UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Effiziente Konvertierung von radiologischen DICOM – Daten in Standardformate

Studienarbeit

Im Studiengang Computervisualistik

vorgelegt von

Rouven Asmus

WS 2005/06

Betreuer: Dipl.-Inform. Matthias Biedermann
(Institut für Computervisualistik, AG Computergraphik)

Koblenz, im Juni 2006

Erklärung

ja nein

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.

Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

Ort, Datum Unterschrift

Inhalt

1. Einleitung - DICOM - Digital Imaging and Communication in Medicine

Mit dem Einzug der Informationstechnologie in den Bereich der Medizin wurde es notwendig einen Standard für ein Datenformat zu entwickeln, mit dem die Interoperabilität zwischen Systemen, die medizinische Bilder erstellen und verwalten, sichergestellt werden konnte. So Entwickelte 1983 die National Electrical Manufacturers Association (NEMA) in Zusammenarbeit mit dem American College of Radiologe (ACR) den ACR NEMA Standard, der dann zu dem heute verwendeten DICOM Standard weiterentwickelt wurde.
Ein DICOM Datensatz dient als Container für verschiedenste Informationen. So enthält er neben Bildinformationen auch verschiedenste Metadaten wie z.B. den Patientenname, den Namen des behandelnden Arztes oder aber auch technische Informationen wie der eingesetzte Gerätetyp. Als offener Standard ermöglicht DICOM eine plattformunabhängige Kommunikation zwischen medizinischen Geräten verschiedenster Hersteller, so dass der Nutzer in der Wahl seines Arbeitsgerätes nicht eingeschränkt wird.
DICOM kann auch dafür eingesetzt werden Arbeitsabläufe in der Radiologie zu optimieren, da es ohne den Einsatz von Photopapier bzw. Film auskommt und so eine Platz sparende, weniger fehleranfällige Alternative für die Archivierung über lange Zeiträume darstellt. So bildet der Standard die Grundlage für das heute in Kliniken und Praxen eingesetzte Bildarchivierungssystem PACS (Picture Archiving and Communication System).
In der vorliegenden Arbeit soll nun untersucht werden, in wie weit es möglich und sinnvoll ist aus einem DICOM Datensatz die vorahndenden Informationen zu extrahieren und sie in gängige Standardformate zu konvertieren.
Da gerade die Konvertierung der Single- und Multiframe – Images, also Standbilder bzw. Echtzeitfilmaufnahmen des Patienten, einen Grossteil der Arbeit ausmachen, sollen zu Beginn einige der am häufigsten verwendeten Bildgebenden Verfahren in der Medizin kurz vorgestellt und erläutert werden. Bei der anschließenden Betrachtung des Standards selbst, möchte ich dann besonders dem 5. (Part 5: Data Structures and Encoding) und 10. (Part 10: Media Storage and File Format for Media Interchange) Teil des insgesamt 18 Teile umfassenden Standards (Teil 9 und 13 sind mittlerweile nicht mehr im Standard integriert) näher erläutern.
Da JPEG und MPEG wohl zu den bekanntesten und am weitesten verbreiteten Formaten für die Standbild- bzw. Bewegtbildkompression gehören und eine Konvertierung in eben genau diese Formate angestrebt wird, werden im weiteren Verlauf beide Kompressionsverfahren und ihre jeweilige Unterstützung im DICOM Standard vorgestellt.
Der sechste Abschnitt der Arbeit befasst sich schließlich mit DiConv, dem im Verlauf dieser Arbeit entwickelt System, welches genannte Extrahierung und Konvertierung durchführen soll, bevor abschließend die Ergebnisse der Bemühungen in der Schlussbetrachtung präsentiert werden.

2. Bildgebende Verfahren in der Medizin

Seit der Entdeckung der Röntgenstrahlen durch Wilhelm Conrad Röntgen im November 1895, für die er sechs Jahre später mit dem Nobelpreis ausgezeichnet werden sollte, haben sich Bildgebende Verfahren in der Medizin zu einem wichtigen Werkzeug für die Diagnose entwickelt. Besonders in den letzten zwanzig Jahren hat ihr Einsatz in den verschiedensten Bereichen der Medizin stark zugenommen. Dies liegt nicht zuletzt an den vielfältigen neuen Entwicklungen im Bereich der medizinischen Bildaufnahme und Weiterverarbeitung, durch die vollkommen neue Einblicke in das Innere des Menschlichen Körpers möglich sind.
Beim bekannten, konventionellen Röntgen wird der zu untersuchende Körperteil zwischen die Röntgenquelle und einem Detektor, der die Strahlungsabschwächung misst, positioniert. Durch unterschiedliche Abschwächungsgrade von Knochen- und Weichgewebe lassen sich auf diese Weise Aufnahmen der knöchernen Anatomie erstellen. Bestanden diese Detektoren früher ausschließlich aus einem speziellem Photopapier, werden heute vermehrt digitale Akquisitionssysteme, eingesetzt die stattdessen mit CCD - Chips ausgerüstet sind.
Die heutigen Möglichkeiten gehen jedoch weit über eine simple Digitalisierung der Bilder hinaus.
Neben neuen Aufnahmetechniken, welche mit Hilfe sehr starker Magnetfelder oder durch den Einsatz radioaktiver Kontrastmittel eine Untersuchung von Weichgewebe und dessen funktionalen Eigenschaften ermöglichen, werden in den Krankenhäusern 3D - Schichtbilder und Echtzeitaufnahmen des schlagenden Herzens eines Patienten erstellt.
Auch die Endoskopie und die Untersuchung per Ultraschall sind moderne Untersuchungsmethoden die den Medizinern wichtige Informationen für eine erfolgreiche Behandlung liefern.
Da das Speichern und Verwalten medizinischer Bilder eine Hauptintention der Entwicklung des DICOM Standards darstellt, sollen nun im Folgenden kurz einige wichtige Bildgebende Verfahren vorgestellt werden.

2.1 Computertomographie

Die Computertomographie (CT) setzt Röntgenstrahlung zur Bildgebung ein. Anders als beim konventionellen Röntgen wird die Abschwächung (Attenuation) der Strahlung von mehreren Detektoren gleichzeitig gemessen. Durch das Drehen der Apparatur und durch das Anfertigen vieler Bilder können aus den daraus gewonnen Daten Volumendatensätze berechnet werden, aus denen sich wiederum beliebige Schnittbilder und 3D Ansichten rekonstruieren lassen.
Auf diese Weise lassen sich sehr aussagekräftige Befunde erreichen, die bei der Entscheidung über die anschließende Behandlung hilfreich sind.
Da die Untersuchung recht viel zeit in Anspruch nimmt wird der Patient einer relativ hohen Strahlungsbelastung ausgesetzt. Fraktale, die durch die Atembewegungen des Patienten über den langen Zeitraum entstehen können, sind ebenfalls ein Problem bei der Computertomografie. Daher werden heute Systeme verwendet, die möglichst viele Aufnahmen zeitgleich erstellen können um so die Untersuchungszeit zu minimieren. Ein Vorteil des CT sind seine vergleichsweise geringen Kosten.

Abbildung 1: modernes Spiral CT, Quelle: www.wikipedia.de

2.2 Magnetresonanztomographie

Die Magnetresonanztomographie (MRT) wird häufig auch als Kernspintomographie bezeichnet, da dieses Verfahren die so genannte Kernspinresonanz ausnutzt. Man hat herausgefunden, dass Protonen einen Eigendrehimpuls (Spin) besitzen, welcher sich nach magnetischen Feldlinien ausrichtet, wenn er in ein ausreichend starkes Magnetfeld gebracht wird. Passiert dies, beginnt der Kern mit einer Präzessionsbewegung, d.h. die Rotationsachse des Kerns dreht sich in Richtung des angelegten Magnetfeldes. Durch das zusätzliche, impulsartige wirken eines zweiten, hochfrequenten Magnetfeldes (HF – Feld) lässt sich rotierende Quermagnetisierung erzeugen, die über die induzierte Spannung messbar ist. Da die Quermagnetisierung von Ort und Gewebeart abhängig ist können auf diese Weise detaillierte Schnittbilder erstellt werden.
Der Vorteil am MRT ist die gute Differenzierbarkeit verschiedener Gewebetypen und der Wegfall der Strahlungsbelastung, da keine ionisierende Strahlung benötigt wird. Nachteilig sind die hohen Anschaffungs- und Betriebskosten der notwendigen Geräte. Des Weiteren können Patienten mit einem Herzschrittmacher oder metallischen Prothesen nicht auf diese Weise untersucht werden. Auch ist diese Methode anfälliger gegenüber dem Auftreten störender Fraktale.

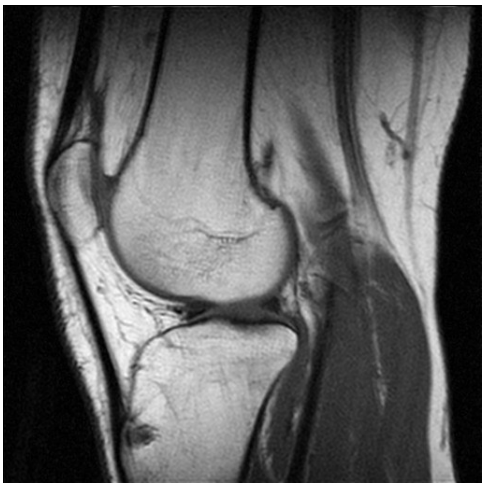

Abbildung 2: MRT eines menschlichen Kniegelenks, Quelle: www.wikipedia.de

Mit einer Variante des MRT, dem fMRT lassen sich mit Hilfe von Kontrastmitteln neben der Morphologie auch Funktionen von Organen untersuchen.

2.3 Positronen – Emissions – Tomografie

Die Positronen – Emissions – Tomographie (PET) ist ein Bildgebendes Verfahren der Nuklearmedizin. Dabei misst man die Verteilung einer radioaktiven Trägersubstanz im Organismus. Zerfällt eine radioaktive Substanz werden in einem Winkel von 180° γ – Quanten ausgestrahlt. Erkennen zwei ebenfalls im Winkel von 180° angebrachte Detektoren eine Vielzahl von beinahe gleichzeitigen Einschlägen (innerhalb von ca. 10 Nanosekunden), kann auf diese Weise wieder ein Schnittbild sowie eine 3D Rekonstruktion berechnet werden. Das PET ermöglicht so präzise Rückschlüsse auf die Funktion von Organen zu ziehen, wobei die Auflösung der Aufnahmen eher gering ist, was jedoch für die Art der Untersuchungen nicht weiter ins Gewicht fällt.

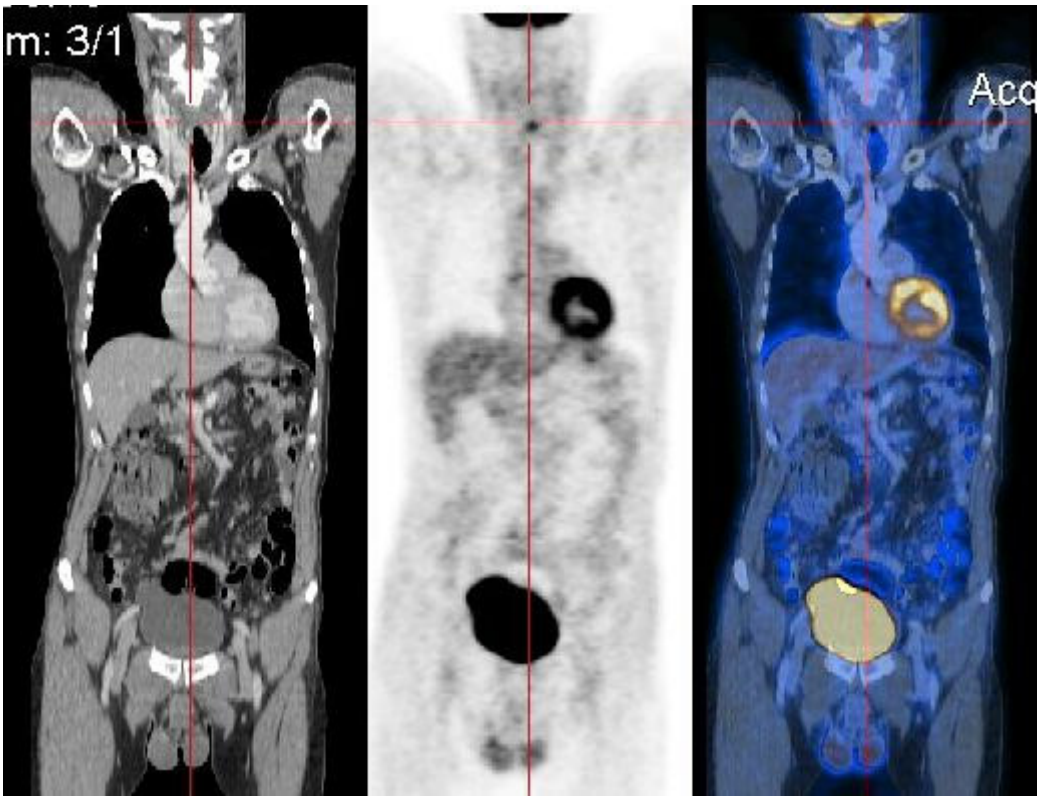

Abbildung 3: Links: CT Aufnahme des Torso, mitte: PET Aufnahme, links: kombiniertes Bild

3. DICOM

Der folgende Abschnitt der Arbeit soll nun einen näheren Blick auf den DICOM Standard ermöglichen. Der Standard selbst besteht aus achtzehn einzelnen Dokumenten die jeweils andere Teile des Standards spezifizieren, wobei der 9. Und 13. Teil mittlerweile nicht mehr dazu gehören. Ihre Funktionen wurden inzwischen durch Zusätze in anderen Teilen realisiert.
Da es hier nicht möglich ist den gesamten Inhalt wiederzugeben, werde ich mich auf die für die Bearbeitung der in der Aufgabenstellung genannten Aufgaben und Fragestellungen benötigten Teile beschränken. Dies sind zunächst die Teile 3 und 4. In ihnen werden die Definitionen für die *Information Objects* - Abstraktionen realer Informationsträger (z.B. CT Bild) - sowie die Spezifikationen der *Service Classes* - Beschreibungen unterstützter Services von DICOM - vorgenommen, die für das weitere Verständnis notwendig sind.
Des Weiteren werden noch der 5. (*Data Structure and Encoding)* und 10. Teil (*Media Storage and File Format For Data Interchange*) näher betrachtet.
Dem ganzen voraus geht jedoch erst einmal eine kurze Zusammenfassung der Entwicklungsgeschichte und der Funktionalität von DICOM.

3.1 Entwicklung ACR NEMA – DICOM

Mit Erfindung der Computertomographie in den 70er Jahren und dem verstärkten Einsatz von Computern in den Krankenhäusern stieg die Notwendigkeit eines Standards um medizinische Bilder und mit ihnen verbundene Informationen zwischen den Geräten verschiedener Hersteller austauschen zu können. So formten 1983 das American College of Radiologe (ACR) und die National Electrical Manufacturers Association (NEMA) ein Komitee um einen Standard zu entwickeln, der

- die Kommunikation digitaler Bildinformationen unabhängig vom Hersteller ermöglicht
- die Entwicklung und Erweiterung des Picture Archiving and Communication Systems (PACS) unterstützt, welches auch mit anderen medizinischen Informationssystemen interagieren kann
- die Aufstellung diagnostischer Informationsdatenbanken erlaubt, welche von verschiedensten, geographisch getrennten Einrichtungen genutzt werden können.

1985 wurde der ACR - NEMA Standard mit der Versionsnummer 1.0 veröffentlicht. Es folgten zwei Revisionen des Standards im Oktober 1986 und im Januar 1988. Im selben Jahr wurde die Version 2.0 veröffentlicht, welche die Version 1.0 enthält und den Standard um einige Funktionen erweiterte. So führte er eine neue Hierarchie zur Bildidentifizierung ein und vergrößerten die Anzahl der Datenelemente die ein Bild beschreiben.
Diese Veröffentlichungen spezifizierten bereits ein Hardwareinterface, minimale Softwarekommandos und eine angemessene Menge an Datenformaten.

Der derzeit vorliegende DICOM Standard (Stand 2004) liefert einige signifikante Erweiterungen zum vorherigen ACR - NEMA Standard:

- Er ist so ausgerichtet, dass er in einem Netzwerk eingesetzt werden kann. Mit ACR NEMA war lediglich eine point-to-point Kommunikation möglich. Um

dies zu realisieren verwendet DICOM das Standard Netzwerk Protokoll TCP/IP.

- Er ist auch offline einsetzbar. Der vorherige Standard spezifizierte kein Dateiformat oder die Wahl eines physischen Mediums bzw. eines Dateisystems. DICOM unterstützt Standard Medien wie CD-R und MOD und logische Dateisysteme wie ISO 9660 und FAT16.

- Wo ACR - NEMA nur den Datentransfer allein behandelte, spezifiziert DICOM wie Geräte standardkonform entwickelt werden können und realisiert durch das Konzept der Serviceklassen die Semantik von Kommandos und Daten.

- Es wird festgelegt, wie Entwickler eine Konformitätserklärung strukturieren müssen um verschiedene Optionen in ihren Entwicklungen zu nutzen. So entstehen verschiedene Stufen der Konformität, wo vorher bei ACR - NEMA lediglich ein Minimum aufgestellt wurde.

- Um die Flexibilität des Standards gegenüber der rasanten Entwicklung des Anwendungsgebietes zu gewährleisten wurde er als *multi-part document* strukturiert, so dass zusätzliche Features schnell integriert werden können.

- Er führt explizite Objekte für Bilder und Grafiken ein.

- Er spezifiziert eine Technik zur eindeutigen Identifizierung eines Objektes. Dies erleichtert eindeutige Definitionen von Beziehungen zwischen Objekten innerhalb des Netzwerks.

Die Änderungen wurden vorgenommen, um den Standard den neuen Gegebenheiten anzupassen. Sie stellen im Großen und Ganzen die Umsetzung der vor der Entwicklung gestellten Ziele dar, welche im ersten Teil des Standards formuliert werden. Aus diesem Grund werden diese an dieser Stelle nicht noch einmal aufgeführt.

Auch heute noch wird der Standard durch das DICOM Komitee weiterentwickelt. Änderungen werden auf Anraten von Mitgliedsorganisationen in Betracht gezogen, die wiederum ihre Ideen von Nutzern des Standards erhalten. Eine Erweiterung ist in jedem Fall nur dann möglich, wenn eine effektive Kompatibilität zu vorherigen Versionen gewährleistet ist. Abbildung 4 zeigt das generelle Kommunikationsmodell des Standards.

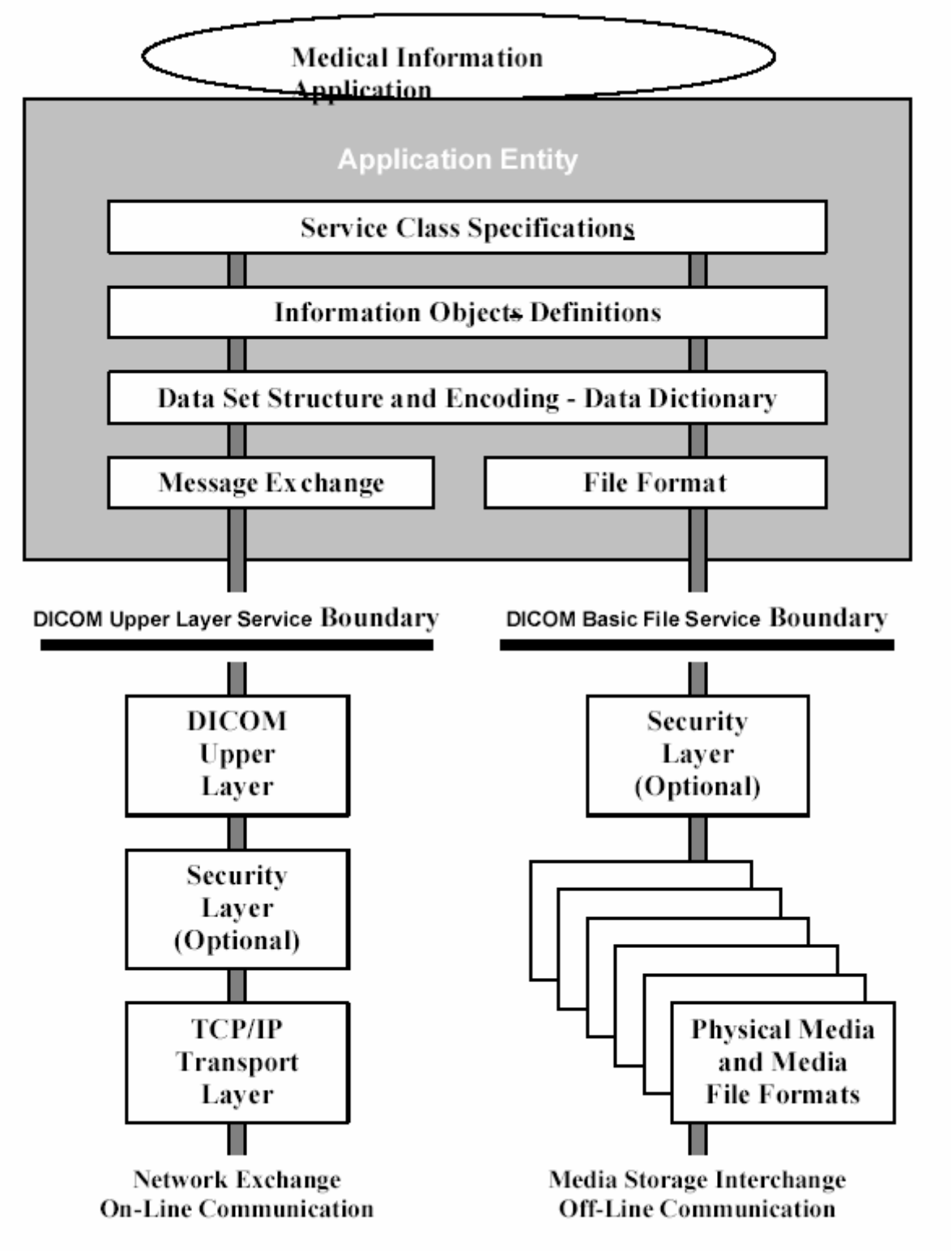

Abbildung 4: General DICOM Communication Model, aus (1)

3.2 Begriffe und Definitionen des DICOM Standards

Der dritte Teil des DICOM Standard spezifiziert die so genannten *Information Object Definitions* (IODs), welche abstrakte Definitionen realer Objekte darstellen. Eingebettet in ein Model zur Strukturierung und Organisation von Informationen medizinischer Bilder *(Abb. 5: Major Structures of DICOM Information Model),* repräsentieren die IODs Klassen realer Objekte mit denselben Eigenschaften.

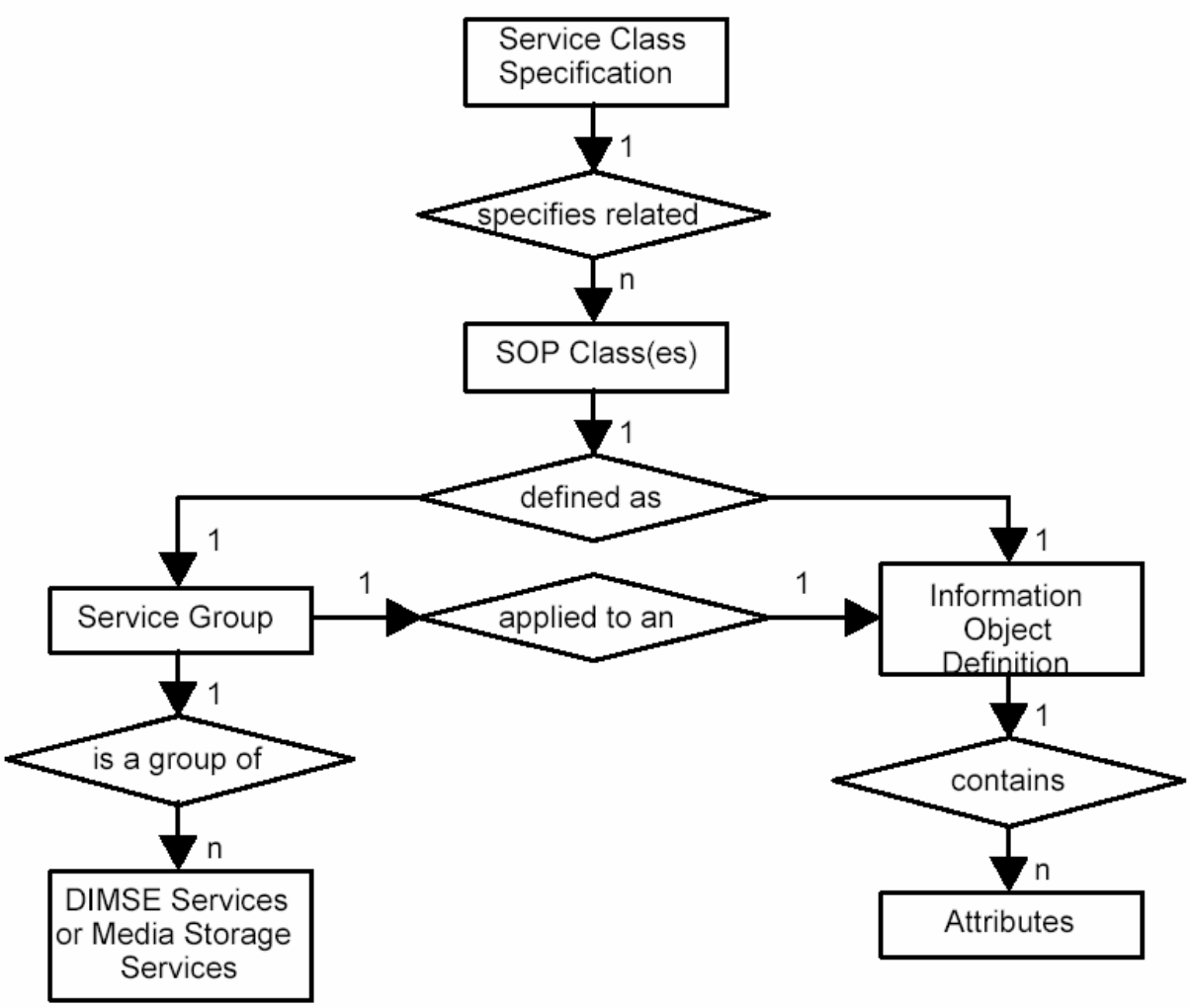

Abbildung 5: Major Structures of DICOM Information Model, aus (2)

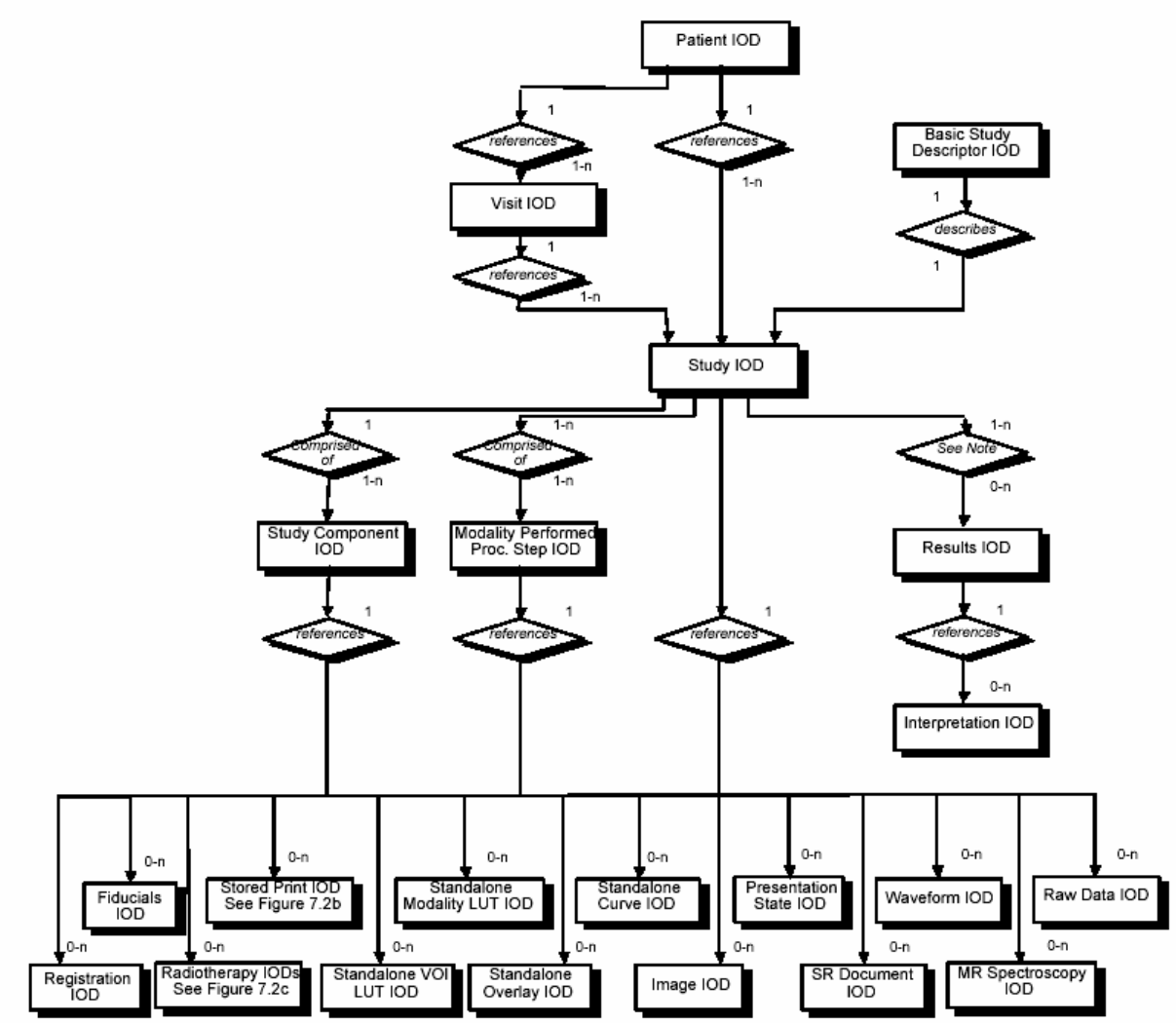

Abbildung 6: DICOM Model of the Real World, aus (2)

IODs stehen für Einheiten die in dem *DICOM Model of the Real - World (Abb. 6)* zusammengefasst werden, welches die relevanten, realen Objekte beinhaltet sowie deren Beziehungen untereinander. Dieses Model wurde für spezifische Aufgaben erweitert, außerdem kann von ihm ein detaillierteres und anwendungsspezifischeres DICOM Information Model abgeleitet werden, wobei nicht immer eine eins-zu-eins Korrespondenz zwischen realen Objekten und IODs erreicht wird.
Es gibt zwei verschiedene Arten von IODs. Die *composite IOD* repräsentiert Teile mehrerer Einheiten innerhalb des *DICOM Model of tue Real World.* Sie besitzt Attribute, die nicht im realen Objekt selbst vorhanden sind, sondern erst in dessen Abstraktion im Model. Bei der Kommunikation einer solchen Instanz wird der gesamte Kontext zwischen den Anwendungen ausgetauscht.
Die *normalized IOD* steht hauptsächlich für eine einzelne Einheit innerhalb des *DICOM Model of the Real - World*. Bei der Kommunikation mit dieser IOD wird der benötigte Kontext über zugehörige Pointer bereitgestellt, jedoch nicht direkt ausgetauscht.
Die *Service - Object Pair (SOP) Class* ist definiert durch die Vereinigung einer IOD und einer *DIMSE Service Group.* Bei letzterem handelt es sich um eine *DICOM Message Service Element Service Group* bzw. einer Spezifikation einer oder mehrerer Operationen welche im 7. Teil des Standards *(Message Exchange)* definiert werden.
Die SOP Klassendefinition enthält alle Regeln die eine Nutzung eines DIMSE Services oder eines IOD Attributs verbieten würden. So wird zwischen den *Application Entities (AE)* - also

den Anwendungen/Geräten die miteinander kommunizieren wollen - eine Menge von Vereinbarungen getroffen die ihre Interaktion unterstützen. Auch bei den SOP Klassen wird wieder zwischen *normalized* und *composite* unterschieden, je nachdem welcher Art die IODs sind, die zusammen mit den DIMSE Services die SOP bilden.
Service Class Specifications definieren eine Gruppe von einer oder mehreren SOP Klassen bezogen auf eine bestimmte Funktion. Anwendungen können ihre SOP Klassen auch anpassen - entweder als *Service Class User* (SCU) oder *Service Class Provider* (SCP).
Nachfolgend werden die genaueren Spezifikationen von IODs und Serviceklassen, welche jeweils in den Anhängen der zugehörigen Dokumente des Standards durchgeführt werden, näher beschrieben.

3.2.1 Service Class Specification

Die Spezifikation der Serviceklassen wird in den Anhängen A bis S im zugehörigen Teil des DICOM Standards vorgenommen. In ihr werden die Klassen im Hinblick auf Arbeitsumfeld, Funktionen, Attribute und Eigenschaften beschrieben. Auch benötigte SOP Klassen, IODs und *DIMSE Service Groups* werden benannt.
Die angebotenen Services decken wohl so ziemlich alle Anwendungsfälle ab, welche in einem heutigen Klinikbetrieb auftreten. Sie reichen von *Verification*- über *Storage*- bis hin zu einer *Media Creation Management Service Class.*
Folgend sei ein knapper Überblick über eine Serviceklasse gegeben, die vor allem das Prinzip von SCU/SCP einmal exemplarisch durchgehen soll. Für eine umfassende Besprechung einer solchen Spezifikation ist an dieser Stelle leider kein Platz.

3.2.1.1 Query/Retrieve Service Class

Diese Service Klasse stellt grundlegende Suchmechanismen bereit. Sie arbeitet nur mit einer kleinen Menge von Schlüsselattributen und soll nicht SQL ähnliche Datenbank Suchmechanismen bereitstellen.
Zusätzlich können dafür jedoch mittels retrieve/transfer - Eigenschaft Anfragen von einer AE zur anderen übertragen und angefordert werden.
Zu diesem Zweck befindet sich eine AE in der SCU - Rolle, die andere in der SCP - Rolle und es werden die DIMSE-C (C für composite) Services C-FIND, C-MOVE und C-GET verwendet.
Der Ablauf einer solchen Anfrage soll am Beispiel von C-Find verdeutlicht werden:

- Der SCU fordert beim SCP an einen Vergleich aller im *Identifier* spezifizierten Schlüssel mit den ihm eigenen Informationen durchzuführen und zwar auf der in der Anfrage selbst festgelegten Ebene (z.B. Patient, Study etc). *Identifier* bezeichnet hier die in Teil 7 des Standards festgelegten *Identifier Service Parameter* der DIMSE-C Services.

- Der SCP generiert eine C-FIND - Antwort für jede Übereinstimmung bezüglich des erhaltenen Schlüssels mit allen bekannten Attributen aus der Anfrage. Diese Antworten gehen zunächst über in einen Wartezustand, welcher anzeigt, dass der Prozess noch nicht komplett abgearbeitet ist.

- Nach Beendigung des Suchvorganges wird eine Antwort zurückgeschickt. Eine verweigerte oder fehlgeschlagene Antwort zeigt an, dass der SCP die Anfrage nicht bearbeiten kann.

- Der SCU kann den Service jederzeit mit C-FIND-CANCEL abbrechen. In diesem Fall unterbricht der SCP den Vorgang und sendet eine entsprechende Nachricht zurück.

3.2.2 IOD Spezifikation

Für jede composite IOD (cIOD) werden spezifiziert:

- IOD Beschreibung
- IOD *Entity - Relationship Model* (E-R)
- IOD Modultabelle und
- optional, Makro Tabelle für so genannte *Multi-Frame Functional Groups*

bei normative IODs, welche im Anhang B definiert werden wird keine E-R Beschreibung angewandt.

Die Beschreibung der IOD besteht aus dem realen Objekt auf welches sie sich bezieht sowie Informationen über die Umgebung des repräsentierten Objekts. Das E-R verdeutlicht die Beziehungen zwischen den Komponenten oder auch *Information Entities* (IE) der jeweiligen IOD. Das E-R stellt also auch ein *Information Model* spezifischen IOD dar. Es stellt den kompletten Kontext bereit, der für die Einordnung der Informationen bei der Übertragung von cIODs *(contextIOD)* benötigt wird.
Für einzelne cIOD Instanzen können Untermengen des Models aufgestellt werden, um den Kontext genauer beschreiben zu können.
Die Modultabelle und die optionalen Makros definieren tabellarisch die Module, die die IOD einschließen. Für jedes Modul ist zu spezifizieren:

- Name des Moduls bzw. der funktionellen Gruppe

- eine Referenz auf den Anhang C des 3. Teil des DICOM Standards, in dem das Modul bzw. die funktionelle Gruppe definiert wird

- Die Nutzungsart des Moduls bzw. der funktionellen Gruppe durch die Unterscheidung in:
  i. Mandatory (M) - zwingende Unterstützung per Moduldefinition
  ii. Conditional (C) - Unterstützung wenn bestimmte Bedingungen erfüllt
  iii. User Option (U) - optional, wird es unterstützt gilt dies auch für die zugehörigen Attribute

Eine informative Übersicht der Module findet sich dritten Teil des DICOM Standards unter *Annex A1.4*, ist aber zu umfangreich, um hier komplett aufgeführt zu werden.
Ebenfalls in besagten Anhang werden eine Vielzahl von IODs definiert. Diese reichen von *Computed Radiography Image IOD* für radiographisch Aufnahmen über *The Multi-Frame True Color Secondary Capture (SC) IOD* für nicht - DICOM Formate die in eine DICOM - abhängige Modalität konvertiert wurden (z.B. gescannte Farbdokumente, synthetische Bilder mit *true color* Unterstützung) und *General Electrocardiogram (ECG) IOD* für digitalisierte elektrische Signale bis hin zu *Hemodynamic IOD* für digitale Signale des zirkulären Systems des Patienten. Es existieren auch IODs für die verschiedenen medizinischen Bildmodalitäten wie z.B. CT, MRT, PET und Ultraschall.
Auch die Moduldefinitionen selbst sind äußerst umfang- und variantenreich, so dass auch nur eine bedingt vollständige Aufzählung hier unmöglich ist. Neben einer natürlich sprachlichen

Beschreibung des Moduls bildet eine Tabelle den eigentlichen Kern der Definition. Beschreibungen der Modulattribute sind optional. Die Tabelle informiert über:

- Attribute Name
- Data Element Tag
- Type Designated
- Attribute Definition

Auf die Bedeutungen des Tags und des Typs soll zu einem späteren Zeitpunkt näher eingegangen werden, nämlich bei der Behandlung des 5. Teils des Standards *Data Structures and Encoding*. Exemplarisch zeigt Tabelle 1 (*General Image Module*) eine solche Tabelle, ohne jedoch den Anspruch auf Vollständigkeit zu erheben.

Attribute Name	**Tag**	**Type**	**Attribute Description**
Instance Number	(0020,0013)	2	A number that identifies this image.
Patient Orientation	(0020,0020)	2C	Patient direction of the rows and columns of the image. Required if image does not require Image Orientation (Patient) (0020,0037) and Image Position (Patient) (0020,0032). See C.7.6.1.1.1 for further explanation.
Content Date	(0008,0023)	2C	The date the image pixel data creation started. Required if image is part of a series in which the images are temporally related.
Content Time	(0008,0033)	2C	The time the image pixel data creation started. Required if image is part of a series in which the images are temporally related.
Image Type	(0008,0008)	3	Image identification characteristics. See C.7.6.1.1.2 for Defined Terms and further explanation.
Acquisition Number	(0020,0012)	3	A number identifying the single continuous gathering of data over a period of time that resulted in this image.
Acquisition Date	(0008,0022)	3	The date the acquisition of data that resulted in this image started
Acquisition Time	(0008,0032)	3	The time the acquisition of data that resulted in this image started
Acquisition Datetime	(0008,002A)	3	The date and time that the acquisition of data that resulted in this image started.
Referenced Image Sequence	(0008,1140)	3	A sequence that references other images significantly related to this image (e.g. post-localizer CT image or Mammographic biopsy or partial view

			images). One or more Items may be included in this sequence.

Tabelle 1: Auszug aus dem „General Image Module“, aus (2)

3.4 Part 5: Data Structures and Encoding

3.4.1 Begriffserklärung und Struktur

Im Folgenden wird die Struktur und die Kodierung des DICOM Datensatzes *(Data Set)* näher betrachtet. Zu diesem Zweck wird näher erläutert:

- die Kodierung von Werten
- die Struktur und Anwendung des Datensatzes
- die Konstruktion und Anwendung von Pixeldaten enthaltenden Datensätzen
- wie Informationen eindeutig Identifiziert werden können

Zu Beginn werden einige Begriffe erläutert und der Zusammenhang der verschiedenen Dokumente des Standards bei der Informationsbeschaffung gezeigt. Im Anschluss daran wird dann eine genauere Betrachtung der Struktur durchgeführt.

Ein Datensatz ist eine Repräsentation eines realen Informationsobjektes, welches durch die IODs abstrahiert wird. Er besteht aus den *Data Elements* (Datenelemente) die wiederum die kodierten Attribute des Objektes beinhalten. Die Datenelemente können eindeutig über das *Data Element Tag* identifiziert werden. Dieses hat die Form (gggg,eeee), wobei gggg für die Gruppennummer und eeee für die Elementnummer steht, welche in hexadezimaler Notation niedergeschrieben werden.
Die *Value Representation* (VR) setzt sich zusammen aus Name, Definition, einem zu Verfügung stehendem Zeichenrepertoire und der nötigen bzw. maximalen Bits, die genutzt werden bzw. genutzt werden können. Neben der VR eines Datenelements wird darüber hinaus die *Value Multiplicity* (VM) angegeben, welche die mögliche Anzahl von Werten die innerhalb des Datenelements kodiert werden können, repräsentiert.
Es werden siebenundzwanzig verschiedene VR in tabellarischer Form spezifiziert, darunter z.B. Code String, Decimal String, Person Name, Short String und Short Text.
Will man also nun Informationen über ein Datenelement erhalten muss man dieses im 6. Teil des Standards ausfindig machen. Dort findet man einen Verweis auf die VR und die VM. Die Bedeutung der VM ist direkt ersichtlich. Sie ist in der Form x - y notiert, wobei x die untere und y die obere Grenze festlegt. Um die VR deuten zu können schlägt man abschließend die genaue Spezifikation im 5. Teil des Standards nach. Diese Vorgehensweise soll an einem Beispiel verdeutlicht werden:

Findet man in einem DICOM Datensatz z.B. das Datenelement - Tag (0010,0020) , so liefert die *Registry of Data Elements* zuerst einmal die Information, dass es sich um die Patienten ID handelt. Des Weiteren erfährt man, dass die VM 1 beträgt, somit ist nur ein Wert gleichzeitig für dieses Element zugelassen. Als VR wird LO angegeben. Eine weitere Recherche bei den VR Spezifikationen liefert letztlich folgende Informationen:

- LO steht für Long String

- Es handelt sich dabei um eine String von Zeichen mit optionalen Leerzeichen vorne oder hinten. Der BACKSLASH "\" (*character code:* 5CH) ist nicht zulässig, da er als Trennzeichen zwischen Werten genutzt wird, sollte das Datenelement multiple Wertzuweisung unterstützen. Als Kontrollzeichen ist nur ESC zugelassen.

- Das zugelassene Zeichenrepertoire stellt das Default Character Repertoire dar - welches dem ISO 8859 Zeichensatz entspricht - sowie eine potentielle Erweiterung, welche jedoch zuvor implementiert werden müsste

- Die maximale Größe beträgt 64 Zeichen, je nach erweitertem Zeichensatz

Auf diese Weise lassen sich mit ein paar Schritten alle Datenelemente nur über die Angabe des eindeutigen Bezeichners - dem Tag - genauer betrachten.

Ein Datenelement setzt sich aus Feldern zusammen. Je nach Struktur des Elements ist die Anzahl der Felder unterschiedlich, mindestens sind es jedoch drei: Data Element Tag, Value Length und Value Field. Das optionale vierte Feld besetzt die VR je nach vereinbarter Transfer Syntax bzw. gewählter Datenelementstruktur (z.B. Data Element Structure with explicit VR).
Abbildung 7 zeigt diesen schematischen Aufbau.

Tag	VR	Value Length	Value Field

Abbildung 7: Aufbau des Data Element Tag, aus (4)

Die Definitionen der einzelnen Felder sind:

Data Element Tag:	paar von 16-bit unsigned Integer, die die Gruppennummer gefolgt von der Elementnummer repräsentieren
Value Representation:	2 Byte String, die die VR des Datenelements enthält
Value Length:	Entweder 16 bzw. 32-bit unsigned Integer (abhängig von der VR) der die Länge des Value Fields enthält oder ein 32-bit Length Field, in welchem eine undefinierte Länge festgesetzt wird. Dies wird für bestimmte Datenelemente z.B. aufgrund der vereinbarten Transfer Syntax benötigt.
Value Field:	Gerade Anzahl Bytes, die die Information des Datenelements beinhalten. Der Typ wird durch die VR angegeben, entweder explizit in dem dafür bestehendem Feld oder im Data Dictionary (Dokument Nr. 6 des Standards). Ist die VM des Elements >1 können mehrere Werte in dem Feld vorhanden sein

Bei der Nutzung einer Struktur mit explizitem VR muss das Element aus vier Feldern bestehen, welche je nach verwendetem VR unterschiedlich aufgebaut sein können. Mit

impliziter VR werden nur drei Felder benötigt - das VR Feld entfällt hier. Verdeutlicht werden die verschiedenen Strukturen durch Tabelle 2 - 4.

Tag		**VR**	**Value Length**	**Value**
Group Number (16-bit unsigned integer)	Element Number (16-bit unsigned integer)	VR (2 byte character string	(16-bit unsigned integer)	Even number of bytes containing the Data Element Value(s) encoded according to the VR and negotiated Transfer Syntax.
2 bytes	2 bytes	2 bytes	2 bytes	Value Length' bytes

Tabelle 2: Variante des Data Elements mit explicit VR, aus (4)

Tag		**VR**		**Value Length**	**Value**
Group Number (16-bit unsigned integer)	Element Number (16-bit unsigned integer)	VR (2 byte character string) of "OB", "OW", "OF", "SQ", "UT" or "UN"	Reserved (2 bytes) set to a value of 0000H	32-bit unsigned integer	Even number of bytes containing the Data Element Value(s) encoded according to the VR and negotiated Transfer Syntax. Delimited with Sequence Delimitation Item if of Undefined Length.
2 bytes	2 bytes	2 bytes	2 bytes	4 bytes	´Value Lengt´bytes if of Explicit Length

Tabelle 3: Variante des Data Elements mit explicit VR OB, OW OF, SQ, UT oder UN, aus (4)

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

Tabelle 4: Variante des Data Elements mit implicit VR, aus (4)

Datenelemente können verschiedenen Typs sein. Der Datenelement - Typ eines Attributs einer IOD oder einer SOP Klasse geben an, ob das Attribut zwingend ist für die Kommunikation zwischen AEs oder optional. Er gibt auch an ob ein Attribut konditional ist, d.h. nur zwingend unter bestimmten Voraussetzungen. Es existieren unterschiedliche Typen:

TYPE 1 REQUIRED DATA ELEMENTS	müssen inkludiert werden und sind zwingend. Das Value Field soll gültige Daten enthalten - welche durch die VR und VM des Elements spezifiziert werden - und nicht leer sein.
TYPE 1C CONDITIONAL DATA ELEMENTS	dürfen nur dann inkludiert werden, wenn bestimmte Bedingungen erfüllt werden. Unterliegen dann den gleichen Bedingungen wie Typ 1 Elemente.
TYPE 2 REQUIRED DATA ELEMENTS	müssen inkludiert werden und sind zwingend. Es ist jedoch möglich keinen Wert anzugeben bzw. ihn auf Null zu setzen, sollte er nicht bekannt sein. Ansonsten ist er anzugeben wie in der VR und VM angegeben.
TYPE 2C CONDITIONAL DATA ELEMENTS	dürfen nur dann inkludiert werden, wenn bestimmte Bedingungen erfüllt werden. Unterliegen dann den gleichen Bedingungen wie Typ 2 Elemente.

Es ist auch möglich, Datenelemente mit einem Value Field welches aus mehreren Items besteht zu definieren. Dazu kann man das Element mit der VR "SQ" (für Sequenz) verwenden. Der Typ gibt hier dann an, wie viele und ob Attribute innerhalb der Sequenz vorhanden sein müssen/dürfen.
Durch die Benutzung von "SQ" können sowohl einfache, sich wiederholende als auch rekursive multi - level Strukturen erstellt werden. Die Items innerhalb des Value Fields werden beginnend mit eins durch nummeriert. Für Sequenzen sind Datenelemente wieder speziell aufgebaut. Sie werden erneut wieder in implizite und explizite VR unterschieden. Nachfolgend zeigt Tabelle 2 ein Beispiel.

Trotz der Vielzahl von verschiedensten verfügbaren Datenelementen kann es durchaus sein, dass spezielle Implementationen die Kommunikation von Informationen verlangen, die nicht durch die Standarddatenelemente abgedeckt werden können. Zu diesem Zweck können *Private Data Elements* verwendet werden. Diese besitzen dieselbe Struktur wie oben angegeben (d.h. Tag - Feld, optionales VR - Feld, Length - Feld und Value - Feld).

3.4.2 Kodierung von Pixel-, Overlay- und Waveform - Daten

Für den Austausch grafischer Bilddaten sollten das Pixeldaten - Element (7FE0, 0010) und das Overlaydaten - Element (60xx, 3000) verwendet werden. Abhängig der vereinbarten Syntax können Pixeldaten komprimiert werden. Die VR von Pixel- bzw. Overlay - Daten ist entweder OB (Other Byte String) oder OW (Other Word String). Eine genauere Spezifikation zeigt Tabelle 5. Der unterschied in der Nutzung besteh lediglich darin, dass OB vom *Byte Ordering* (*Little-* und *Big Endian* sind möglich) nicht affektiert wird.

OB Other Byte String	A string of bytes where the encoding of the contents is specified by the negotiated Transfer Syntax. OB is a VR which is insensitive to Little/Big Endian byte ordering (see Section 7.3). The string of bytes shall be padded with a single trailing NULL byte value (00H) when necessary to achieve even length.	not applicable	see Transfer Syntax definition
OW Other Word String	A string of 16-bit words where the encoding of the contents is specified by the negotiated Transfer Syntax. OW is a VR that requires byte swapping within each word when changing between Little Endian and Big Endian byte ordering (see Section 7.3).	not applicable	see Transfer Syntax definition

Tabelle 5: VR Spezifikation zweier VR, aus (4)

Die Struktur der Pixeldaten wird durch drei Datenelemente beschrieben:

- Bits Allocated (0028,0100)
- Bits Stored (0028,0101)
- High Bit (0028,0102)

Ein Pixel kann aus einem oder mehreren *Pixel Sample Values* bestehen (z.B. bei Farbbilder oder Bildern mit mehreren Ebenen). Die Werte können entweder als Zweierkomplement - Integer oder als binärer unsigned Integer angegeben werden. Die Anzahl von Bits eines Pixelsamples wird über Bits Stored festgelegt. Eine Pixelzelle (*Pixel Cell*) existiert für jeden einzelnen *Pixel Sample Value* und ist eine Art Behältnis, sowohl für diese als auch für zusätzliche Informationen. Die Größe der Zelle wird mit Bits - Allocated angegeben, welche größer bzw. gleich groß ist wie Bits Stored. High Bit gibt die Position des *Pixel Sample Values* an.
Standardmäßig hat das Pixel - Datenelement die VR OW und ist gepackt. Dies kann man sich vorstellen als einen Stream von Bits, beginnend mit dem *least significant bit*
der ersten Zelle und endend mit *most significant bit* der Letzten. So folgt dem *most significant bit* einer Zelle immer das *least significant bit* der darauf folgenden. So lassen sich die Pixeldaten in physikalisch 16-bit Wörter zerlegen. Dies stellt die Anordnung der *default DICOM Transfer Syntax* dar. Andere, *non - default DICOM Transfer Syntaxes* nutzen die explizite VR von OB. Dabei die Anzahl Bits - spezifiziert durch Bits Allocated - kleiner gleich acht. Auch hier werden die Daten gepackt, können jedoch nur in physikalische 8-bit Wörter zerlegt werden. Nachfolgend ein Beispiel zur Verdeutlichung :

Bestimmungen über akzeptierbare Werte von Bits Allocated, Bits Stored und High Bit werden in den jeweiligen IODs vorgenommen. Des Weiteren sollte die Anzahl der Bytes des Value Fields, wie auch bei allen anderen Datenelementen, gerade sein.

Die Kodierung von Overlaydaten hat immer eine Tiefe von einem Bit. Sie können mit den Bits kodiert werden die nicht für die *Pixel Sample Value* verwendet werden.
Angenommen wir hätten Pixeldaten mit Bits Allocated = 16 Bits, Bits Stored = 12 Bits und High Bit = 11. Ein Pixelsample wäre pro 16 Bit Wort kodiert, wobei die 4 *most significant bits* keine Pixeldaten enthalten würden. Diese könnten für die Overlaydaten verwendet werden.

Zu diesem Zweck existieren Datenelemente mit denen ungenutzte Bits den Overlaydaten zugewiesen werden könne:

- Overlay Bits Allocated (60xx,0100)
- Overlay Bit Position (60xx,0102)

Für das obige Beispiel hieße das: die Overlay Bit Position = 15 und Overlay Bits Allocated = 16. Werden ungenutzte Bits zur Kodierung verwendet gilt immer:

Bits Allocated = Overlay Bits Allocated.

Pixeldaten können komprimiert oder im Originalzustand versendet werden. *Value Representation* im unkomprimierter Form ist meist OW, OB wird in Ausnahmen verwendet, wie bei Datenelementstrukturen mit expliziter VR wenn gleichzeitig gilt Bits Allocated <= 8. Die VR bei Komprimierung ist OB. Die Pixeldaten werden in ein oder mehrere Fragmente zerlegt, wobei das Ende des Datenstroms begrenzt wird, um auch Methoden zu ermöglichen bei denen die Länge erst am Ende des Kodierungsprozesses bekannt ist. Sowohl *Single - Frame* als auch *Multi - Frame Images* werden unterstützt.
Durch diese Kapselung der Daten ist auch JPEG Kompression möglich. Dabei wird sowohl verlustbehaftete als auch verlustfreie Kompression unterstützt. Um die Interoperabilität zwischen Implementationen sicherzustellen die konform dem DICOM Standard entwickelt werden wurden Richtlinien definiert die es einzuhalten gilt:

"- Any implementation which conforms to the DICOM Standard and has elected to support any one of the Transfer Syntaxes for lossless JPEG Image Compression, shall support the following lossless compression: The subset (first-order horizontal prediction [Selection Value 1) of JPEG Process 14 (DPCM, non-hierarchical with Huffman coding) (see Annex F).

- Any implementation which conforms to the DICOM Standard and has elected to support any one of the Transfer Syntaxes for 8-bit lossy JPEG Image Compression, shall support the JPEG Baseline Compression (coding Process 1).

- Any implementation which conforms to the DICOM Standard and has elected to support any one of the Transfer Syntaxes for 12-bit lossy JPEG Image Compression, shall support the JPEG Compression Process 4.

Note: The DICOM conformance statement shall differentiate whether or not the implementation is capable of simply receiving or receiving and processing JPEG encoded images..."[1]

Neben JPEG ist auch MPEG2 MP@ML Kompression möglich. Mit der Frage in wie weit ein inhärent verlustreiches Verfahren medizinisch akzeptabel ist geht der Standard folgendermaßen um:

"Note: MPEG2 compression is inherently lossy. The context where the usage of lossy compression of medical images is clinically acceptable is beyond the scope of the DICOM Standard. The policies associated with the selection of appropriate compression parameters (e.g. compression ratio) for MPEG2 MP@ML are also beyond the scope of this standard..."[2]

Gleiches gilt ebenfalls für die Datenreduktion bei JPEG. Folgend die Datenelemente, die für die MPEG2 Kompression zur Verfügung gestellt werden:

- Samples per Pixel (0028,0002)

[1] Aus (4), Seite 48
[2] Aus (4), Seite 51

- Photometric Interpretation (0028,0004)
- Bits Allocated (0028,0100)
- Bits Stored (0028,0101)
- High Bit (0028,0102)
- Pixel Representation (0028,0103)
- Planar Configuration (0028,0006)
- Rows (0028,0010), Columns (0028,0011), Cine Rate (0018,0040) und Frame Time (0018,1063) oder Frame Time Vector (0018,1065) sollten nach folgender Tabelle angegeben werden.

Video Typ	Spatial resolution	Frame Rate	Frame Time	Maximum Rows	Maximum Columns
525-line PAL	FULL	30	33.33ms	480	720
625-line NTSC	FULL	25	33.33ms	576	720

Tabelle 6: Parameterdaten für MPEG, aus (4)

Audiodaten innerhalb des Bit - Stream sollte folgenden Vorgaben genügen:

- CBR MPEG-1 LAYER III (MP3) Audio Standard
- bis zu 24 Bits
- 32 kHz, 44,1 kHz oder 48 kHz für den Hauptkanal
- ein Hauptkanal mono oder stereo und optional ein oder mehrere Nebenkanäle

Zum Austausch zeitbasierter Signale oder Kurvendaten steht das *Waveform Data Element (5400, 1010)* zur Verfügung. Waveform Bits Allocated (5400, 1004) gibt die Größe jedes Datensamples an, wobei 8 und 16 Bit die erlaubten Eingabemöglichkeiten sind. Folgende zugehörigen Datenelemente sollten mit der Selben VR kodiert werden wie die Kurvendaten selbst:

- Channel Minimum Value (5400,0110)
- Channel Maximum Value (5400,0112)
- Waveform Padding Value (5400,100A).

Zuletzt sei noch kurz das Prinzip der *Unique Identifiers* (UIDs). Sie garantieren die weltweite Einzigartigkeit unzähliger Items. Das Schema basiert auf der *OSI Object Identification* - definiert in ISO 8824. Eine UID besteht aus zwei teilen - <org root> und <suffix>. Erstes bezeichnet die Organisation, zweites einen einzigartigen, numerischen Code innerhalb dieser. Die genannte Organisation hat für die tatsächliche Einzigartigkeit des <suffix> Sorge zu tragen.

3.5 Part 10: Media Storage and File Format for Media Interchange

Im 10. Teil des DICOM Standards wird ein Model zur Speicherung medizinischer Bilddaten auf austauschbare Medien spezifiziert. Es wird ein Framework erläutert, mit dem der Austausch medizinischer Bilder und deren Metadaten auf physikalischen Speichermedien möglich ist. Zu diesem Zweck wird unter anderem spezifiziert:

- ein Model zur Speicherung mit dem Konzept der *Media Storage Application Profiles*

- das *DICOM File Format* zur Kapselung von IODs spezifiziert

- ein *DICOM File Service,* welcher die Unabhängigkeit zum Medium sicherstellt

Zur Regelung des Datenaustausches in einem Netzwerk und von Speichermedien wurde das *General DICOM Communication Model* (Abb. 8) aufgestellt. DICOM AEs könne entweder den OSI Upper Layer Service für physische Netzwerk Kommunikation oder den DICOM File Service für physikalische Speichermedien verwenden.

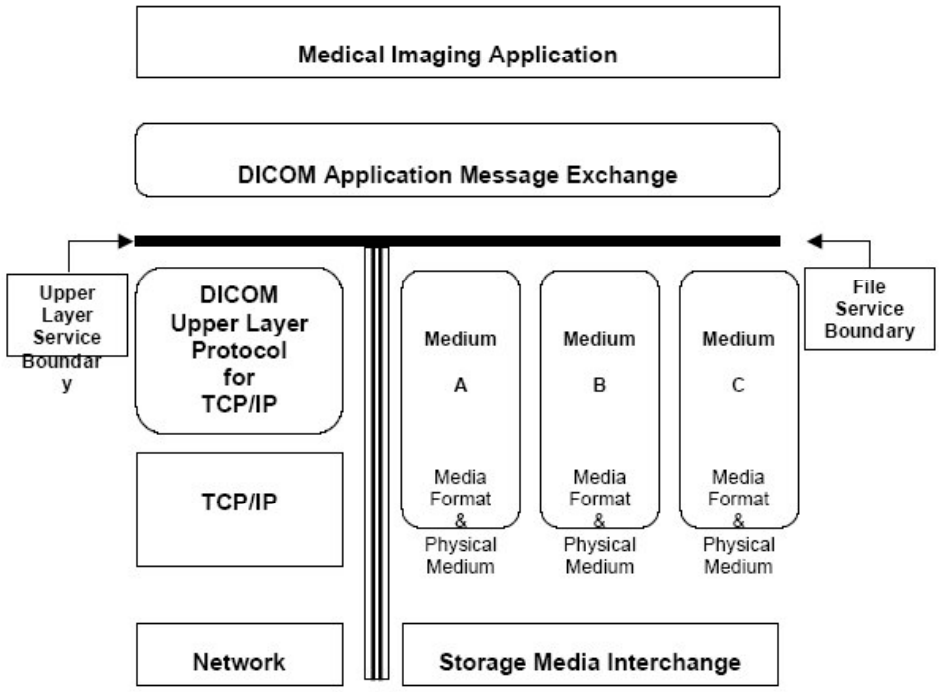

Abbildung 8: General DICOM Communication Model, aus (6)

Erweitert wird das Model von dem *DICOM Media Storage Model* (Abb. 9)*,* welches Aspekte regelt, die direkt mit Austausch von Daten über ein austauschbares Speichermedium zu tun haben. Es umfasst drei Layer:

I. Physical Media Format Layer Eigenschaften des physikalischen Mediums, wie z.B. mechanische- und Aufnahmeeigenschaften werden definiert.

II. Media Format Layer
Bitströme werden in einer spezifischen Struktur organisiert, welche oft vom verwendeten Betriebssystem bestimmt wird

III. DICOM Data Format Layer
Er enthält eine Vielzahl von Spezifikationen:

- DICOM Media Storage SOP Klassen und zugehörige IODs

- DICOM File Format
- Secure DICOM File Format
- DICOM Media Storage SOP Klasse
- DICOM Media Storage Application Profiles
- DICOM Security Profiles for Media Storage

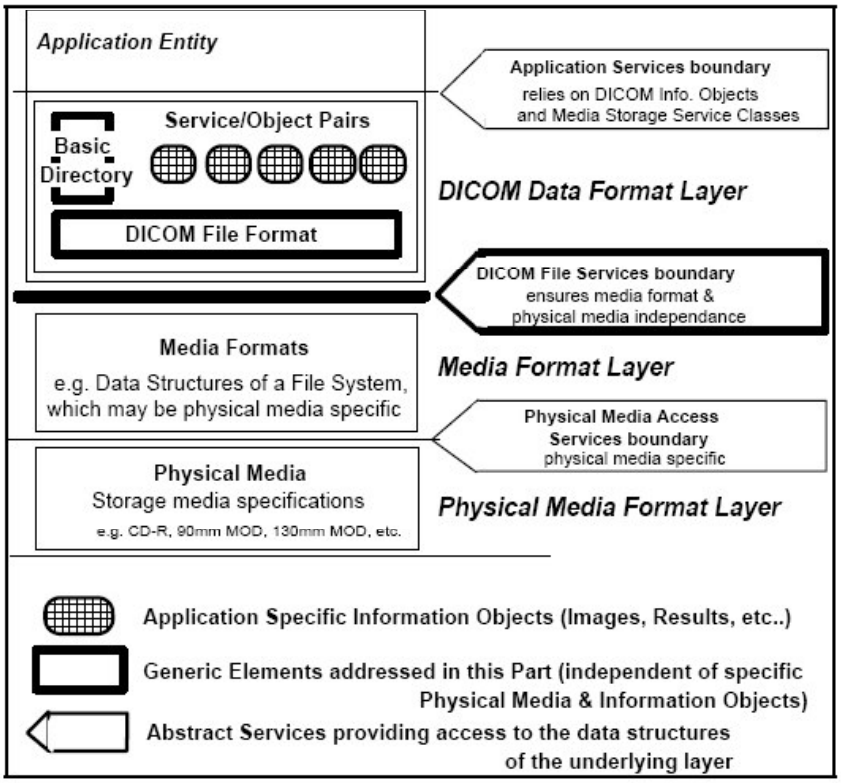

Abbildung 9: DICOM Media Storage Model, aus (6)

Die SOP Klassen und ihre IODs liefern wichtige Darstellungsinformationen. Die IODs sollten den DICOM File Service verwenden, wenn sie das Ziel haben Daten auf ein Medium zu speichern.
In den *DICOM Media Storage Application Profiles* werden verschiedene Angaben über die Beschaffenheit der Layer gemacht, je nach Anwendung sollte es enthalten:

- Eine Beschreibung über den Anwendungszweck des Profils (z.B. Angioraphy etc.).
- Auf Datenformatebene die Auswahl an SOP Klassen und deren IODs. Es sollte angegeben werden ob diese optional sind oder nicht. Sie werden über die UID angesprochen.

- Die Spezifikation des Mediums. Dies geschieht anhand von Referenzen die im 12. Teil des Standards zu finden sind.

- Eine angebrachte Transfer Syntax.

- Die Angabe eines Sicherheitsprofils, referenziert im 15. Teil des Standards.

- Andere Definitionen die eine Interoperabilität erlauben. (z.B. maximale Dateigröße)

Das DICOM File Format kapselt das *Data Set.* Zuvor wird die Datei von einem Header eingeleitet, in dem sich mehrere Metainformationen befinden. Dieser Header besteht aus einer 128 Byte großen Präambel gefolgt von einem 4 Byte DICOM Präfix gefolgt von mehreren *File Meta Elements.* In der Präambel müssen nicht zwangsweise Informationen vorhanden sein. Sie ist vielmehr dazu gedacht von spezifischen Implementationen je nach Anwendungsprofil genutzt werden zu können. Das Präfix sollte die Buchstaben "DICM" enthalten. Weitere Metaelemente zeigt folgende Tabelle 4.

Attribute Name	**Tag**	**Type**	**Attribute Describtion**
Group Length	(0002,0000)	1	Number of bytes following this File Meta Element (end of the Value field) up to and including the last File Meta Element of the Group 2 File Meta Information
File Meta Information Version	(0002,0001)	1	This is a two byte field where each bit identifies a version of this File Meta Information header. In version 1 the first byte value is 00H and the second value byte value is 01H. Implementations reading Files with Meta Information where this attribute has bit 0 (lsb) of the second byte set to 1 may interpret the File Meta Information as specified in this version of PS 3.10. All other bits shall not be checked.
Media Storage SOP Class UID	(0002,0002)	1	Uniquely identifies the SOP Class associated with the Data Set. SOP Class UIDs allowed for media storage are specified in PS 3.4 of the DICOM Standard - Media Storage Application Profiles.
Media Storage SOP Instance UID	(0002,0003)	1	Uniquely identifies the SOP Instance associated with the Data Set placed in the file and following the File Meta Information.
Transfer Syntax UID	(0002,0010)	1	Uniquely identifies the Transfer Syntax used to encode the following Data Set. This Transfer Syntax does not apply to the File Meta Information.
Implementation Class UID	(0002,0012)	1	Uniquely identifies the implementation which wrote this file and its content. It provides an unambiguous identification of the type of implementation which last wrote the file in the event of interchange problems. It follows the same policies as defined by PS 3.7 of the DICOM Standard (association negotiation).
Implementation Version Name	(0002,0013)	3	Identifies a version for an Implementation Class UID (0002,0012) using up to 16 characters of the repertoire identified in Section 8.5. It follows the same policies as defined by PS 3.7 of the DICOM Standard (association negotiation).
Source Application Entity Title	(0002,0016)	3	The DICOM Application Entity (AE) Title of the AE which wrote this file's content (or last updated it). If used, it allows the tracing of the source of errors in the event of media interchange problems. The policies associated with AE Titles are the same as those defined in PS 3.8 of the DICOM Standard.

Private Information Creator UID	(0002,0100)	3	The UID of the creator of the private information (0002,0102).
Private Information	(0002,0102)	1C	Contains Private Information placed in the File MetaInformation. The creator shall be identified in (0002,0100). Required if Private Information Creator UID (0002,0100) is present.

Tabelle 7: Header Metaelemente, aus (6)

Der *DICOM File Service* ermöglicht einen abstrakten Blick auf die Dateien aus der Sicht eines Service - Users auf dem *Data Format Layer.* Er ist einfach gehalten, so dass er mit vielen bekannten Formaten und Dateisystemen kompatibel ist, jedoch umfangreich genug um Dateien zu verwalten und Zugang zu ihren Daten zu bekommen.
Über den Service lassen sich ein oder mehr Dateien durch ein *File-set* erstellen. Diese haben keine besondere Ordnung innerhalb des Sets, können aber durch ihre *File-ID* identifiziert werden. Das Set wird über einen UID referenziert, die sich auch nicht ändert, sollte der Inhalt des Sets sich ändern. In jedem File-set sollte eine Datei mit der ID DICOMDIR vorhanden sein. Hier werden grundlegende Informationen über das *File-Set* abgelegt. Für den Informationsaustausch zwischen AEs stehen über den File Service mehrer *Media Storage Services* zur Auswahl.

a) M-WRITE, neue Datei im Set erstellen und ID zuweisen;
b) M-READ, zum Lesen von Dateien anhand ihrer ID;
c) M-DELETE, zum Löschen von Dateien anhand ihrer ID;
d) M-INQUIRE FILE-SET, zum Prüfen von freiem Speicher zur Erstellung neuer Dateien
e) M-INQUIRE FILE, zum Abfragen von Erstellungszeitpunkt (Datum, Uhrzeit) einer Datei

Eine AE kann mehrere Rollen einnehmen. Ein *File-set Creator (FSC)* erstellt Dateien (auch DICOMDIR) mittels M-WRITE. Als *File-set Reader (FSR)* ist nur das Lesen nicht aber das Manipulieren der Daten möglich. Letzteres übernimmt der *File-set Updater (FSU),* jedoch nur in der DICOMDIR - Datei. Jedoch ist es ihm möglich neue Dateien zu erstellen oder alte zu löschen. Jede DICOM konforme Implementierung müssen eine der Rollen wählen, wobei auch Kombinationen von Rollen sind möglich (z.B. FSC und FSR). Tabelle 5 zeigt welche Rolle bzw. welche Kombinationen, die Operationen voraussetzen.

Rollen	**M-WRITE**	**M-READ**	**M-DELETE**	**M-INQUIRE FILE-SET**	**M-INQUIRE FILE-SET**
FSC	zwingend	-	-	zwingend	-
FSR	-	zwingend	-	-	zwingend
FSC+FSR	zwingend	zwingend	-	zwingend	zwingend
FSU	zwingend	zwingend	zwingend	zwingend	zwingend
FSU+FSC	zwingend	zwingend	zwingend	zwingend	zwingend
FSU+FSR	zwingend	zwingend	zwingend	zwingend	zwingend
FSU+FSC+FSR	zwingend	zwingend	zwingend	zwingend	zwingend

Tabelle 8: Rolleverteilung, aus (6)

4.JPEG

In der heutigen Zeit, in der das Internet einer großen Anzahl der Bevölkerung zur Verfügung steht, ist JPEG sicherlich vielen Nutzern des WWW ein Begriff. Bekannt ist es vor allem als effizientes Grafikformat mit denen Bilder auf Webseiten dargestellt werden. Weniger bekannt dürfte jedoch sein, dass JPEG ursprünglich die Bezeichnung des Komitees war und noch heute ist, das den Standard entwickelte und ihn 1993 unter der Bezeichnung ISO/IEC IS 10918-1 | ITU-T Recommendation T.81 veröffentlichte.
Ziel des Komitees war es einen Standard zu schaffen der die Bildkompression

- mit akzeptabler Komplexität
- in Unabhängigkeit von der Bildbeschaffenheit und
- mit vom Nutzer beeinflussbarer Qualität

ermöglichen sollte.
Ursprünglich handelte es sich bei JPEG um ein verlustbehaftetes Kompressionsverfahren, was es für eine Nutzung im Bereich der professionellen Bildbearbeitung ungeeignet machte. Aus diesem Grund existieren heute verschiedene Weiterentwicklungen wie JBIG oder JPEG2000 die eine Verbesserung der Bildqualität und eine verlustfreie (lossless) Kompression möglichen machen.
Im Folgenden sollen nun sowohl die Funktionsweise des lossy- als auch des lossless Modus der JPEG – Kompression vorgestellt werden. Auch die Unterstützung beider Varianten innerhalb des DICOM Standards soll Thema des Abschnitts sein.

4.1 JPEG – DCT – based coding

Die erste JPEG Spezifikation wurde vor dem Hintergrund entwickelt natürliche Bilder möglichst so zu komprimieren, dass der visuelle Qualitätsunterschied zwischen Original und komprimiertem Äquivalent möglichst gering ausfällt. Der Anspruch einer verlustlosen Kommpression bestand zu diesem Zeitpunkt noch nicht. Es zeigte sich, dass der Einsatz der Diskreten Kosinus Transformation (DCT) für ein solches Vorhaben zu guten Ergebnissen führt. Alle Kodierungsprozesse die auf der DCT basieren sind verlustbehaftet, allerdings liegt dies nicht an ihr selbst, da eine inverse DCT existiert. Vielmehr ist es die vorbereitende Tiefpassfilterung sowie die nachfolgende Quantisierung der DCT – Ergebnisse die zu dem Datenverlust führen. Die Kompression erfolgt anhand verschiedener Arbeitsschritte:

- Umrechnung in YUV Farbraum
- Tiefpassfilterung
- DCT
- Quantisierung
- Entropiekodierung

Abbildung 10 zeigt ein vereinfachtes, generelles Schema der ablaufenden Kodierungsprozesse, die im Folgenden näher erklärt werden.

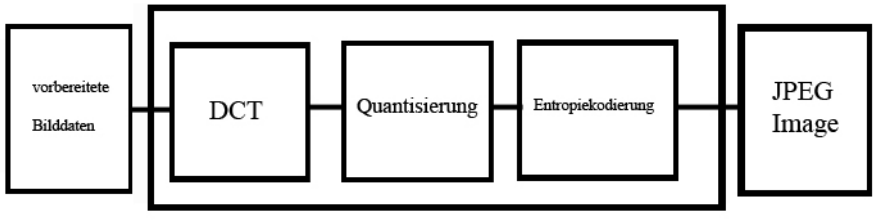

Abbildung 10: Genereller Aufbau der JPEG Kompression

4.1.1 Farbraumumrechnung

Ein für die Kompression geeigneter Farbraum ist YUV. Er liefert die Farbinformationen anhand von zwei Komponenten: der Luminanz Y (Lichtstärke pro Fläche) und der Chrominanz (Farbanteil) die wiederum aus den beiden Komponenten U und V besteht. U steht hierbei für den Farbton und V für die Farbsättigung. Ein Vorteil dieses Farbmodells ist die Nutzung des so genannten *Chroma Subsamplings* (s. unten), bei der es möglich ist das Chrominanz – Signal mit einer niedrigeren Frequenz abzutasten als die Luminanz, ohne einen sichtbaren Qualitätsunterschied zu verursachen – eine Eigenschaft, die sich mit den Zielen der JPEG Kompression gut vereinbaren lässt. Das man in diesem Farbmodell so vorgehen kann liegt an der Beschaffenheit des menschlichen Sehapparates, der Änderungen der Helligkeit eher wahrnimmt als eine Variation der Farbe, was an Verhältnis und Lage von Stäbchen (Hell-\ Dunkelsehen) und Zapfen (Farbsehen) liegt.
Da als Quelle meist das bei Farbbildschirmen verwendete RGB – Farbmodell Verwendung findet sei hier kurz die Umrechnungsformel genannt:

$$Y = 0{,}299*R + 0{,}587*G + 0{,}114*B$$
$$U = B - Y$$
$$V = R - Y$$

Die unterschiedlichen Gewichtungen bei der Berechnung der Luminanz erklären sich erneut durch die Eigenschaften des menschlichen Sehens, welches sich z.B. für Blau weniger empfindlich zeigt als für Grün.

4.1.2 Tiefpassfilterung

Die sich an der Farbraumumrechnung anschließende Tiefpassfilterung nutzt nun das oben genannte *Chroma Subsampling* um die Datenmenge zu reduzieren. Dazu werden die beiden Chrominanz – Signale – also die Farbinformation – niedriger abgetastet als das Signal für die Lichtstärke. Eine Unterabtastung im Verhältnis YUV 4:2:2 würde z.B. eine horizontal halbierte Chrominanz – Auflösung bedeuten, wohingegen ein YUV 4:4:4 Verhältnis eine identische Auflösung beider Hauptkomponenten darstellt.
Typisch für die JPEG – Kodierung ist eine Unterabtastung im Verhältnis YUV 4:2:0. In diesem Verhältnis liegt eine Unterabtastung der Chrominanz in horizontaler und vertikaler Richtung um den Faktor vier vor. Die Null bedeutet also keinesfalls das Weglassen eines Wertes, sondern vielmehr das zeilenweise hin- und herwechseln zwischen Beiden. Abbildung 2 soll die Abtastpositionen für verschiedene Fälle verdeutlichen.

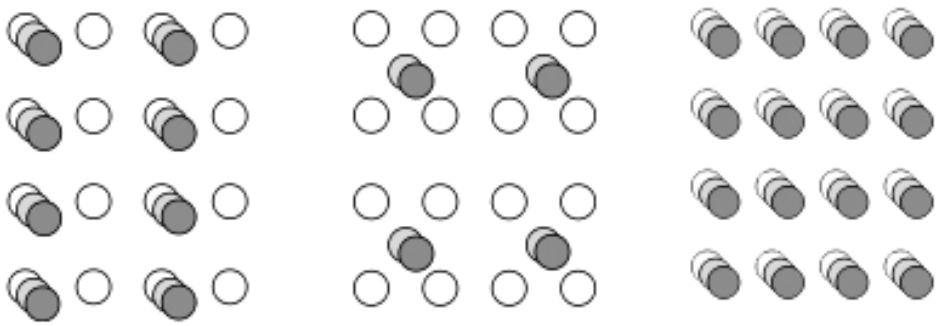

Abbildung 11: Sampling Positionen, weiß = Y, grau = U,V; Quelle: www.wikipedia.de

Der Schritt der Tiefpassfilterung ist nicht reversibel und so findet hier ein nicht rückgängig zu machender Informationsverlust statt.

4.1.3 DCT

Um die Daten für die folgende Kompression vorzubereiten, wird nun eine Diskrete Cosinus Transformation (DCT) durchgeführt. Sie ist eine orthogonale Transformation welche ein Zeitdiskretes Signal vom Orts- in den Frequenzraum überführt und heute eine der am häufigsten verwendeten Transformationen zur Redundanzreduktion (verlustfrei) von Bildsignalen. Letzteres ist vor allem darin begründet, dass sie die Daten in eine Form überführt, die gut komprimiert werden kann.
Zur Durchführung der DCT wird jede Komponente (Y, U, V) in 8x8 – Blöcke eingeteilt. Die DCT besitzt 64 Basisfunktionen und so erhält man 64 Koeffizienten, die jeweils angeben wie stark die einzelnen Basisfunktionen an dem zu kodierenden Block beteiligt sind.
Die Koeffizienten werden auch DCT – Koeffizienten genannt.
Durch die DCT werden die Daten nicht komprimiert sondern nur vor verarbeitet. So findet an diesem Punkt auch kein Datenverlust statt.

4.1.4 Quantisierung

Durch die Quantisierung der vorher berechneten DCT – Koeffizienten tritt erneut ein Datenverlust auf. Dabei werden sie durch eine Quantisierungsmatrix geteilt und auf ganze Zahlen gerundet. So gehen irrelevante Informationen verloren. Aus diesem Grund spricht man hier auch von einer Irrelevanzreduktion (verlustbehaftet). Die Matrix, die bei diesem Schritt zum Einsatz kommt bestimmt maßgeblich Qualität und Kompressionsrate des Ergebnisses. Sie wird im Header der JPEG – Datei abgespeichert.
Bei der Wahl der Matrix sollte erneut wieder Rücksicht auf die Eigenschaften des menschlichen Auges genommen werden um ein visuell zufrieden stellendes Ergebnis zu bekommen. So sollten die Quantisierungswerte für solche Frequenzen niedriger sein, auf die das Auge besonders empfindlich reagiert.
Um die abschließende Entropiekodierung zu erleichtern werden die Koeffizienten der Frequenz nach sortiert, beginnend mit dem Gleichstromanteil (DC) mit der Frequenz 0. Alle anderen Koeffizienten werden als AC (Wechselstromanteil) – Koeffizienten bezeichnet.

4.1.5 Entropiekodierung

Für die Entropiekodierung wird häufig die Huffman – Kodierung angewendet. Zu diesem Zweck werden den Zeichen zur Kodierung je nach Häufigkeit unterschiedlich viele Bits zugewiesen. Bekommt also das am häufigsten vorkommende Zeichen die wenigsten Bits bzw. das seltenste Zeichen die Meisten, so resultiert daraus eine Datenkompression.

4.2 JPEG - 2000

Der JPEG – 2000 Standard bietet neben einer verlustbehafteten auch die Möglichkeit einer verlustfreien Kompression. Zusätzlich zur grundlegenden Kompressionsfunktionalität bietet er weitere Funktionen:

- Schrittweise Rückgewinnung eines Bildes durch Genauigkeit oder Auflösung

- *Region of Interest Coding* bei dem Bildteile mit verschiedener Genauigkeit kodiert werden können.
- willkürlicher Zugang zu einzelnen Bildregionen ohne die komplette Kodierung dekodieren zu müssen.
- ein flexibles Dateiformat in dem Informationen über Deckkraft oder Bildfolgen spezifiziert werden können.
- gute Ausfallsicherheit bei Auftreten von Fehlern.

Grund für die Entwicklung des neuen Standards war neben den genannten neuen Funktionen auch der Wunsch die Schwächen des ersten Formats zu vermeiden. Da man unter anderem der freien Verfügbarkeit des JPEG Standards seinen Erfolg zuschreibt, sollte auch die neue Entwicklung ohne das Zahlen von Lizenzgebühren möglich sein.
Im Folgenden werde ich mich auf den ersten Teil des JPEG – 2000 Standards beschränken. In ihm wird die geringste Funktionalität zu Kodierung spezifiziert.
Das JPEG – 2000 Codec basiert auf der so genannten Wavelet oder auch Subband Kodierungstechnik.
Bevor das Codec detailliert beschrieben wird ist es zunächst einmal notwendig zu verstehen was für ein Bildmodell der Kodierung zu Grunde liegt.
Ein Bild ist hierbei in ein oder mehrere Komponenten zerlegt (höchstens 2^14), die wiederum in ein rechteckiges Array von Samples zerlegt sind, dessen Werte vom Typ Integer sind (1 bis 38 bits/sample). Alle Komponenten haben die gleiche räumliche Ausdehnung tragen jedoch verschiedene Informationen in sich. So besteht z.B. ein RGB Farbbild aus drei Komponenten die jeweils die Farbinformationen der jeweiligen Kanäle tragen. Die verschiedenen Komponenten müssen jedoch nicht mit derselben Auflösung abgetastet werden, wie es z.B. auch bei der unterschiedlichen Abtastrate von Luminanz und Chrominanz im YUV Farbmodell der Fall ist (s.o.). Die Geometrie der verschiedenen Komponenten wird anhand eines Referenzrasters *(reference grid*) beschrieben, dessen Ursprung sich in der oberen, linken Ecke befindet. Innerhalb des Rasters, sozusagen aufgespannt an der unteren, rechten Ecke befindet sich die Bildebene (*image area*), welche kleiner ist als das Raster selbst. Auch gibt es für die Größe des Referenzrasters eine Beschränkung, womit gleichzeitig die maximale Größe eines Bildes bestimmt wird, die das Codec noch verarbeiten kann. Die einzelnen Komponenten eines Bildes werden also auf die Bildebene gemapped, wobei verschiedene Auflösungen einzelner Komponenten für das Mapping die Bereitstellung zusätzlicher Informationen bedeuten.
Um nicht dazu gezwungen zu sein immer das Bild als atomare Einheit betrachten zu müssen – was gerade bei großen Bildern im Hinblick auf den für das Codec verfügbaren Speicher hinderlich wäre – gibt es die Möglichkeit das Bild aufzuteilen, wobei jeder Teil separat kodiert werden kann. Das Bild kann also in rechteckige, disjunkte Regionen unterteilt werden welche als Kacheln (tiles) bezeichnet werden.
Die generelle Struktur des Codec zeigt folgende Abbildung.

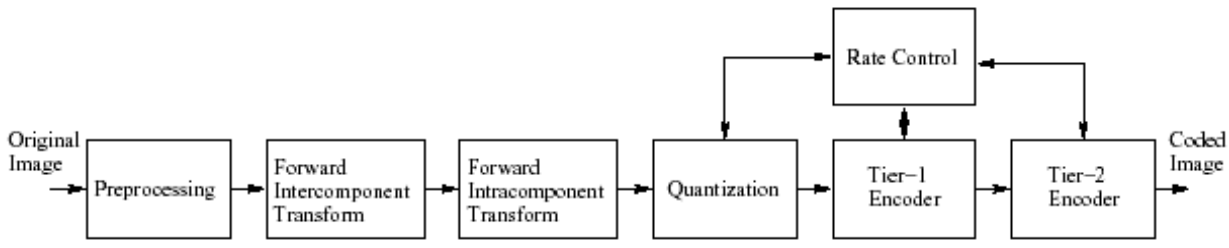

Abbildung 12: JPEG-2000 Kodierung, aus (7)

Zu Beginn findet eine Vorverarbeitung *(Preprocessing)* der Bilddaten statt. Das Codec erwartet Samplewerte die in einer gewissen Spanne um den Nullpunkt herum liegen. Ist dies nicht der Fall, so wird diese Bedingung hergestellt. Dies wird z.B. notwendig wenn die Samplewerte kein Vorzeichen haben.
Als nächstes wird eine Interkomponententransformation *(intercomponent transformation)* vorgenommen. Dies sind Transformationen die auf allen Komponenten gleichzeitig durchgeführt werden, was zu einer erhöhten Effizienz der Kodierung führen soll. Im baseline Codec sind lediglich zwei solche Transformationen definiert: eine irreversible (ICT) und eine reversible Farbtransformation. Beide Transformationen bilden Informationen vom RGB- in den YCrCb - Farbraum ab. Sie werden auf den ersten drei Komponenten eines Bildes ausgeführt, mit der Annahme, dass diese die rote, grüne und blaue Farbebene repräsentieren. Die Komponenten müssen für eine solche Transformation in derselben Auflösung vorliegen. ICT kann hierbei benutzt werden um verlustbehaftete Kompression zu erreichen, da sie auf einer real-to-real Transformation beruht, wohingegen mit der RCT auch eine verlustfreie Variante implementiert werden kann, was an der ihr zu Grunde liegenden integer-to-integer Transformation liegt, welche reversibel ist. Sie stellt eine Annäherung an die RCT dar.
Nach der Interkomponententransformation werden die Daten jeder Komponente getrennt behandelt. Dies geschieht mit den Intrakomponententransformation, welche Transformationen auf individuellen Komponenten ermöglicht. Durch die Anwendung der Wavelet - Transformation werden die Komponenten in mehrere Frequenzbänder, so genannte Subbänder, zerlegt. Aufgrund statistischer Eigenschaften dieser Subbandsignale lassen sich die Daten effizienter kodieren. Auch für diese Transformation sind wieder jeweils die real-to-real (9/7 Transformation) und integer-to-integer (5/3 Transformation) Transformationen im baseline Codec verfügbar. Zur Realisierung wird die so genannte 2-D UMDFB (uniformly-maximally-decimated filter bank) verwendet. Angenommen eine (R – 1)-level Wavelet – Transformation soll durchgeführt werden, so wird die 2-D UMDBF iterativ auf die gekachelten Daten der einzelnen Komponenten angewendet. Dies resultiert in verschiedene Arten von Subbandsignalen:

1. horizontal/vertikal Tiefpass (LL)
2. horizontal Tiefpass / vertikal Hochpass (LH)
3. horizontal Hochpass / vertikal Tiefpass (HL)
4. horizontal/vertikal Hochpass (HH)

Des Weiteren gehören zu einer (R - 1)-level Wavelet – Transformation R verschiedene Auflösungsstufen. Jedes Subband der Zerlegung kann also anhand seiner Orientierung (LL, HL, LH, HH) und seiner Auflösung identifiziert werden. Auf jeder Stufe der Auflösung, außer der Niedrigsten, wird das LL Subband weiter unterteilt in die geringeren Auflösungsstufen.

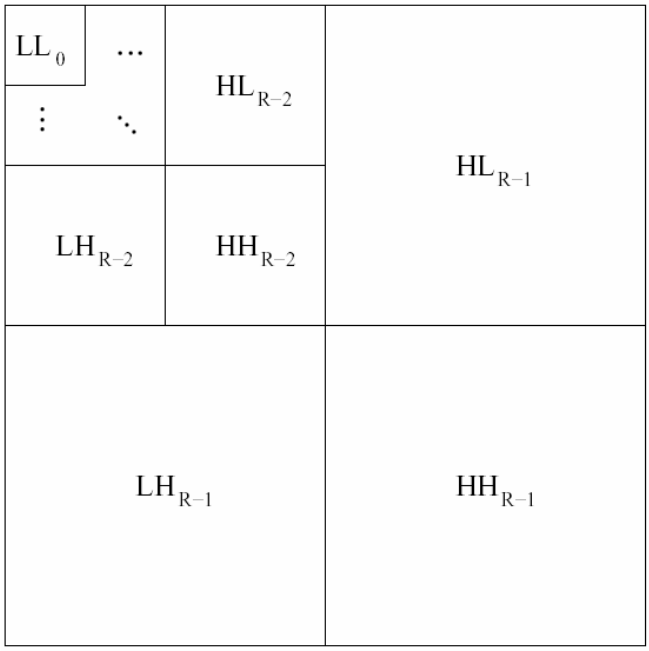

Abbildung 13: Zerlegung des Subbandes, aus (7)

Als nächste folgt die Quantisierung der Daten. Sie wird für die Koeffizienten jedes Subbandes durchgeführt, anhand einer eigenen Schrittweite, welche von der Wahl des Modus (real oder integer) bzw. von dynamischen Bandbreite des Signals abhängt. Zur Dekodierung wird versucht die Quantisierungsschritte zurück zu verfolgen bis eine Schrittgröße kleiner gleich null erreicht wird.
Als nächste wird der erste zweier *Coding Stages* angewandt: das *Tier-1 Coding*. Zu diesem Zweck werden die Indizes der Quantisierung jedes Subbandes wieder zerlegt und zwar in rechteckige Codeblöcke. Die Größe der Blöcke ist ein freier Parameter des Kodierungsprozesses der jedoch beschränkt ist (Höhe und Breite müssen Zweierpotenzen sein, Höhe * Breite <= 4096).
Mit Hilfe des *Bit-Plane Coding* werden die einzelnen Blöcke kodiert. Pro Bitebene gibt es drei *Coding Passes:* 1) *significance*, 2) *refinement* und 3) *cleanup.* Sie tasten die Blöcke alle auf die gleiche Weise ab (Abb. 14).

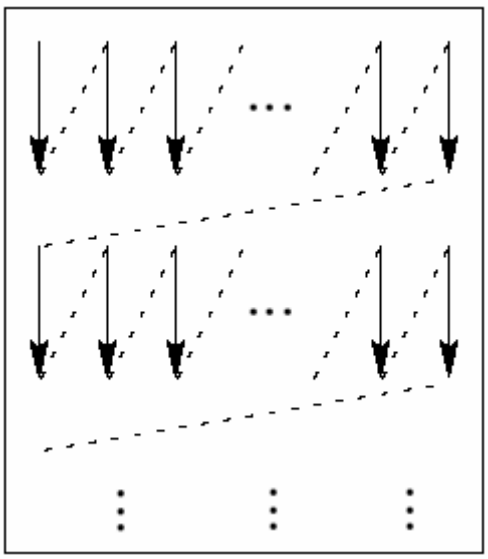

Abbildung 14: Abtastrichtung der Data Passes, aus (7)

Als erstes wird der *Significance Pass* angewandt. Er wird dazu verwendet um signifikante Samples zu erkennen, die bisher noch nicht gefunden wurden und/oder für die eine hohe Wahrscheinlichkeit zur Signifikanz für den Prozess bisher lediglich angenommen wurde. Die Signifikanz wird in einem einzigen binären Symbol kodiert. Als Pseudocode kann der Algorithmus wie folgt beschrieben werden:

ALG I[3]:

```
for each sample in code block do
        if sample previously insignificant and predicted to become significant
        during current bit plane then
                code significance of sample /* 1 binäres Symbol */
                if sample significant then
                        code sign of sample /* 1 binäres Symbol */
                endif
        endif
endfor
```

Im zweiten Schritt dem *Refinement Pass* werden die dem signifikantesten Bit (most significant bit) nachfolgenden Bits ausfindig gemacht. Sie können nach Kodierung durch ein binäres Symbol, wie es im Übrigen auch im ersten Schritt der Fall ist, weiter arithmetisch kodiert werden. Nachfolgend der Algorithmus als Pseudecode:

ALG II[4]:

```
for each sample in code block do
        if sample found significant in previous bit plane then
                code next most significant bit in sample /* 1 binäres Symbol */
        endif
endfor
```

Abschließend wird der *Cleanup Pass* durchgeführt. Er unterscheidet sich nicht sehr vom ersten Durchgang. Der Unterschied besteht darin, dass hier Informationen über Samples verarbeitet werden die mit hoher Wahrscheinlichkeit insignifikant bleiben werden. Außerdem werden hier die Samples teilweise in Gruppen verarbeitet und nicht ausschließlich individuell wie beim *Significance Pass.* Man spricht daher auch von der Nutzung so genannter *vertical scans,* die in ihrer Form und Größe den vertikalen Pfeilen in Abb. 14 entsprechen. Hier noch zur Vervollständigung den Algorithmus in Pseudocode:

ALG III[5]:

```
for each vertical scan in code block do
        if four samples in vertical scan and all previously insignificant and unvisited
        and none have significant 8-connected neighbor then
                code number of leading insignificant samples via aggregation
                skip over any samples indicated as insignificant by aggregation
        endif
        while more samples to process in vertical scan do
                if sample previously insignificant and unvisited then
                        code significance of sample if not already implied by run /* 1 binäres Symbol */
                        if sample significant then
                                code sign of sample /* 1 binäres Symbol */
                        endif
```

[3] aus (7), S. 8
[4] aus (7), S. 9
[5] aus (7), S.9

```
            endif
        endwhile
endfor
```

Anschließend folgt das *tier-2 encoding*. Hier werden die Daten aus dem vorherigen Schritt gepackt. Die daraus resultierenden Pakete stellen den finalen Datenstrom dar. Sie schließen zwei Teile ein: den Header und den Body. Der Header legt fest welcher Art die enthaltenen Daten sind, der Body enthält dann die Daten an sich. Die Regelung der Datenrate ist in Schichten *(quality layer)* organisiert. Sie umfassen L Schichten, durchnummeriert von 0 bis L – 1. Wichtige, signifikante Daten werden in niedrigen Layern kodiert, während die Details auf den höheren untergebracht werden. Je nach gewünschter Qualität werden auch Entscheidungen über das ausrangieren von bestimmten Daten getroffen, was jedoch nur bei Verlust behafteter Kodierung passiert.
Für jede 4-tupel Kombination von Komponente, Auflösung, Layer und Codeblockvereinigung (Umfeld *(precinct)* genannt) wird ein Paket „geschnürt", dabei sind leere Pakete durchaus erlaubt, schließlich kommt es vor, dass durch das Wegwerfen von Daten eine Kombination des 4-tupels leer bleiben muss.
Die Anordnung der Pakete kann auf verschiedene Weise geschehen:

- layer-resolutioncomponent-position ordering
- resolution-layer-componentposition ordering
- resolution-position-component-layer ordering
- position-component-resolution-layer ordering
- component-position-resolution-layer ordering

Die Anordnung der Pakete wird hier jeweils in der angegebenen Reihenfolge organisiert.
Die Ordnungen werden Progressionen genannt. Neben diesen fünf vordefinierten ist es auch möglich Nutzerdefinierte Varianten zu implementieren.
Wie auf Abb. 12 dargestellt interagiert die *Rate Control* mit den Kodierungsschritten Quantisierung, tier-1 und tier-2. Sie kann auf zwei Arten in die Kodierung involviert werden:

1. durch die Bestimmung der Größe der Quantisierungsschritte
2. durch die Auswahl der zu kodierenden Daten

Werden die Schritte bei der Quantisierung vergrößert, häufen sich Bildstörungen und Fehler zu Gunsten der Kompressionsrate. Diese Kontrollmöglichkeit erscheint zunächst äußerst simpel, hat jedoch einen entscheidenden Nachteil, denn jedes Mal wenn die Parameter der Quantisierung verändert werden ändert sich auch die Indexierung, welche den nachfolgenden Schritten zugrunde liegt. So muss das *tier-1 coding* erneut durchgeführt werden.
Der zweite Schritt ist im Gegenzug wesentlich flexibler und erlaubt eine Kodierung die durch die richtige Auswahl an Daten Störungen im Zielbild minimiert.
Das Codec erlaubt es verschiedene Regionen eines Bildes in unterschiedlicher Qualität zu kodieren. Dieses Feature wird *region – of – interest (ROI) coding* genannt und erlaubt eine solche Vorgehensweise anhand einer einfachen Technik. Man muss lediglich die Koeffizienten der ROI identifizieren und alle oder auch nur einige von ihnen dann mit höherer Präzision kodieren.

Neben den nun hier aufgeführten Möglichkeiten des baseline Codecs, gibt es für weitereichende Funktionen noch eine Vielzahl von *Extensions.* Sie werden im 2. Teil des Standards beschrieben und bieten z.B.

1. *independent regions,* eine allgemeiner gefasste Form der Kachelung, welche überlappende oder ungleichmäßig geformte Regionen möglich machen
2. zusätzliche Intrakomponententransformationen
3. zusätzliche Interkomponententransformationen
4. überlappende Transformationen
5. zusätzliche Quantisierungsmethoden
6. erweiterte ROI Funktionen
7. erweitertes Dateiformat (wurde hier nicht weiter erwähnt)

5. MPEG

Genau wie JPEG sollte auch MPEG so ziemlich jedem Internetnutzer ein Begriff sein. Auch hier handelt es sich nicht ausschließlich um einen Standard, sondern bezeichnet ebenfalls das Komitee, welche sich mit der Entwicklung beschäftigt. Über die Jahre sind so mehrere Standards veröffentlicht worden, die sich mit der Kompression von Videodaten beschäftigen. Auch die Kompression von Audiodaten und die Entwicklung so genannter Container Formate gehört zu ihrem Aufgabenbereich.
Spezifiziert wurde zunächst nur der Bitstream und der Dekodierer, so dass bei der Kodierung Raum für effizientere Methoden bleibt.
Nach einer ausführlichen Beschreibung des Videostreams wird das generelle Model von MPEG2 näher erläutert, da DICOM dieses Format unterstützt.

5.1 MPEG – Videostream

Der MPEG – Videostream besteht aus mehreren Sequenzen, die wiederum in sechs verschiedenen Layer aufgeteilt sind:

1. Sequence Layer
2. GOP Layer
3. Picture Layer
4. Slice Layer
5. Macroblock Layer
6. Block Layer

5.1.1 Sequence Layer

Der Sequence Layer ist der äußerste Layer. In seinem Header befinden sich sowohl Video- als auch Bitstream - Parameter. Erstes sind Informationen über Bildbreite, Bildhöhe, Bildrate und Seitenverhältnis, während im Zweiten Angaben zu Bitrate, Puffergröße und verschiedenen Flags gemacht werden können.

5.1.2 GOP Layer

Eine GOP *(Group of Pictures)* stellt Informationen über die Kodierreihenfolge einzelner Bildtypen bereit. Im Header dieser Schicht befindet sich der *Video Time Code* sowie Parameter, die die GOP als offen – mit verweisen zu anderen GOPs – oder geschlossen charakterisiert.

5.1.3 Picture Layer

Hier sind nun weitere Informationen über die Bilder vorhanden. Hier wird unter anderem festgelegt, ob der Datenfluss als Voll- oder Halbbild kodiert werden soll. Auch werden die Eigenschaften der Bildtypen der GOP behandelt.

- Intra-Coded Picture (I-Frame)
  Es handelt sich um ein kodiertes Einzelbild. Somit könnte man es durchaus auch als Standbild bezeichnen, womit es unabhängig gegenüber vorherigen oder nachfolgenden Bildern ist. Zur Kompression, die sehr niedrig ist, wird die aus der JPEG Kompression bekannte DCT verwendet. In einer GOP die nur aus I-Frames entspräche die Qualität

der Kompression dem Motion – JPEG Verfahren. Ein I-Frame steht immer am Anfang einer GOP.

- Predictive-Coded Picture (P-Frame)
  Zur Kodierung solcher Bilder werden vorherige P- oder I-Frames benötigt. Wie der Name vermuten lässt werden die P-Frames anhand vorhergehender Bilder vorausgesagt. Dies bedeutet, dass man nach einer Zerlegung des Bildes in Blöcke nur die in dem Frame abspeichert, in denen sich etwas verändert hat. Solche Bildveränderungen werden als Bewegungsvektoren interpretiert. Für die restlichen, unveränderten Bildteile werden die vorherigen Bilder herangezogen. Die Kompression von P-Frames ist höher als die der I-Frames

- Bidirectionally Predictive-Coded Picture (B-Frame)
  B-Frames benötigen neben Informationen aus vorhergehenden Bildern auch die Information über das, was nach ihm folgt (bidirektionale Information). Daraus ergibt sich ein Unterschied zwischen der *Display Order* und der *Stream Order*. Mögliche Reihenfolgen wären:

  i. I P B B P B B P… als Datenfluss und
  ii. I B B P B B P B… als Reihenfolge der Darstellung

  Um die unterschiedliche Reihenfolge zu kompensieren muss also Zeit versetzt auf die Bilder zugegriffen werden könne, zu welchem Zweck Bildpuffer Verwendung finden.
  B-Frames können auch zur künstlichen Erhöhung der Framerate verwendet werden, indem man zusätzliche Frames innerhalb einer GOP mit Hilfe eines *Enhancement Layers* kodiert *(Temporal Scalability)*

5.1.4 Slice Layer

Ein Bild wird in Slices unterteilt. Der Header enthält hier die Anfangsposition des Slice, eine Quantisierungstabelle sowie die jeweilige Anzahl von Makroblöcken aus denen das Slice aufgebaut ist. Einer dieser Blöcke besteht aus vier Arrays von 8x8 Helligkeitswerten (Luminanz) und zwei Arrays von 8x8 Farbwerten (Chrominanz). Die Werte liegen also wie bei JPEG im Verhältnis 4:2:0 vor, auch wenn die Abtastpositionen leicht verändert sind. An diesem Punkt wird nun auch die DCT angewendet, die erneut in Koeffizienten resultiert, die wiederum einer Quantifizierung unterzogen werden.
Slices können immer unterschiedliche Längen annehmen, solange ein komplettes Bild mit ihnen ausgefüllt werden kann.

5.1.5 Makroblock Layer

Hier werden Typen für die Makroblöcke und das *Coded Block Pattern* (CBP) angegeben.
CBP zeigt, welche Blöcke kodiert werden müssen. Der Typ gibt auch den des Bewegungsvektors an. Es gibt vier verschiedene Typen:

- Intra Type
  wird unabhängig von anderen Makroblöcken kodiert und kann in jedem Bildtyp (I-, P- oder B-Frame) eingesetzt werden

- Backward-Predicted Type
  wird in P- oder B-Frames verwendet. Enthält den Bewegungsvektor, der die Veränderungen zu einem vorausgegangenen Bild beschreibt.

- Forward-Predicted Type
  Nur einsetzbar in B-Frames. Hier gibt der Bewegungsvektor folgerichtig die Änderung zu einem folgenden Bild an.

- Average Type
  Ist ebenfalls nur in B-Frames einsetzbar. Hier werden zwei Bewegungsvektoren verwendet die jeweils die Bildveränderung zu einem vorausgegangenem und einem Nachfolgendem Bild enthalten. Aus den Informationen wird dann ein Mittelwert gebildet.

5.1.6 Block Layer

Ein Block stellt die kleinste Einheit im MPEG – Standard dar und ist 8x8 Bildpunkte groß. Hier werden schließlich auch die DCT Koeffizienten gespeichert.

5.2 MPEG-2

MPEG-2 ist der offizielle Nachfolger von MPEG-1. Der Standard soll an dieser Stelle noch einmal gesondert betrachtet werden, da es derjenige ist, der von DICOM unterstützt wird. Ursprünglich wurde es entwickelt um bei der Übertragung von digitalem Fernsehen eingesetzt zu werden. Die Qualität der Darstellung von Videodaten wurde im Vergleich zum Vorgänger verbessert und eine höhere Datenrate wurde erreicht. Auch das Ausstrahlen von Interlace - Signalen wurde nun unterstützt.
Seine weite Verbreitung verdankt der Standard wohl vor allem der DVD. Doch auch in seinem geplanten Einsatzgebiet, wie z.B. dem DVB-T, wird es heute bevorzugt benutzt.
Der Standard besteht insgesamt aus neun Teilen, die im Folgenden kurz erläutert werden.
Der erste Teil des Standards befasst sich mit der Kombination von elementaren Video- und Audiodatenströmen. Des Weiteren werden auch zusätzliche Informationen integriert die nützlich für das Speichern oder Übertragen von Video- bzw. Audiodaten sind. Dies wird auf zwei Arten spezifiziert: Im *Program Stream* und im *Transport Stream.*
Der *Program Stream* ist dem von MPEG-1 ähnlich. Er besteht aus der Kombination einer oder mehrerer *Packetised Elementary Streams* (PES), die gleich getaktet sind, in einen Einzigen. Sie sind relativ lang und werden dort eingesetzt, wo relativ wenig Fehler zu erwarten sind.
Der *Transport Stream* kombiniert hingegen PES, die nicht gleich getaktet sind. Wo Fehler zu erwarten sind, z.B. bei der Datenübertragung in Verlust behaftetes Medium, wird dieser eingesetzt.
Der zweite Teil des Standards baut auf der Videokompression von MPEG-1 auf. Die Tools wurden in Profile zusammengefasst um verschiedene Funktionen anzubieten. Zu den ursprünglichen Profilen sind nach der ersten Veröffentlichung im November 1994 weitere hinzugekommen, wie z.B. das *Multiview Profile* für Stereoaufnahmen.
Der dritte Teil befasst sich mit einer erweiterten Version des MPEG-1 Audio Standards.
Teil 4: *Conformance testing* und Teil 5: *Software simulation* entsprechen den Dokumenten des MPEG-1 Standards und deren Betrachtung ist hier nicht weiter interessant.
Teil 6 spezifiziert die *Digital Storage Media Command and Control* (DSM-CC). Dies sind eine Menge von Protokollen die die Funktionen und Operationen zur Verwaltung der Datenströme realisieren. In dem Modell wird eine Server/Client Umgebung beschrieben.

DSM-CC definiert eine logische Einheit, den *Session and Resource Manager* (SRM) durch den die Verwaltung innerhalb eines Netzwerks zentralisiert werden kann. Das Prinzip veranschaulicht Abbildung 15.

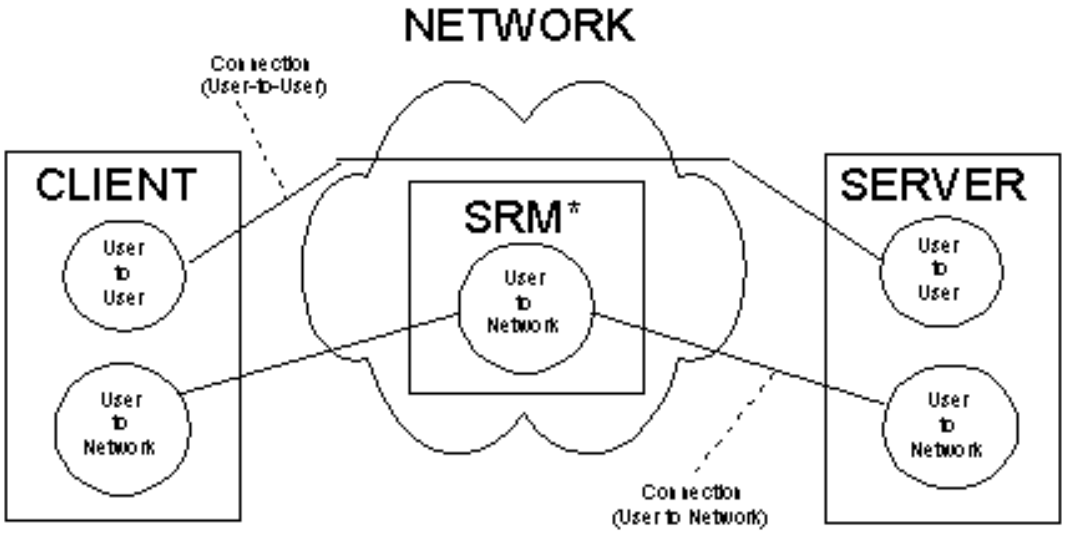

Abbildung 15: Client-Server Struktur mit SRM, Quelle: www.chiariglione.org

Im siebten Teil wurde ein *multichannel - audio* Kodierungsalgorithmus entwickelt, den es bei MPEG-1 vorher nicht gegeben hat. Gerade auf DVDs jedoch wird heute meist eine andere Kodierung wie z.B. AC-3 verwendet. Der achte Teil wurde nie beendet, da das Interesse der Industrie an der Möglichkeit 10 bits/sample Datenströme zu kodieren eher gering gewesen sei[6].
Teil neun definiert schließlich ein *Real-time Interface* für *Transport Stream* Dekodierer, welches von Netzwerken, die solche Daten verarbeiten müssen leicht adaptiert werden kann und der Neunte stellt das *conformance testing* der DSM-CC bereit.
Diese Zusammenfassung sollte einen kurzen Einblick in den Aufbau des Standards geben. Auch wenn er heute noch genutzt wird existieren heute wohl bereits zukunftsorientiertere Entwicklungen, nicht zuletzt MPEG-4.

[6] L. Chiariglione – Convenor, „Short MPEG-2 description“, 2000, aus (10)

6. DiConv – Eine Anwendung zum Konvertieren und Extrahieren von Informationen aus DICOM – Dateien

Im letzten Abschnitt vor der Schlussbetrachtung soll das im Rahmen der Arbeit entwickelte System betrachtet werden. Dazu wird sowohl auf die Architektur als auch auf die Methoden zur Realisierung näher betrachtet.

6.1 DCMTK – Referenzimplementierung eines DICOM – Toolkits

Zu Begin der Implementierung stellte sich die Frage welche Software verwendet werden sollte. Vorgabe war das Programm mit Visual Studio 2005 in C++ zu realisieren und dabei auf die Document – View – Architektur zurückzugreifen. Darüber hinaus blieb noch zu klären in wie fern ein zusätzliches Toolkit verwendet werden könnten. Um die Qualität von Software beurteilen zu könne kann man verschiedenen Aspekte betrachten. Einige wie z.B. Wartbarkeit, Nutzerfreundlichkeit und Effizienz sind eher von der Implementierung selbst abhängig. Die Zuverlässigkeit kann jedoch durch den Einsatz erprobter Software einfacher und zeitsparender realisiert werden. Die Portabilität spielte keine große Rolle, da eine Überführung auf ein anderes System nicht geplant und eher unwahrscheinlich ist.
Neben der Zeitersparnis kann mit einem Toolkit auch der Umfang des Programms übersichtlicher gefasst werden.
Da beim Einsatz von Software durchaus auch zusätzliche, hier unerwünschte Kosten anfallen könnten, war darauf zu achten, dass es sich um Open Source Software handelt.
Es sollt natürlich auch Möglichst viele Funktionen von denen bereitstellen die Implementiert werden sollten, die da sind:

- Konvertierung nach JPEG und MPEG
- Konvertierung nach XML

Diese Vorraussetzungen schien das DICOM Toolkit von Office zu erfüllen. Dies ist eine Sammlung von Bibliotheken unter anderem zur Untersuchung, Erzeugung und Konvertierung von DICOM – Bilddaten. Es wird in vielen Kliniken und Instituten weltweit eingesetzt auch in kommerziellen Produkten. Es ist kompatibel zu vielen Betriebssystemen, darunter Linux, Solaris und Windows. Dies alles und die Tatsache, dass es sich um Open Source Software handelt sind Gründe dafür das Toolkit zu benutzen.

6.2 MFC – Document – View – Architektur

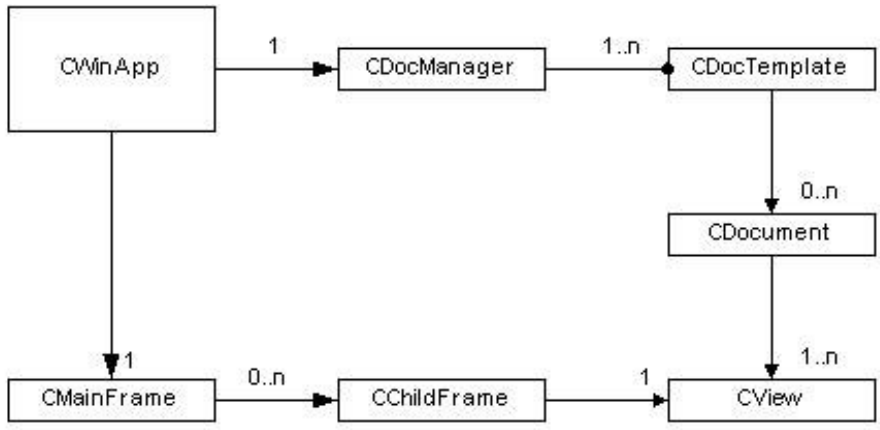

Abbildung 16: MFC – Document – View Architektur, aus (12)

Die Document – View – Architektur wird dazu verwendet ein komplettes Rahmenfenster mit Menü-, Werkzeug- und Statuszeile zu implementieren ohne die dafür verantwortliche *CFrameWnd* - Klasse direkt benutzen zu müssen. Über den *AppWizard* werden zwei Klassen erstellt. Die *CDocument* – Klasse verwaltet die Daten eines Dokumentes, wobei dies frei programmierbar ist. Die *CView* – Klasse ist realisiert die Sicht auf das Document – also das Fenster – weshalb sie auch von *CWnd* abgeleitet ist. An diesem Punkt werden darüber hinaus auch Benutzeraktionen registriert, welche das Dokument verändern können. Sie ist jedoch nicht als die Darstellung des dem Benutzer zur Verfügung gestellten Windows – Fenster zu sehen, sie kümmert sich vielmehr um den Inhalt des Fensters. Was der Nutzer sieht, regelt *CMainFrame* – Klasse bzw. *CChildFrame* für mehrere Fenster. Initialisiert wird die Anwendung von der *CWinApp* – Klasse. Sie zeigt weiterhin ein Dialogfenster an.
CDocTemplate enthält Vorlagen für neu zu erstellende Documents oder Views und ein Objekt der CDocManager verwaltet diese.
Zentral in dem Document – View – Konzept ist das Zusammenspiel von Document, View und Anwendung. Ein Document kann mehrere Views besitzen, die alle einzeln adressierbar sind. Vom View aus lässt sich ebenfalls aufs Document zugreifen (*CView::GetDocument*).
Über Änderungen am Document informiert der View selbiges. Dies sollte man nach Möglichkeit jedoch nicht über die Benutzung von Document – Membern machen, um die Trennung der Klassen einzuhalten und da ein Document auch mehrere Views haben kann, und somit wissen muss, dass es Verändert wurde um darauf alle Views auf den neusten Stand zu bringen (*CDocument::UpdateAllViews*). Auf den einzelnen Views wird mittels der Funktion *CView::OnDraw* gezeichnet.
Buttons und Menüeinträge können einfach durch die Entwicklungsumgebung erstellt werden. Sie bekommen eine eigene ID zugewiesen, über die man sie identifizieren kann, um Ihnen Methoden zuzuweisen.
Nutzereingaben werden über das Abfangen der jeweiligen *Windows Messages* registriert die in der *MESSAGEMAP* im View deklariert werden.
In der Implementierung des Programms dieser Studienarbeit wurde außerdem die *CImage* – Klasse, welche Bilddateien verschiedener Formate erstellen und verwalten kann, verwendet.
Von ihr wurde die Klasse CImageDCM abgeleitet welche Methoden zur Konvertierung von DICOM Daten enthält. Sie wird im Folgenden beschrieben.

6.3 CImageDCM – Erweiterung der MFC – Klasse CImage

Die entwickelte Applikation DiConv enthält neben den Klassen der MFC - Document - View Architektur noch die von CImage abgeleitete Klasse CImageDCM. In ihr befínden sich vier Methoden zur Konvertierung der Daten.

- LoadDCM (const *char filename)
- Dump2XML()
- Conv2JPEG()
- Conv2MPEG()

Mit *CImageDCM::LoadDCM* wird das DICOM Data Set geladen. Zu diesem Zweck wird ein Objekt der Klasse *DcmFileFormat* erstellt. In dieses wird der Datensatz mittels *loadFile* geladen. Der Methode werden mehrere Parameter Übergeben.

- fileName — Name der Datei
- readXfer — Transfersyntax die verwendet werden soll um die Datei zu lesen (Es stehen eine ganze Reihe von Transfersyntaxen zur

Verfügung, z.B. EXS_JPEG2000, EXS_MPEG2MainProfileAtMainLevel oder EXS_JPEGLSLossy)

- groupLength flag, das angibt wie gleichnamige Tags behandelt werden
- maxReadLength maximal Anzahl Bytes eines Elements
- readMode einlesen der Datei mit oder ohne Header

Die geladenen Daten werden in der Membervariablen *m_file* gespeichert.
In einem nächsten Schritt werden die Bilddaten in ein weiteres Objekt der Klasse *DicomImage* geladen, mit folgenden Paramtern:

- object Zeiger auf eine DICOM Datenstruktur (z.B. vom Typ DcmFileFormat)
- xfer Transfersyntax
- flags Konfigurationsflags
- fstart erster zu verarbeitender Frame
- fcount Anzahl Frames

Auf diese weise ist es also möglich die Daten auszulesen. Für die Konvertierung nach XML kann nun einfach die Methode *DcmFileFormat::writeXML* verwendet werden. Dies geschieht über die genannte Memberfunktion. Aufgerufen wird diese, wird der Button DUMP2XML gedrückt. Die dabei versendete Windows - Message wird abgefangen und führt die zugehörige Funktion der *CDiConvDoc* - Klasse aus.
Die äquivalenten Methoden für MPEG und JPEG funktionieren nach dem gleichen Schema.
Ebenfalls in dieser Klasse aufgerufen wird auch die *CImageDCM::LoadDCM* Methode, und zwar in *CDiConvDoc::OnOpenDocument().*
Die Darstellung der Daten erfolgt über den View bzw. der *OnDraw* Methode.
Zu diesem Zweck werden weitere MFC Klassen verwendet (z.B. CImage).
Auf diese Weise lässt sich recht einfach die Konvertierung in die gewünschten Formate realisieren.

7. Schlussbetrachtung

Laut Aufgabenstellung sollte ein weg zur effizienten Konvertierung von DICOM Formaten in Standardformate gefunden werden. Bei der zur Verfügung stehenden Kompressionsrate wurden die Funktionen des DCMTK, welches mehrere Transfersyntaxen anbietet, verwendet, die für die Erfüllung der Aufgabe von DiConv, nämlich der Vorbereitung der Datensätze zur Archivierung, ausreichend sind.
Der DICOM Standard unterstützt mehrer Möglichkeiten und spezifiziert mehr als eine Transfersyntax. Für als am effizientesten würde man vielleicht auf Anhieb die Varianten halten, die am meisten Kompression verspricht. Jedoch ist ein solches Vorgehen nicht immer sinnvoll. Gerade im medizinischen Gebrauch ist eine Unverfälschtheit der Daten wichtig.
Daher wäre es zum Beispiel Sinnvoll das Kompressionsverfahren und dessen Rate wählbar zu gestallten, so dass der Benutzer immer angesichts der Umstände entscheiden kann. Für medizinische Zwecke ist natürlich eine möglichst hohe Nähe zum Original bedeutsam, während für die Archivierung die Größe der Daten eine Rolle spielt. So sind zum Beispiel auch Systeme interessant, die die Daten für eine bestimmte Zeitspanne verlustfrei speichern und nach Ablauf die Daten für die Archivierung Komprimieren.
Die effizienteste Lösung ist also nicht immer gleich die Beste. Da mit dem System jedoch nur einen Archivierung ermöglicht werden soll, wurde die bestmögliche Kompressionsrate des Toolkits verwendet.

Quellenangaben

(1) The DICOM Standard, PS 3.1: Introduction and Overview, 2006, NEMA, Virginia, USA
(2) The DICOM Standard, PS 3.3: Information Object Definitions, 2006, NEMA, Virginia, USA
(3) The DICOM Standard, PS 3.4: Service Class Specifications, 2006, NEMA, Virginia, USA
(4) The DICOM Standard, PS 3.5: Data Structures and Encoding, 2006, NEMA, Virginia, USA
(5) The DICOM Standard, PS 3.6: Data Dictionary, 2006, NEMA, Virginia, USA
(6) The DICOM Standard, PS 3.10: Media Storage and File Format for Data Interchange, 2006, NEMA, Virginia, USA
(7) M.D. Adams, "The JPEG-2000 Still Image Compression Standard", 2001, University of British Columbia, Vancouver, Kanada
(8) Andreas Kinzler, "MFC-Programmierung, Teil 3: das Document-View-Konzept", www.heise.de
(9) www.jpeg.org
(10) www.chiariglione.org (Umfangreiche MPEG Homepage)
(11) www.wikipedia.de
(12) R. Allen, „MFC Document/View documentation and enhancements", www.codeproject.com