e a linha 5 declara a variável que será utilizada. As linhas 7 e 8 declaram o construtor do contrato, que será executado uma única vez, recebendo como parâmetro uma mensagem inicial.

Por fim, temos duas funções definidas no contrato. A função `setMessage`, na linha 11, quando invocada, recebe como parâmetro uma mensagem nova e atualiza a variável do contrato. A função `getMessage`, na linha 15, retorna o valor armazenado na variável `message`. Observe que, enquanto a primeira mensagem modifica o valor de uma variável do contrato, a segunda apenas lê o seu conteúdo.

Isto implica que a função `setMessage`, quando for invocada, vai gerar custos para executar a transação e será necessário desembolsar *ethers* da carteira para que a transação seja completada. Já a função `getMessage` não tem custo nenhum para ser executada.

A função `getMessage` está presente neste contrato apenas como exemplo didático. Todas as variáveis declaradas no contrato, automaticamente, terão uma função `get` associada no momento da compilação.

2.3. Desafios da Blockchain e IoT na área de Saúde

O setor de saúde tem particularidades associadas à segurança e privacidade devido aos requisitos legais para proteger as informações médicas dos pacientes. A Internet compartilha registros e dados armazenando-os na nuvem. Com a adoção de dispositivos móveis na saúde, o risco de ataques maliciosos e de informações privadas serem comprometidas à medida que são compartilhadas tornam-se, portanto, evidentes.

As informações ficam mais fáceis de serem obtidas, principalmente por meio do uso de componentes de IoT acoplados a pacientes ou a equipamentos médicos, portanto o compartilhamento e a privacidade destas informações são uma preocupação adicional.

É neste cenário que a blockchain e IoT precisam se engajar, proporcionando um ambiente adequado para celebrar esta união. Para explorar melhor estes conceitos, divide-se esta seção em três partes: Principais requisitos, Desafios técnicos e Protocolos de Consenso.

2.3.1. Principais Requisitos

A literatura classifica os dados de saúde em dois grupos básicos: Registros Pessoais de Saúde (PHR, do inglês Personal Health Record) e Registros Eletrônicos de Saúde (EHR, do inglês Electronic Health Records), que são controlados por hospitais, e não por pacientes. Estas informações possuem requisitos exclusivos da área de saúde que são evidenciados aqui. Segundo [McGhin e outros 2019], são: Controle de acesso, autenticação e não repudiação; Interoperabilidade; Compartilhamento de dados; e Mobilidade.

Controle de Acesso, Autenticação e não repudiação

Os dados médicos podem ser obtidos por intermédio de sensores corporais ou registros médicos com informações sobre o paciente. Estes registros, quando em formato digital, precisam garantir itens de segurança como integridade, não repudiação, confidencialidade e disponibilidade. Isto permite, por exemplo, que os pacientes armazenem e compartilhem com segurança seus EHRs em um servidor na nuvem para que médicos ou cuidadores acessem. Os médicos podem encaminhar o prontuário dos pacientes a outros especialistas para diagnósticos e pesquisas, sempre que necessário, garantindo que as informações dos pacientes permaneçam privadas [Yüksel e outros 2017, Au e outros 2017].

Interoperabilidade

Segundo [Azaria e outros 2016], o processo de compartilhamento e transferência de dados entre diferentes fontes é a definição para interoperabilidade. Entre as principais limitações que dificultam este processo está o emprego de armazenamento centralizado dos dados médicas em bancos de dados. Outras fontes que atuam neste sentido são diferenças de padrões entre sistemas e a legislação de proteção de dados entre diferentes nações.

Compartilhamento de dados

O PHR/EHR estão dispersos em clínica, hospitais e laboratórios. Isto impede que informações abrangentes e atualizadas sobre os pacientes possam ser compartilhadas [Roehrs e outros 2017]. Paralelamente, a ausência de um identificador comum entre as bases é outro fator que dificulta o compartilhamento dos dados.

Mobilidade

A questão da mobilidade está relacionada com aplicações de saúde móveis, dispositivos de IoT e redes sem fio. Existem muitas soluções que aplicam o conceito de aplicações móveis em saúde, usando massivamente as redes sem fio e os dispositivos de IoT, mas não possuem conhecimentos adequados para isto [Kotz e outros 2016]. Entre os principais requisitos estão a disponibilidade da rede, autenticação de dados, confiabilidade e localização. Os dispositivos inteligentes e sensores que registram e enviam dados vitais de saúde ao médico para visualização e avaliação remotas das condições, como por exemplo relógios inteligentes, lentes de contato, pulseiras de fitness, microchips sob a pele e sensores sem fio, às vezes, não se preocupam tanto com a segurança [Zhang e outros 2017].

2.3.2. Desafios Técnicos

A seguir, listamos alguns desafios técnicos da tecnologia blockchain, quando empregada na área de saúde [De Aguiar e outros 2020, Hoy 2017, Yli-Huumo e outros 2016, McGhin e outros 2019]:

Latência O processo para validar um bloco na plataforma Ethereum leva cerca de 15 segundos. Este tempo de espera pode ser prejudicial uma vez que os sistemas de saúde são dinâmicos e devem ser acessados o tempo todo. Uma alternativa seria o uso de blockchains permissionadas como a Hyperledger, cujo tempo necessário para geração de blocos é determinístico, com base na latência da rede, que deve operar na casa de milissegundos [Cachin e outros 2016].

Vazão Os sistemas de saúde, em alguns casos, necessitam de alto rendimento com um tempo de resposta muito curto, pois isto pode afetar negativamente um diagnóstico que e pode envolver vidas. Com o incremento do número de transações e o tempo de bloqueio para plataformas que usam algum tipo de prova, em especial o PoW descrito na Seção 2.3.3), a vazão pode ser um fator proibitivo para algumas aplicações.

Consumo de energia: Para blockchains que empregam o protocolo PoW, é preciso ponderar o alto consumo energético durante o processo de mineração dos blocos. A tendência é a adoção, por parte de algumas plataformas, de protocolos de consenso mais eficientes sob este ponto de vista ou de então de alternativas a protocolos que não usem provas para alcançar o consenso, como a plataforma IOTA [Silvano e Marcelino 2020, Popov e outros 2020].

Centralização: Alguns protocolos de consenso, apesar dos esforços e mecanismos para manter a justiça, tendem a centralizar os mineradores, e como resultado, isso reduz o nível de confiabilidade da rede. [Zheng e outros 2018, De Aguiar e outros 2020].

Privacidade: As blockchains públicas fornecem um certo grau de anonimidade e privacidade, mas todos os seus registros são públicos e auditáveis. Devido às leis e regulamentações de privacidade, os sistemas baseados em blockchain devem estar em conformidade com o Regulamento Geral de Proteção de Dados (GDPR). Uma possível alternativa seria o emprego de redes permissionadas, onde os dados armazenados na blockchain não são públicos.

2.3.3. Protocolos de Consenso

O consenso é um problema fundamental em computação distribuída e permite com que um conjunto de participantes (ou nós) numa rede chegue a um acordo sobre um conjunto de transações, ou sobre um determinado estado do sistema, apesar da ocorrência de falhas ou da presença de nós maliciosos, que podem subverter o sistema [Greve e outros 2018].

O consenso portanto mantém o estado consistente das réplicas e a disponibilidade do sistema. No contexto da saúde, o consenso da blockchain precisa ser bem elaborado para atender aos requisitos acima enumerados. Desta forma, as aplicações que envolvem dados de saúde e que contemplam dispositivos de IoT podem ser resolvidas com a introdução de um mecanismo de consenso distribuído adequado.

Esta seção apresenta os protocolos de consenso PoW, PoS, Variantes do PoW e PoS, BFT e Grafo Direcionado Acíclico (DAG, do inglês (*Directed Acyclic Graph*), ilustrados na figura 2.10. Onde for possível, correlaciona-se estes protocolos com aplicações na área de saúde [Al Omar e outros 2017, Ramachandran e outros 2020, Azaria e outros 2016, Patel 2019]. O termo PoX (*Proof of Somethings*) é uma forma de referir-se genericamente aos protocolos que necessitam de alguma prova para alcançar o consenso [Lao e outros 2020].

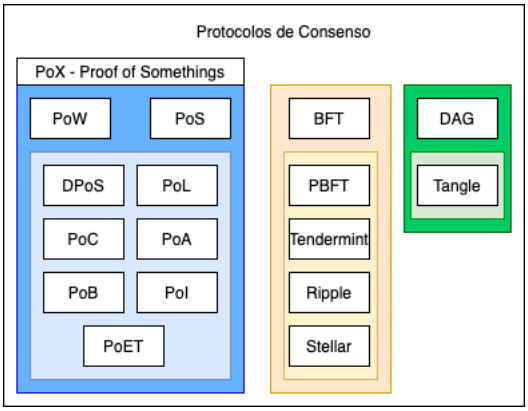

Figura 2.10. Diagrama com os pricipais protocolos de consenso.

2.3.3.1. Prova de Trabalho - Proof of Work (PoW)

O protocolo de consenso *Proof of Work* (PoW) surge com a blockchain do Bitcoin, no famoso artigo [Nakamoto 2008]. Desde então, diversas variações apareceram. No geral, o PoW adota a seguinte estratégia: cada nó da rede precisa resolver um desafio computacional para poder propor à rede um bloco de transações. Assim, através de um mecanismo de competição, em um processo exaustivo, o nó que resolver o quebra-cabeça matemático obterá uma recompensa na forma de criptomoeda, o bitcoin. Após a formação do bloco, o nó irá encaminhá-lo à rede, e todos os nós irão agregá-lo a uma "blockchain", estrutura de dados contendo toda a cadeia de blocos até então acordada pelos nós, de tal forma que o bloco recentemente transmitido aponta para o anterior.

Desta forma, observa-se dois princípios básicos que fazem o consenso PoW funcionar [Lao e outros 2020]:

(i) *A regra da cadeia mais longa*: o nó considera a cadeia mais longa como a cadeia certa. Isso porque, como mais de um nó pode resolver o puzzle ao mesmo tempo, mais de um bloco é transmitido na rede para estender a cadeia. Por princípio, os nós irão sempre estender a cadeia mais longa, e portanto, após um tempo, todos estarão com a mesma estrutura de blockchain, obtendo-se assim o acordo.

(ii) *A regra de incentivo*: um nó será recompensado ao encontrar um bloco adequado. Desta forma, os nós estarão motivados a despender recursos computacionais participando da competição (ou mineração de blocos).

Estas premissas são a base de funcionamento da rede Bitcoin. Elas garantem a exatidão e exclusividade da cadeia de blocos, evitando o duplo gasto e a manipulação da cadeia de blocos (ou livro razão) por um nó malicioso. Evidentemente, há outros desafios a serem tratados em um sistema complexo que movimenta ativos digitais tão valiosos. Para uma melhor base sobre esses elementos de segurança, recomendamos o livro [Narayanan e outros 2016].

O MedRec oferece aos pacientes registros imutáveis e de fácil acesso em locais de tratamento, gerenciando autenticação, confidencialidade, responsabilidade e compartilhamento de dados. Utiliza o fornecimento de dados agregados e anônimos como recompensas para pesquisadores, autoridades de saúde públicas, hospitais e clinicas que aceitem ser mineradores de uma rede baseada em PoW [Azaria e outros 2016].

O MedBChain é um sistema de gerenciamento de dados de saúde centrado no paciente usando Blockchain com base em PoW como armazenamento para obter privacidade. O pseudo anonimato é garantido pelo uso de funções criptográficas para proteger os dados do paciente [Al Omar e outros 2017].

A validade dos EHRs encapsulados na blockchain, com uso de PoW, empregando um esquema de assinatura baseada em atributo com várias autoridades, no qual um paciente endossa uma mensagem de acordo com um atributo, sem divulgar nenhuma informação, além da evidência que ele atestou. é descrito em [Guo e outros 2018].

A implementação de uma arquitetura de informação em larga escala para acessar (EHRs) com base em Contratos Inteligentes como mediadores de informação é explicado em [da Conceição e outros 2018], baseado em uma arquitetura de blockchain que também emprega o protocolo PoW.

2.3.3.2. Prova de Participação - *Proof of Stake* (PoS)

Prova de Participação (ou Prova de Posse) - *Proof of Stake* (PoS) [Kiayias e outros 2017] é um dos algoritmos em ascensão para muitas aplicações de blockchain. Após anos de uso do PoW, algumas desvantagens ficaram evidentes, como segurança (ataques de duplo gasto possíveis), alto consumo energético e desperdício de recursos (no processo de competição/mineração) ou baixa vazão - *throughput* (pouca quantidade de transações acordadas no tempo).

De forma simplista, o PoS baseia-se na hipótese de que os usuários com a posse de mais moedas (ou recursos computacionais) são mais propensos a garantir a confiabilidade

do sistema e têm menos probabilidade de se comportar como nós maliciosos. Assim, o algoritmo PoS considera a porcentagem do número total de moedas (ou recursos) que um nó envolvido na competição detém, e eventualmente considera o tempo que o nó leva com o montante de moedas (ou recursos) para estabelecer uma porcentagem de direito de participação no consenso. Quanto mais moedas (ou recursos), mais probabilidade o nó terá de participar do consenso para decidir sobre os blocos.

A fim de permitir que cada bloco seja gerado mais rapidamente, o mecanismo PoS elimina o processo exaustivo de resolução do quebra-cabeça criptográfico. Mas, há problemas que também tonaram-se ou podem se tornar evidentes, como aconteceu com o PoW. Por exemplo, os usuários com mais moedas por um longo período têm maior possibilidade de serem selecionados pelo sistema para gerar o próximo bloco, ocasionando um elitismo e centralização das decisões.

As referência [Patel 2019] apresenta um *framework* para compartilhamento de imagens entre domínios que usa um blockchain, com base em PoS, como um armazenamento de dados distribuído, para estabelecer um livro-razão de estudos radiológicos e permissões de acesso definidas pelo paciente.

2.3.3.3. Variantes do PoW e PoS

Existe uma coletânea de protocolos que alcançam o consenso exigindo recursos computacionais de cada nó participante da rede, empregando mecanismos probabilísticos específicos e sem obrigação de informações completas sobre as operações do nó no sistema. A premissa é que há mais nós benignos, os quais podem ter mais recursos. Dentre estes protocolos, estão:

Prova de Participação Delegada - DPoS (*Delegated Proof of Stake*)

O DPoS [Larimer 2014] resolve o problema de centralização através da introdução do mecanismo de delegação. Os nós da rede elegem nós especiais, os super nós ou delegados, que passam a gerar e assinar os blocos. Com esta estratégia, o tempo de se confirmar uma transação melhora, pois o protocolo DPoS elimina a necessidade de se aguardar a confirmação de nós não confiáveis. Parece um paradoxo tentar centralizar decisões em um sistema descentralizado, no entanto qualquer nó pode ser alçado à condição de delegado. Quando um deles viola quaisquer regras do protocolo, seus direitos são negados e outro delegado será eleito.

Como exemplo de uso do DPoS temos as plataformas *BitShares* [1], *Steem* [2] e EoS [3], que apresentam 101, 21 e 21 delegados, respectivamente.

Comparado com o PoW, o DPoS é mais rápido e eficiente. Além disso, é mais democrático e flexível do que o PoS.

Prova de Sorte - PoL (*Proof of Luck*)

Este protocolo emprega funções TEE(*Trustworthy Execution Environment*) para

[1] https://wallet.bitshares.org/#/

[2] https://steem.com/

[3] https://eos.io/

fornecer justiça de mineração, segurança de tempo e certeza da identidade do nó. Surge como uma alternativa ao PoW, que exige cada vez mais poder de processamento e consumo de energia. Através da geração de um número aleatório executado em um TEE, estas funções bloqueiam as plataformas que podem ser usadas para processamento das operações como forma de limitar o poder desigual da computação. Após a geração do número, o PoL escolhe um líder para o consenso, e, com isto, obtém-se economia no consumo de energia, baixa latência para confirmação de transações e equidade na mineração [Milutinovic e outros 2016].

Prova de Capacidade ou Prova de Espaço - PoC (*Proof of Capacity* ou *Proof of Space*)

O PoC [Dziembowski e outros 2015] usa o espaço disponível no disco rígido para definir privilégios ao invés do poder computacional dos nós concorrentes. A probabilidade de propor um bloco é proporcional ao espaço de armazenamento cedido à rede por um nó minerador. Quanto maior a capacidade de armazenamento em disco, maior o domínio sobre o consenso.

Prova de Atividade - PoA (*Proof of Activity)*

Os algoritmos de PoA contam com um conjunto de N nós confiáveis chamados de autoridades. Cada autoridade é identificada por um único id e a maioria delas é considerada honesta, ou seja, pelo menos N / 2 + 1. As autoridades chegam a um consenso para ordenar as transações emitidas pelos clientes. O consenso em algoritmos de PoA depende de um esquema de rotação de mineração, uma abordagem amplamente usada para distribuir de forma justa a responsabilidade da criação de blocos entre as autoridades. O tempo é dividido em etapas, cada uma das quais tem uma autoridade eleita como líder de mineração [De Angelis e outros 2018].

As principais implementações de PoA são o *Clique* e o *Aura*. Ambas têm um primeiro turno onde o novo bloco é proposto pelo líder atual (proposta de bloco); então o *Aura* requer uma nova rodada (aceitação do bloco), enquanto o *Clique* não.

Prova de Queima - PoB (*Proof of Burn*)

Neste protocolo de consenso, os mineradores devem comprovar que queimaram algumas moedas, enviando-as para alguns endereços onde não podem ser gastos. A quantidade dessas moedas destruídas determina a probabilidade de um minerador emitir um novo bloco. O PoB funciona como uma espécie de mineração virtual, queimando moedas virtuais [Frankenfield 2018].

Prova de Importância - PoI (*Proof of Importance*)

Empregando o conceito de importância, este protocolo consegue medir a capacidade de uma conta de minerar um bloco. Para isto, a quantidade de moedas que possui e o número de transações realizadas são considerados. Há uma certa similaridade com o PoS, sob o ponto de vista do saldo em criptomoedas quando se observa o uso do saldo como critério decisivo para eleger nó, no entanto há de se observar que no PoI também se considera o volume de transação realizada. Para minerar um blocoos nós devem realizar transações ativamente. Ao aplicar PoI, a blockchain ganha vantagens de eficiência energética e alta taxa de transação [Bach e outros 2018].

Prova de Tempo Decorrido - PoET (*Proof of Elapsed Time*)

O algoritmo de consenso PoET [PoET 2018] faz com que os nós eleitos estocasticamente aguardem um tempo de espera aleatório criado pelo sistema. O nó que primeiro esgotar o tempo será eleito o líder para a criação do novo bloco. O PoET é um algoritmo semelhante a uma loteria que atende à justiça, ao investimento e à verificação. Para evitar trapaças, dois requisitos precisam ser verificados: O primeiro é que o líder realmente espera por um tempo aleatório em vez de um curto período de tempo para vencer. O segundo é que o líder realmente espera pelo tempo de espera determinado pelo protocolo.

2.3.3.4. Tolerância a Falhas Bizantinas - BFT(*Byzantine Fault Tolerance*)

Os protocolos baseados em BFT pertencem a uma classe que conseguem obter um acordo em um sistema, onde os processadores podem falhar de forma arbitrária, denominado Problema Geral Bizantino [Lamport e outros 1982]. A seguir, define-se os seguintes protocolos baseados em BFT: PBFT, *Tandermint*, *Ripple* e *Stellar*.

PBFT (*Practical Byzantine Fault Tolerance*)

O PBFT [Castro e outros 1999] foi o primeiro algoritmo prático a tolerar falhas bizantinas e adaptou-se para ser usado em ambientes assíncronos. O *BFT-Smart* é um outro projeto promissor em bom estágio de maturidade [Bessani e outros 2014]. O PBFT é oferecido pelo Hyperledger Fabric como camada de acordo (ordenação de transações). Além disso, o *BFT-Smart* [Sousa e outros 2018] também foi recentemente incorporado ao projeto [Greve e outros 2018].

A referencia [Dubovitskaya e outros 2017] propõe uma estrutura para gerenciar e compartilhar dados para atendimento a pacientes com câncer, apresentando um protótipo que garante privacidade, segurança, disponibilidade e controle de acesso refinado sobre os dados em uma blockchain com protocolo de consenso PBFT.

O AuditChain é um protótipo que aproveita a tecnologia de blockchain do Hyperleger Fabric para resolver problemas de interoperabilidade, conteúdo, estrutura e consolidação de log de auditoria. Especificamente, usa o livro razão e contratos inteligentes para padronizar o conteúdo, simplificar o acesso e garantir que os logs de auditoria contenham todas as informações necessárias e úteis [Anderson 2018].

O MedChain oferece uma solução de blockchain, com base no PBFT, e armazenamento distribuído para EMRs e informações de saúde protegidas através de uma arquitetura extensível [Sandgaard e Wishstar 2018].

O Medicalchain permite que o paciente forneça aos profissionais de saúde acesso aos registros e exames médicos de forma auditável, transparente e segura empregando tokens chamadas MedTokens [Albeyatti 2018].

Tendermint

O *Tendermint* [Kwon 2014] é um protocolo de consenso BFT quase assíncrono, baseado em validadores, propondo blocos de transações e votando neles. Ele requer apenas duas rodadas de votação para chegar a um consenso. Em cada rodada, há três etapas (ou seja, propor, prevenir, pré-comprometer). Quando mais de 2/3 dos votos pré-comprometidos forem recebidos para alcançar o consenso em uma rodada, o consenso

para a próxima rodada começará.

Ripple

O *Ripple Protocol Consensus Algorithm* (RPCA) [Todd 2015] utiliza sub-redes confiáveis coletivamente dentro da rede maior para chegar a um consenso para o Problema Geral Bizantino. No Ripple, a Unique Node List (UNL) é um conjunto de outros servidores mantidos por cada servidor, que desempenha um papel importante quando um servidor faz consultas para determinar o consenso. Apenas os votos dos servidores na UNL são considerados na determinação do consenso. Esta é uma diferença óbvia de muitos algoritmos de consenso. A UNL representa um subconjunto da rede que exige sabedoria coletiva para chegar a um consenso. A premissa da RPCA é que cada servidor confia nos outros servidores da UNL e acredita que eles não entrarão em conluio. A RPCA procede em várias rodadas para chegar a um consenso. Em cada rodada, cada servidor primeiro coleta o máximo de transações para se preparar para o consenso e torná-las públicas na forma de "conjunto de candidatos". Em seguida, cada servidor faz uma união dos conjuntos candidatos dos servidores em seu UNL e vota em cada transação. De acordo com o resultado da votação, as transações que obtiverem votos abaixo de um percentual mínimo serão descartadas ou colocadas em candidatos definidos no próximo consenso para o próximo bloco do livro-razão, enquanto aqueles que obtiverem votos suficientes irão para o próximo turno [Wu e outros 2019]

Stellar

O *Stellar Consensus Protocol* (SCP) [Mazieres 2015] é um protocolo do acordo bizantino federado (FBA). Ele é considerado o primeiro mecanismo de consenso comprovadamente seguro a desfrutar simultaneamente de quatro propriedades principais: controle descentralizado, baixa latência, confiança flexível e segurança assintótica. Segurança assintótica significa que a segurança do SCP depende de assinaturas digitais e famílias de *hash* cujos parâmetros podem ser ajustados de forma realista para proteger contra adversários com um poder de computação inimaginavelmente vasto.

2.3.3.5. Grafo Direcionado Acíclico - DAG (*Directed Acyclic Graph*)

Existe uma categoria de protocolos, a exemplo do Dagcoin [Lerner 2015] e do Tangle [Popov 2018], que apresentam uma estratégia visando explorar o paralelismo do sistema tradicional de blockchain de cadeia única, e empregam uma estrutura de dados de grafo direcionado acíclico para conectar blocos. O mecanismo de consenso, distinto dos demais abordados anteriormente, consiste em que cada transação fique vinculada aos dois registros de transações anteriores através do grafo. Desta forma, a conformidade da transação atual pode ser comprovada referenciando-se às transações anteriores. Se comparado com outros protocolos de consenso que estabelecem algum tipo de prova, observa-se que o DAG preocupa-se apenas com as transações vinculadas, constituindo um modo bem mais simples do que as diferentes provas a que se submetem os nós. O IOTA é uma plataforma de blockchain que através do Tangle utiliza este conceito.

2.3.3.6. Características de protocolos de consenso para IoT

Os dispositivos de IoT possuem limitações computacionais, energéticas e restrições de armazenamento de dados. Os protocolos de consenso precisam, além destas características, de um ambiente distribuído para garantir validade e consistência. Atingir este equilíbrio é o desafio para tornar este casamento duradouro e estável. A alta eficiência energética e um processo de consenso leve podem atenuar estes problemas.

Aplicações que envolvem blockchain-IoT precisam então driblar tais limitações e herdar os benefícios da blockchain. Se as restrições dos dispositivos de IoT forem premissas, nem os clientes leves nem os mineradores são indicados.

A blockchain IOTA é um exemplo de plataforma desenvolvida para IoT. Ela adota o consenso Tangle, baseado em DAG. Esta rede não possui mineradores, portanto o consumo energético é mais eficiente. Cada nó participante desta blockchain que necessita criar/enviar transações, primeiramente deve participar ativamente do processo de consenso aprovando duas transações anteriores.

A rede no grafo Tangle é composta por nós que são entidades que emitem e validam transações, e cada nó também representa uma transação [Popov e outros 2020]. Para que um nó adicione uma transação à rede, primeiro ele deve escolher duas transações para que possa aprová-las, conforme o algoritmo *Markov Chain Monte Carlo* (MCMC); Em seguida, o nó verifica se as transações escolhidas estão em conflito. Se isto ocorre, o nó deve desaprovar as transações conflitantes, e assim prevenir o gasto duplo; o próximo passo é resolver uma espécie de prova de trabalho (PoW), encontrando um valor *nonce*, tal que, seu *hash* seja concatenado com alguns dados da transação aprovado. É necessário destacar que este esforço computacional é uma versão muito mais leve do que a realizada pelos protocolos PoW tradicionais; Após conseguir este valor, o usuário envia sua transação para a rede, e ela se torna um *tip* (transação não aprovada); Por fim, o *tip* aguarda a confirmação por meio de aprovação direta ou indireta até que seu peso acumulado atinja o limite predefinido.

2.4. Pesquisas e Aplicações Recentes

O emprego de blockchain, contratos inteligentes e IoT oferecem novas possibilidades tecnológicas na saúde. A literatura apresenta muitas pesquisas e sistemas que exploram esta convergência. Esta seção apresenta dez classificações de pesquisas e aplicações recentes nesta área, exemplificando-as. Em seguida, apresenta-se a RNDS, uma das maiores redes de saúde com blockchain em operação no planeta.

2.4.1. Aplicações de blockchain na área de saúde

Segundo Ahmad [Ahmad e outros 2021], as aplicações e pesquisas em blockchain podem ser classificadas de acordo com a Figura 2.11. A seguir, comenta-se sobre cada uma delas.

Gerenciamento de dados clínicos do paciente

A eficácia do atendimento e do monitoramento da saúde de pacientes à distância depende da integridade e da manutenabilidade de seus registros digitais de acompanhamento médico, que incluem, entre outros: imagens, prescrições de medicação, análi-

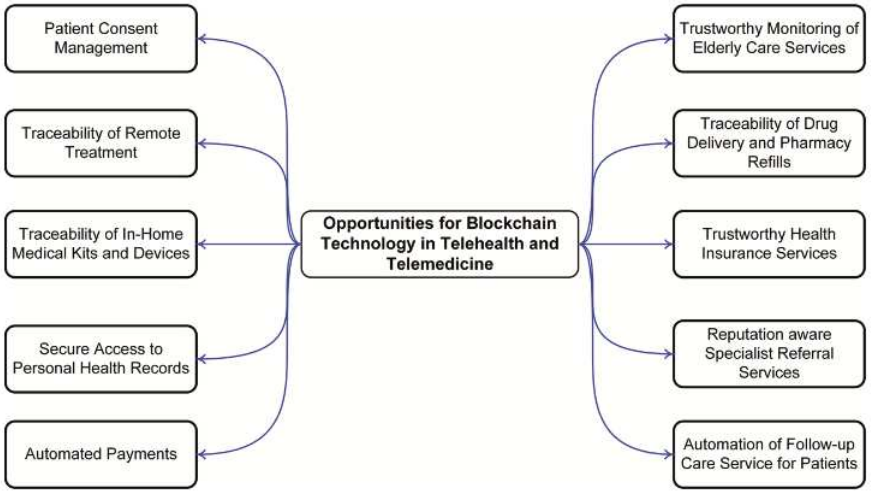

Figura 2.11. Diagrama com as principais oportunidades de aplicação da tecnlogia blockchain na área de saúde. Fonte: [Ahmad e outros 2021]

ses e históricos médicos, planos de tratamento e resultados. Esses registros são informações altamente confidenciais que precisam ser compartilhadas com segurança entre profissionais da área da saúde (médicos, farmacêuticos, hospitais, etc.) [Albeyatti 2018, Saweros e Song 2019]. Os sistemas que gerenciam tais registros enfrentam vários desafios, como por exemplo: na limitação em conduzir testes confiáveis de auditoria e na confiabilidade dos servidores (de terceiros) que armazenam tais dados. As características de imutabilidade, rastreabilidade e transparência oferecidas pela blockchain podem ajudar a conduzir testes de auditoria que verifiquem a conformidade dos dados clínicos dos pacientes. Igualmente, a tecnologia de blockchain pode ajudar a reforçar o nível de confiabilidade na manipulação e proteção de tais informações, uma vez que a terceira parte de confiança deste processo pode ser eliminada.

Rastreabilidade de tratamento remoto

O acompanhamento remoto de pacientes utiliza plataformas computacionais que permitem a troca de dados digitalizados que ajudam especialistas na área de saúde a diagnosticar o estado clínico de seus pacientes, auxiliando-os em suas consultas e procedimentos cirúrgicos [Mannaro e outros 2018, Hussien e outros 2021]. Nos sistemas de telemedicina existentes, clínicas, hospitais e especialistas não são capazes de compartilhar as bases de dados de seus pacientes. Para superar esse problemática, a tecnologia blockchain pode fornecer uma visão única e coerente das informações clínicas de pacientes em todos os nós participantes da cadeia de dados. A visibilidade e a transparência dos registros de saúde permitem que os participantes da blockchain rastreiem o histórico médico de cada paciente em análise, de forma a se produzir um tratamento adequado, com uma visão unificada dos especialistas envolvidos. A blockchain também pode permitir que auditorias sejam realizadas para descobrir quem acessou certos registros clínicos, e quais as transações foram realizadas nestas consultas.

Rastreabilidade de kits e dispositivos médicos residenciais

A adoção de kits e dispositivos de teste para uso domiciliar auxiliam no tratamento precoce de doenças, reduzindo os custos associados a exames médicos que antes

eram realizados apenas em laboratórios e hospitais [Weissman e outros 2018]. Tais dispositivos ajudam o paciente, em sua casa, a realizar auto-exames que podem o ajudar a precocemente detectar níveis indesejados de certas substâncias em seu sangue, como por exemplo, na avaliação do nível de glicose diretamente relacionado à variação de corrente elétrica resultante de reações eletroquímicas em dispositivos de sensoriamento digital (glicosímetros) [Sobreira 2005]. Entretanto, em sistemas de telessaúde centralizados tradicionais, a falta de transparência, visibilidade e proveniência de alguns fabricantes de tais kits, leva pacientes e médicos a adotarem apenas dispositivos de empresas farmacêuticas mundialmente renomadas. Esta problemática poderia ser atenuada pelo uso da blockchain, uma vez que esta tecnologia permite registrar, de forma imutável e transparente, as transações relacionadas à propriedade e ao desempenho de tais kits no livro-razão distribuído da cadeia. Contratos inteligentes poderiam ser usados para registrar pontuações de reputação para todos os dispositivos médicos residenciais, com base em suas avaliações de desempenho. Estas informações poderiam ajudar pacientes e médicos na escolha de dispositivos precisos e confiáveis (não necessariamente construídos por fabricantes renomados), segundo as informações presentes na blockchain.

Acesso seguro a registros pessoais de saúde

Em sistemas usados para oferecer serviços virtuais de armazenamento de informações de saúde, um paciente geralmente possui: um registro de acompanhamento médico, que contém informações clínicas mais detalhadas, devendo ser criado e gerenciado por laboratórios e/ou hospitais (anteriormente discutido), e um registro de saúde pessoal, que armazena seu histórico clínico, devendo ser criado e mantido pelo próprio paciente [of Health e outros 2008]. Tais sistemas são geralmente baseados em plataformas em nuvem, que são menos confiáveis devido ao fato de serem gerenciadas por uma única entidade certificadora. A natureza descentralizada da tecnologia blockchain permite que o proprietário dos dados médicos mantenha a privacidade destes. Contratos inteligentes podem registrar e autorizar usuários a acessar tais informações, em conformidade com a política de consentimento de seus proprietários [Shahnaz e outros 2019, Guo e outros 2019].

Pagamentos automatizados

Planos de saúde geralmente empregam serviços (centralizados) terceirizados para liquidar os pagamentos e dividendos de seus clientes (clínicas, médicos e pacientes). Entretanto, estes procedimentos normalmente são não-transparentes, relativamente lentos, vulneráveis a ataques cibernéticos, e geralmente caros, não sendo viáveis à pagamentos com baixos valores financeiros. A blockchain pode ser utilizada como plataforma para a realização de micro-pagamentos no setor da telessaúde, através da transferência direta de tokens de criptomoeda de forma rápida, segura, transparente, auditável e sem a necessidade de serviços de mediação central [Albeyatti 2018, Halamka e outros 2019]. Além disso, na blockchain as transações financeiras são assinadas digitalmente, o que garante às partes envolvidas a não-repudiação futura das transações realizadas. A tecnologia blockchain também pode ser usada para reduzir as chances de fraude em processos de "pagamento na entrega", como por exemplo, na operacionalização do serviço de entrega remota de medicamentos por tele-farmácias, onde contratos inteligentes podem ser programados para manter e transferir os tokens de criptomoeda para a carteira da farmácia apenas quando os produtos são recebidos com sucesso pelo paciente, em sua residência.

Monitoramento confiável de serviços de atendimento a idosos

Graças aos avanços tecnológicos proporcionados por arquiteturas computacionais embarcadas e pelo conceito de IoT, pacientes idosos podem permanecer em suas residências e serem monitorados remotamente por biossensores acoplados em seus corpos [Kazmi e outros 2019, Salah e outros 2020]. Tais dispositivos podem continuamente armazenar, processar e enviar dados de pacientes (temperatura corporal, indicadores de pressão arterial, etc.), ajudando profissionais da saúde a analisar e tomar decisões em relação aos seus estados clínicos. De forma a otimizar este serviço de atendimento domiciliar, a tecnologia blockchain e contratos inteligentes poderiam ser os responsáveis por proativamente notificar médicos e farmácias para a compra de medicamentos (quando necessário), e por disparar alertas a hospitais/centros de saúde (em situações de emergência).

Rastreabilidade da entrega de medicamentos

Os mecanismos de distribuição e renovação de receitas médicas entre profissionais da saúde, pacientes e farmácias podem ser realizados através do uso da blockchain auxiliada por contratos inteligentes. Farmacêuticos cadastrados podem acessar a prescrição de medicamentos armazenada na blockchain para verificar, preparar e enviar a encomenda aos respectivos pacientes. Contratos inteligentes podem periodicamente disparar a renovação de receitas e novos pedidos de envio de medicamentos às farmácias parceiras, segundo a especificação e os critérios pré-definidos na prescrição receitada pelo médico responsável pelo paciente. Em resposta, a farmácia pode autenticar e validar a atualização da receita, enviar os medicamentos e atualizar os registros de saúde do paciente com tais informações. Os medicamentos em trânsito podem ser rastreados pela farmácia e pelo paciente, que poderão acompanhar o trajeto realizado pela encomenda. Por fim, médicos e pacientes podem verificar a legitimidade de um medicamento por meio da análise da proveniência de seus dados [Thatcher e Acharya 2018].

Serviços de plano de saúde confiáveis

Devido a políticas rígidas de preservação da privacidade, pacientes não possuem o hábito de informar os dados de seus registros médicos às seguradoras de seus planos de saúde. Como consequência, podemos observar cotidianamente vários tipos de fraudes realizadas por pacientes mal-intencionados, que não informam as suas doenças pré-existentes, ou ainda, que realizam pedidos de indenização médica de exames e especialidades não previstas em suas apólices de seguro. A tecnologia blockchain pode ajudar os planos de saúde a minimizar tais fraudes através do acesso aos registros médicos digitais de seus clientes (com base em consentimento). Como incentivo a esta prática, várias seguradoras solicitam acesso a estas informações e oferecem em contrapartida tokens de criptomoeda a clientes que mantenham um seu estilo de vida saudável, através, por exemplo, de visitas a academias de ginástica (dispositivos inteligentes presentes na academia, ou conectados ao paciente, podem realizar transações na blockchain para a persistência de tais informações) [Raikwar e outros 2018, Albeyatti 2018].

Serviços especialistas de recomendação baseados em reputação

A telemedicina permite que dados remotos de pacientes possam ser acessados e analisados à distância por especialistas multidisciplinares [Lee 2019]. Em uma solução baseada em blockchain, o provedor de assistência médica pode armazenar os documentos

de referência dos pacientes em um servidor distribuído, que retorna o *hash* (dos registros) a serem armazenados na blockchain. Por meio do *hash* armazenado na cadeia, é possível se identificar se o documento armazenado no servidor distribuído foi alterado. O médico pode examinar o relatório de saúde do paciente para, em seguida, armazená-lo no livro-razão da blockchain.

Automação do serviço de acompanhamento de pacientes

O serviço de acompanhamento virtual permite que médicos monitorem remotamente a saúde de seus pacientes. Em certos casos, este serviço exige que o paciente compartilhe os relatórios de seus exames de sangue e de urina antes da realização da consulta virtual. A tecnologia blockchain pode automatizar este serviço por meio de contratos inteligentes que poderiam, por exemplo, disparar lembretes ao paciente para a submissão destes exames no sistema, ou ainda, para que ele não se esqueça do dia do encontro remoto com o seu médico [Siyal e outros 2019]. Além disso, ao usar servidores distribuídos que hospedam relatórios de exames clínicos, o paciente pode usar um contrato inteligente para registrar e compartilhar o *hash* de seu registro digital com o seu médico, que poderá então acessar os relatórios de saúde de seu paciente de uma forma transparente e segura.

2.4.2. Rede Nacional de Pesquisa

A Rede Nacional de Dados em Saúde (RNDS) é a plataforma nacional de interoperabilidade de dados em saúde. Um projeto estruturante do Conecte SUS, programa do Ministério da Saúde (MS) do Governo Federal, cujo objetivo é promover a troca de informações entre os pontos da Rede de Atenção à Saúde, permitindo a transição e continuidade do cuidado nos setores público e privado [da Saúde 2020].

Para que isto seja possível, existe integração das seguintes informações: resumo de atendimento; sumário de alta; imunização; medicamentos dispensados e exames realizados. Essa integração é possível devido à padronização de interoperabilidade, por meio de padrões médicos (SNOMED-CT, TISS, DICOM, LOINC, ISBT, 128, CID, CIAP-2, TUSS, CBHPM); padrões arquiteturais (FHIR - RES, REST, JSON); padrões de interoperabilidade (HL7 FHIR) e padrões dados dos pacientes (IHE PIX).

Isso promove a quebra dos silos de saúde, uma vez que os documentos clínicos federados estão disponíveis juntamente com a linha do tempo do paciente, a qual está distribuída por meio da tecnologia blockchain. Assim, torna-se possível trabalhar com os documentos clínicos em cada um dos estabelecimentos de origem, porém com uma visão de linha do tempo unificada para o cidadão. A ideia é ter informações otimizadas por meio do *Fast Healthcare Interoperability Resources* (FHIR), que é um padrão para troca de dados de saúde [Bender e Sartipi 2013].

Isto potencializa análises clínicas, integração de dados, atendimento otimizado para o cidadão (sumário inteligente do paciente e interoperabilidade nativa com a nuvem do MS), bem como o *e-patient*, por intermédio do qual o paciente gerencia a sua própria saúde. Em função disso, será possível falar de medicina preventiva com dispositivos inteligentes e sensores de IoT.

Atualmente, a RNDS é o maior case de blockchain em saúde do mundo com mais de 1 bilhão de transações e apoia-se em três pilares: confiabilidade; distribuição

e rastreabilidade. Totalmente enquadrada na Lei nº 13.709 - Lei Geral de Proteção a Dados (LGPD), utiliza blockchain para garantir segurança, privacidade e consentimento dos pacientes. A estratégia de Saúde Digital para o Brasil trata-se do uso de recursos de TIC para produzir e disponibilizar informações confiáveis, sobre o estado de saúde para quem precisa no momento que precisa.

2.5. Integrando sistemas Web, IoT e blockchain

Esta seção contempla a parte prática proposta neste capítulo. Os participantes podem fazer esta prática junto com os apresentadores ou fazer os tutoriais posteriormente acessando um material complementar em [Abijaude e outros 2021].

A prática proposta é criar uma DApp que ilustre o rastreamento do envio de vacinas entre a origem e o destino, coletando a temperatura durante o percurso. Este exemplo didático abrange o desenvolvimento de um contrato inteligente e de uma DApp que interagem com dispositivos de IoT e um middleware.

A Figura 2.12 ilustra a tela principal da DApp. Estes dados são enviados para o contrato no momento de sua criação através do uso de funções construtoras.

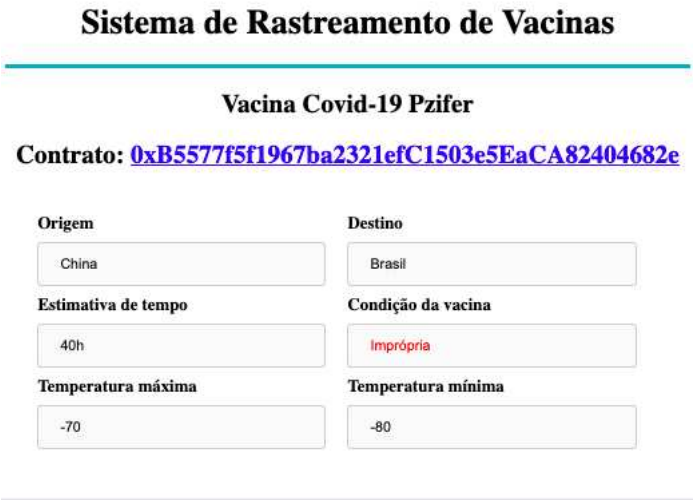

Figura 2.12. Tela da DApp que faz o rastreamento da vacina com base no contrato inteligente.

Os sensores captam o valor de temperatura durante o transporte da vacina e enviam os dados para um middleware, que então encaminha para um CI hospedado na plataforma Ethereum. Caso a temperatura esteja fora das especificações, o contrato vai considerar a vacina imprópria para consumo humano. Quando isto ocorrer, o contrato envia para a DApp uma informação e o registro do local, data, hora e temperatura coletados, conforme ilustrado na Figura 2.13.

Como se trata de um exemplo didático que deve ser replicado em outros ambientes, substituímos a geração dos dados dos sensores por mensagens REST geradas manualmente em programas como Insominia ou Postman. Estas mensagens serão enviadas para um endereço que emula um middleware capaz de recepcionar os dados e enviá-los diretamente para a blockchain.

Figura 2.13. Informação enviada a DApp sobre quando e onde ocorreu uma violação

No entanto, considerando uma melhor técnica para explicar o fluxo de informações, o middleware vai encaminhar as mensagens temporariamente para a aplicação, onde o usuário poderá verificar estes dados e então liberar o envio para o contrato inteligente, conforme mostrado na Figura 2.14. Ressalta-se que esta técnica é meramente didática, pois em um sistema real, o envio dos dados coletados pelos sensores são enviados para o middleware e deste diretamente para os contratos.

Figura 2.14. Informação enviada a DApp sobre quando e onde ocorreu uma violação

Todos os detalhes da construção da aplicação, do contrato inteligente, do middleware, do hardware e dos detalhes de compilação e implementação dos contratos estão na página web com o material complementar.

2.5.1. Ambiente de desenvolvimento

Ainda não há um Ambiente de Desenvolvimento Integrado (IDE, do inglês *Integrated Development Environment*) ou um ambiente amigável para o desenvolvimento dos contratos e de DApps. Pode-se usar editores on-line, como por exemplo o Remix ou preparar um ambiente de desenvolvimento local.

O uso de um ambiente local para desenvolvimento permite mais liberdade ao desenvolvedor. Esta solução é a utilizada pelos autores e já testada em cursos na Universidade Estadual de Santa Cruz (UESC), Universidade Estadual do Sudoeste da Bahia (UESB) e Universidade Federal da Bahia (UFBA).

Este ambiente local é composto por um editor de código de sua preferencia, o Node.js e os seguintes pacotes adicionais: a) `solc`: compilador *Solidity*; b) `mocha`: *framework* para testar os contratos antes de implementá-los em uma rede blockchain; c) `web3`: coleção de bibliotecas que permite interagir com um nó *Ethereum* local ou remoto usando HTTP; d) `ganache-cli`: é uma blockchain pessoal para desenvolvimento rápido de aplicativos distribuídos *Ethereum* e *Corda* em um ambiente seguro e determinístico; e) `truffle-hdwallet-provider`: para realizar as assinaturas usando as palavras mnemônicas. Estas palavras são informadas ao usuário no momento da instalação do metamask e devem ser guardadas, pois através delas conseguiremos autorizar as transações.

Os projetos que estão disponíveis para a prática estão com todas as dependências configuradas e com um tutorial que permite instalar todas elas automaticamente.

Ao realizar o download e descompactar o arquivo, será criado um diretório chamado `Vacina`, com as subpastas `api`, `deploy` e `frontend`. Estas três pastas representam os projetos que serão detalhados adiante.

Antes porém, o usuário deverá abrir um terminal, ir até a cada uma destas pastas e e executar o comando `npm install` para baixar as dependências e configurar o ambiente do projeto. Isto deverá ser feito individualmente em cada uma das pastas.

A pasta `deploy` contém o projeto relacionado à escrita, compilação e implementação dos CIs. Dentro dela está a pasta `contracts` com os arquivos `Vacina.sol` e `vacina.abi.json`. O primeiro, mostrado na Figura 2.15 é o contrato inteligente escrito em Solidity. O segundo é a ABI já recuperada, e que será utilizada no projeto do *front-end*. A seguir, explica-se a lógica do contrato.

As primeiras linhas do contrato, que não aparecem na figura, declaram as variáveis e definem a função construtora, que recebe como parâmetros nome da vacina, local de origem, local de destino, duração prevista para o transporte e temperaturas máxima e mínima. A parte do código entre as linhas 40 e 62 representam a lógica do negócio. Na linha 41 está a função *insertRegistro()* que, ao ser invocada pela DApp, recebe os parâmetros digitados na tela da DApp e executa duas verificações: a primeira é para garantir que o valor da temperatura está dentro do intervalo esperada; a segunda, verifica se em algum registro adicionado anteriormente, este intervalo foi já violado. Caso estas condições tenham sido violadas, registra-se então os valores que tornaram a vacina imprópria por ter sido armazenada fora dos padrões exigidos. Em seguida os dados são armazenados em uma matriz de endereços.

A função `getRegistro()`, na linha 51, retorna todas as etapas da viagem enviando a matriz `registros`. A função `close()`, na linha 55, destrói o contrato, desde que seja invocada pelo endereço que o implementou. Observe que na declaração desta função, ela recebe um indicativo que terá de consultar a função verificadora antes de sua execução, através da palavra `verificaOwner()`. Isto desvia o fluxo de execução do


```
40      // Cadastra uma etapa do transporte
41      function insertRegistro(int16 _value, uint64 _time, string memory _local)
42          // Verifica se a temperatura esta entre a maxima e a mininma entre
43          // se nao teve perda antes dessa inserção então execute o if
44          if((_value > tempMax || _value < tempMin) && perdaVacina.value == 0){
45              perdaVacina = Registro(_value, _time, _local);
46              condicao = false;
47          }
48          registros.push(Registro(_value, _time, _local));
49      }
50      // Retorna etapas do transporte
51      function getRegistro() public view returns(Registro[] memory _registro) {
52          return registros;
53      }
54      // Destroi contrato
55      function close() public verificaOwner {
56          address payable addr = payable(msg.sender);
57          selfdestruct(addr);
58      }
59      modifier verificaOwner(){
60          require(msg.sender == owner);
61          _;
62      }
```

Figura 2.15. Parte do contrato inteligente `Vacina.sol`.

contrato para a linha 59, executando a função modificadora até encontrar o sinal de "_", quando então retorna para execução da função `close()`, a partir da linha 56.

Ainda na pasta do projeto `deploy`, existem três arquivos importantes localizados na raiz da pasta. São eles o `.env`, e os *scripts* `compile.js` e `deploy.js`. O primeiro é um arquivo de configuração que possui apenas duas linhas. A primeira linha deve conter as palavras mnemônicas e a segunda linha o endereço do nó que permite a conexão com a blockchain. Este endereço adquire-se ao acessar o site `www.infura.io`, seguindo o tutorial que está na página web complementar.

O segundo arquivo, `compile.js`, é responsável pela compilação do projeto, e consequentemente pela produção da ABI e dos *bytecodes*. Este *script* pode ser reaproveitado para outros contratos, bastando apenas atualizar as variáveis `contractName` e `contractFileName`.

O terceiro arquivo é o *script deploy.js*. Ele invoca o *script* de compilação e manipula os resultados de modo a enviar para a o endereço especificado no `.env` os bytecodes, e assim consequentemente, implementar o contrato na rede blockchain. Para executá-lo, digite no terminal o comando `node deploy.js` e aguarde. No término de sua execução, o terminal recebe como resultado a ABI gerada pelo compilador, que neste caso, foi copiada para o arquivo `vacina.abi.json`. O endereço da conta utilizada para implementação e o endereço que identifica o contrato na rede blockchain. Guarde o endereço do contrato, pois vocÊ irá precisar dele para informar à DApp.

Este *script* também pode ser utilizado por outros contratos. Para tanto deve ser ajustada a variável `contract`. Esta variável configura uma parte da transação a ser enviada, e, em particular, neste caso, envia os parâmetros esperados pela função construtora.

A pasta `frontend` possui a DApp que será executada. Na pasta `src` há a pasta `contracts` e os arquivos `index.js` e `App.js`.

A pasta `frontend/src/contracts` possui dois arquivos. O arquivo `web3.js` configura a `web3` para ser usado pela DApp. O arquivo `vacina.contracts.js` possui as variáveis `address`, cujo valor deve ser substituído pelo endereço do contrato implementado quando você executou o *script* `deploy.js` e a variável `abi` cujo valor deve ser a ABI, também impressa na tela no processo de implementação. Para facilitar, o arquivo `/deploy/contracts/vacina.abi.js` já possui este conteúdo.

O arquivo `index.js` é o arquivo padrão que será lido pelo servidor web. Este arquivo importa o `App.js` que contém de fato a DApp.

Este arquivo é divido basicamente em duas partes: Uma que declara variáveis e funções e outra que prepara a tela a ser exibida para o usuário. A Figura 2.16 ilustra, entre as linhas 92 e 98, invocações às funções do contrato. Para que isto ocorra, a DApp utiliza as informações na ABI, fornecida ao projeto pelo arquivo `vacina.contracts.js`, para enviar uma solicitação à rede blockchain. Cada uma dessas funções descritas no contrato tem a finalidade de retornar um valor.

```
89  const pegaInfoContrato = async () => {
90    try {
91      // Pega informações do contrato
92      const _nome = await vacina.methods.nome().call();
93      const _origem = await vacina.methods.origem().call();
94      const _destino = await vacina.methods.destino().call();
95      const _duracao = await vacina.methods.duracao().call();
96      const _tempMax = await vacina.methods.tempMax().call();
97      const _tempMin = await vacina.methods.tempMin().call();
98      const _condicao = await vacina.methods.condicao().call();
```

Figura 2.16. Parte do arquivo `App.js` que recupera dados do contrato.

A Figura 2.17, entre as linhas 147 e 152, mostra a variável `responseTrx` que recebe o resultado do envio de dados para o CI. Para que isto seja possível, novamente a ABI é consultada pela aplicação para saber como encaminhar para a função `insertRegistro()` os valores da temperatura (`_value`), data/hora (`_time`) e local (`_local`). As taxas deste envio serão debitadas no endereço representado pela variável `contas[0]`.

```
147     const responseTrx = await vacina.methods
148       .insertRegistro(_value, _time, _local)
149       .send({
150         // Diz a carteira que está enviando os dados
151         from: contas[0],
152       });
```

Figura 2.17. Parte do arquivo `App.js` que recupera dados do contrato.

Para deixar o servidor web com o *front-end* ativo, digite na pasta do projeto `npm start` e aguarde. O navegador padrão será aberto e a tela da DApp exibida.

A pasta `api` contém os arquivos para emular um middleware, na porta TCP 3333, para recepcionar os dados enviados por um sensor ou por um gerador de mensagens HTTP e reencaminhá-los para a aplicação. O endereço utilizado para enviar as mensagens é `localhost:3333/insert-data`. Estas mensagens possuem o formato JSON `{"value":-75,"local":"Brasil"}`. O terceiro parâmetro, referente a data/hora é recuperado automaticamente pelo sistema. Para ativar este serviço, vá até o terminal e digite na pasta do projeto o comando `npm start`. Para enviar mensagens use na linha de comando o aplicativo `curl` ou um programa como o Postman ou Insominia. Nós recomendamos fortemente que seja utilizado um destes dois programas, em especial o insomnia, cujo tutorial está publicado na página web deste capítulo.

Ao acessar a página do curso, teremos um tutorial completo para a execução da prática. Esta prática pode ser feita de duas formas: usando um dispositivo de IoT com interface de rede ou através de um programa que envie mensagens como o Postman ou Insomnia. Todos os códigos estão comentados e explicados de forma bem didática.

2.6. Como montar um curso de Blockchain e IoT aplicado à saúde

A programação de aplicações para blockchain e IoT não é uma tarefa trivial. Envolve conceitos, propriedades e conhecimentos que vão além da blockchain, dos contratos inteligentes e da linguagem de programação *Solidity* .

Existe uma escassez de material teórico e prático para o ensino de tecnologias emergentes como blockchain, CIs e desenvolvimento de DApps. Uma alternativa para isto são as plataformas MOOC (*Massive Open Online Course*), como Udemy, Coursera, edX. Algumas Universidades promovem cursos de extensão ou treinamentos para seus alunos [Rao e Dave 2019, Dettling 2018, Araujo e outros 2019].

Os autores deste capítulo elaboraram e ministraram um curso em três universidades na Bahia: A Universidade Estadual de Santa Cruz (UESC), a Universidade Estadual do Sudoeste da Bahia (UESB) e a Universidade Federal da Bahia (UFBA).

Esta seção objetiva esclarecer e auxiliar a criação de cursos de extensão ou disciplinas de graduação/pós-graduação que pretendem explorar este tema através de três subseções: Infraestrutura, conteúdo programático e sugestão de práticas.

2.6.1. Infraestrutura

O hardware empregado para o desenvolvimento dos laboratórios é bastante simples, uma vez que não há plataformas que necessitem de muitos recursos computacionais. Nos 3 treinamentos ministrados havia computadores equipados com processadores que vão desde o Core 2 Duo com 4 Gb de RAM, até o Core i7 com 32 Gb de RAM. O espaço em disco também não é um fator limitante, uma vez que a maioria das máquinas possui espaço de armazenamento suficiente para os experimentos realizados.

O conjunto de softwares necessários à realização de todas as atividades práticas do treinamento é composto por navegadores, extensões para navegadores, ferramentas, pacotes e aplicativos hospedados em sites. Os softwares são compatíveis com praticamente todos os sistemas operacionais, como por exemplo Windows, Linux, Unix e MacOs.

2.6.2. Conteúdo a ser abordado

Sugere-se três módulos para serem ministrados aos alunos. O primeiro, básico, aborda conceitos de BC, CIs e DApps, criando um contrato simples e uma DApp. O curso básico pode ser ministrado com carga horária de 16h. Seriam 2 dias de aula, com 4 horas no período da manhã e 4 horas no período da tarde. No primeiro dia, durante o período da manhã foi abordada toda a parte teórica e conceitual da blockchain com ênfase na plataforma Ethereum, além da apresentação do editor de contratos on-line *Remix*. Durante o período da tarde, configuramos as máquinas locais e realiza-se rotinas de testes.

O segundo, explora mais profundamente as funções da linguagem *Solidity*, construindo um contrato bem mais complexo, utilizando técnicas de engenharia de software, e criando uma DApp multi página. Neste nível intermediário, durante o período da manhã apresenta-se mais um pouco de teoria com foco na interação com as redes *Ethereum*. Aprende-se mais um pouco sobre a linguagem de programação *Solidity* e escreve-se, compila-se, testa-se e implementa-se um contrato na rede *Rinkeby*. No período da tarde, desenvolve-se uma DApp que interage com o contrato implementado.

O terceiro módulo serviu de base para a criação deste capítulo e prevê a integração com a Internet das Coisas, coletando dados dos sensores, armazenando-os em CIs e disparando ações quando determinadas situações forem alcançadas.

É possível encontrar mais recursos na internet, como por exemplo:

a) *CryptoZombies*: uma plataforma online onde o intuito é ensinar sobre contratos inteligentes de forma interativa. O usuário desenvolve um jogo com foco em zumbis onde a logística é administrada por um contrato e com interação visual através de html, css e javascript. Disponível em `https://cryptozombies.io/pt/`.

b) *Ethernaut*: uma plataforma online que apresenta diversos tutoriais voltados para jogos. Ela foca puramente na criação de contratos, sem implementação visual. Alguns exemplos possibilitam a interação através da ferramenta do desenvolvedor do navegador. Disponível em `https://ethernaut.openzeppelin.com/`.

c) *Vyper Tutorials*: semelhante à ideia do CryptoZombies, esta plataforma propõe a criação de um jogo de *pokémon*. Atualmente está em fase de desenvolvimento, mas já é possível aprender como funciona a criação de contratos. Futuramente serão adicionadas interações através de interface assim como *CryptoZombies*. Disponível em `https://vyper.fun/#/`.

d) *Ethereum Studio*: uma ferramenta para desenvolvedores que desejam aprender sobre como construir aplicações na rede *Ethereum*. Os modelos ensinam como escrever um contrato inteligente, implementá-lo e interagir com os CIs por meio de um aplicativo baseado na web. Disponível em `https://studio.ethereum.org/`.

2.6.3. Práticas Propostas

O primeiro conjunto de práticas propostas teria a finalidade de apresentar o Remix e desenvolvimento do primeiro contrato. Sugere-se um exemplo didático que sirva para o aluno se familiarizar com o ambiente e começar a entender os conceitos de compilação, implementação, rede de testes, etc.

A segunda prática, recomenda-se que seja a montagem de ambiente local de desenvolvimento, ressaltando as vantagens e desvantagens desta abordagem. Nessa prática, o aluno vai lidar com scripts de compilação e implementação, compreender conceitos de ABI, bytecodes, instalar uma carteira Ethereum e abastecê-la com ethers sem valor comercial.

O conceitos de testes de contratos e seus benefícios devem ser introduzidos como o terceiro momento da prática. É importante deixar claro que os contratos são imutáveis, e uma vez implementados não é possível modificá-los e nem mesmo apagá-los. Aqui o aluno aprende a configurar o ambiente de testes, montar uma rede blockchain local em sua máquina e executar testes básicos.

Na sequência, as práticas evoluem para contratos mais sofisticados, que representam exercícios mais elaborados e envolvem a transferência de moedas entre as contas. Novos testes também são propostos aqui e, ao final, a integração com um sistema Web simples, com apenas uma página, criando a primeira DApp.

Após a criação deste contrato, sugere-se criar contratos mais complicados, que envolvem conceitos de engenharia de software, como padrão *fabric*. Questões de quem vai pagar por eventuais operações nos contratos e recursos mais avançados da linguagem solidity serão tratados nesta prática. Depois de novas rotinas de teste, constrói-se uma DApp multipágina, que representa um sistema mais elaborado e com grau de dificuldade maior.

A última e mais desafiadora prática e construir um pequeno sistema que seja capaz de enviar dados de dispositivos IoT para um contrato e exibí-los em um sistema, empregando, por exemplo o estilo arquitetural REST. Caso não seja possível ter os dispositivos de IoT, pode-se optar pela geração de dados que simule tais equipamentos.

Como mecanismo de avaliação sugere-se que os alunos pesquisem e elaborem um projeto que contemple os conhecimentos adquiridos, para em seguida, apresentá-lo a todos da sala, detalhando as atividades realizadas.

2.7. Desafios, Perspectivas e Conclusão

Este capítulo abordou a IoT, blockchain e contratos inteligentes aplicados à saúde. Após uma sessão de nivelamento, onde conceitos sobre estes temas e sobre o desenvolvimento de DApps foram tratados, os autores se aprofundaram na plataforma ethereum e nos contratos inteligentes.

Na sequência, foram abordados os desafios da IoT e blockchain na área de saúde, enumerando os principais requisitos e os desafios técnicos impostos pela tecnologia. Os principais protocolos de consenso foram agrupados em PoW, PoS, BFT e suas variantes, e sempre que possível, citadas referências que aplicações na área de saúde para exemplificar o emprego de tais protocolos.

O emprego da blockchain e IoT possui várias pesquisas e aplicações em andamento, conforme descrito na Seção 1.4, no entanto ainda há desafios de pesquisa para resolver ou aprimorar questões como desafios organizacionais para adição da blockchain; segurança e vulnerabilidade dos contratos inteligentes; grande e crescente volume de da-

dos na área de saúde e Interoperabilidade e suporte para transações entre plataformas [Ahmad e outros 2021].

Os Desafios organizacionais para a adoção de blockchain esbarra nos sistemas tradicionais de telemedicina, que dependem principalmente de métodos desatualizados para armazenar, manter e proteger os dados dos pacientes, o que pode limitar as oportunidades de colaboração entre os participantes e provedores de saúde. Conforme foi abordado, a tecnologia Blockchain garante que o histórico médico completo e confiável de um paciente possa ser mantido e rastreado pelos usuários autorizados por meio de registros imutáveis. No entanto, a falta de consciência, imaturidade da tecnologia e indisponibilidade de padrões de segurança e privacidade impedem os atores de empregar plenamente os benefícios da blockchain, inclusive os relacionados aos incentivos monetários para as organizações participantes [Kolan e outros].

Os contratos inteligentes ainda possuem riscos de adulteração e segurança e isto pode afetar significativamente o histórico médico de um paciente, como um ataque de vulnerabilidade [Liu e outros 2018]; um contrato inteligente, que tem privilégios exclusivos para se comunicar com outro contrato, pode alterar o EHR de um paciente, ou pode recuperar fundos da carteira de um usuário legítimo. Há ferramentas de diagnóstico, como ZeppolinOS, SolCover e Oyente. Essas ferramentas ajudam a identificar as características vulneráveis de contratos inteligentes para ajudar os desenvolvedores a propor contramedidas contra ameaças externas. No entanto, as soluções propostas são inadequadas para identificar todos os tipos de vulnerabilidades e bugs em um contrato inteligente. Portanto, os testes rigorosos são fundamentais para detectar vulnerabilidades antes de sua implementação.

O histórico médico consistente e atualizado de um paciente é sempre crescente e isto pode gerar uma enorme quantidade de dados que requer processamento rápido e constante. No entanto, para as plataformas atuais de blockchain, a grande quantidade de dados de saúde é um problema para as taxas de transação e o tempo total de mineração. O emprego de *sidechains* ou uma camada na névoa pode ajudar a minimizar a taxa de transação [Debe e outros 2019].

Construir plataformas de blockchain interoperáveis é desafiador devido a vários problemas, como diferenças nas linguagens suportadas e protocolos de consenso das plataformas de blockchain [Herlihy 2018]. Isto é um empecilho para o suporte à interoperabilidade dos sistemas de saúde que precisam ser interoperáveis e exigem transações seguras nas plataformas de blockchain. Apesar

Por fim, espera-se que este trabalho motive e contribua positivamente para a comunidade de Computação Aplicada à Saúde. Aqui apresentou-se uma visão geral e esclarecedora de um conjunto de tecnologias, que para operarem em conjunto, não possuem métodos triviais.

Referências

[Abijaude e outros 2021] Abijaude, J., Serra, Henrique Bezerra, A., Barretto, R., Sobreira, P., e Greve, F. (2021). Internet das coisas, blockchain e contratos inteligentes aplicados à saúde. `https://github.com/lifuesc/sbcas2021`. Acessado

em 09/06/2021.

[Adibi 2015] Adibi, S. (2015). A mobile health network disaster management system. In *2015 seventh international conference on ubiquitous and future networks*, pages 424–428. IEEE.

[Ahmad e outros 2021] Ahmad, R. W., Salah, K., Jayaraman, R., Yaqoob, I., Ellahham, S., e Omar, M. (2021). The role of blockchain technology in telehealth and telemedicine. *International Journal of Medical Informatics*, page 104399.

[Al Omar e outros 2017] Al Omar, A., Rahman, M. S., Basu, A., e Kiyomoto, S. (2017). Medibchain: A blockchain based privacy preserving platform for healthcare data. In *International conference on security, privacy and anonymity in computation, communication and storage*, pages 534–543. Springer.

[Albeyatti 2018] Albeyatti, A. (2018). Meddicalchain white paper. https://medicalchain.com/Medicalchain-Whitepaper-EN.pdf.

[Alphand e outros 2018] Alphand, O., Amoretti, M., Claeys, T., Dall'Asta, S., Duda, A., Ferrari, G., Rousseau, F., Tourancheau, B., Veltri, L., e Zanichelli, F. (2018). Iotchain: A blockchain security architecture for the internet of things. In *2018 IEEE wireless communications and networking conference (WCNC)*, pages 1–6. IEEE.

[Anderson 2018] Anderson, J. (2018). Securing, standardizing, and simplifying electronic health record audit logs through permissioned blockchain technology.

[Antonopoulos 2017] Antonopoulos, A. (2017). *Mastering Bitcoin: Programming the Open Blockchain*. O'reilly, 2nd edition.

[Araujo e outros 2019] Araujo, P., Viana, W., Veras, N., Farias, E. J., e de Castro Filho, J. A. (2019). Exploring students perceptions and performance in flipped classroom designed with adaptive learning techniques: A study in distributed systems courses. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 30, page 219.

[Au e outros 2017] Au, M. H., Yuen, T. H., Liu, J. K., Susilo, W., Huang, X., Xiang, Y., e Jiang, Z. L. (2017). A general framework for secure sharing of personal health records in cloud system. *Journal of Computer and System Sciences*, 90:46–62.

[Azaria e outros 2016] Azaria, A., Ekblaw, A., Vieira, T., e Lippman, A. (2016). Medrec: Using blockchain for medical data access and permission management. In *2016 2nd International Conference on Open and Big Data (OBD)*, pages 25–30. IEEE.

[Bach e outros 2018] Bach, L. M., Mihaljevic, B., e Zagar, M. (2018). Comparative analysis of blockchain consensus algorithms. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1545–1550. IEEE.

[Bender e Sartipi 2013] Bender, D. e Sartipi, K. (2013). Hl7 fhir: An agile and restful approach to healthcare information exchange. In *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*, pages 326–331.

[Bessani e outros 2014] Bessani, A., Sousa, J., e Alchieri, E. E. (2014). State machine replication for the masses with bft-smart. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 355–362. IEEE.

[Buterin e outros. 2014] Buterin, V. e outros. (2014). A next-generation smart contract and decentralized application platform. *white paper*.

[Cachin e outros 2016] Cachin, C., Caro, A. D., Christidis, K., e Yellick, J. (2016). Architecture of the hyperledger blockchain fabric. Technical report, IBM Research - Zurich.

[Castro e outros 1999] Castro, M., Liskov, B., e outros. (1999). Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186.

[Coinbase 2018] Coinbase (2018). Coinbase wallet. `https://wallet.coinbase.com/`. Acessado em 03/03/2021.

[da Conceição e outros 2018] da Conceição, A. F., da Silva, F. S. C., Rocha, V., Locoro, A., e Barguil, J. M. (2018). Eletronic health records using blockchain technology. *arXiv preprint arXiv:1804.10078*.

[da Saúde 2020] da Saúde, M. (2020). Rede nacional de dados em saúde. `https://rnds.saude.gov.br/`. Acessado em 05/06/2021.

[De Aguiar e outros 2020] De Aguiar, E. J., Faiçal, B. S., Krishnamachari, B., e Ueyama, J. (2020). A survey of blockchain-based strategies for healthcare. *ACM Computing Surveys (CSUR)*, 53(2):1–27.

[De Angelis e outros 2018] De Angelis, S., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., e Sassone, V. (2018). Pbft vs proof-of-authority: Applying the cap theorem to permissioned blockchain.

[Debe e outros 2019] Debe, M., Salah, K., Rehman, M. H. U., e Svetinovic, D. (2019). Iot public fog nodes reputation system: A decentralized solution using ethereum blockchain. *IEEE Access*, 7:178082–178093.

[Dettling 2018] Dettling, W. (2018). How to teach blockchain in a business school. In *Business Information Systems and Technology 4.0*, pages 213–225. Springer.

[Dorri e outros 2017] Dorri, A., Kanhere, S. S., Jurdak, R., e Gauravaram, P. (2017). Blockchain for iot security and privacy: The case study of a smart home. In *2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)*, pages 618–623. IEEE.

[Dubovitskaya e outros 2017] Dubovitskaya, A., Xu, Z., Ryu, S., Schumacher, M., e Wang, F. (2017). Secure and trustable electronic medical records sharing using blockchain. In *AMIA annual symposium proceedings*, volume 2017, page 650. American Medical Informatics Association.

[Dziembowski e outros 2015] Dziembowski, S., Faust, S., Kolmogorov, V., e Pietrzak, K. (2015). Proofs of space. In *Annual Cryptology Conference*, pages 585–605. Springer.

[Ethereum 2014] Ethereum, W. (2014). A secure decentralised generalised transaction ledger [j]. *Ethereum project yellow paper*, 151:1–32.

[Ethfiddle 2017] Ethfiddle (2017). Ethfiddle editor rinkeby. `https://ethfiddle.com/`. Acessado em 03/03/2021.

[Frankenfield 2018] Frankenfield, J. (2018). Proof of burn. `https://www.investopedia.com/terms/p/proof-burn-cryptocurrency`. Acessado em 04/05/2021.

[Greve e outros 2018] Greve, F. G., Sampaio, L. S., Abijaude, J. A., Coutinho, A. C., Valcy, Í. V., e Queiroz, S. Q. (2018). Blockchain e a revolução do consenso sob demanda. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)-Minicursos*.

[Greve 2005] Greve, F. G. P. (2005). Protocolos fundamentais para o desenvolvimento de aplicaçoes robustas. In *Minicursos SBRC 2005: Brazilian Symposium on Computer Networks*, pages 330–398.

[Guo e outros 2018] Guo, R., Shi, H., Zhao, Q., e Zheng, D. (2018). Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems. *IEEE access*, 6:11676–11686.

[Guo e outros 2019] Guo, R., Shi, H., Zheng, D., Jing, C., Zhuang, C., e Wang, Z. (2019). Flexible and efficient blockchain-based abe scheme with multi-authority for medical on demand in telemedicine system. *IEEE Access*, 7:88012–88025.

[Halamka e outros 2019] Halamka, J. D., Alterovitz, G., Buchanan, W. J., Cenaj, T., Clauson, K. A., Dhillon, V., Hudson, F. D., Mokhtari, M. M., Porto, D. A., Rutschman, A., e outros. (2019). Top 10 blockchain predictions for the (near) future of healthcare. *Blockchain in Healthcare Today*.

[Herlihy 2018] Herlihy, M. (2018). Atomic cross-chain swaps. In *Proceedings of the 2018 ACM symposium on principles of distributed computing*, pages 245–254.

[Hoy 2017] Hoy, M. B. (2017). An introduction to the blockchain and its implications for libraries and medicine. *Medical reference services quarterly*, 36(3):273–279.

[Hussien e outros 2021] Hussien, H. M., Yasin, S. M., Udzir, N. I., Ninggal, M. I. H., e Salman, S. (2021). Blockchain technology in the healthcare industry: Trends and opportunities. *Journal of Industrial Information Integration*, 22:100217.

[Jaxx 2018] Jaxx (2018). Jaxx safely manager ethereum. `https://jaxx.io/`. Acessado em 03/03/2021.

[Jiang e outros 2016] Jiang, J., Wen, S., Yu, S., Xiang, Y., e Zhou, W. (2016). Identifying propagation sources in networks: State-of-the-art and comparative studies. *IEEE Communications Surveys & Tutorials*, 19(1):465–481.

[Kazmi e outros 2019] Kazmi, H. S. Z., Nazeer, F., Mubarak, S., Hameed, S., Basharat, A., e Javaid, N. (2019). Trusted remote patient monitoring using blockchain-based smart contracts. In *International Conference on Broadband and Wireless Computing, Communication and Applications*, pages 765–776. Springer.

[Kiayias e outros 2017] Kiayias, A., Russell, A., David, B., e Oliynykov, R. (2017). Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer.

[Kolan e outros] Kolan, A., Tjoa, S., e Kieseberg, P. Medical blockchains and privacy in austria-technical and legal aspects.

[Kotz e outros 2016] Kotz, D., Gunter, C. A., Kumar, S., e Weiner, J. P. (2016). Privacy and security in mobile health: A research agenda. *Computer*, 49(6):22–30.

[Kwon 2014] Kwon, J. (2014). Tendermint: Consensus without mining. *Draft v. 0.6, fall*, 1(11).

[Lamport e outros 1982] Lamport, L., Shostak, R., e Pease, M. (1982). The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401.

[Lao e outros 2020] Lao, L., Li, Z., Hou, S., Xiao, B., Guo, S., e Yang, Y. (2020). A survey of iot applications in blockchain systems: Architecture, consensus, and traffic modeling. *ACM Computing Surveys (CSUR)*, 53(1):1–32.

[Larimer 2014] Larimer, D. (2014). Delegated proof-of-stake (dpos). *Bitshare whitepaper*, 81:85.

[Lee 2019] Lee, C. K. (2019). Blockchain application with health token in medical & health industrials. In *2nd International Conference on Social Science, Public Health and Education (SSPHE 2018)*, pages 233–236. Atlantis Press.

[Lee e Lee 2015] Lee, I. e Lee, K. (2015). The internet of things (iot): Applications, investments, and challenges for enterprises. *Business Horizons*, 58(4):431–440.

[Lerner 2015] Lerner, S. D. (2015). Dagcoin: a cryptocurrency without blocks. *White paper*.

[Liu e outros 2018] Liu, C., Liu, H., Cao, Z., Chen, Z., Chen, B., e Roscoe, B. (2018). Reguard: finding reentrancy bugs in smart contracts. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*, pages 65–68. IEEE.

[Mannaro e outros 2018] Mannaro, K., Baralla, G., Pinna, A., e Ibba, S. (2018). A blockchain approach applied to a teledermatology platform in the sardinian region (italy). *Information*, 9(2):44.

[Mazieres 2015] Mazieres, D. (2015). The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Development Foundation*, 32.

[McGhin e outros 2019] McGhin, T., Choo, K.-K. R., Liu, C. Z., e He, D. (2019). Blockchain in healthcare applications: Research challenges and opportunities. *Journal of Network and Computer Applications*, 135:62–75.

[Metamask 2018] Metamask (2018). Metamask crypto wallet and gateway. `https://metamask.io/`. Acessado em 03/03/2021.

[Milutinovic e outros 2016] Milutinovic, M., He, W., Wu, H., e Kanwal, M. (2016). Proof of luck: An efficient blockchain consensus protocol. In *proceedings of the 1st Workshop on System Software for Trusted Execution*, pages 1–6.

[MyCrypto 2019] MyCrypto (2019). Mycrypto. `https://mycrypto.com/`. Acessado em 03/03/2021.

[Myetherwallet 2019] Myetherwallet (2019). Myetherwallet original wallet. `https://www.myetherwallet.com/`. Acessado em 03/03/2021.

[Nakamoto 2008] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Technical report, Bitcoin Org.

[Narayanan e outros 2016] Narayanan, A., Bonneau, J., Felten, E., Miller, A., e Goldfeder, S. (2016). *Bitcoin and cryptocurrency technologies*. Princeton University Press.

[of Health e outros 2008] of Health, U. D., Services, H., e outros. (2008). Personal health records and the hipaa privacy rule. *Washington, DC. URL: https://www. hhs. gov/sites/default/files/ocr/privacy/hipaa/understanding/special/healthit/phrs. pdf [accessed 2016-06-20][WebCite Cache]*.

[Patel 2019] Patel, V. (2019). A framework for secure and decentralized sharing of medical imaging data via blockchain consensus. *Health informatics journal*, 25(4):1398–1411.

[PoET 2018] PoET (2018). Poet 1.0 specification. `https://sawtooth.hyperledger.org/docs/core/releases/1.0/architecture/poet.html`. Acessado em 04/05/2021.

[Popov 2018] Popov, S. (2018). The tangle, iota whitepaper. `https://iota.org/IOTA\_Whitepaper.pdf`. Acessado em 03/03/2021.

[Popov e outros 2020] Popov, S., Moog, H., e et al. (2020). The coordicide. *white paper Iota Foundation*.

[Raikwar e outros 2018] Raikwar, M., Mazumdar, S., Ruj, S., Gupta, S. S., Chattopadhyay, A., e Lam, K.-Y. (2018). A blockchain framework for insurance processes. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–4. IEEE.

[Ramachandran e outros 2020] Ramachandran, S., Kiruthika, O. O., Ramasamy, A., Vanaja, R., e Mukherjee, S. (2020). A review on blockchain-based strategies for management of electronic health records (ehrs). In *2020 International Conference on Smart Electronics and Communication (ICOSEC)*, pages 341–346. IEEE.

[Rao e Dave 2019] Rao, A. R. e Dave, R. (2019). Developing hands-on laboratory exercises for teaching stem students the internet-of-things, cloud computing and blockchain applications. In *2019 IEEE Integrated STEM Education Conference (ISEC)*, pages 191–198. IEEE.

[Remix 2015] Remix (2015). Remix ide. `https://remix.ethereum.org/`. Acessado em 03/03/2021.

[Roehrs e outros 2017] Roehrs, A., Da Costa, C. A., e da Rosa Righi, R. (2017). Omniphr: A distributed architecture model to integrate personal health records. *Journal of biomedical informatics*, 71:70–81.

[Salah e outros 2020] Salah, K., Alfalasi, A., Alfalasi, M., Alharmoudi, M., Alzaabi, M., Alzyeodi, A., e Ahmad, R. W. (2020). Iot-enabled shipping container with environmental monitoring and location tracking. In *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE.

[Samaniego e outros 2016] Samaniego, M., Jamsrandorj, U., e Deters, R. (2016). Blockchain as a service for iot. In *2016 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*, pages 433–436. IEEE.

[Sandgaard e Wishstar 2018] Sandgaard, J. e Wishstar, S. (2018). Medchain white paper. http://www.medchain.us/doc/Medchain%20Whitepaper%20v1.0.pdf. Acessado em 04/06/2021.

[Santos e outros 2016] Santos, B. P., Silva, L. A., Celes, C., Borges, J. B., Neto, B. S. P., Vieira, M. A. M., Vieira, L. F. M., Goussevskaia, O. N., e Loureiro, A. (2016). Internet das coisas: da teoria à prática. *Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuidos*, 31.

[Saweros e Song 2019] Saweros, E. e Song, Y.-T. (2019). Connecting heterogeneous electronic health record systems using tangle. In *International Conference on Ubiquitous Information Management and Communication*, pages 858–869. Springer.

[Shahnaz e outros 2019] Shahnaz, A., Qamar, U., e Khalid, A. (2019). Using blockchain for electronic health records. *IEEE Access*, 7:147782–147795.

[Silvano e Marcelino 2020] Silvano, W. F. e Marcelino, R. (2020). Iota tangle: A cryptocurrency to communicate internet-of-things data. *Future Generation Computer Systems*, 112:307–319.

[Siyal e outros 2019] Siyal, A. A., Junejo, A. Z., Zawish, M., Ahmed, K., Khalil, A., e Soursou, G. (2019). Applications of blockchain technology in medicine and healthcare: Challenges and future perspectives. *Cryptography*, 3(1):3.

[Sobreira 2005] Sobreira, P. (2005). Desenvolvimento de uma arquitetura microcontrolada para tratamento de sinais biológicos. Master's thesis, Universidade Federal de Pernambuco.

[Sousa e outros 2018] Sousa, J., Bessani, A., e Vukolic, M. (2018). A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In *2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*, pages 51–58. IEEE.

[Status 2019] Status (2019). Status private, secure communication. `https://status.im/`. Acessado em 03/03/2021.

[StudioEthereum 2019] StudioEthereum (2019). Studio ethereum. `https://studio.ethereum.org//`. Acessado em 03/03/2021.

[Szabo 1997] Szabo, N. (1997). Formalizing and securing relationships on public networks. *First Monday*, 2(9).

[Sztajnberg e outros 2018] Sztajnberg, A., da Silva Macedo, R., e Stutzel, M. (2018). Protocolos de aplicação para a internet das coisas: conceitos e aspectos práticos. *Sociedade Brasileira de Computação*.

[Thatcher e Acharya 2018] Thatcher, C. e Acharya, S. (2018). Pharmaceutical uses of blockchain technology. In *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–6. IEEE.

[Todd 2015] Todd, P. (2015). Ripple protocol consensus algorithm review. *Ripple Labs Inc White Paper (May, 2015) https://raw. githubusercontent. com/petertodd/rippleconsensus-analysis-paper/master/paper. pdf*.

[Trust 2019] Trust (2019). Trust wallet - secure crypto wallet. `https://trustwallet.com/`. Acessado em 03/03/2021.

[Web3js 2016] Web3js (2016). Ethereum javascript api. `https://web3js.readthedocs.io/en/v1.3.0/index.html`. Acessado em 03/03/2021.

[Weissman e outros 2018] Weissman, S. M., Zellmer, K., Gill, N., e Wham, D. (2018). Implementing a virtual health telemedicine program in a community setting. *Journal of genetic counseling*, 27(2):323–325.

[Wu e outros 2019] Wu, M., Wang, K., Cai, X., Guo, S., Guo, M., e Rong, C. (2019). A comprehensive survey of blockchain: From theory to iot applications and beyond. *IEEE Internet of Things Journal*, 6(5):8114–8154.

[Yli-Huumo e outros 2016] Yli-Huumo, J., Ko, D., Choi, S., Park, S., e Smolander, K. (2016). Where is current research on blockchain technology?—a systematic review. *PloS one*, 11(10):e0163477.

[Yüksel e outros 2017] Yüksel, B., Küpçü, A., e Öznur Özkasap (2017). Research issues for privacy and security of electronic health services. *Future Generation Computer Systems*, 68:1–13.

[Zhang e outros 2017] Zhang, Y., Liu, T., Li, K., e Zhang, J. (2017). Improved visual correlation analysis for multidimensional data. *Journal of Visual Languages and Computing*, 41:121–132.

[Zheng e outros 2018] Zheng, Z., Xie, S., Dai, H.-N., Chen, X., e Wang, H. (2018). Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4):352–375.

Capítulo 3

Fundamentals of IEC 62304 with an Agile Software Development Model

Johnny Marques, Lilian Barros, Sarasuaty Yelisetty, Talita Slavov

Abstract

In 2006, a working group from the International Electrotechnical Commission (IEC) defined IEC 62304:2006. Thus, the use of IEC 62304 has become fully harmonized in the United States and Europe. The purpose of IEC 62304 is to provide requirements for Healthcare Systems manufacturers with Software to demonstrate their ability to provide Software that consistently meets customer requirements and applicable regulatory requirements. Objective: This chapter aims to present the fundamentals of IEC 62304 with an Agile Software Development Model. Justification and motivation: In 2021, IEC 62304 completes 15 years. Thus, the authors believe that there are already works that report experiences, analyzes and difficulties in its use. These results can be interesting to direct further research and definitions of methods, models, guides or other materials to comply with IEC 62304. Conclusion: This book chapter provided a lecture with fundamentals of IEC 62304, including an Agile Software Development Model.

3.1. Introduction

The standards published by committees, international technical entities or regulatory agencies influence the development of Software in Regulated Environments through guidelines for Software processes and products [Munch et al. 2012], considering the risk mentioned above. In addition, there are several similarities between the standards for software development in the aviation, health, and railway areas [Marques and Cunha 2019].

Faced with the challenges that involve the insertion of informatics in the health field, specialized organizations in the area decided to produce guidelines and policies that can help in the process of implantation and evaluation of systems with software [Mauer and Marin 2017].

A Medical System (MS) is composed of several physical and logical parts. One or more Medical Devices (MDs) are part of a Medical System (MS), and they are responsible for the necessary information controls [Marques et al. 2021]. A Medical Device Software

(MDS) is loaded and operated into a Medical Device (MD). Each MDS is composed of Software Items (SIs), which are any identifiable part of a computer program. According to Magnusson (2012), all Medical Devices need to meet the regulations to ensure the safety of the user and the patient. In addition, with the increased use of Software on these systems, entities such as the Food and Drug Administration (FDA) in the United States have identified the need for specific regulation.

In 2006, a working group from the International Electrotechnical Commission (IEC) defined IEC 62304:2006 [IEC 2006]. Thus, the use of IEC 62304 has become fully harmonized in the United States and Europe. The purpose of IEC 62304 is to provide requirements for Healthcare Systems manufacturers with Software to demonstrate their ability to provide Software that consistently meets customer requirements and applicable regulatory requirements.

Figure 3.1 presents an illustrative association of these concepts. IEC 62304 operates within the scope of the MDS, SIs and Software Units (SUs). The IEC 62304:2006 [IEC 2006] and its amendment IEC 62304:2006/AMD 1:2015 [IEC 2015] present the requirements associated with the rigor of the MDS development and maintenance processes, according to the safety risk [Marques 2019].

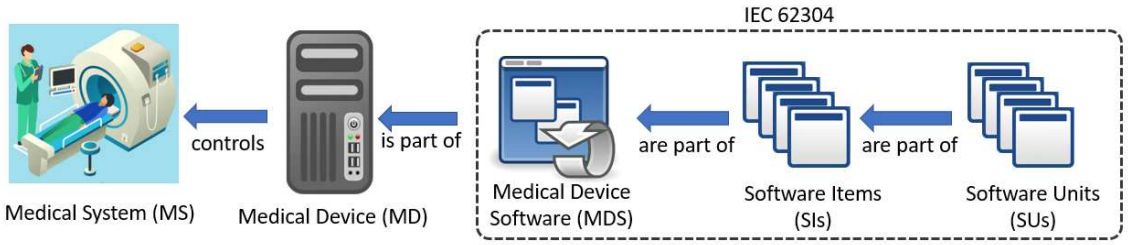

Figura 3.1. Scope of IEC 62304 [Marques et al. 2021]

In 2021, IEC 62304 completes 15 years. Thus, the authors believe that there are already works that report experiences, analyzes and difficulties in its use. These results can be interesting to direct further research and definitions of methods, models, guides or other materials to comply with IEC 62304. Therefore, this chapter aims to present the fundamentals of IEC 62304 with an Agile Software Development Model.

In addition to this section 1, this book chapter has another 11 (eleven) sections. Section 2 presents the acronyms, and section 3 presents the definitions. Section 4 presents the general requirements. Section 5 describes the software development process. Section 6 presents the software maintenance process. Section 7 presents the software risk process. Section 8 describes the software configuration management process. Section 9 presents the software problem resolution process. Section 10 briefly describes the relationship between IEC 62304 and other standards. Section 11 presents an overview of agile methods and Scrum. Section 12 describes our Agile Software Development Model. Section 13 presents the related work identified during a Systematic Literature Mapping. Finally, section 14 presents the final considerations.

3.2. Acronyms

Table 3.1 presents the acronyms required for this book chapter.

Tabela 3.1. Acronyms

Acronym	Definition
ASD	Adaptive Software Development
Cat	Categories
CMMI	Capability Maturity Model Integration
Co	Contribution
CR	Change Request
DSDM	Dynamic Software Development Method
FDA	Food and Drug Administration
FDD	Feature Driven Development
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
MAD	Manifesto for Agile Development
MD	Medical Device
MDS	Medical Device Software
MS	Medical System
PR	Problem Report
SDP	Software Development Plan
SI	Software Item
SLM	Systematic Literature Mapping
SOUP	Software of Unknown Provenance
SU	Software Unit
TDD	Test Driven Development
Var	Variability
XP	Extreme Programming

3.3. Definitions

Table 3.2 presents some definitions required for this book chapter.

3.4. General Requirements

The IEC 62304 defines the life cycle requirements for MDS. The structure of the standard follows a hierarchy composed by processes, activities, and tasks, establishing a common framework for MDS life cycle processes. The hierarchy is illustrated as part of Figure 3.2.

Compliance with the IEC 62304 is defined as implementing all the processes, activities, and tasks identified in this standard following the software safety class. In addition, the MDS manufacturer must identify the safety risks to users (patients and medical staff) regarding software misbehavior. These risks are directly correlated with the software safety classes defined by IEC 62304.

Tabela 3.2. Terms and definitions

Term	Definition
Activity	A set of one or more interrelated or interacting tasks.
Anomaly	Any condition that deviates from the expected based on requirements specifications, design documents, standards, or from someone's perceptions or experiences.
Configuration item	Entity that can be uniquely identified at a given reference point.
Legacy software	Medical Device Software which was legally placed on the market with insufficient objective evidence that it was developed in compliance with the current version of this standard.
Manufacturer	Organization, which conducts software development, maintenance, and other processes, activities, and tasks in the scope of IEC 62304.
Problem report	A record of the actual or potential behavior of the MDS that a user or other interested person believes to be unsafe, inappropriate for the intended use or contrary to specification.
Process	A set of interrelated or interacting activities that transform inputs into outputs.
Risk management file	A set of records and other documents, not necessarily contiguous, that is produced by a Risk assessment process.
Safety	Freedom from unacceptable risk.
Serious injury	Injury or illness that: is life-threatening results in permanent impairment of a body function or permanent damage to a body structure or necessitates medical or surgical intervention.
Software item	Any identifiable part of a computer program.
Software system	Integrated collection of software items organized to accomplish a specific function or set of functions.
Software unit	Software item that is not subdivided into other items.
Task	A single piece of work that needs to be done.

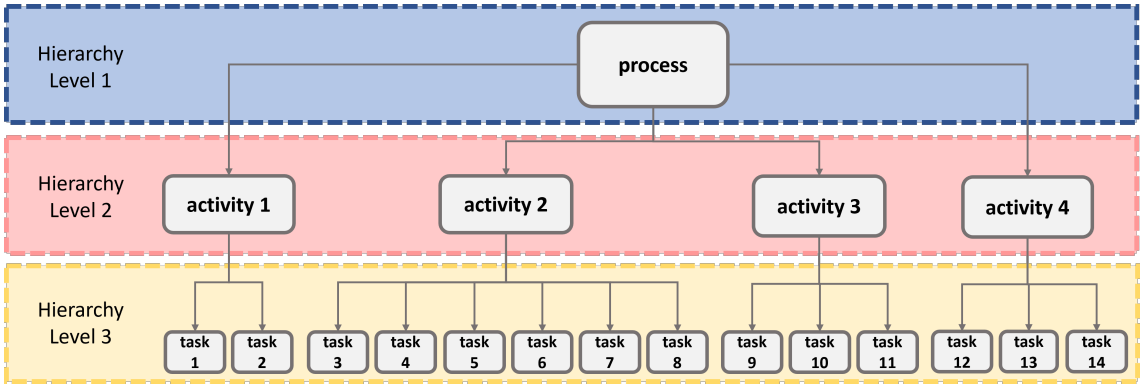

Figura 3.2. Hierarchy of the MDS life cycle

The Medical Device Software manufacturer must apply a risk management pro-

cess using ISO 14971:2019 [ISO 2019], where safety risks are identified to users (patients and medical staff) regarding software misbehavior. This analysis will define the software classes (A, B and C) provided by IEC 62304. The safety classes determine the rigor in the software development process through the activities. Marques et al. (2021) presented the number of activities associated to each class, as shown in Table 3.3.

Tabela 3.3. Software safety classes, impacts and total of activities required

Class	Impact	Required Tasks
A	No injury or damage to health is possible.	48
B	Non-serious injury is possible.	85
C	Death or serious injury is possible.	89

IEC 62304 had an amendment introduced in 2015 and is under review for improvement. The expectation is that IEC 62304 will have a revision in the year 2021. One of the main contributions of the amendment was the flow of risk classification for the Software Systems existing in a Health System. Initially, all analysis begins with the most severe category (Class C). After following the flow provided in Figure 3.3, the classification can be changed to Class A or B if the conditions existing in the risk analysis are met. The amendment also brought editorial and conceptual changes that needed updating.

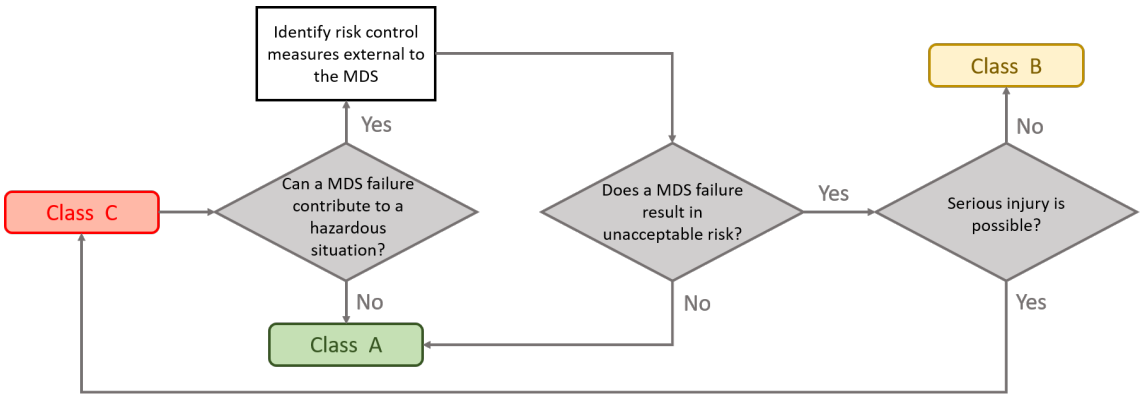

Figura 3.3. Assignment of the software class from the safety risk analysis - adapted from [IEC 2015]

IEC 62304 describes 5 (five) processes: Software Development Process, Software Maintenance Process, Software Risk Management Process, Software Configuration Management Process, and Software Problem Resolution Process, as shown in Figure 3.4. The Software Development Process contains 8 (eight) activities.

The Software of Unknown Provenance (SOUP) is a SI that is already developed and generally available and has not been developed to be incorporated a MD or Software previously developed for which adequate records of the development processes are not available.

3.5. Software Development Process

Some software life cycle models were created over the years. Although they define different strategies for executing the software development and verification, it is possible to see

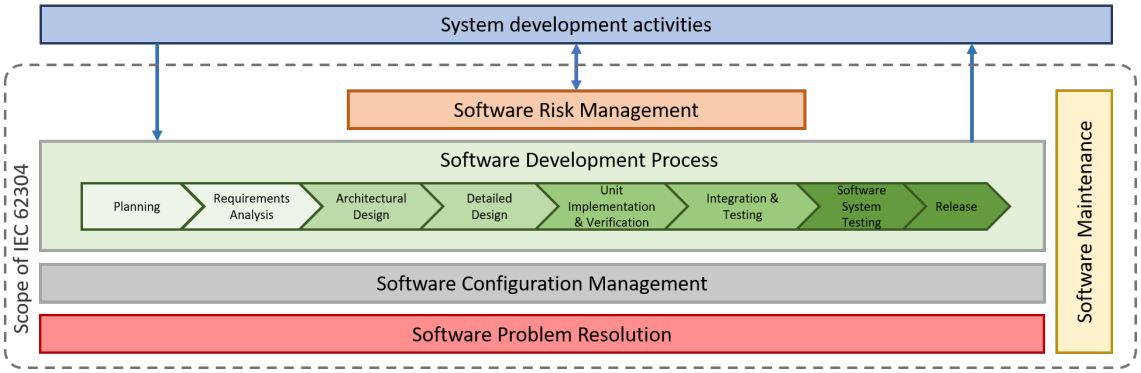

Figura 3.4. Overview of IEC 62304 Processes

that such models differ only in the granularity and the involvement of the user in the evaluation of the software [Sommerville 2015][Pressman and Maxim 2015][Tsui et al. 2015]. The software development is generally divided into 7 (seven) subprocesses: Requirements, Architecture, Design, Implementation, Tests, Verification, and Release [Wasson 2015]. IEC 62304 contains 8 (eight) activities as part of the *Software development process*:

1. *Software development plan*;
2. *Software requirements analysis*;
3. *Software architectural design*;
4. *Software detailed design*;
5. *Software unit implementation and verification*;
6. *Software integration and integration testing*;
7. *Software system testing*; and
8. *Software release*.

3.5.1. Software Development Planning

The *Software development planning* activity contains 11 (eleven) tasks:

1. *Software development plan* (clause 5.1.1);
2. *Keep software development plan updated* (clause 5.1.2);
3. *Software development plan reference to system design and development* (clause 5.1.3);
4. *Software development standards, methods and tools planning* (clause 5.1.4);
5. *Software integration and integration testing planning* (clause 5.1.5);
6. *Software verification planning* (clause 5.1.6);

7. *Software risk management planning* (clause 5.1.7);
8. *Documentation planning* (clause 5.1.8);
9. *Software configuration management planning* (clause 5.1.9);
10. *Supporting items to be controlled* (clause 5.1.10);
11. *Software configuration item control before verification* (clause 5.1.11); and
12. *Identification and avoidance of common software defects* (clause 5.1.12).

Table 3.4 presents the applicability of each task inside the software classes.

Tabela 3.4. Summary of tasks in *Software development planning* activity

Clause	Class A	Class B	Class C
5.1.1	X	X	X
5.1.2	X	X	X
5.1.3	X	X	X
5.1.4			X
5.1.5		X	X
5.1.6	X	X	X
5.1.7	X	X	X
5.1.8	X	X	X
5.1.9	X	X	X
5.1.10		X	X
5.1.11		X	X
5.1.12		X	X

As part of the activity *Software development plan*, the manufacturer shall establish a Software Development Plan (SDP) for conducting the activities of the software development process appropriate to the scope, magnitude, and Software safety classifications of the Software system to be developed. The software development lifecycle shall either be fully defined in the SDP. The plan shall address the following:

1. The processes to be used in the development of the MDS;
2. The artifacts of the activities and tasks;
3. The traceability between System Requirements, Software System Requirements, and System Tests;
4. Software configuration and change management, including SOUP configuration items and software, used to support development; and
5. The software problem resolution for handling problems detected in the deliverables and activities at each stage of the life cycle.

As part of the task *Keep software development plan updated*, the manufacturer shall update the plan as development proceeds as appropriate.

As part of the task *Software development plan reference to system design and development*, System Requirements, as inputs for software development, shall be referenced in the SDP. In addition, the manufacturer shall include or reference the SDP procedures for coordinating the software development and the design and development validation necessary to satisfy customer requirements and applicable regulatory requirements.

As part of the task *Software development standards, methods and tools planning*, the manufacturer shall include or reference in the SDP the standards, methods, and tools associated with the development of Software Items of Class C.

As part of the task *Software integration and integration testing planning*, the manufacturer shall include or reference in the SDP a plan to integrate the SIs (including SOUP) and perform testing during integration. It is acceptable to combine integration testing and Software System testing into a single plan and set of activities.

As part of the task *Software verification planning*, the manufacturer shall include or reference in the software development plan the following verification information:

1. Artifacts requiring verification;
2. The required verification tasks for each lifecycle activity;
3. Milestones at which the artifacts are verified; and
4. The acceptance criteria for each artifact verification.

As part of the task *Software risk management planning*, the manufacturer shall include or reference in the SDP a plan to conduct the activities and tasks of the software Risk Management Process, including the management of risks relating to SOUP.

As part of the task *Documentation planning*, the manufacturer shall include or reference in the SDP information about the documents to be produced during the software development lifecycle. For each identified document or type of document, the following information shall be included or referenced:

1. Title, name or naming convention;
2. Purpose; and
3. Intended audience of the document.

As part of the task *Software configuration management planning*, the manufacturer shall include or reference software configuration management information in the SDP. The software configuration management information shall include or reference:

1. The classes, types, categories or lists of items to be controlled;
2. The software configuration management activities and tasks;

3. The organization(s) responsible for performing software configuration management and activities;
4. Their relationship with other organizations, such as software development or maintenance;
5. When the items are to be placed under configuration control; and
6. When the problem resolution process is to be used.

As part of the task *Supporting items to be controlled*, the items to be controlled shall include tools, items or settings, used to develop and which could impact the MDS.

As part of the task *Software configuration item control before verification*, the manufacturer shall plan to place configuration items under configuration management control before they are verified.

As part of the task *Identification and avoidance of common software defects*, the manufacturer shall include or reference in the SDP a procedure for:

- Identifying categories of defects that may be introduced based on the selected programming technology that are relevant to their Software System; and
- Documenting evidence that demonstrates that these defects do not contribute to unacceptable risk.

3.5.2. Software Requirements Analysis

The *Software requirements analysis* activity captures and defines software requirements from the product level that are implemented by software [Sommerville 2015] [Pressman and Maxim 2015].

The *Software requirements analysis* activity contains 6 (six) tasks:

1. *Define and document software system requirements from system requirements* (clause 5.2.1);
2. *Software system requirements content* (clause 5.2.2);
3. *Include risk control measures in software system requirements* (clause 5.2.3);
4. *Re-evaluate medical device risk analysis* (clause 5.2.4);
5. *Update system requirements* (clause 5.2.5); and
6. *Verify software system requirements* (clause 5.2.6).

Table 3.5 presents the applicability of each task inside the software classes.

As part of the task *Define and document software system requirements from system requirements*, for each MDS, the manufacturer shall define and document Software System Requirements from the System Requirements.

Tabela 3.5. Summary of tasks in *Software requirements analysis* activity

Clause	Class A	Class B	Class C
5.2.1	X	X	X
5.2.2	X	X	X
5.2.3		X	X
5.2.4	X	X	X
5.2.5	X	X	X
5.2.6	X	X	X

As part of the task *Software system requirements content*, the manufacturer shall define and document Software System Requirements from the System Requirements. The manufacturer shall include in the Software System Requirements:

- Functional and capability requirements;
- Software system inputs and outputs;
- Interfaces between the software system and other systems;
- Software-driven alarms, warnings, and operator messages;
- Security requirements;
- User interface requirements implemented by software;
- Data definition and database requirements;
- Installation and acceptance requirements of the delivered MDS at the operation and maintenance site or sites;
- Requirements related to methods of operation and maintenance;
- Requirements related to IT-network aspects;
- User maintenance requirements; and
- Regulatory requirements, according to the nature of the MS.

As part of the task *Include risk control measures in software requirements*, the manufacturer shall include risk control measures implemented in the MDS.

As part of the task *Re-evaluate medical device risk analysis*, the manufacture shall re-evaluate the medical device risk analysis when software requirements are established and update it as appropriate.

As part of the task *Update system requirements*, the manufacturer shall ensure that existing requirements, including System Requirements, are re-evaluated and updated as appropriate as a result of the *Software requirements analysis* activity.

As part of the task *Verify Software System requirements*, the manufacturer shall verify and document that the Software System Requirements:

- Implement System Requirements;
- Do not contradict one another;
- Are expressed in terms that avoid ambiguity;
- Are stated in terms that permit the establishment of test criteria and performance of tests to determine whether the test criteria have been met;
- Can be uniquely identified; and
- Are traceable to System Requirements or another source.

Figure 3.5 presents an overview of the *Software requirements analysis* activity.

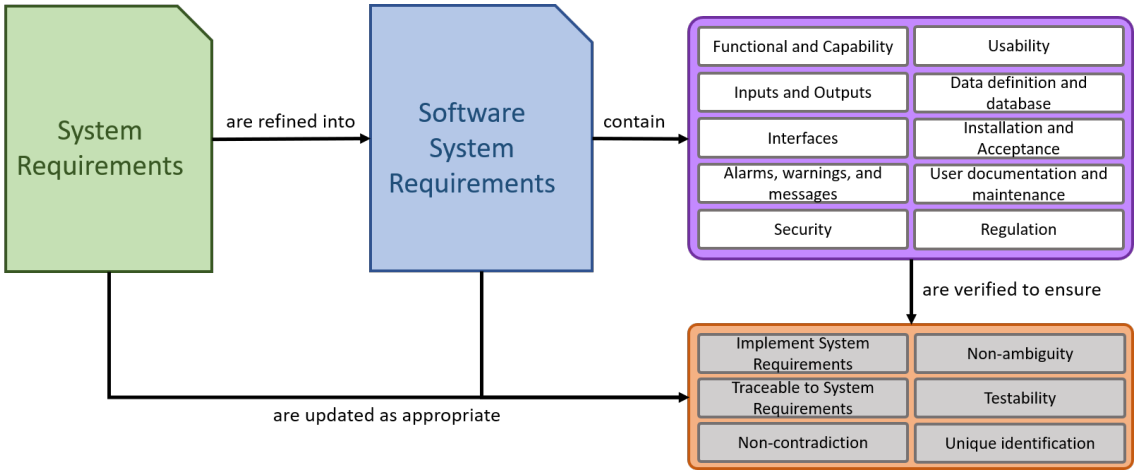

Figura 3.5. Overview of the *Software requirements analysis activity*

3.5.3. Software Architectural Design

The *Software architectural design* activity uses outputs of the software requirements analysis activity to develop the Architecture by creating Software Items and allocating functions expressed by Software System Requirements.

The *Software architectural design* activity contains 6 (six) tasks:

1. *Transform software system requirements into a software architecture* (clause 5.2.1);
2. *Develop an architecture for the interfaces of software items* (clause 5.2.2);
3. *Specify functional and performance requirements of SOUP item* (clause 5.2.3);
4. *Specify system hardware and software required by SOUP item* (clause 5.2.4);
5. *Identify segregation necessary for risk control* (clause 5.2.5); and
6. *Verify software architecture* (clause 5.2.6).

Tabela 3.6. Summary of tasks in *Software architectural design* activity

Clause	Class A	Class B	Class C
5.3.1		X	X
5.3.2		X	X
5.3.3		X	X
5.3.4		X	X
5.3.5			X
5.3.6		X	X

Table 3.6 presents the applicability of each task inside the software classes.

As part of the task *Transform software systems requirements into a software architecture*, the manufacturer transforms the Software System Requirements for the MDS into a documented Architecture that describes the Software's structure and identifies the software items.

As part of the task *Develop a software architecture for the interfaces of software items*, the manufacturer shall develop and document an Architecture for the interfaces between the SIs.

As part of the task *Specify functional and performance requirements of SOUP item*, if a SI is identified as SOUP, the manufacturer shall specify functional and performance requirements for the SOUP item that is necessary for its intended use.

As part of the task *Specify system hardware and Software required by SOUP item*, if a SI is identified as SOUP, the manufacturer shall specify the system hardware and software necessary to support the proper operation of the SOUP item.

As part of the task *Identify segregation necessary for risk control*, the manufacturer shall identify any segregation between SIs that is necessary for risk control, and state how to ensure that such segregation is effective.

As part of the task *Verify software architecture*, the manufacturer verifies and documents that the Architecture of the software implements Software System Requirements and that the Architecture can support interfaces between SIs and between SIs and hardware. The manufacturer shall verify and document that the Architecture:

- Is complete, allocating the Software System Requirements correctly;
- Supports interfaces between Software Items and hardware; and
- Supports the operation of any SOUP items.

Figure 3.6 presents an overview of the *Software architecture design* activity.

3.5.4. Software Detailed Design

The *Software detailed design* activity contains 5 (five) tasks:

1. *Refine software detailed architecture into software units* (clause 5.4.1);

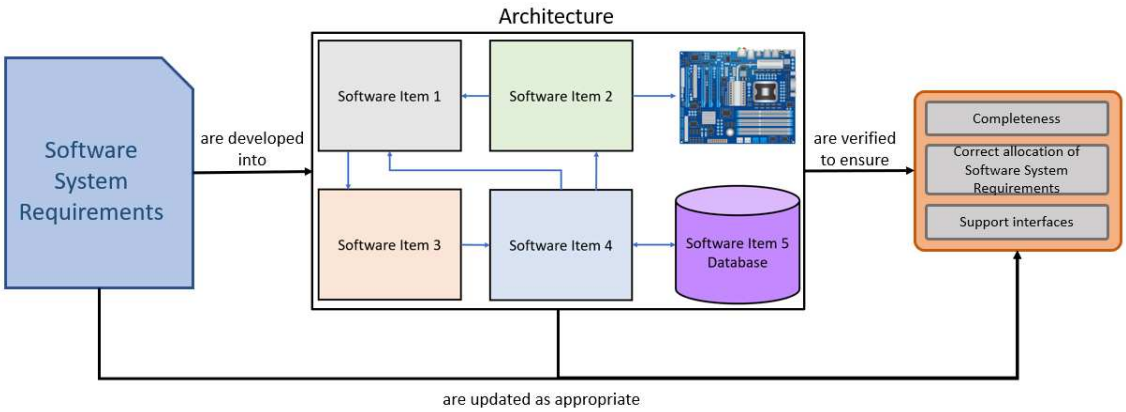

Figura 3.6. Overview of the Software architecture design activity

2. *Develop detailed design for each software unit* (clause 5.4.2);
3. *Develop detailed design for interfaces* (clause 5.4.3); and
4. *Verify detailed design* (clause 5.4.4).

Table 3.7 presents the applicability of each task inside the software classes.

Tabela 3.7. Summary of tasks in *Software detailed design* activity

Clause	Class A	Class B	Class C
5.4.1		X	X
5.4.2			X
5.4.3			X
5.4.4			X

As part of the task *Refine software architecture into software units*, the manufacturer shall refine the Software Architecture until Software Units represent it and develop and document a detailed design for each SU of the SI. The number of levels is defined by the manufacturer, as presented in Figure 3.7.

As part of the task *Develop detailed design for each software unit*, the manufacturer shall document a design with enough detail to allow correct implementation of each SU.

As part of the task *Develop detailed design for interfaces*, the manufacturer shall document a design for any interfaces between the SU and external components (hardware or software), as well as any interfaces between SUs, detailed enough to implement each SU and its interfaces correctly.

As part of the task *Verify detailed design*, the manufacturer shall verify and document that the software detailed design:

- Implements the Architecture;

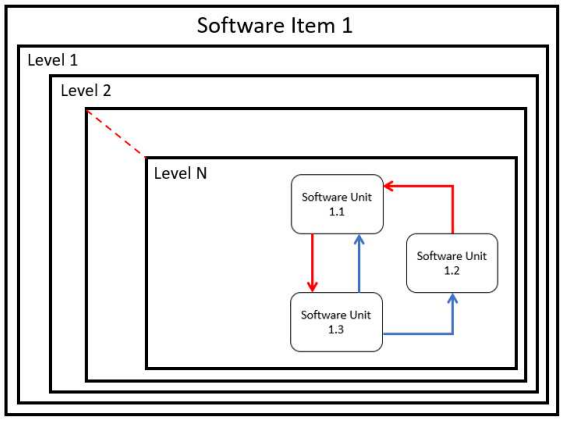

Figura 3.7. Overview of levels of Architecture

- Has correct data and control flow among Software Units;
- Presents the detailed internal logic confirming the refinement from Software System requirements; and
- Is free from contradiction with the Architecture.

Figure 3.8 presents an overview of the *Software detailed design* activity.

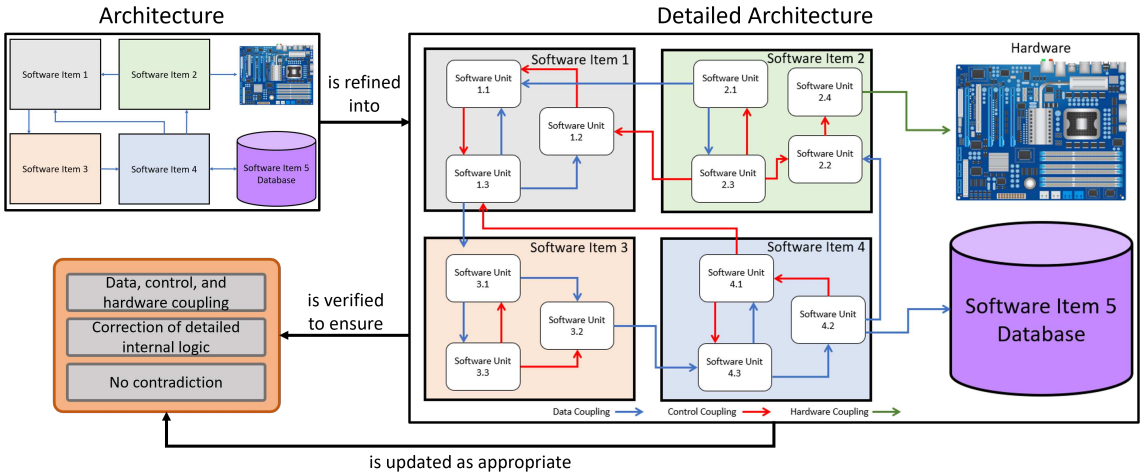

Figura 3.8. Overview of the Software detailed design activity

3.5.5. Software Unit Implementation and Verification

The *Software unit implementation and verification* activity uses outputs of the Software detailed design activity, generating source and executable codes to be loaded inside the selected hardware [Sommerville 2015] [Pressman and Maxim 2015].

The *Software unit implementation and verification* activity contains 5 (five) tasks:

1. *Implement each software unit* (clause 5.5.1);

2. *Establish software unit verification process* (clause 5.5.2);
3. *Software unit acceptance criteria* (clause 5.5.3);
4. *Additional software unit acceptance criteria* (clause 5.5.4); and
5. *Software Unit verification* (clause 5.5.5).

Table 3.8 presents the applicability of each task inside the software classes.

Tabela 3.8. Summary of tasks in *Software unit implementation and verification* activity

Clause	Class A	Class B	Class C
5.5.1	X	X	X
5.5.2		X	X
5.5.3		X	X
5.5.4			X
5.5.5		X	X

As part of the task *Implement each software unit*, the manufacturer shall implement each Software Unit. In the task *Establish software unit verification process*, the manufacturer shall establish strategies, methods and procedures for verifying each Software Unit is required, where verification is done by testing. The test procedures shall be evaluated for correctness.

As part of the task *Software unit acceptance criteria*, the manufacturer shall establish acceptance criteria for Software Units prior to integration into more oversized Software Items as appropriate and ensure that Software Units meet acceptance criteria.

The manufacturer shall include additional acceptance criteria, according to task *Additional software unit acceptance criteria* as appropriate for:

- Proper event sequence;
- Data and control flow;
- Planned resource allocation;
- Fault handling (error definition, isolation, and recovery);
- Initialization of variables;
- Self-diagnostics;
- Memory management and memory overflows; and
- Boundary conditions.

As part of the task *Software unit verification*, the manufacturer shall execute the test and document the results. Figure 3.9 presents an overview of the *Software unit implementation and verification* activity.

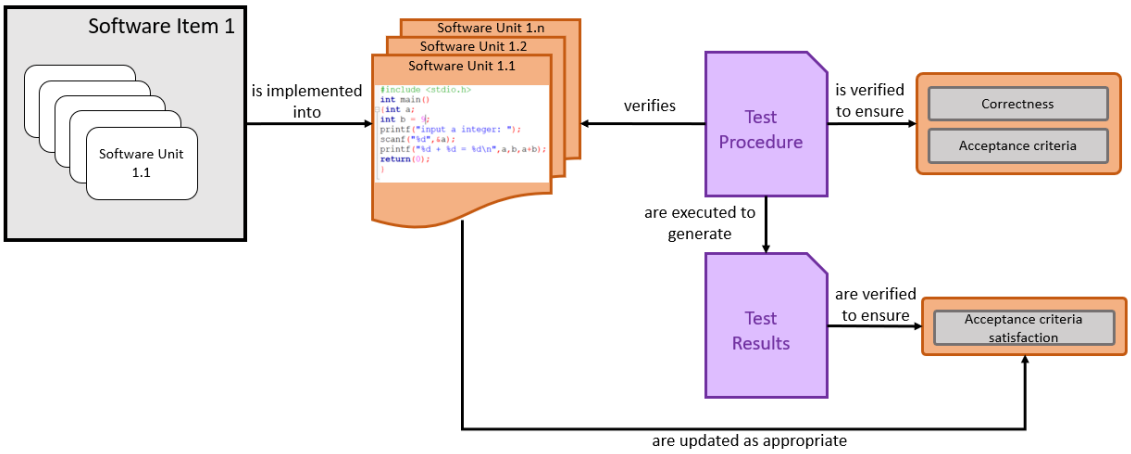

Figura 3.9. Overview of the *Software unit implementation and verification* activity

3.5.6. Software Integration and Integration Testing

The *Software integration and integration testing* activity contains 8 (eight) tasks:

1. *Integrate software units* (clause 5.6.1);
2. *Verify software integration* (clause 5.6.2);
3. *Software integration testing* (clause 5.6.3);
4. *Software integration testing content* (clause 5.6.4);
5. *Evaluate software integration test procedures* (clause 5.6.5);
6. *Conduct regression tests* (clause 5.6.6);
7. *Integration test record contents* (clause 5.6.7); and
8. *Use software problem resolution process* (clause 5.6.8).

Table 3.9 presents the applicability of each task inside the software classes.

Tabela 3.9. Summary of tasks in *Software integration and integration testing* activity

Clause	Class A	Class B	Class C
5.6.1		X	X
5.6.2		X	X
5.6.3		X	X
5.6.4		X	X
5.6.5		X	X
5.6.6		X	X
5.6.7		X	X
5.6.8		X	X

As part of the task *Integrate software units*, the manufacturer shall integrate the Software Units following the integration plan, ensuring the Software Units have been integrated into Software items and the Software system.

As part of the task *Verify software integration*, the manufacturer shall verify and record the following aspects of the software integration following the integration plan:

1. The SUs have been integrated into SIs and the MDS; and
2. The hardware items, SIs, and support for manual operations (e.g., human-equipment interface, online help menus, speech recognition, voice control) of the MS have been integrated.

As part of the task *Software integration testing*, the manufacturer shall test the integrated software items following the integration plan and document the results. As part of the task *Software integration testing content*, the manufacturer shall address whether the integrated SIs performs as intended. Examples to be considered are:

1. The required functionality of the Software;
2. Specified timing and other behavior;
3. Specified functioning of internal and external interfaces; and
4. Testing under abnormal conditions including foreseeable misuse.

As part of the task *Evaluate software integration test procedures*, the manufacturer shall evaluate the integration Test Procedures for correctness.

As part of the task *Conduct regression tests*, when software items are integrated, the manufacturer shall conduct regression testing appropriate to demonstrate that defects have not been introduced into previously integrated Software.

As part of the task *Integration test record contents*, the manufacturer shall:

- Document the test result (pass/fail and a list of anomalies);
- Retain sufficient records to permit the test to be repeated; and
- Identify the tester.

As part of the task *Use software problem resolution process*, the manufacturer shall enter anomalies found during software integration and integration testing into a software problem resolution process.

Figure 3.10 presents an overview of the *Software integration and integration testing* activity.

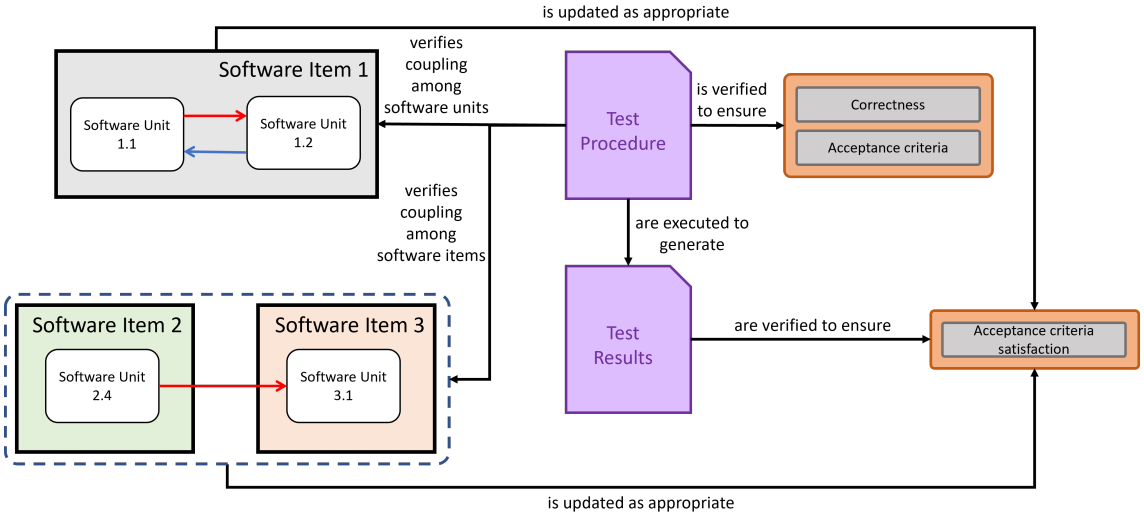

Figura 3.10. Overview of the *Software integration and integration testing* activity

3.5.7. Software System Testing

The *Software system testing* activity contains 5 (five) tasks:

1. *Establish tests for software system requirements* (clause 5.7.1);
2. *Use software problem resolution process* (clause 5.7.2);
3. *Retest after changes* (clause 5.7.3);
4. *Evaluate software system testing* (clause 5.7.4); and
5. *Software system test record contents* (clause 5.7.5).

Table 3.10 presents the applicability of each task inside the software classes.

Tabela 3.10. Summary of tasks in *Software system testing* activity

Clause	Class A	Class B	Class C
5.7.1	X	X	X
5.7.2	X	X	X
5.7.3	X	X	X
5.7.4	X	X	X
5.7.5	X	X	X

As part of the task *Establish tests for software system requirements*, the manufacturer shall establish and perform a set of tests, expressed as input stimuli, expected outcomes, pass/fail criteria and procedures, for conducting Software system testing such that all software requirements are covered. Separate tests for each requirement and tests of combinations of requirements can be performed, primarily if dependencies between requirements exist.

As part of the task *Use software problem resolution process*, the manufacturer shall enter anomalies found during Software System testing into a software problem resolution process.

As part of the task *Retest after changes*, when changes are made during Software system testing, the manufacturer shall:

1. Repeat tests, perform modified tests or perform additional tests, as appropriate, to verify the effectiveness of the change in correcting the problem;
2. Conduct testing appropriate to demonstrate that unintended side effects have not been introduced; and
3. Perform relevant risk management activities.

As part of the task *Verify software system testing*, the manufacturer shall verify that:

- Software System test procedures trace to software requirements;
- All software requirements have been tested or otherwise verified; and
- Test results meet the required pass/fail criteria.

As part of the task *Software system test record contents*, the manufacturer shall:

- A reference to test case procedures showing required actions and expected results;
- The test result (pass/fail and a list of anomalies);
- The version of software tested;
- Relevant hardware and software test configurations;
- Relevant test tools;
- Date tested; and
- The identity of the person responsible for executing the test and recording the test results.

Figure 3.11 presents an overview of the *Software system testing* activity.

3.5.8. Software Release

The *Software release* activity contains 8 (eight) tasks:

1. *Ensure software verification is complete* (clause 5.8.1);
2. *Document known residual anomalies* (clause 5.8.2);

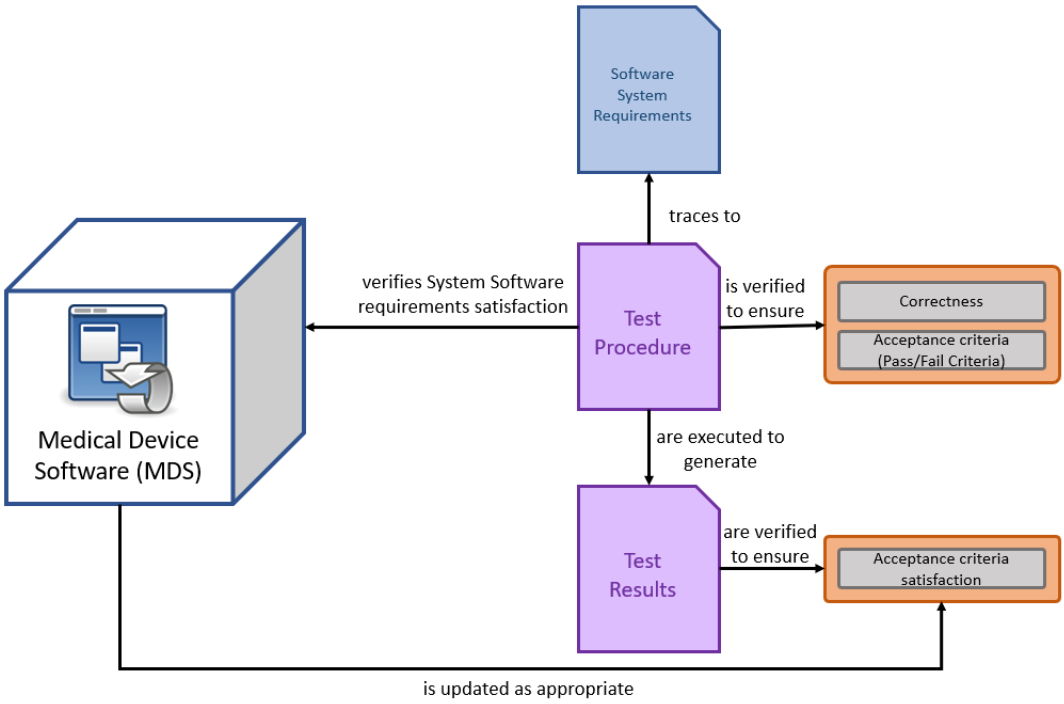

Figura 3.11. Overview of the *Software system testing* activity

3. *Evaluate known residual anomalies* (clause 5.8.3);
4. *Document released versions* (clause 5.8.4);
5. *Document how released software was created* (clause 5.8.5);
6. *Ensure activities and tasks are complete* (clause 5.8.6);
7. *Archive software* (clause 5.8.7); and
8. *Assure reliable delivery of released software* (clause 5.8.8).

Table 3.11 presents the applicability of each task inside the software classes.

Tabela 3.11. Summary of tasks in *Software release* activity

Clause	Class A	Class B	Class C
5.8.1	X	X	X
5.8.2	X	X	X
5.8.3		X	X
5.8.4	X	X	X
5.8.5		X	X
5.8.6		X	X
5.8.7	X	X	X
5.8.8	X	X	X

As part of the task *Ensure software verification is complete*, the manufacturer shall ensure that all software verification activities have been completed and the results evaluated before the Software is released.

As part of the task *Document known residual anomalies*, the manufacturer shall document all known residual anomalies. Additionally, the manufacturer shall document all known residual anomalies to ensure that they do not contribute to an unacceptable risk as part of the task *Evaluate known residual anomalies*.

As part of the task *Document released versions*, the manufacturer shall document the version of the MDS that is being released. Furthermore, the manufacturer shall document the procedure and environment used to create the released software, as presented in task *Document how released Software was created*.

As part of the task *Ensure activities and tasks are complete*, the manufacturer shall ensure that all activities and tasks are complete along with all the associated documentation.

As part of the task *Archive software*, the manufacturer shall archive:

- The MDS and source-code; and
- The generated artifacts.

The archival is required for at least a period determined as the length of the device's lifetime as defined by the manufacturer or a time specified by relevant regulatory requirements.

As part of the task *Assure reliable delivery of released software*, the manufacturer shall establish procedures to ensure that the released MDS can be reliably delivered to the point of use without corruption or unauthorized change. These procedures shall address the production and handling of media containing the MDS, including as appropriate:

- Replication;
- Media labelling;
- Packaging;
- Protection;
- Storage; and
- Delivery.

Figure 3.12 presents an overview of the *Software release* activity.

3.6. Software Maintenance Process

The *Software maintenance process* contains 3 activities:

1. *Establish software maintenance plan*;

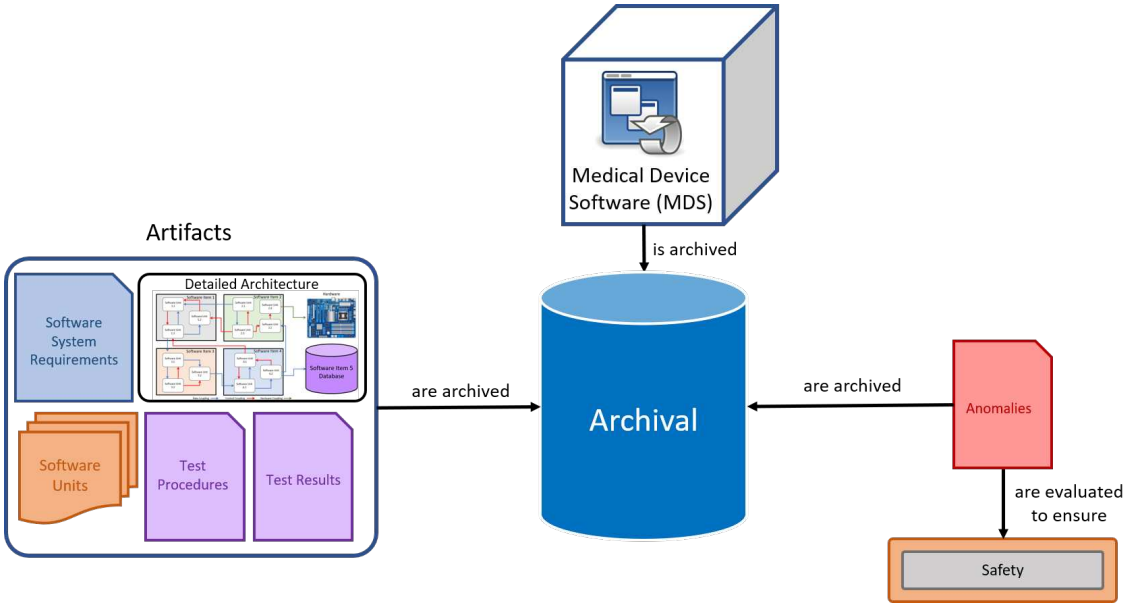

Figura 3.12. Overview of the *Software release* activity

2. *Problem and modification analysis*; and
3. *Modification implementation*.

3.6.1. Establish software maintenance plan

The Establish software maintenance plan (clause 6.1) contains 1 (one) task with the same name of the activity. As part of the task *Establish software maintenance plan*, the manufacturer shall establish a software maintenance plan (or plans) for conducting the activities and tasks of the maintenance process. The plan shall address the following:

- Procedures for feedback arising after the release of the MDS:
  1. Receiving;
  2. Documenting;
  3. Evaluating;
  4. Resolving; and
  5. Tracking.
- Criteria for determining whether the feedback is considered to be a problem;
- Use of the software problem resolution process for analyzing and resolving problems arising after the release of the MDS;
- Use of the software configuration management process for managing modifications to the existing software system; and
- Procedures to evaluate and implement:

1. Upgrades;
2. Bug fixes;
3. Patches; and
4. Obsolescence of SOUP.

Table 3.12 presents the applicability of each task inside the software classes.

Tabela 3.12. Summary of tasks in *Establish software maintenance plan* activity

Clause	Class A	Class B	Class C
6.1	X	X	X

3.6.2. Problem and modification analysis

The *Problem and modification analysis* activity contains 5 (five) tasks:

1. Document and evaluate feedback (clause 6.2.1);
2. Use software problem resolution process (clause 6.2.2);
3. Analyze change request (clause 6.2.3);
4. Change request approval (clause 6.2.4); and
5. Communicate to users and regulators (clause 6.2.5).

Table 3.13 presents the applicability of each task inside the software classes.

Tabela 3.13. Summary of tasks in *Problem and modification analysis* activity

Clause	Class A	Class B	Class C
6.2.1	X	X	X
6.2.2	X	X	X
6.2.3	X	X	X
6.2.4	X	X	X
6.2.5	X	X	X

The task *Document and evaluate feedback* is broken into 3 subtasks:

1. Monitor feedback (clause 6.2.1.1);
2. Document and evaluate feedback (clause 6.2.1.2); and
3. Evaluate problem report's affects on safety (clause 6.2.1.3).

As part of the subtask *Monitor feedback*, the manufacturer shall monitor feedback on MDS released for intended use.

As part of the subtask *Document and evaluate feedback*, the feedback shall be documented and evaluated to determine whether a problem exists in a released MDS. Any such problem shall be recorded as a Problem Report (PR). The PR shall include actual or potential adverse events and deviations from specifications and must be evaluated, as part of the subtask *Evaluate Problem Report's effects on safety*, to determine how it affects the safety of a released MDS and whether a change is needed to correct the problem.

As part of the task *Use software problem resolution process*, the manufacturer shall use the software problem resolution process to address PRs.

As part of the task *Analyze change requests*, the manufacturer shall analyze each Change Request (CR) for its effect on the organization, released MDS, and MS with which it interfaces.

As part of the task *Change request approval*, the manufacturer shall evaluate and approve CRs which modify released MDS. As required by local regulation, and presented by the task *Communicate to users and regulators*, the manufacturer shall inform users and regulators about:

- Any problem in released MDS and the consequences of continued unchanged use; and
- The nature of any available changes to released MDS and how to obtain and install the changes.

3.6.3. Modification implementation

The *Modification implementation* activity contains 2 (two) tasks:

1. *Use established process to implement modification* (clause 6.3.1); and
2. *Re-release modified MDS* (clause 6.3.2).

Table 3.14 presents the applicability of each task inside the software classes.

Tabela 3.14. Summary of tasks in *Modification implementation* activity

Clause	Class A	Class B	Class C
6.3.1	X	X	X
6.3.2	X	X	X

As part of the task *Use established process to implement modification*, the manufacturer shall use the software development process, briefly describe in Section 3.5 to implement the modifications needed by PRs or CRs. Modifications may be released as part of a complete re-release of a MDS or as a modification kit comprising changed SIs and the necessary tools to install the changes as modifications to an existing MDS, as presented in the task *Re-release modified MDS*.

3.7. Software Risk Process

The *Software risk process* contains 4 (four) activities:

1. *Analysis of software contributing to hazardous situations*;
2. *Risk control measures*;
3. *Verification of risk control measures*; and
4. *Risk management of software changes*.

3.7.1. Analysis of Software contributing to hazardous situations

The *Analysis of software contributing to hazardous situations* activity contains 4 (four) tasks:

1. *Identify software items that could contribute to a hazardous situation* (clause 7.1.1);
2. *Identify potential causes of contribution to a hazardous situation* (clause 7.1.2);
3. *Evaluate published SOUP anomaly lists* (clause 7.1.3); and
4. *Document potential causes* (clause 7.1.4).

Table 3.15 presents the applicability of each task inside the software classes.

Tabela 3.15. Summary of tasks in *Analysis of software contributing to hazardous situations* activity

Clause	Class A	Class B	Class C
7.1.1		X	X
7.1.2		X	X
7.1.3		X	X
7.1.4		X	X

As part of the task *Identify Software Items that could contribute to a hazardous situation*, the manufacturer shall identify SIs that could contribute to a hazardous situation identified in the Medical Device Risk Analysis of ISO 14971:2019 [ISO 2019]. The hazardous situation could be the direct result of software failure or failure of a risk control measure implemented in Software.

As part of the task *Identify potential causes of contribution to a hazardous situation*, the manufacturer shall identify potential causes of the SI identified above contributing to a hazardous situation. The manufacturer shall also consider potential causes including, as appropriate:

1. Incorrect or incomplete specification of functionality;
2. Software defects in the identified SI functionality;

3. Failure or unexpected results from SOUP;

4. Hardware failures or other software defects that could result in unpredictable software operation; and

5. Reasonably foreseeable misuse.

As part of the task *Evaluate published SOUP anomaly lists*, if a failure or unexpected results from SOUP is a potential cause of the SI contributing to a hazardous situation, the manufacturer shall evaluate as a minimum any anomaly list published by the supplier of the SOUP item relevant to the version of the SOUP item used in the MDS to determine if any of the known anomalies result in a sequence of events that could result in a hazardous situation.

As part of the task *Document potential causes*, the manufacturer shall document in the risk management file the potential causes of the SI contributing to a hazardous situation.

3.7.2. Risk control measures

The *Risk control measures* activity contains 2 (two) tasks:

1. *Define risk control measures* (clause 7.2.1); and

2. *Risk control measures implemented in software* (clause 7.2.2).

Table 3.16 presents the applicability of each task inside the software classes.

Tabela 3.16. Summary of tasks in *Risk control measures* activity

Clause	Class A	Class B	Class C
7.2.1		X	X
7.2.2		X	X

As part of the task *Define risk control measures*, for each potential cause of the software item contributing to a hazardous situation documented in the risk management file, the manufacturer shall define and document risk control measures. The risk control measures can be implemented in hardware, Software, working environment or user instruction.

As part of the task *Risk control measures implemented in software*, if a risk control measure is implemented as part of the functions of a SI, the manufacturer shall:

1. Include the risk control measure in the software requirements;

2. Assign a software safety class to the SI based on the possible effects of the hazard that the risk control measure is controlling; and

3. Develop the SI in accordance with the Software development process.

3.7.3. Verification of risk control measures

The *Verification of risk control measures* activity contains 2 (two) tasks:

1. *Verify risk control measures* (clause 7.3.1); and
2. *Document traceability* (clause 7.3.3).

Table 3.17 presents the applicability of each task inside the software classes.

Tabela 3.17. Summary of tasks in *Verification of risk control measures* activity

Clause	Class A	Class B	Class C
7.3.1		X	X
7.3.3		X	X

As part of the task *Verify risk control measures*, the implementation of each risk control measure documented shall be verified, and this verification shall be documented.

As part of the task *Document traceability*, the manufacturer shall document traceability of software hazards from:

1. The hazardous situation to the SI;
2. The SI to the specific software cause;
3. The software cause to the risk control measure; and
4. The risk control measure to its.

3.7.4. Risk management of software changes

The *Risk management of software changes* activity contains 3 (three) tasks:

1. *Analyze changes to Medical Device Software with respect to safety* (clause 7.4.1);
2. *Analyze impact of software changes on existing risk control measures* (clause 7.4.2); and
3. *Perform risk management activities based on analyses* (clause 7.4.3).

Table 3.18 presents the applicability of each task inside the software classes.

Tabela 3.18. Summary of tasks in *Risk management of software changes* activity

Clause	Class A	Class B	Class C
7.4.1	X	X	X
7.4.2		X	X
7.4.3		X	X

As part of the task *Analyze changes to Medical Device Software with respect to safety*, the manufacturer shall analyze changes to the MDS to determine whether:

1. Additional potential causes are introduced contributing to a hazardous situation; and

2. Additional software risk control measures are required.

As part of the task *Analyze the impact of software changes on existing risk control measures*, the manufacturer shall analyze changes to the Software, including changes to SOUP, to determine whether the software modification could interfere with existing risk control measures.

As part of the task *Perform risk management activities based on analyses*, the manufacturer shall perform all risk management activities defined in this section based on these analyses.

3.8. Software Configuration Management Process

The *Software configuration management process* contains 3 (three) activities:

1. *Configuration identification*;

2. *Change control*; and

3. *Configuration status accounting*.

3.8.1. Configuration identification

The *Configuration identification* activity contains 3 (three) tasks:

1. *Establish means to identify configuration items* (clause 8.1.1);

2. *Identify SOUP* (clause 8.1.2); and

3. *Identify system configuration documentation* (clause 8.1.3).

Table 3.19 presents the applicability of each task inside the software classes.

Tabela 3.19. Summary of tasks in *Configuration identification* activity

Clause	Class A	Class B	Class C
8.1.1	X	X	X
8.1.2	X	X	X
8.1.3	X	X	X

As part of the task *Establish means to identify configuration items*, the manufacturer shall establish a scheme for the unique identification of configuration items and their versions to be controlled for the project. This scheme shall include other MDS or entities such as SOUP and documentation.

As part of the task *Identify SOUP*, for each SOUP configuration item being used, including standard libraries, the manufacturer shall document:

1. The title,
2. The manufacturer, and
3. The unique SOUP designator.

As part of the task *Identify System configuration documentation*, the manufacturer shall document the set of configuration items and their versions that comprise the MDS configuration.

3.8.2. Change control

The *Change control* activity contains 3 (three) tasks:

1. *Approve change requests* (clause 8.2.1);
2. *Implement changes* (clause 8.2.2);
3. *Verify changes* (clause 8.2.3); and
4. *Provide means for traceability of change* (clause 8.2.4).

Table 3.20 presents the applicability of each task inside the software classes.

Tabela 3.20. Summary of tasks in *Change control* activity

Clause	Class A	Class B	Class C
8.2.1	X	X	X
8.2.2	X	X	X
8.2.3	X	X	X
8.2.4	X	X	X

As part of the task *Approve change requests*, the manufacturer shall provide configuration items only in response to an approved Change Request.

As part of the task *Implement changes*, the manufacturer shall implement the change as specified in the Change Request. In addition, the manufacturer shall identify and perform any activity that needs to be repeated as a result of the change, including changes to the software safety classification of MDS and SIs.

As part of the task *Verify changes*, the manufacturer shall verify the change, including repeating any verification that a change has invalidated.

As part of the task *Provide means for traceability of change*, the manufacturer shall create an audit trail whereby each of the following items are evaluated:

1. Change Request (CR);
2. Relevant Problem Report (PR); and
3. Approval of the CR.

3.8.3. Configuration status accounting

The *Configuration status accounting* (clause 8.3) contains 1 (one) task with the same name of the activity. As part of the task *Configuration status accounting*, the manufacturer shall retain retrievable records of the history of controlled configuration items, including System configuration. Table 3.21 presents the applicability of each task inside the software classes.

Tabela 3.21. Summary of tasks in *Configuration status accounting* activity

Clause	Class A	Class B	Class C
8.3	X	X	X

3.9. Software Problem Resolution Process

The *Software problem resolution* process contains 8 (eight) tasks:

1. *Prepare problem reports* (clause 9.1);
2. *Investigate the problem* (clause 9.2);
3. *Advise relevant parties* (clause 9.3);
4. *Use change control process* (clause 9.4);
5. *Maintain records* (clause 9.5);
6. *Analyze problems for trends* (clause 9.6);
7. *Verify software problem resolution* (clause 9.7); and
8. *Test documentation contents* (clause 9.8).

Table 3.22 presents the applicability of each task inside the software classes.

Tabela 3.22. Summary of tasks in *Software problem resolution* process

Clause	Class A	Class B	Class C
9.1	X	X	X
9.2	X	X	X
9.3	X	X	X
9.4	X	X	X
9.5	X	X	X
9.6	X	X	X
9.7	X	X	X
9.8	X	X	X

As part of the task *Prepare problem reports*, the manufacturer shall prepare a PR for each problem detected in a MDS. Problem Reports shall include a statement of criticality (for example, effect on performance, safety, or security) as well as other information

that may aid in the resolution of the problem (for example, devices affected, supported accessories affected).

As part of the task *Investigate the problem*, the manufacturer shall:

1. Investigate the problem and, if possible, identify the causes;
2. Evaluate the problem's relevance to safety using the software risk management process;
3. Document the outcome of the investigation and evaluation; and
4. Create a Change Request(s) for actions needed to correct the problem or document the rationale for taking no action.

As part of the task *Advise relevant parties*, the manufacturer shall advise relevant parties of the existence of the problem, as appropriate.

As part of the task *Use change control process*, the manufacturer shall approve and implement all Change Requests, observing the requirements of the Change control process.

As part of the task *Maintain records*, the manufacturer shall maintain records of PRs and their resolution, including their verification. Additionally, the manufacturer shall perform analysis to detect trends in Problem Reports.

As part of the task *Verify software problem resolution*, the manufacturer shall verify resolutions to determine whether:

1. Problem Report has been resolved, and the Problem Report has been closed;
2. Adverse trends have been reversed;
3. Change Requests have been implemented in the appropriate MDS and activities; and
4. Additional problems have been introduced.

As part of the task *Test documentation contents*, when testing, retesting or regression testing SIs and MDS following a change, the manufacturer shall include in the test documentation:

1. Test results;
2. Anomalies found;
3. The version of Software tested;
4. Relevant hardware and software test configurations;
5. Relevant test tools;
6. Date tested; and
7. Identification of the tester.

3.10. Relationship with other Standards

As described before, the IEC 62304 applies to the development and maintenance of MDS. The Software is considered a part of the MD. The IEC 62304 should be used together with other appropriate standards when developing an MD. Medical device management standards such as ISO 13485:2016 [ISO 2016a] and ISO 14971:2019 [ISO 2019] provide a management environment that lays a foundation for an organization to develop products. Safety standards such as IEC 60601-1-12:2014/AMD 1:2020 [IEC 2020], IEC 61010-1:2010 [IEC 2010a], and IEC 82304-1:2016 [IEC 2016] give specific direction for creating safe Medical Devices. When Software is a part of these Medical Devices, IEC 62304 provides more detailed direction on developing and maintaining safe MDS. Many other standards such as ISO/IEC/IEEE 12207:2017 [ISO 2017], IEC 61508-3:2010 [IEC 2010b], and ISO/IEC/IEEE 90003:2018 [ISO 2018] can be looked to as a source of methods, tools and techniques that can be used to implement the requirements in IEC 62304.

Figure 3.19 shows the relationship among these standards.

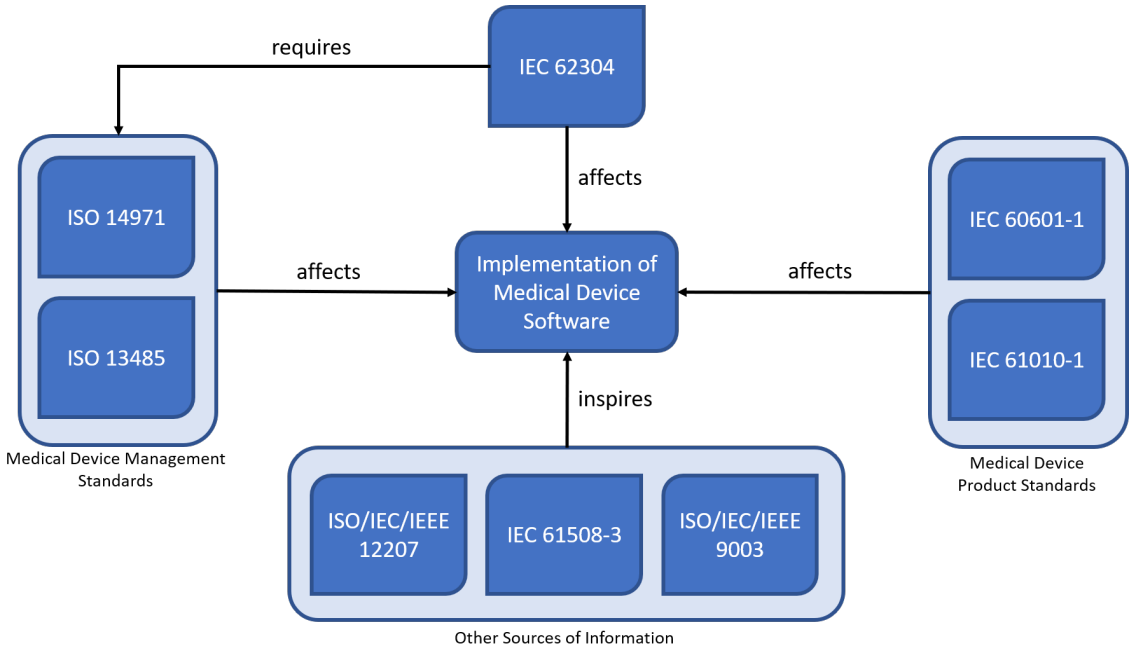

Figura 3.13. Relationship with other standards

3.11. Agile Methods and Scrum

According to Davis (2013), in 2001, a group of 17 professionals met and produced a document that established the principles of agile development. This document, called the Manifesto for Agile Development (MAD), was prepared broadly and generically. These general terms and concepts started to guide the agile way of managing projects.

Several proponents of agile methods agreed with the MDA [Beck et al. 2001], shown in Figure 3.14. It synthesizes the origins of a set of lightweight methodologies, such as Scrum [Schwaber and Beedle 2001], Extreme Programming (XP) [Beck 2000],

Crystal [Cockburn 2004], Feature Driven Development (FDD) [Palmer and Felsing 2002], Test Driven Development (TDD) [Astels 2003], Dynamic Software Development Method (DSDM) [Stapleton 1997] and Adaptive Software Development (ASD) [Highsmith 2000].

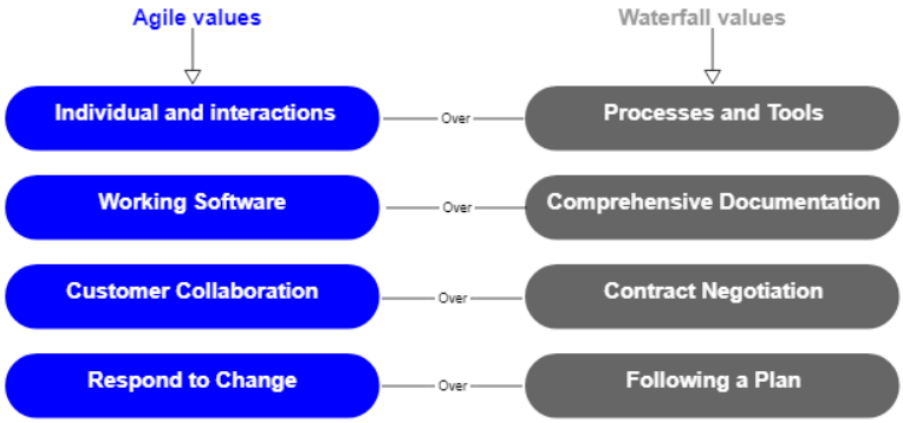

Figura 3.14. Manifesto for Agile Development (MAD) [Beck et al. 2001]

In addition to these fundamental values identified, the participants of the MAD also created 12 principles that guide the agile development of Software [Davis 2013]:

1. Our highest priority is to satisfy the customer through the early and continuous delivery of valuable Software;
2. Accept changing requirements, even at the end of development. Agile processes are adapted to changes so that the client can gain competitive advantages;
3. Deliver Software frequently running, on the scale of weeks to months, with a preference for shorter periods;
4. Business-related people and developers must work together daily throughout the project;
5. Build projects around motivated individuals. Giving them the necessary environment and support and trusting that they will do their job;
6. The most efficient and effective method of transmitting information to a development team is through face-to-face conversation;
7. Functional Software delivery is the primary measure of progress;
8. Agile processes promote a sustainable environment. Sponsors, developers and users must be able to maintain constant steps indefinitely;
9. Continuous attention to technical excellence and good design increases agility;
10. Simplicity;
11. The best architectures, requirements and designs emerge from self-organizing teams; and

12. At regular intervals, the team reflects on becoming more effective, so it adjusts and optimizes its behavior accordingly.

Principles 1, 2 and 7 are strongly correlated. In principle 2, breaking the paradigm between agile and traditional development stands out since traditional development avoids and makes it challenging to change requirements. In traditional projects, it is essential to follow a plan, and any variation can mean a significant risk to the project's success. In agile projects, on the other hand, the customer is the one who dictates the priorities, and if the changes add more value to the customer, these are welcome, as stated in principle 1. In principle 7, as functional Software is the primary measure of progress, the added value to the customer is the same as the working Software.

In principle 3, agile developments work with the concept of iterations, with cyclical efforts of fixed duration. The work to be done is selected and prioritized before the start and then delivered at the end. This principle strongly correlates with 9, providing a well-designed software with incremental delivery and modularity.

Principles 4, 5, 6 and 11 are associated with human relationships. Principle 4 focuses on free and unhindered communication from any barrier. Principles 5 and 11 provide that teams are self-organizing and do not need someone by their side to tell them what to do. Principle 6 establishes a preference for verbal and informal communication overwritten and formal communication.

The rationale for principle 8 is that projects can keep pace indefinitely without the team experiencing fatigue. Principle 10 focuses on keeping things simple and eliminating what is considered unnecessary. Finally, principle 12 appears in several methods with "Retrospective" calls. At the end of each iteration, the team looks at the completed work and reflects on what went right and what went wrong.

According to Stober and Hansmann (2010), agile thinking is an attempt to simplify things, reducing planning complexity, focusing on customer value and shaping a favorable climate for participation and collaboration. Furthermore, Vuori(2011) points out that there is a tendency in companies to transform their Software practices and product development to a more incremental way, using agile development.

Sutherland (2010) defined Scrum as an iterative and incremental framework for application development, and its structure is defined in work cycles, which happen as a race, called Sprint. One Sprint typically has 1 (one) to 4 (four) weeks, ending on a specific date, regardless of the work being completed, and is never extended. Scrum has a series of defined roles with different responsibilities, as shown in Table 3.23.

The Scrum Master protects the team by ensuring that it does not over-commit itself to what it can accomplish during a sprint. He also acts as a facilitator and becomes responsible for removing any obstacles that Scrum Team raises during these meetings.

A meeting called Sprint Planning takes place at the beginning of each Sprint. The Product Owner is a role of important responsibility and visibility in the Scrum method. This represents the customer in decisions and prioritization, considering the value added to the product.

The Product Owner and Scrum team review the product backlog and discuss the

Tabela 3.23. Roles of Scrum

Role	Description
Product Owner	Defines the items that make up the product backlog and prioritizes them in sprint planning.
Scrum Master	Ensures that the team respects and follows Scrum values and practices. It also protects the team by ensuring that it does not overcommit itself to what it is capable of accomplishing during a sprint.
Scrum Team	Formed by the development team. There is not necessarily a functional division through traditional roles such as programmer, test analyst or architect. Everyone on the project works together to complete the set of work they have jointly committed to for a sprint.

goals and context for the items. In addition, the Scrum Team selects the items from the product backlog and commits to completing, by the end of the Sprint, forming the sprint backlog.

A Release is the delivery of one or more product increments, generated in one or more successive sprints, for use. Scrum projects perform frequent releases. Performing releases throughout a project gets frequent feedbacks and promote a sense of development evolution.

The Product Backlog is a list containing all the desired functionality for a product. The Product Owner defines the content of this list. The Product Backlog does not need to be complete at the beginning of a project. Instead, it can start with whatever is most apparent at first. Over time, the Product Backlog grows and changes as the team learns more about the product and its users.

The Sprint Backlog is a list of tasks that the Scrum Team undertakes to do in a Sprint. The Sprint backlog items are extracted from the Product Backlog by the Scrum Team based on the priorities set by the Product Owner and the team's perception of the time needed to complete the various functionality.

After the Sprint is completed, there is a Sprint Retrospective, where the team and stakeholders discuss and review the results, identifying applicable improvements. Figure 3.15 presents the Scrum structure.

In a Scrum project, the team monitors its progress against a plan, updating a Burndown Chart at the end of each Sprint. The vertical axis of a Burndown Chart shows the amount of work that still needs to be done at the beginning of each Sprint. The horizontal axis represents the measure of time. Figure 3.16 presents an example of the Burndown Chart.

3.12. Agile Software Development Model

The Agile Software Development Model presented in Figure 3.17 consists of 7 stages. Stage A is planning, where the two plans required for IEC 62304 are generated. These plans should describe the other stages of the agile software development model proposed in this work, including the roles, responsibilities and tools used in software development.

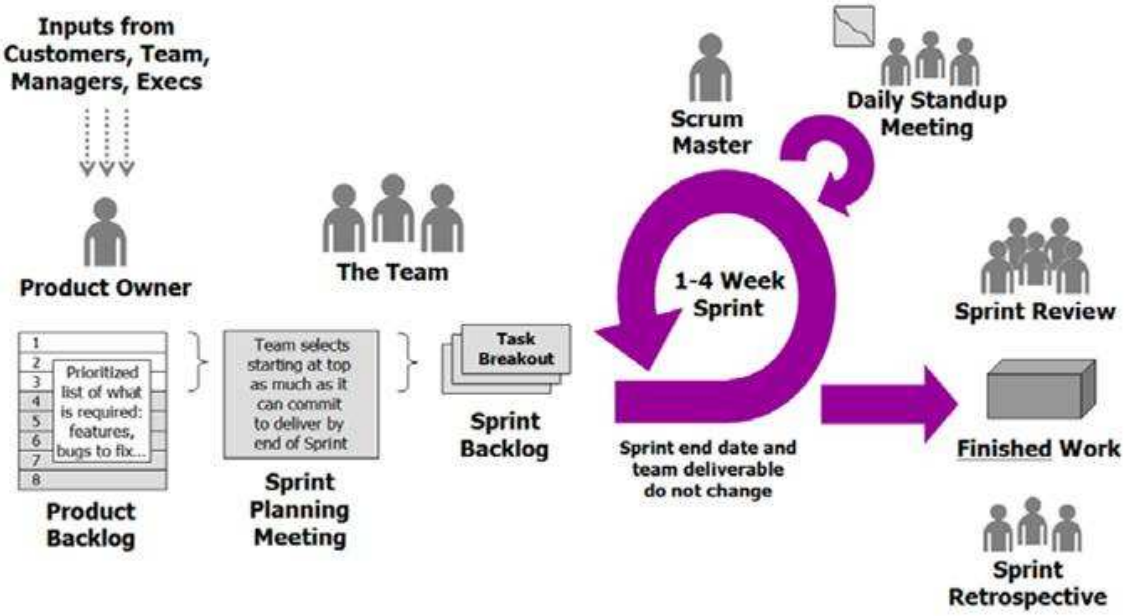

Figura 3.15. Scrum framework [Beck et al. 2001]

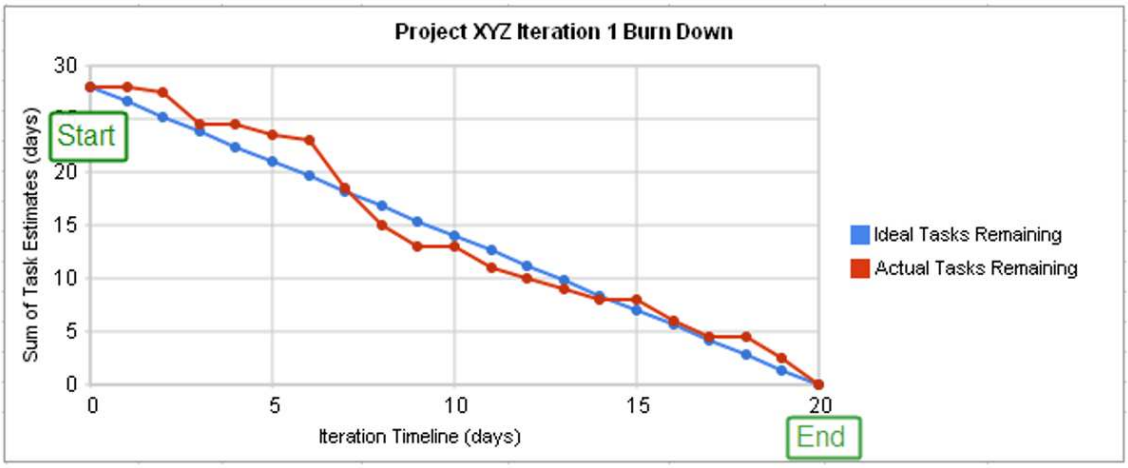

Figura 3.16. Example of Burn Down Chart [Ambler 2002]

Stage B represents the Architecture Design, involving the software items and accommodating the future allocation of Software System requirements. Stage A and B represent the inputs needed for the repetitive execution of sprints.

Stages C and D are within the Sprint. Stage C represents the Sprint Planning with a focus on prioritization of the system requirements. In stage D, the specification of Software Requirements is from the refinement of the prioritized System Requirements. With the Software Requirements and the Architecture, a Detailed Architecture is generated specifying the software units for this Sprint's software requirements implementation, thus generating an implementation of each software unit. Also, within the sprints, a set of testing procedures for the software units is built and executed, thus causing the test results at the unit level. Finally, Sprints are repeated until all software units are architecturally

detailed, implemented and tested.

Once the implementations of the software units are completed, stage E is the integration implementation and testing. At this stage, the Software Units built within the Sprints will be integrated into Software Items. In this same stage, the data and control couplings between the Software Units belonging to the same software item will be evaluated, respecting the detailed specification of the architecture. In stage F, the system testing will be done, involving integrating the Software Items and the hardware designated for the MDS. Finally, in stage G, the MDS will be released for its use.

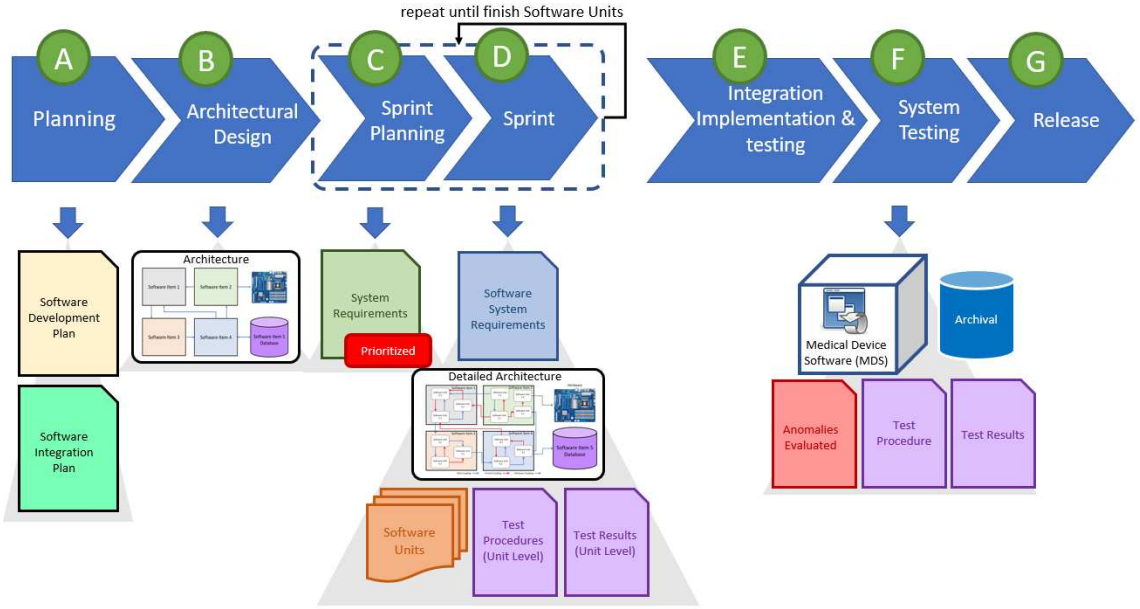

Figura 3.17. Agile software development model

During the sprint planning meeting (Figure 3.18 - stage C), the system requirements are evaluated. The team chooses those system requirements that will be implemented by Software, which are prioritized into the Sprint. Once the System Requirements set is selected, the scrum master monitors the progress of its refinements in software requirements and other development process artifacts, using the Burn Down Chart. During the Sprint (Figure 3.18 - stage D), which can last up to 4 weeks, the development of the Software System requirements, the detailed Architecture, implementation of the Software Units, and their Test procedures and Test Results are carried out in this time. The Weekly Meeting evaluates the progress of refining the system requirements prioritized for this Sprint. The Sprint Review ensures that generated artifacts have been verified at the Software Unit level. The team will also carry out a Sprint Retrospective that will record and evaluate the possible Anomalies identified in the artifacts generated in this Sprint, scheduling the correction of these anomalies for a future Sprint. Stages C and D are detailed in Figure 3.18.

According to Figure 3.19, after all the software units are developed within the numerous sprints performed, the integration implementation Testing, stage E, performs the verification of the integration between the software units belonging to the same software item. Therefore, test procedures at the integration level will be created and generated

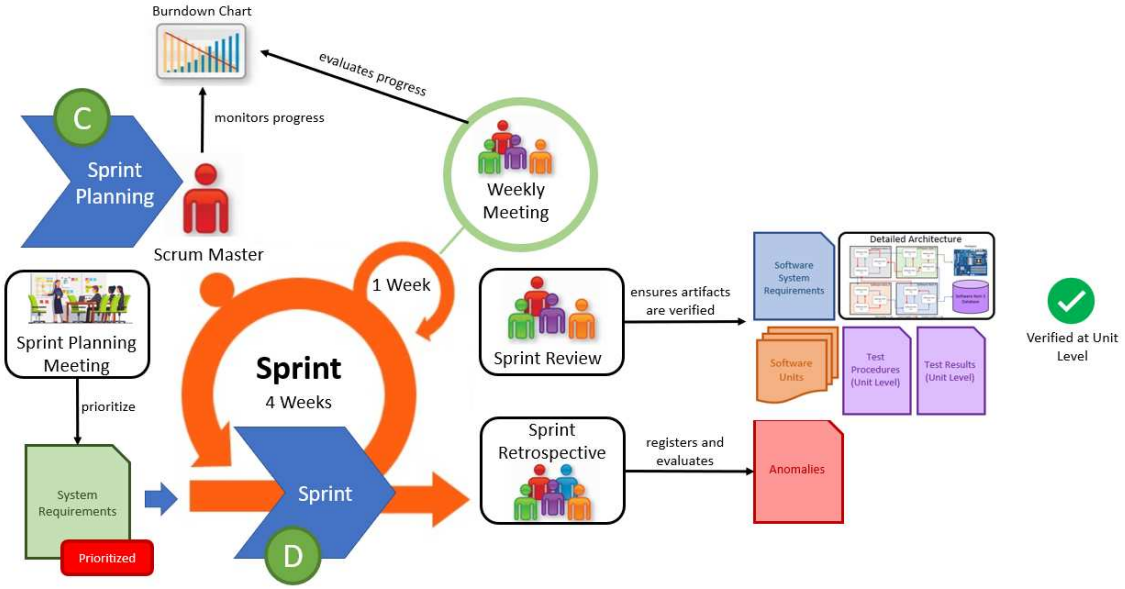

Figura 3.18. Sprint structure

from these tests when executed. In system testing, stage F, on the other hand, performs the verification of the integration between the software items with the definition of test procedures that exercise the software requirements and that, when executed, generate test results. Finally, in stage G, the manufacturer archives the artifacts generated during the Medical device software and evaluate the possible anomalies identified regarding the safety aspect.

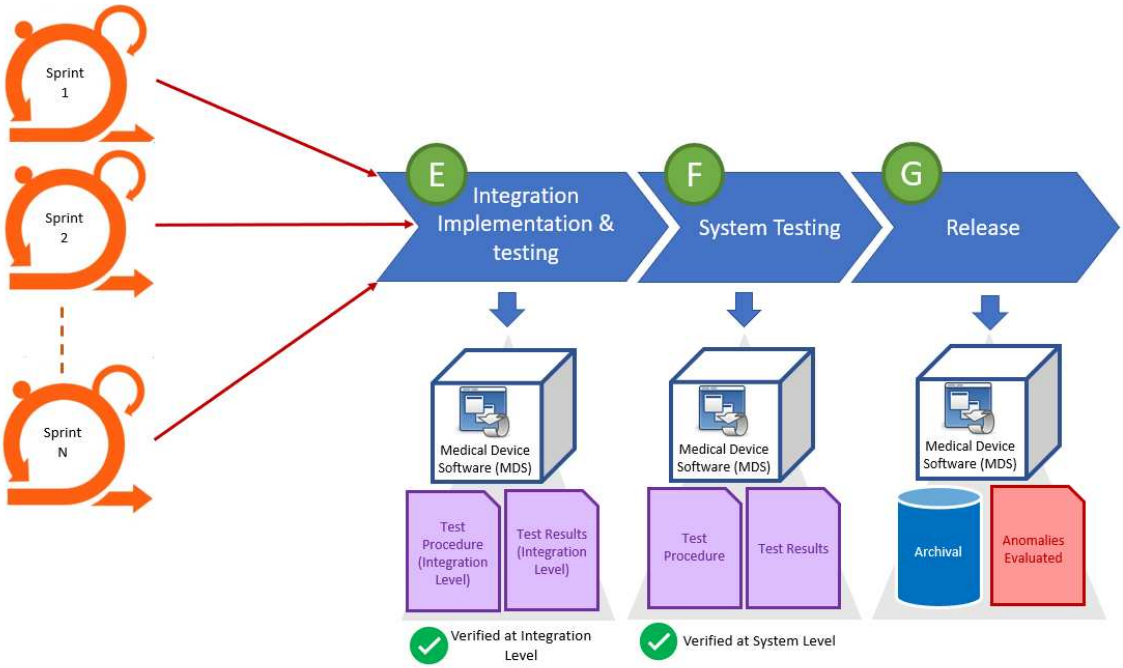

Figura 3.19. Integration, system testing, and release

3.13. Related Work

For the identification of the related work, we executed a Systematic Literature Mapping (SLM) that was published in March 2021 in Journal of Health Informatics [Marques et al. 2021].

3.13.1. Systematic Literature Mapping (SLM)

Our previous work [Marques et al. 2021] has identified works reporting the usage of IEC 62304. We also identified the advantages and difficulties of using IEC 62304. We found and classified 22 (twenty-two) works that met the inclusion criteria, as part of our SLM. Using the instructions suggested by Petersen et al. (2015), 4 Categories (Cat) were defined and identified (Cat1...4):

- Conceptual Analysis (Cat1): works that discuss a theoretical concept or a new approach, but without validating it;
- Experimental Analysis (Cat2): works that discuss a theoretical concept or a new approach with validation;
- Experience Report (Cat3): works that report an industrial experience without declaring research questions or theoretical concepts; and
- Survey (Cat4): works that collect data based on a questionnaire.

The IEC 62304 standard presents software development and support processes such as configuration and risk control. Thus, the works usually present contributions that describe, support, elucidate their processes and activities through guides, models, methods or comparatives. Thus, the Contribution (Co) axis identified 4 (four) types of contribution (Co1...4), as follows:

- Guidance (Co1): works that support the understanding of IEC 62304 processes and activities;
- Model (Co2): works that present an extension and detailing of the processes and activities of IEC 62304, with reusable tools, methods and checklists;
- Method (Co3): works that present methods to meet only one IEC 62304 process or activity; and
- Comparative (Co4): works that present a comparison of IEC 62304 in some perspective.

IEC 62304 describes 5 (five) processes: Software Development Process, Software Maintenance Process, Software Risk Management Process, Software Configuration Management Process, and Software Problem Resolution Process. The Software Development Process contains 8 (eight) activities, as presented in sections 3.5. For the Variability (Var) axis, the team leading the SLM decided to group 4 (four) processes and 8 (eight) activities of the Software Development Process into 4 (four) groups with identification (Var1...4). We did not find any work associated with the Software Maintenance Process:

- Planning and Requirements (Var1): works that address the Development Planning and Requirements Analysis activities that belong to the Software Development Process;
- Design and Implementation (Var2): works that address the Architectural Design, Detailed Design and Unit Implementation and Verification activities that also belong to the Software Development Process;
- Tests (Var3): works that address the Integration and Integration Testing and Software System Testing activities that belong to the Software Development Process; and
- Risks (Var4): works that address the Software Risk Management process.

3.13.2. Results

Figure 3.20 presents the 22 works grouped into the 3 (three) axis. We mapped some works to more than one possibility within the same axis.

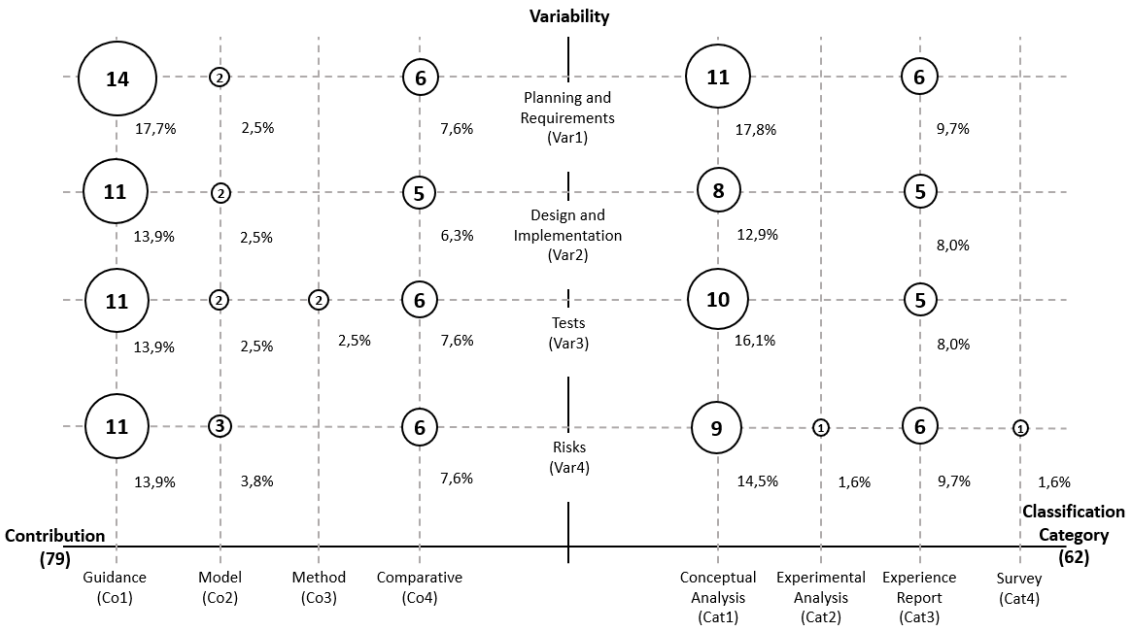

Figura 3.20. SLM bubble chart

Jordan (2006) and Varri et al. (2019) described IEC 62304 presenting their software processes and classes. Jordan (2006) was the first to deal with IEC 62304 after its issuance. Varri et al. (2019) plays a similar role but updated with the IEC 62304 amendment issued in 2015.

Huhn and Zechner (2010) proposed a method to evaluate arguments centred on quality and engineering in reliability cases (assurance cases) to guarantee software development, according to IEC 62304.

Mc Caffery et al. (2010) compared the depth of current medical equipment regulations, focusing on IEC 62304, concerning Capability Maturity Model Integration (CMMI)

in specifying which risk management practices companies should adopt when developing software medical devices. Bianco (2011) describes a quality system that integrates as main processes those specified by IEC 62304 and applies a risk-oriented approach and supporting processes such as contract and supplier management.

Cruciani and Vicario (2011) identified the need for the extensive testing effort necessary to comply with IEC 62304 prescriptions, presenting how data flow analysis can identify an appropriate set of constraints explored in the verification stage at reducing the set of tests, preserving the coverage. Finally, Larson et al. (2012) presented an initial proposal for a real-time and critical computing platform to integrate heterogeneous devices, identifying the absence of requirements in IEC 62304 for this purpose.

McHugh et al. (2012) identified how regulations affect medical device software development companies, and they made recommendations on compliance with IEC 62304. Wong and Callaghan (2012) described an approach taken to manage the baselines of software requirements for medical devices in need of compliance with IEC 62304.

Regan et al. (2013) described the extent and diversity of traceability requirements in medical device standards and guidelines at each stage of the software development life cycle, as required by IEC 62304.

Höss et al. (2014) described the first experiences with the implementation of IEC 62304 to guarantee the quality of a radiotherapy unit, being the only work with an industry report.

Rust et al. (2016) described a roadmap that assists small and medium-sized companies in developing medical Software by IEC 62304, offering design patterns to generate pre-established artifacts and models to demonstrate compliance. They also presented a software development plan to help organizations in which the use of IEC 62304 can be problematic because they are new organizations or have limited experience in the medical field. They also present a roadmap, divided into two levels: (a) the high level consists of the activities and tasks necessary for implementing IEC 62304, and (b) the low level contains the artifacts of design standards and instructions related to the tasks. The script involved a consultation, by questionnaire, with 6 (six) experts performing the evaluation.

Laukkarinen et al. (2017) examined the obstacles and benefits of using DevOps to develop Software for medical devices. Finally, Hatcliff provided an overview of the life cycle problems of interoperable medical devices not sufficiently addressed in existing medical device standards, including IEC 62304.

Kasisopha and Meananeatra (2019) presented a directive for Very Small Entities (VSE) that employ ISO TR 29110 [ISO 2016b] and aspire to apply the IEC 62304 processes in constructing Software for medical devices. Marques and Yelisetty (2019) analyzed the characteristics of specification of software requirements in regulated environments such as medical, aeronautical and rail. The four characteristics identified are consistency (internal and external), unambiguity, verifiability and traceability. The document also describes the three standards used in these regulated environments (RTCA DO-178C, IEC 62279 and IEC 62304). It examines their similarities and differences from the point of view of the requirements' specification.

Table 3.24 identified the difficulties inside the 22 works.

Tabela 3.24. Difficulties of using IEC 62304 [Marques et al. 2021]

Difficult	Description
DIF 1	Challenging to select and use automated tools to comply with the standard.
DIF 2	High initial effort with a learning curve to overcome.
DIF 3	There is a lack of recommendations of methods and techniques.
DIF 4	High test effort.
DIF 5	Difficult to use in small and medium-sized companies.
DIF 6	Lack of how to handle the interoperability of various medical products.
DIF 7	Continuous integration is complex.
DIF 8	Lack of integration with CMMI [Chrissis et al. 2011].
DIF 9	Need for qualified personnel.

Table 3.25 identified the advantages inside the 22 works.

Tabela 3.25. Advantages of using IEC 62304 [Marques et al. 2021]

Advantage	Description
ADV 1	Present rigorous criteria equivalent to the norms of other critical domains.
ADV 2	Do not prescribe a specific lifecycle, only its processes.
ADV 3	Facilitate competitiveness among companies.
ADV 4	Determine rigor according to safety impact.
ADV 5	Control of software planning, programming, testing and documentation.
ADV 6	Present a process for handling software updates.
ADV 7	Emphasize the importance of requirements management and traceability.

3.14. Final Considerations

This chapter presented the fundamentals of IEC 62304 with an Agile Software Development Model. In 2021, IEC 62304 completed 15 years. Thus, the authors believe that there are already works that report experiences, analyzes and difficulties in its use. These results can be interesting to direct further research and definitions of methods, models, guides or other materials to comply with IEC 62304.

The main contributions of this chapter are:

1. The summary of IEC 62304 processes, activities, and tasks, as presented in Sections 3.5, 3.6, 3.7, 3.8, and 3.9;
2. The illustration of Figures 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, and 14 that contributes to a visual understanding of the Software Development Process;
3. The Agile Software Development Model providing an adaptation of Scrum, focusing on IEC 62304 compliance; and
4. The summary of the Systematic Literature Mapping performed.

We believe that our chapter helps the difficulty identified during SLM and presented in Table 3.24. Furthermore, by creating a lecture involving IEC 62304, we are helping to solve DIF 2 *High initial effort with a learning curve to overcome* and DIF 9 *Need for qualified personnel*, because this chapter allows readers to better understand with a qualification in the IEC 62304.

We also believe that our Agile Software Development Model is helpful to solve some difficulties identified during SLM, as presented in Table 3.24. We are helping to solve DIF 3 *There is a lack of recommendations of methods and techniques* and DIF 5 *Difficult to use in small and medium-sized companies*. The usage of an adaptation of Scrum, which focuses on small and medium teams (4 to 9 participants), we adapted the compliance using the Agile Software Development Model presented in Section 3.12, facilitating the competitiveness among companies. We identified that our Agile Software Development Model reaffirms the ADV 2 *Do not prescribe a specific lifecycle, only its processes* and ADV 3 *Facilitate competitiveness among companies*.

Referências

[Ambler 2002] Ambler, S. (2002). *Agile Modeling*. Wiley, Nova Iorque, Estados Unidos.

[Astels 2003] Astels, D. (2003). *Test-driven Development: A Practical Guide*. Pearson Education.

[Beck 2000] Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. Addison-Wesley.

[Beck et al. 2001] Beck, K., Fulano, Beltrano, and Ciclano (2001). Manifesto for agile software development.

[Bianco 2011] Bianco, C. (2011). Integrating a risk-based approach and iso 62304 into a quality system for medical devices. In *Nineteenth Safety-Critical Systems Symposium*.

[Caffery et al. 2010] Caffery, F. M., Burton, J., and Richardson, I. (2010). Risk management capability model for the development of medical device software. *Software Quality Journal*, 18(1):81–107.

[Chrissis et al. 2011] Chrissis, M. B., Konrad, M., and Shrum, S. (2011). *CMMI for Development: Guidelines for Process Integration and Product Improvement*. Software Engineering Institute.

[Cockburn 2004] Cockburn, A. (2004). *Crystal Clear: a Human-powered Methodology for Small Teams*. Addison-Wesley.

[Cruciani and Vicario 2011] Cruciani, F. and Vicario, E. (2011). Reducing complexity of data flow testing in the verification of a iec-62304 flexible workflow system. In *30th International Conference (SAFECOMP)*.

[Davis 2013] Davis, B. (2013). *Agile Practices for Waterfall Projects*. J.ROSS.

[Highsmith 2000] Highsmith, J. A. (2000). *Adaptive Software Development: a Collaborative Approach to Managing Complex Systems*. Dorset House Publishing.

[Huhn and Zechner 2010] Huhn, M. and Zechner, A. (2010). Arguing for software quality in an iec 62304 compliant development process. In *4th International Symposium on Leveraging Applications*.

[Höss et al. 2014] Höss, A., Lampe, C., Panse, R., Ackermann, B., Naumann, J., and Jäkel, O. (2014). First experiences with the implementation of the european standard en 62304 on medical device software for the quality assurance of a radiotherapy unit. *Radiat Oncol*, 9(79):1–10.

[IEC 2006] IEC (2006). Iec 62304:2006 medical device software - software life-cycle processes – amendment 1. Technical report, International Electrotechnical Commission.

[IEC 2010a] IEC (2010a). Iec 61010-1:2010 safety requirements for electrical equipment for measurement, control, and laboratory use - part 1: General requirementssafety requirements for electrical equipment for measurement, control, and laboratory use - part 1: General requirements. Technical report, International Electrotechnical Commission.

[IEC 2010b] IEC (2010b). Iec61508-3:2010 functional safety of electrical/electronic/programmable electronic safety related sysyste - software requirements.

[IEC 2015] IEC (2015). Iec 62304:2006/amd 1:2015 medical device software - software life-cycle processes – amendment 1. Technical report, International Electrotechnical Commission.

[IEC 2016] IEC (2016). Iec 82304-1:2016 health software - part 1: General requirements for product safety. Technical report, International Electrotechnical Commission.

[IEC 2020] IEC (2020). Iec 60601-1-12:2014/amd 1:2020 medical electrical equipment part 1-12: General requirements for basic safety and essential performance — collateral standard: Requirements for medical electrical equipment and medical electrical systems intended for use in the emergency medical services environment. Technical report, International Electrotechnical Commission.

[ISO 2016a] ISO (2016a). Iso 13485:2016 medical devices — quality management systems — requirements for regulatory purposes. Technical report, International Standardization.

[ISO 2016b] ISO (2016b). Iso tr 29110:2016 systems and software engineering — lifecycle profiles for very small entities (vses). Technical report, International Standardization Organization.

[ISO 2017] ISO (2017). Iso/iec/ieee 12207:2017 systems and software engineering — software life cycle processes. Technical report, Internation Standardization Organization.

[ISO 2018] ISO (2018). Iso/iec/ieee 90003:2018 software engineering — guidelines for the application of iso 9001:2008 to computer software. Technical report, International Standardization.

[ISO 2019] ISO (2019). Iso 14971:2019 medical devices — application of risk management to medical devices. Technical report, International Standardization.

[Jordan 2006] Jordan, P. (2006). Standard iec 62304 - medical device software - software lifecycle processes. In *IET Seminar on Software for Medical devices*.

[Kasisopha and Meananeatra 2019] Kasisopha, N. and Meananeatra, P. (2019). Applying iso/iec 29110 to iso/iec 62304 for medical device software sme. In *2nd International Conference on Computing and Big Data*.

[Larson et al. 2012] Larson, B., Hatcliff, J., Procter, S., and Chalin, P. (2012). Requirements specification for apps in medical application platforms. In *4th International Workshop on Software Engineering in Health Care (SEHC)*.

[Laukkarinen et al. 2017] Laukkarinen, T., Kuusinen, K., and Mikkonen, T. (2017). Devops in regulated software development: Case medical devices. In *2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track (ICSE-NIER)*.

[Magnuson 2012] Magnuson, A. (2012). Iec/iso 62304 regulations for the development of medical software devices. Master's thesis, Chalmers University of Technology.

[Marques 2019] Marques, J. (2019). Uma análise das características de especificação de requisitos de software em normas de ambientes regulados. In *22º Workshop de Engenharia de Requisitos (WER 2019)*.

[Marques and Cunha 2019] Marques, J. and Cunha, A. (2019). Ares: An agile requirements specification process for regulated environments. *International Journal of Software Engineering and Knowledge Engineering (IJSEKE)*, 29(10):1403–1438.

[Marques et al. 2021] Marques, J., Yelisetty, S., and Barros, L. (2021). Um mapeamento sistemático da literatura no uso da iec 62304. *Journal of Health Informatics*.

[Mauer and Marin 2017] Mauer, T. and Marin, H. (2017). Instrumento de avaliação de implantação de sistemas de informação em saúde. *Journal of Health Informatics*, 9(4):111–118.

[Mchugh et al. 2012] Mchugh, M., Caffery, F. M., and Casey, V. (2012). Software process improvement to assist medical device software development organizations to comply with the amendments to the medical device directive. *IET Software*, 6(5):431–437.

[Munch et al. 2012] Munch, J., Armbrunt, O., Kowalczyk, M., and Soto, M. (2012). *Software Process Definition and Management*. Springer-Verlag, Berlim, Germany.

[Palmer and Felsing 2002] Palmer, S. R. and Felsing, J. M. (2002). *A Practical Guide to Feature Driven Development*. Pearson Educational.

[Peterson et al. 2015] Peterson, K., Vakkalanka, S., and Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1–18.

[Pressman and Maxim 2015] Pressman, R. and Maxim, B. (2015). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education.

[Regan et al. 2013] Regan, G., Caffery, F. M., Daid, K. M., and D. Flood, D. (2013). Medical device standards' requirements for traceability during the software development lifecycle and implementation of a traceability assessment model. *Computer Standards Interfaces*, 36(1):3–9.

[Rust et al. 2016] Rust, P., Flood, D., and McCaffery, F. (2016). Creation of an iec 62304 compliant software development plan. *Journal of Software Evolution and Process*, 28(11):1.1–1.10.

[Schwaber and Beedle 2001] Schwaber, K. and Beedle, M. (2001). *Agile Software Development with SCRUM*. Prentice-Hall.

[Sommerville 2015] Sommerville, I. (2015). *Software Engineering*. Pearson.

[Stapleton 1997] Stapleton, J. (1997). *DSDM - Dynamic Systems Development Method*. Addison-Wesley.

[Stober and Hansmann 2010] Stober, T. and Hansmann, U. (2010). *Agile Software Development - Best Practices for Large Software Projects*. Springer.

[Sutherland 2010] Sutherland, J. (2010). *SCRUM Handbook,*. Scrum Training Institute Press.

[Tsui et al. 2015] Tsui, F., Karam, O., and Bernal, B. (2015). *Essentials of Software Engineering*. Jones Bartlett Learning.

[Varri and de la Cruz 2019] Varri, A. and de la Cruz, P. K.-Z. R. (2019). Software life cycle standard for health software. *Stud Health Technol Inform*, 264:868–872.

[Vuori 2011] Vuori, M. (2011). Agile development of safety-critical software. Technical report, Tampere University of Technology.

[Wasson 2015] Wasson, C. (2015). *System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices*. Wiley Series in Systems Engineering and Management.

[Wong and Callaghan 2012] Wong, K. and Callaghan, C. (2012). Managing requirements baselines for medical device software development. In *2012 IEEE International Systems Conference (SysCon)*.

