

Otto-von-Guericke-Universität Magdeburg



*Fakultät für Informatik  
Institut für Verteilte Systeme  
Arbeitsgruppe Softwaretechnik*

## **Diplomarbeit**

### **Implementierung und Robustheitsanalyse eines automatischen Verfahrens in der Medizinischen Bildverarbeitung**

Themensteller: Prof. Dr.-Ing. habil. R. Dumke  
Betreuer: Dipl.-Inform. Martin Kunz  
Michael Hamm (Siemens Medical Solutions)

vorgelegt von: Matthias Ludewig  
Badeteichstrasse 10  
39126 Magdeburg  
Matrikelnummer: 159987

Ludewig, Matthias:

*Implementierung und Robustheitsanalyse eines automatischen Verfahrens in der Medizinischen Bildverarbeitung*

Diplomarbeit, Otto-von-Guericke-Universität Magdeburg, 2007

## Kurzfassung

Der Trend zur Automatisierung zeigt sich nicht nur in der Automobilbranche oder in der Informatik, sondern auch zunehmend in den Fachbereichen der Medizin. Hier wurde das Potenzial einer Automatisierung richtig erkannt, umgesetzt und wird ständig weiter entwickelt. An Stellen, an denen noch vor einiger Zeit eine Auswertung des bestehenden Datenmaterials manuell ausgeführt werden musste, werden immer mehr Vorgänge durch einen automatischen Ablauf ersetzt. Ein solches Vorgehen hat auch bei der Bestimmung der individuellen Herz-Leistungsfähigkeit bzw. bei der Erkennung von Herzkrankheiten am Menschen Einzug gehalten.

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung und Implementierung eines automatisierten Verfahrens zur Ablösung eines manuellen Prozesses bei der medizinischen Bildverarbeitung. Zur Beurteilung der Softwarequalität wird die ISO-Norm 9126 unter Betrachtung der Funktionalität, Zuverlässigkeit und Effizienz auf den Quellcode angewendet, ausgewertet und beurteilt.

Unter Verwendung bestehender Algorithmen zur Erkennung der linken Herzkammer in medizinischen Bildern wird ein geeignetes Software-Modul entwickelt, welches parallel in einem laufenden bildgebenden Verfahren der Magnetresonanz-Tomographie eine Auswertung der medizinischen Daten vornimmt. Zu den Vorteilen des automatischen Ablaufs gehören ein geringerer zeitlicher Aufwand bei der medizinischen Befundung sowie eine Benutzerunabhängigkeit und eine stark verminderte Fehleranfälligkeit. Die Reproduzierbarkeit der Auswertungsergebnisse wird bei einer automatischen Routine um ein Vielfaches gesteigert. Möglich geworden durch die eminente Steigerung der Rechenleistung sind online arbeitende Bildverarbeitungsprozesse. Die Schnelligkeit und Fehlerunanfälligkeit von automatisch arbeitenden Analyseverfahren führen dazu, dass diese Verfahren einen routinemäßigen Einzug in die Diagnostik halten.

Während der Realisierung werden verschiedene Vorgehensweisen bei der Softwareprüfung vorgestellt und diese, besonders im Hinblick auf die Entwicklung von Medizinischer Software, untersucht. Mit Hilfe der ermittelten Ergebnisse wird die Verbesserung zur bisherigen manuellen Arbeitsweise aufgezeigt und die Vorteile für eine zukünftige Anwendung hervorgehoben.

# Inhaltsverzeichnis

<b>Abkürzungsverzeichnis.....</b>	<b>VI</b>
<b>Abbildungsverzeichnis.....</b>	<b>VII</b>
<b>Tabellenverzeichnis.....</b>	<b>IX</b>
<b>1 Einleitung.....</b>	<b>10</b>
1.1 Motivation.....	10
1.2 Wirtschaftliche Aspekte.....	11
1.3 Zielstellung.....	12
1.4 Aufbau der Diplomarbeit.....	13
<b>2 Medizinische und Technische Grundlagen.....</b>	<b>15</b>
2.1 Das menschliche Herz.....	15
2.1.1 Aufbau und Funktion des Herzens.....	16
2.1.2 Herzdiagnose durch Analyse des linken Ventrikels.....	18
2.2 Die Magnetresonanz-Tomographie.....	19
2.2.1 Theoretische Grundlagen der Magnetresonanz.....	20
2.2.2 Aufbau eines Magnetresonanz-Tomographen.....	23
2.2.3 Medizinische Bildgebung in der Magnetresonanz-Tomographie.....	28
2.3 Herzuntersuchung am MRT.....	32
2.3.1 Derzeitiges Auswertungsverfahren der aquirierten Schnittbilder.....	34
2.3.2 Verbesserung durch automatischen Konturfundung in den Bilddaten.....	34
<b>3 Grundlagen der Softwareentwicklung.....</b>	<b>35</b>
3.1 Lebenszyklusmodell von Software.....	35
3.2 Beschreibung einzelner Phasen des Softwarelebenszyklus.....	36
3.3 Vorgehensmodelle bei der Softwareentwicklung.....	40
3.4 Anforderungen an die Software-Qualität.....	45
<b>4 Entwurf und Design der Argus-Algorithmen und des Inline Cardio Moduls.....</b>	<b>48</b>
4.1 Algorithmus zur Erkennung der linken Herzkammer.....	48
4.2 Das Inline Cardio Modul (ICM).....	50

4.3 Integration des ICM in die MR Bildverarbeitung.....	52
<b>5 Implementierung der Segmentierungsalgorithmen und des ICM.....</b>	<b>54</b>
5.1 Algorithmen für die Segmentierung.....	54
5.1.1 Anpassungen der Segmentierungsalgorithmen.....	54
5.1.2 Plattformabhängige Besonderheiten.....	57
5.1.3 Erkennung und Vererbung von Geweberegionen.....	60
5.2 Einbindung des ICM in die MR Herzbild-Nachverarbeitung.....	61
5.2.1 Integration in das bestehende Softwaresystem.....	62
5.2.2 Funktionen des ICM.....	65
<b>6 Test und Robustheitsanalyse.....</b>	<b>68</b>
6.1 Test des Software Moduls.....	68
6.2 Unit-Test an einer Simulationsanlage.....	70
6.3 Integrations-Test am MRT.....	72
6.4 System-Test.....	74
6.5 Wissenschaftliche Vergleichsdaten.....	74
6.6 Beurteilung der Qualität des ICM nach ISO 9126.....	77
6.7 Zusammenfassung.....	79
<b>7 Fazit und Ausblick.....</b>	<b>81</b>
<b>Literaturverzeichnis.....</b>	<b>84</b>
<b>Anhang A - Designspezifikation ICM.....</b>	<b>87</b>
<b>Anhang B - Makefile.....</b>	<b>89</b>
<b>Anhang C - Software für abs() / fabs() Funktion.....</b>	<b>92</b>
<b>Anhang D - Inline Cardio Software-Modul.....</b>	<b>93</b>
<b>Anhang E - Metrik Lines of Code.....</b>	<b>99</b>
<b>Selbständigkeitserklärung.....</b>	<b>101</b>

## Abkürzungsverzeichnis

AG	Aktien Gesellschaft
AV	Atrioventrikular
CT	Computertomographie
CD-R	Compact Disc Rewritable
DB	Datenbank
DLL	Dynamic Link Library
ED	Enddiastolisch
EDV	Enddiastolisches Volumen
EF	Ejection Fraction
ES	Endsystolisch
ESV	Endsystolisches Volumen
GB	GigaByte
GUI	Graphic User Interface
HF	Hochfrequenz
ICM	Inline Cardio Modul
ISO	International Organization for Standardization
LOC	Lines of Code
MR	Magnetresonanz
MRT	Magnetresonanz-Tomograph
MTA	Medizinisch Technische(r) Assistent(in)
NMR	Nuclear Magnetresonanz
PC	Personal Computer
PET	Positronen Emissions Tomographie
RAM	Random Access Memory
SE	Software Entwicklung
SV	Stroke Volumen
SCR	Siemens Corporate Research
SCSI	Small Computer System Interface

## Abbildungsverzeichnis

Abbildung 2.1: Das Menschliche Herz (Querschnitt).....	16
Abbildung 2.2: Der Herzzyklus.....	18
Abbildung 2.3: Relaxation des Wasserstoffkerns.....	21
Abbildung 2.4: Aufbau der Längsmagnetisierung (T1 Relaxation).....	22
Abbildung 2.5: Abbau der Quermagnetisierung (T2 Relaxation).....	22
Abbildung 2.6: Konstanten für T1 und T2 Relaxation.....	23
Abbildung 2.7: Offener MRT (Concerto).....	24
Abbildung 2.8: MRT mit supraleitendem Magneten (Magnetom Symphony).....	25
Abbildung 2.9: Veränderung des Magnetfeldes durch den magnetischen Feldgradient.	25
Abbildung 2.10: Steuersoftware an einem Magnetresonanz-Tomographen.....	28
Abbildung 2.11: Bestimmung der Schichtposition.....	29
Abbildung 2.12: Schräge Schichten.....	30
Abbildung 2.13: Bestimmung der Schichtdicke.....	30
Abbildung 2.14: Einzelne Bildzeile mit verschiedenen Grau- Intensitäten.....	31
Abbildung 2.15: Rohdaten-Matrix, verkürzt auf 8x8 Pixel.....	31
Abbildung 2.16: 2-Kammer-Blick.....	32
Abbildung 2.17: 4-Kammer-Blick.....	33
Abbildung 2.18: Kurzachsen-Blick.....	33
Abbildung 3.1: Prototyping nach Marciniak 1994 [9].....	41
Abbildung 3.2: Wasserfallmodell nach Royce (1970).....	42
Abbildung 3.3: V-Modell nach Boehm [12].....	43
Abbildung 3.4: Übersicht über den Prozess der Softwareentwicklung.....	44
Abbildung 3.5: Qualitative Anforderungen bei der Softwareentwicklung [10].....	46
Abbildung 4.1: Abgrenzung der Linken Herzkammer vom Myokard.....	50
Abbildung 4.2: UML Diagramm für das ICM.....	52
Abbildung 5.1: Aufbau eines make-Files für ein C++ Projekt.....	56
Abbildung 5.2: Falsche Kontur bei der Segmentierung unter Linux.....	57

Abbildung 5.3: if(...) Abfrage nach Fortran-to-C Konvertierung.....	59
Abbildung 5.4: Vergleich der Funktionen abs( ) und fabs( ).....	59
Abbildung 5.5: if (...) Abfrage modifiziert für eine Verwendung unter Linux.....	60
Abbildung 5.6: Kette von mehreren Modulen einschließlich des Cardio-Moduls.....	62
Abbildung 5.7: Konturen um die Linke Kammer und Kammermuskel.....	64
Abbildung 5.8: Ergebnissbild der vollautomatischen Auswertung.....	65
Abbildung 5.9: Schichtbilder angeordnet nach Zeitlicher Auflösung (X-Achse) und entlang der Schichten durch das Herz (Y-Achse).....	66
Abbildung 6.1: Gegenüberstellung von Entwicklungs- und Testphasen bei Siemens Medical.....	68
Abbildung 6.2: Simulationsanlage Teil 1.....	71
Abbildung 6.3: Simulationsanlage Teil 2.....	71
Abbildung 6.4: Erzeugtes Rauschbild während einer Messung an einer Simulationsanlage.....	71
Abbildung 6.5: Medizinische Kenngrößen von männl. Probanden (vgl. auch [3] ).....	75
Abbildung 6.6: Medizinische Kenngrößen von weibl. Probanden (vgl. auch [3]).....	76
Abbildung 6.7: Ergebnisse der automatischen Auswertung im direkten Vergleich mit wissenschaftlichen Daten von [3].....	76
Abbildung 6.8: Modell für die Entwicklung und dem Test von Software bei Siemens.	79

## **Tabellenverzeichnis**

Tabelle 5.1: Neue Verzeichnisstruktur der zugeliferten Segmentierungsalgorithmen..	55
Tabelle 5.2: Verschiedene Schreibweisen von Datentypen unter Windows und Linux.	57
Tabelle 5.3: Test der zugeliferten Algorithmen auf verschiedenen Betriebssystemen..	58
Tabelle 5.4: Struktur des Software-Moduls.....	63
Tabelle 6.1: Reihenfolge der einzelnen aquirierten Schichten bei der Verarbeitung.....	73
Tabelle 6.2: Ergebnisse einzelner Qualitätsmerkmale von ISO 9126.....	78

# 1 Einleitung

## 1.1 Motivation

Seit ihrem Beginn unterliegen die Softwartechnik und ihre Produkte einem rasanten Fortschritt und Wandel. Ein Aspekt dieser Veränderung sind Programme, die vor Jahrzehnten noch manuell gesteuert wurden und dem Benutzer eine Vielzahl von Eingaben und Entscheidungen abforderten und heute weitestgehend voll-automatisch ablaufen. Dieser Trend zur Automatisierung zeigt sich unter anderem in der Datenauswertung, wo sich die Vorteile nicht nur auf eine enorme Leistungssteigerung und Benutzerunabhängigkeit, sondern auch auf eine mögliche Fehlerreduktion auswirken.

In der medizinischen Bildverarbeitung hat man dieses Potential erkannt und versucht nun, es sich zunutze zu machen. Durch den rasanten Fortschritt bei den bildgebenden Systemen, die immer bessere und schärfere Bilder von anatomischen Strukturen erzeugen, stehen der nachfolgenden Auswerte-Software ständig verbesserte Eingangsdaten zur Verfügung.

Diese Tatsachen sind auch der Hintergrund und die Motivation für die vorliegende Arbeit. Die Vermessung der linken Herzkammer stellt ein etabliertes Verfahren zur Bestimmung der Herz-Leistungsfähigkeit bzw. zur Erkennung von Koronarkrankheiten beim Menschen dar. Die Herzbilder für die Vermessung können mit verschiedenen bildgebenden Systemen wie z.B. Ultraschall (Sonographie), Nuklearmedizin, Röntgen (Koronarangiographie), Computer-Tomographie und Kernspinresonanz-Tomographie erzeugt werden, wobei aus ärztlicher Sicht die Kernspinresonanz-Tomographie, auch bekannt als Magnetresonanz-Tomographie, entscheidende Vorteile in der zeitlichen Auflösung besitzt.

Bei medizinischen Bildern der Magnetresonanz-Tomographie wird derzeit die Vermessung der linken Herzkammer manuell oder semi-automatisch durchgeführt. Dieses Vorgehen ist relativ zeitaufwendig (dadurch letztendlich auch teuer),

benutzerabhängig und fehleranfällig.

Durch ständig weiterentwickelte medizinische Geräte, Technologien und die damit verbundene Erzeugung von Bildern mit hoher zeitlicher und räumlicher Auflösung, besteht die Möglichkeit, das bisherige manuelle Auswertungsverfahren durch eine voll-automatische Form zu ersetzen. Damit verlieren die oben genannten Nachteile ihre Relevanz. Das Verfahren wird von den Anwendern akzeptiert und kann mit einem verbesserten Patientennutzen eingesetzt werden.

## **1.2 Wirtschaftliche Aspekte**

Die Betrachtung des im Kapitel 1.1 beschriebenen Problems von einem wirtschaftlichen Standpunkt ist für diese Arbeit nicht unerheblich und zeigt weitere Möglichkeiten für die Motivation.

Die Untersuchung eines Patienten und die anschließende Auswertung der Herzbilder durch einen Arzt oder Medizinisch Technischen Assistenten (im folgenden als MTA bezeichnet) erfordert in erster Linie eine geeignete Schulung. Jedoch ist diese Maßnahme sowohl mit finanziellen Kosten als auch mit einem gewissen Zeitaufwand verbunden. In jedem wirtschaftlich arbeitenden Betrieb ist es das Ziel, viel Leistung mit wenig Aufwand zu erbringen um einen möglichst hohen Gewinn zu erwirtschaften. Im medizinischen Bereich kommt hierzu noch der Wunsch nach einer gesteigerten Diagnosesicherheit.

Das Ergebnis dieser Diplomarbeit soll dazu beitragen, dass zusätzliche Kosten für spezielle Kardio<sup>1</sup>-Schulungen bei MTAs durch den Einsatz eines voll-automatischen Prozesses optimiert und die Diagnosesicherheit verbessert wird.

---

<sup>1</sup> Kardio bezeichnet im medizinischen Sinn das Herz

### 1.3 Zielstellung

Das Ziel dieser Diplomarbeit ist der Entwurf und die Implementierung eines voll-automatischen Verfahrens zur Bestimmung der Herz-leistungsfähigkeit (Ejection Fraction) im Laufe einer Magnetresonanz (MR) Herzuntersuchung, um die bisher verwendeten manuellen Vorgehensweisen zu ersetzen. Hierfür soll ein Software-Modul entwickelt werden, welches die automatische Verarbeitung der aquirierten Bilder realisiert und basierend auf den ermittelten medizinischen Kenngrößen eine Auswertung zur Herz-Leistungsfähigkeit durchführt.

Ein weitere Bestandteil der Zielstellung ist die Implementierung der vorhandenen semi-automatischen Segmentierungsalgorithmen<sup>2</sup> der linken Herzkammer in eine eigenständige Bibliothek. Durch den bisherigen Einsatz nur unter Windows ist eine Portierung der Algorithmen in das Betriebssystem Linux zwingend notwendig und erfordert eine entsprechende Anpassung aller vorhandenen Klassen und Funktionen.

Nach dem Design und der erfolgreichen Implementierung sollen verschiedene Test- und Analyseverfahren Aufschluss über die Reproduzierbarkeit der Auswertungsergebnissen geben, sowie zu einer Verbesserung von Robustheit und Performance im Bereich der medizinischen Bildverarbeitung beitragen. Mit den Ergebnissen soll gezeigt werden, dass die entwickelte Software eine Qualitätsverbesserung darstellt und gefahrlos im Produktcode – und damit am Menschen – eingesetzt werden kann.

Für die Aufbereitung und Darstellung der berechneten medizinischen Kenngrößen ist eine geeignete und übersichtliche Form zu wählen. Es ist vor allem darauf zu achten, dass eine einfache Portierung der Ergebnisse in die Datenbank (DB) möglich ist.

---

<sup>2</sup> Die Erzeugung von inhaltlich zusammenhängenden Regionen durch Zusammenfassung benachbarter Pixel oder Voxel entsprechend einem bestimmten Homogenitätskriterium wird als Segmentierung bezeichnet.

## 1.4 Aufbau der Diplomarbeit

Die vorliegende Diplomarbeit beschäftigt sich mit der Ablösung eines bestehenden manuellen Verfahrens zur Bestimmung der Herz-Leistungsfähigkeit durch ein vollautomatisches Verfahren. Um einen Überblick über den Aufbau der Arbeit zu geben, sollen an dieser Stelle die einzelnen Kapitel kurz vorgestellt und beschrieben werden.

Zur Erläuterung und dem besseren Verständnis des verwendeten medizinischen Verfahrens werden im zweiten Kapitel die Grundlagen zur Anatomie des Herzens, besonders der Aufbau und die Funktion der linken Herzkammer, beschrieben und einige technische Hintergründe zur Bildgebung in der Magnetresonanztomographie dargelegt. Um die grundsätzlichen Probleme bei der Auswertung der gemessenen Bilder zu verdeutlichen, wird das derzeitige Vorgehen kurz beschrieben und die Möglichkeiten zur Verbesserung genannt.

Einige für die Softwareentwicklung benötigten Grundlagen werden im Anschluss an das zweite Kapitel vermittelt. Dabei werden besonders der Software-Lebenszyklus, die einzelnen Phasen des Lebenszyklus und verschiedene Modellformen beschrieben. Weiterhin findet die Überleitung zur Softwareentwicklung in der Siemens AG anhand eines erläuterten Modells statt. Das Kapitel schließt mit der Einführung des Qualitätsmodells und der Beschreibung von Softwarequalität ab.

Nachdem ein geeignetes Modell für die Entwicklung der Software ausgewählt wurde, beschäftigt sich das vierte Kapitel mit dem Design des zu entwickelnden Software-Moduls. In dieser Phase wird der Aufbau und die Funktionsweise diskutiert und eine Anpassung an die bereits vorhandenen Hard- und Softwareschnittstellen vorgenommen. Weiterhin wird im Design die Vorgehensweise bei der Verarbeitung der übergebenen Bilddaten und die Struktur der Software festgelegt. Mögliche Fehler bei der späteren Implementierung lassen sich durch ein sinnvolles und durchdachtes Design von vornherein ausschließen.

Nachdem mit dem vorangegangenen Kapitel die Planung des neuen Software-Moduls

abgeschlossen ist, folgt im fünften Kapitel die Implementierung. Dabei wird auf einen gravierenden Fehler bei der Konvertierung von Fortran Code zu C-Code eingegangen und eine Lösung basierend auf der entsprechenden Fehleranalyse vorgestellt. Im zweiten Teil des Kapitels findet die Implementierung des Software-Moduls für die automatische Auswertung statt.

In Kapitel 6 werden verschiedene Verfahren von angewendeten Software-Tests, basierend auf den Vorgaben des V-Modells, zur Überprüfung der korrekten Implementierung des Software-Moduls dargestellt. Das Kapitel enthält zudem noch wissenschaftliche Vergleichsdaten und die Einordnung der durch die automatische Auswertung gewonnenen Ergebnisse.

Abschließend wird im letzten Kapitel ein Fazit über die Diplomarbeit gezogen und ein Ausblick auf eine Weiterführung des Projekts gegeben.

## 2 Medizinische und Technische Grundlagen

Dieses Kapitel enthält spezielle Grundlagen über die Anatomie und Funktion des menschlichen Herzens sowie einen Überblick über ein bildgebendes Verfahren der Magnetresonanztomographie. Zur Vervollständigung der Grundlagen wird ein derzeitiges Verfahren zur Auswertung von Herzschnittbildern vorgestellt und die Problematik bei diesem Verfahren aufgezeigt.

### 2.1 Das menschliche Herz

Das Herz bildet das Zentrum des Herz-Kreislauf-Systems (Kardiovaskuläres System). Es stellt die zentrale Pumpe für alle Transportvorgänge in den Blutgefäßen dar und versorgt den ganzen Körper mit Sauerstoff und Nährstoffen. Gleichzeitig findet über die Blutgefäße der Abtransport von Kohlendioxid statt.

Im Normalfall ist das Herz etwa so groß, wie die geschlossene Faust seines Trägers. Es wiegt durchschnittlich 300 Gramm und liegt zwischen den beiden Lungenflügeln im Mittelfellraum (Mediastinum). Das Herz wird vorn vom Brustbein (Sternum) und hinten von der Speiseröhre und der Aorta<sup>3</sup> begrenzt. Unten liegt das Herz auf dem Zwerchfell auf. Zwei Drittel des Herzens befinden sich in der linken Brustseite, ein Drittel auf der rechten Seite (vgl. auch [5]).

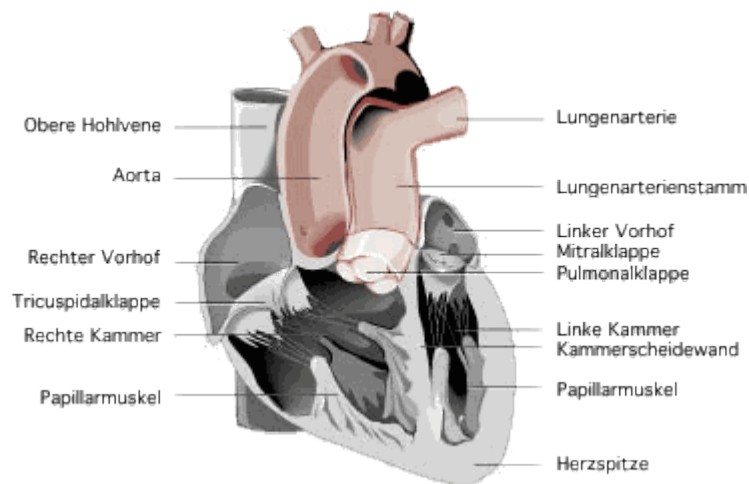
Abbildung 2.1 zeigt einen Schnitt durch das Herz entlang der Längsachse. Das Herz besteht aus dem linken und rechten Vorhof, linker und rechter Kammer sowie den Atrioventrikularklappen<sup>4</sup> (AV-Klappen), Sehnenfäden und Papillarmuskeln<sup>5</sup>. Weiterhin erkennbar sind die herznahen Gefäße wie Hohlvene, Lungenschlagader, Aorta, Lungenvenen und Lungenarterien.

---

3 Die Aorta ist die Hauptschlagader im menschlichen Körper.

4 Herzklappen zwischen Vorkammer und Kammer des Herzens.

5 Der Papillarmuskel ist eine Vorstülpung des Herzmuskels im Herzinnere, der über Sehnenfäden mit zwei der Segel einer Herzklappe zwischen Vorhof (Atrium) und Herzkammer (Ventrikel) verbunden ist.



*Abbildung 2.1: Das Menschliche Herz (Querschnitt)*

### **2.1.1 Aufbau und Funktion des Herzens**

Der Blutkreislauf bildet ein in sich geschlossenes System, bei dem das Blut über ein aus Arterien und Venen bestehendes Gefäßsystem ständig zu allen Punkten des Körpers hin und rücktransportiert wird. Im Mittelpunkt dieses Transportsystems steht das Herz als kombiniertes Druck-Saug-Organ, das für die kontinuierliche Strömung und den Rücktransport des Blutes sorgt. Es ist durch die Herzscheidewand (Septum) in eine rechte und eine linke Herzhälfte aufgeteilt. Jede Herzhälfte verfügt über eine Herzkammer (Ventrikel) und einen Vorhof (Atrium). Damit das Blut beim Pumpvorgang des Herzens in die richtige Richtung fließt, sind die Ein- und Ausgänge beider Kammern mit entsprechenden Klappen versehen, die ein Zurückfließen des Blutes verhindern und es nur in eine Richtung durchlassen. Die Klappe zwischen Vorhof und Kammer wird links Mitralklappe und rechts Trikuspidalklappe genannt. Diese Klappen sind zur Stabilisierung über Sehnenfäden mit den Papillarmuskeln innerhalb ihrer jeweiligen Kammer verbunden. Die Klappen zwischen den Kammern und den großen Schlagadern werden im Allgemeinen als Taschenklappen bezeichnet. Auf der rechten Seite befindet sich die so genannte Pulmonalklappe und auf der linken Seite (zwischen linker Kammer und Aorta) die Aortenklappe.

Die eigentliche Leistung des Herzens wird überwiegend von der Muskelschicht des Herzens (Myokard) erbracht (vgl. auch [5]).

Im Herz befindet sich ein spezielles Reizbildungs- und Reizleitungssystem, in dem die elektrische Erregung entsteht und sich ausbreitet. Dadurch wird die Kontraktion des Herzens überhaupt erst möglich. Die Erregung beginnt im Sinusknoten, einem bestimmten Bereich im rechten Vorhof des Herzens, welcher der primäre elektrische Taktgeber der Herzaktion ist. Der Sinusknoten treibt das Herz mit einer bestimmten Frequenz an. Vom Sinusknoten gelangt der elektrische Impuls über die Muskulatur der beiden Vorhöfe des Herzens auf den AV-Knoten und breitet sich von dort über das Reizleitungssystem auf die Muskulatur der beiden Herzkammern aus (vgl. auch [16]).

### **Der Herzzyklus**

Während der Herzzyklen wird das Blut in regelmäßigen Abständen in den Körper- und Lungenkreislauf gepumpt. Man unterscheidet hierbei die Phase der Kontraktion (Systole<sup>6</sup>) und die Phase der Erschlaffung (Diastole<sup>7</sup>). Beide Phasen werden abwechselnd nacheinander durchlaufen.

### **Systole (Kontraktionsphase )**

Die Systole ist die Kontraktionsphase des Herzens. Sie unterteilt sich in folgende zwei Teilphasen:

- Isovolumetrische Anspannungszeit
- Austreibungszeit

In der isovolumetrischen Anspannungszeit spannt sich das Herz an, ohne das Blut auszutreiben. In der Austreibungszeit kommt es zum Auswurf des Blutvolumens sowohl von der rechten Kammer in die Lunge (Pulmonalarterie) als auch von der linken Kammer in die Aorta.

---

6 Die Systole ist die Anspannungs- und Auswurfphase des Herzens.

7 Bei der Diastole handelt es sich um die Entspannungs- und Füllungsphase.

### Diastole (Erschlaffungsphase)

Die Diastole ist die Erschlaffungsphase des Herzens. Sie wird ebenfalls unterteilt in zwei Teilphasen:

- isovolumetrische Entspannungszeit
- Füllungszeit

Während der isovolumetrischen Entspannungszeit entkrampft sich der Herzmuskel, ohne sich mit Blut zu füllen. In der anschließenden Füllungszeit kommt es zur Füllung beider Kammern mit Blut aus den Vorhöfen, wobei die letzte Phase dieser Füllung durch eine aktive Kontraktion der Vorhöfe zustande kommt. Die Füllung der Vorhöfe findet während der Systole durch eine Sogwirkung statt. Der rechte Vorhof füllt sich mit Blut aus oberer und unterer Hohlvene, der linke Vorhof mit Blut aus der Lungenvene (siehe dazu Abbildung 2.2).

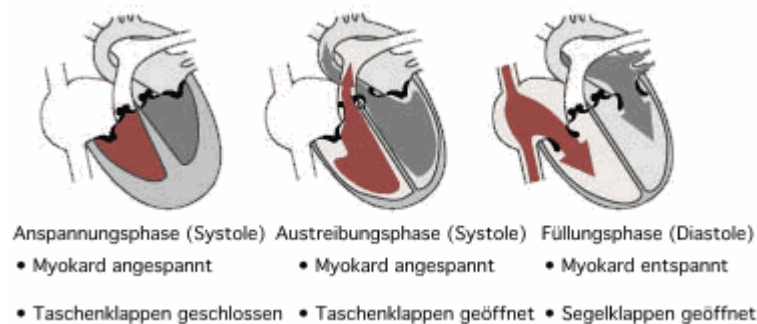


Abbildung 2.2: Der Herzzyklus

### 2.1.2 Herzdiagnose durch Analyse des linken Ventrikels

Die Beschreibung der Herz-Leistungsfähigkeit basiert häufig auf einer Reihe von Kennzahlen, die auf den Blutvolumina des linken und rechten Ventrikels während Systole und Diastole beruhen. Für eine Bestimmung der Kennzahlen wird ein kompletter Herzzyklus mit Hilfe eines radiologischen bildgebenden Verfahren aufgenommen und anschließend ausgewertet. Hier können Computer-Tomographie (im folgenden kurz: CT) oder Magnetresonanztomographie zum Einsatz kommen. Für diese Arbeit wird nur der linke Ventrikel betrachtet.

Aus den akquirierten Bilddaten (siehe dazu Abbildung 5.9, S.66) kann zunächst das enddiastolische Blutvolumen (im folgenden kurz: EDV) und das endsystolische Blutvolumen (im folgenden kurz: ESV) der linken Herzkammer ermittelt werden.

Das Schlagvolumen (engl.: Stroke Volume, im folgenden kurz: SV) entspricht der Blutmenge, die pro Herzzyklus aus der linken Kammer gepumpt wird. Es wird wie folgt ermittelt:

$$SV = EDV - ESV$$

Um die Leistungsfähigkeit des Herzens besser bewerten zu können, muss noch eine weitere wichtige Kenngröße berechnet werden. Es handelt sich hierbei um die Ejection Fraction (im folgenden kurz: EF) genannt. Sie entspricht dem Anteil des enddiastolischen Blutvolumens, das von der linken Herzkammer während der Systole ausgeworfen wird und errechnet sich über folgende Formel:

$$EF = \frac{EDV - ESV}{EDV} * 100 \%$$

Für ein gesundes Herz werden in den meisten Fällen für die EF Werte größer als 50% angenommen und liegt im Schnitt bei 65% bis 67% (vgl. auch [1] und [3]).

## 2.2 Die Magnetresonanz-Tomographie

Mit der Magnetresonanz-Tomographie (vgl. auch [14]) steht neben der Positronen-Emissions-Tomographie (PET) ein weiteres nichtinvasives<sup>8</sup> bildgebendes Verfahren in der Medizin zur Verfügung.

Der Hauptanwendungsbereich der Magnetresonanz-Tomographie ist die Darstellung von Gewebestrukturen in einer Schnittbildserie durch den Körper.

---

<sup>8</sup> Bezeichnet eine Prozedur, bei denen Geräte oder Katheder entweder gar nicht (nichtinvasiv) oder in geringerem Maße als üblich (minimal-invasiv) in den Körper eindringen

Dabei zeichnet sich die Bildgebung vor allem durch drei wesentliche Vorteile aus:

- Es lässt sich ein hervorragender Kontrast bei Weichteilen sowie eine hohe Bildauflösung erreichen. Der Weichteilkontrast ist besonders bei der Bilderzeugung von Organen und Gewebe von Vorteil.
- Eine Darstellung der Aufnahmen ist in mehreren Schichten möglich und unterstützt eine schräge Schnittführung.
- Während der Untersuchung wird keine ionisierende Strahlung bzw. keine radioaktiven Tracer<sup>9</sup> benötigt.

Die Magnetresonanz-Tomographie gilt damit als eines der wichtigsten medizinischen Verfahren mit einer sehr hohen diagnostischen Aussagekraft und hat sich im klinischen Umfeld zu einem Routineverfahren in der Schnittbilddiagnostik entwickelt.

### 2.2.1 Theoretische Grundlagen der Magnetresonanz

Ein Atomkern mit einer ungerade Anzahl an Protonen und Neutronen besitzt einen Kernspin und damit eine Rotation um die eigene Achse. Atomkerne mit einer geraden Anzahl von Protonen und Neutronen verhalten sich magnetisch neutral.

Der Kernspin ist die entscheidende Ursache für die Fähigkeit zur Magnetresonanz, weil ein Atomkern mit Spin immer magnetisch ist. Ungefähr zwei Drittel der in der Natur vorkommenden Atomkerne besitzen einen Kernspin. Sie alle könnten rein theoretisch für die Magnetresonanz-Tomographie genutzt werden.(vgl. auch [6])

Der Wasserstoffkern<sup>10</sup> eignet sich für die Magnetresonanz-Tomographie am besten, weil er gebunden in Wassermolekülen sehr häufig im menschlichen Körper vorkommt. Durch ein statisch angelegtes Magnetfeld  $B_0$  beginnen die Wasserstoffkerne um die Richtung des Magnetfeldes zu präzedieren – oder einfacher gesagt „zu kreiseln“. Die

---

<sup>9</sup> Tracer sind Stoffe, die dem Patienten vor der Untersuchung gespritzt werden. Diese Stoffe reichern sich an den besonders aktiven Stellen im Körper an. Mit einer Spezialkamera wird diese Strahlung gemessen (vgl. auch [15])

<sup>10</sup> Der Wasserstoffkern besteht aus einem einzelnen Proton.

meisten Kerne richten sich dabei parallel zum angelegten Magnetfeld aus. Zusätzlich dazu wird ein Hochfrequenzfeld eingestrahlt, welches senkrecht zu dem statischen Magnetfeld  $B_0$  steht. Die Frequenz des eingestrahlten Feldes stimmt dabei mit der Präzessionsfrequenz der einzelnen Wasserstoffkerne übereinstimmt. Auf diese Weise befindet es sich mit dem statischen Magnetfeld in Resonanz. Die Präzessionsfrequenz  $\omega_0$  (auch Resonanzfrequenz oder Lamorfrequenz genannt) wird mit dem Lamortheorem berechnet:

$$\omega_0 = \gamma * |B_0|$$

Dabei ist  $\gamma$  das gyromagnetische Verhältnis der Atomkerne und  $B_0$  bezeichnet die Feldstärke des statischen Magnetfeldes.

Nach Abschalten des zusätzlich eingestrahlten Hochfrequenzfeldes geraten die einzelnen Wasserstoffkerne zunächst wieder außer Phase. Die Quermagnetisierung  $M_{xy}$  geht dadurch auf 0 zurück. Das Abklingen der Quermagnetisierung resultiert aus der gegenseitigen Beeinflussung der Spins (Spin-Spin-Relaxation). Die Längsmagnetisierung  $M_z$  hingegen nimmt wieder zu und kehrt so in ihre Ausgangslage zurück. Die Kerne nehmen wieder ihre ursprüngliche parallele, damit energetisch günstigere, Ausrichtung ein. Die Rückkehr der Wasserstoffkerne in ihren ursprünglichen Zustand wird als Relaxation bezeichnet. (siehe dazu Abbildung 2.3)

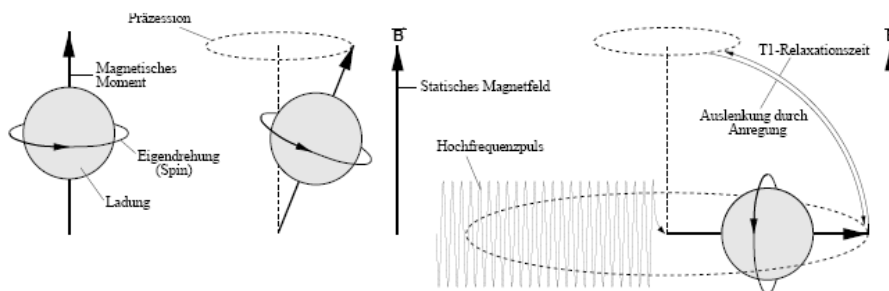


Abbildung 2.3: Relaxation des Wasserstoffkerns

Die Abnahme der Quermagnetisierung und die Zunahme der Längsmagnetisierung kann von einer entsprechenden Empfängerspule gemessen werden. Die Zeit zur Angleichung

der Längsmagnetisierung zu 63% an  $M_z$  wird als  $T_1$ -Relaxation bezeichnet. Nach 5 mal  $T_1$  hat sich die Längsmagnetisierung wieder vollständig aufgebaut (siehe Abbildung 2.4).

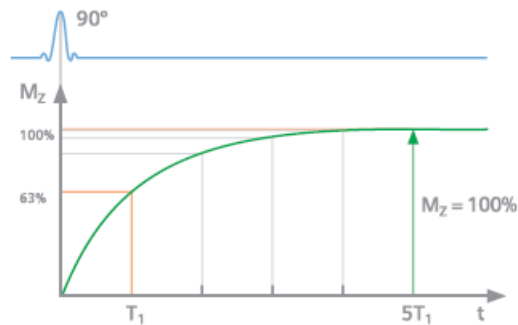


Abbildung 2.4: Aufbau der Längsmagnetisierung ( $T_1$  Relaxation)

Hingegen wird die Zeit, welche die Quermagnetisierung zum Abklingen auf 37% benötigt, als  $T_2$ -Relaxation bezeichnet. Nach 5 mal  $T_2$  ist die Quermagnetisierung vollständig verschwunden (siehe Abbildung 2.5).

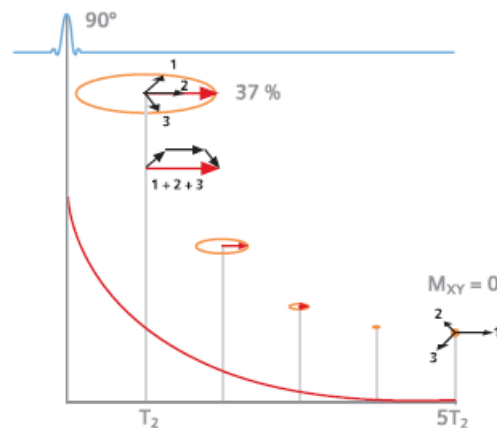


Abbildung 2.5: Abbau der Quermagnetisierung ( $T_2$  Relaxation)

Durch die unterschiedlichen Relaxationszeiten können die verschiedenen Gewebearten bestimmt werden, was die Basis für einen guten Bildkontrast bei MR darstellt. In Abbildung 2.6 sind die  $T_1$ - und  $T_2$ -Konstanten für die Relaxationszeit von verschiedenem Gewebe abgebildet. Dabei zeigt sich auch, dass die  $T_1$ -Konstante

abhängig von der benutzten Feldstärke ist, die  $T_2$ -Konstante weitestgehend nicht.

$T_1$ -Konstanten (in ms)				$T_2$ -Konstanten (in ms)	
	0,2 Tesla	1,0 Tesla	1,5 Tesla		
Fett		240		Fett	84
Muskel	370	730	863	Muskel	47
Weißer Substanz	388	680	783	Weißer Substanz	92
Graue Substanz	492	809	917	Graue Substanz	101
Liquor	1400	2500	3000	Liquor	1400

Abbildung 2.6: Konstanten für  $T_1$  und  $T_2$  Relaxation

Durch die unterschiedlichen Relaxationszeiten bei verschiedenen Gewebetypen kann die MR-Bildgebung den Unterschied als Bildkontrast darstellen. Diese Tatsache kann für die Diagnose verwendet werden, um gesundes Gewebe von pathologischem zu unterscheiden. Pathologisches Gewebe besitzt eine andere Wasserkonzentration als das umliegende gesunde Gewebe.

Weiterführende Grundlagen zur Kernmagnetischen Resonanz können unter [20] und [7] nachgelesen werden.

## 2.2.2 Aufbau eines Magnetresonanz-Tomographen

Ein Magnetresonanz-Tomograph (MRT) besteht aus drei wesentlichen Kernelementen

1. dem Hauptfeld – Magneten
2. dem Gradientensystem
3. und dem Hochfrequenzsystem

Zu jedem MRT gehört der Vollständigkeit halber noch ein Computersystem, welches zusammen mit den aufgezählten Subsystemen im Folgenden näher erläutert werden soll.

### Der Hauptfeld-Magnet

Für die MR - Bildgebung wird ein homogenes Magnetfeld benötigt, welches durch einen starken Magneten erzeugt wird. Dabei unterscheidet man zwischen zwei Typen von Magneten, den Permanentmagneten und den supraleitenden Magneten.

Permanentmagnete besitzen Feldstärken<sup>11</sup> von 0,01 bis 0,5 Tesla<sup>12</sup> [6] und werden vorzugsweise auf Grund ihrer Form, die einem „C“ ähnelt, in so genannten offenen Systemen wie zum Beispiel einem Magnetom Concerto der Firma Siemens eingesetzt (siehe Abbildung 2.7). Permanentmagnete benötigen eine stabile Betriebstemperatur, damit ein ausreichend hohes homogenes Feld erzeugt werden kann.



Abbildung 2.7: Offener MRT (Concerto)

Ein anderer Typ sind die supraleitenden Magneten, welche Feldstärken von 0,5 bis 3,0 Tesla erreichen, im Bereich von Forschung und Entwicklung sind Anlagen mit Feldstärken von bis zu 7 Tesla und mehr im Einsatz (vgl. auch [6]). Bei einem supraleitenden Magneten handelt es sich um einen Elektromagneten, bei dem das starke Magnetfeld durch große, stromdurchflossene Spulen erzeugt wird. Solche supraleitenden Magneten findet man vorwiegend in röhrenförmigen Systemen, wie einem Magnetom Symphony oder Avanto von Siemens (siehe Abbildung 2.8). Hier besteht der Leitdraht der Spule aus einer in Kupfer eingebetteten tiefgekühlten Niob-Titan-Legierung. Um die Spule ausreichend kühlen zu können, wird flüssiges Helium verwendet. Ein supraleitender Magnet muss nur einmal bis zur gewünschten Feldstärke mit Strom geladen werden. Dieser fließt dann auf Grund der widerstandsfreien Leitung über einen sehr langen Zeitraum mit konstanter Stromstärke (teilweise > 400 Ampere).

---

<sup>11</sup> Die Feldstärke beschreibt die Magnetische Induktion eines Magneten

<sup>12</sup> Tesla ist die Einheit für die Feldstärke (1 Tesla ist etwa 20000 mal so stark wie das Magnetfeld der Erde)



Abbildung 2.8: MRT mit supraleitendem Magneten (Magnetom Symphony)

### Das Gradientensystem

Für eine Beschreibung des Gradientensystems müssen wir zuerst die Frage klären, was überhaupt ein Gradient ist. Ein Magnetischer Feldgradient ist eine Änderung des Magnetfeldes in eine bestimmte Richtung – eine lineare Zu- bzw. Abnahme. Gradienten werden verwendet, um direkt nach einem Hochfrequenz-Puls das Magnetfeld so zu verändern, dass es in der einen Richtung kleiner und in der Gegenrichtung größer wird (siehe auch Abbildung 2.9). Damit „kreiseln“ die Spins im schwachen Magnetfeld langsamer und im starken schneller.

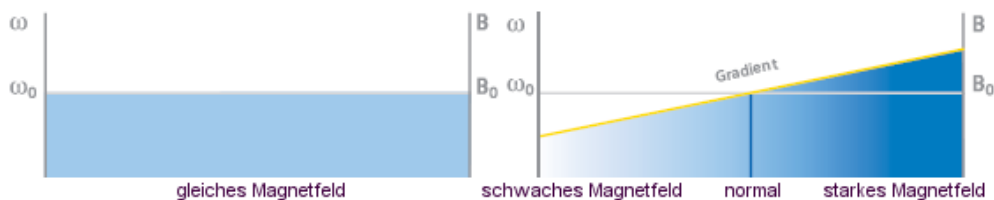


Abbildung 2.9: Veränderung des Magnetfeldes durch den magnetischen Feldgradient

Ein MR - System verfügt über drei Gradienten-Spulenordnungen. Jede Spulenordnung ist für eine der drei Raumrichtungen (X, Y und Z) zuständig. Die

Gradientenspulen erzeugen kein permanentes Magnetfeld und werden auch nur kurzzeitig während einer Untersuchung zugeschaltet. Die individuelle Leistungsfähigkeit eines Gradientensystems wird durch die maximale Leistung (Amplitude) und minimale Anstiegszeit (Zeit in der die Leistung erreicht wird) charakterisiert. Aus den Werten der Leistung und Anstiegszeit lässt sich die Anstiegsrate berechnen, welche zum Vergleich der Leistungsfähigkeit bei Gradientensystemen verwendet wird (vgl. auch [6]).

### **Das Hochfrequenzsystem**

Hohe Frequenzen, die zum Beispiel bei elektromagnetischen Feldern auftreten, werden als Hochfrequenz (im folgenden kurz: HF) bezeichnet.

Durch gepulste magnetische HF-Felder werden die Kernspins des Körpergewebes angeregt. Diese HF-Pulse werden gesendet und das von den Spins abgegebene MR - Signal wird anschließend empfangen. Für diesen Vorgang kommt das HF-System des MRT zum Einsatz.

Ein solches HF-System besteht in der Regel aus den HF-Antennen (Spulen oder auch Resonatoren genannt), einem HF-Sendeverstärker und einem HF-Empfangsverstärker. Die HF-Antennen werden zum Senden und Empfangen von Resonanzanregungen verwendet und können in ihrer Größe und Form unterschiedlich sein. Der HF-Sendeverstärker wird in zwei Stufen unterteilt, den Vorverstärker, welcher die Signale erzeugt und den Sendeverstärker, der die Signale auf die erforderliche Stärke bringt.

Am Ende wird das bei seinem Empfang sehr schwache MR - Signal in einem rauscharmen Verstärker entsprechend verstärkt, bevor es digitalisiert und weiterverarbeitet werden kann. Je stärker und klarer ein Signal von der Spule empfangen werden kann, umso besser ist das Signal.

## Das Computersystem

Die stetig fortschreitende Entwicklung der Rechnerleistung gestattet heute die Unterbringung der Bild- und Steuerrechner in einem bzw. mehreren PC's im Schaltraum. Das Computersystem besteht aus einem Bildrechner, einem Steuerrechner und der Auswertungssoftware, welche auf dem Steuerrechner installiert ist.

Der Bildrechner ist für die Rekonstruktion der verschiedenen MR – Bilder zuständig. Die Rekonstruktion erfolgt mit Hilfe der zweidimensionalen Fourier-Transformation<sup>13</sup>. Ein moderner Bildrechner kann pro Sekunde etwa 100 Bilder mit einer Matrix von 256 x 256 Bildpunkten rekonstruieren (vgl. auch [6]).

Um die Leistungsfähigkeit des Rechners zusätzlich zu erhöhen, wird Hauptspeicher (RAM<sup>14</sup>) im Gigabyte-Bereich und schnelle SCSI<sup>15</sup>-Festplatten mit einer hohen Kapazität für die Speicherung von Roh- und Bilddaten verwendet.

Der Steuerrechner kontrolliert und überwacht das gesamte System. Dazu gehört die Dateneingabe, Messablauf und Bilddarstellung. Zur Bewältigung der oft parallelen Aufgaben werden mehrere Prozessoren der neuesten Generation verbaut. Die Fähigkeit zum Multitasking<sup>16</sup> ist dabei unerlässlich und wird zum Beispiel vom Anwender benutzt, um bereits während einer laufenden Messung schon vorliegende Ergebnisse zu betrachten.

Die Schnittstelle zwischen MR – System und Anwender ist die Steuer- und Auswertungssoftware (siehe Abbildung 2.10).

---

13 Die Fourier-Transformation ist eine Integral-Transformation, die einer Funktion eine andere Funktion (ihre Fourier-Transformierte) zuordnet. Sie ordnet einer Struktur oder einem Signal die einzelnen Frequenzen zu, aus denen es sich zusammensetzt.

14 Random Access Memory : Speicher mit wahlfreiem Zugriff, wobei jede Speicherzelle über die Speicheradresse direkt angesprochen werden kann

15 Small Computer System Interface, ist eine parallele Schnittstelle zur Datentübertragung zwischen Geräten auf einem Computer-Bus

16 Fähigkeit eines Betriebssystems, mehrere Aufgaben (Tasks) nebenläufig auszuführen.

Sie ist grundsätzlich modular aufgebaut und enthält:

- die Patientenverwaltung
- die Organisation und Steuerung des Messsystems
- die Messdatenerfassung und –verarbeitung
- die Darstellung der Bilddaten
- eine Möglichkeit zur Bildnachverarbeitung
- und Funktionen zur Sicherung und Dokumentation der Bilddaten.



*Abbildung 2.10: Steuersoftware an einem Magnetresonanz-Tomographen*

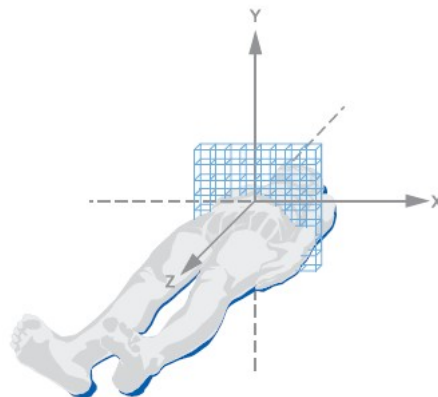
Die Steuersoftware übernimmt die Organisation und Steuerung des Messsystems und bietet dem Anwender die Möglichkeit, genau für die Untersuchung abgestimmte Messprotokolle auszuwählen. Diese Protokolle können bei Bedarf modifiziert und abgespeichert werden, um bei einer späteren Verwendung wieder zum Einsatz zu kommen.

### **2.2.3 Medizinische Bildgebung in der Magnetresonanz-Tomographie**

In der medizinischen Bildgebung werden Schnittbilder aus verschiedenen Bereichen des menschlichen Körpers benötigt. Um ein Schnittbild messen zu können, muss zuerst die entsprechende Schichtposition festgelegt werden. Die drei möglichen Schichtpositionen

können durch „Schalten“ des entsprechenden Gradienten erreicht werden (vgl. dazu „Das Gradientensystem“ Kapitel 2.2.2).

An der Stelle, an der das Magnetfeld nach seiner Veränderung durch den Gradienten noch immer die normale Stärke hat, befindet sich die zu messende Schicht. Für eine sagittale Schicht wird der x – Gradient benötigt, eine koronare Schicht lässt sich durch den y – Gradient und eine transversale Schicht durch den z – Gradient bestimmen. Die zu messende Schicht befindet sich immer senkrecht zum Gradienten. Abbildung 2.11 verdeutlicht den Zusammenhang zwischen einer Schicht und dem Gradienten. Dabei wird die Schicht in der xy – Ebene, welche sich senkrecht zur z – Achse befindet, durch „Schalten“ des z – Gradienten bestimmt. Die hier gemessene Schicht ist also eine transversale Schicht.



*Abbildung 2.11: Bestimmung der Schichtposition*

Durch gleichzeitiges Schalten von Gradienten kann man demzufolge auch schräge Schichten erhalten (siehe Abbildung 2.12).

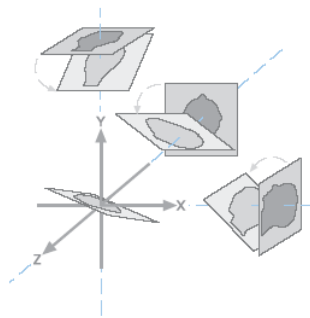


Abbildung 2.12: Schräge Schichten

Nach der Auswahl der gewünschten Schicht wird die Schicht-Dicke bestimmt. Nur durch die Schicht-Position allein würde eine hauchdünne Schicht mit wenigen angeregten Protonen entstehen, in der nicht genügend Informationen enthalten sind, um die Darstellung eines Bildes zu ermöglichen. Die Schichtdicke kann über die Stärke des geschalteten Gradienten bestimmt werden. Wenn die Stärke des Gradienten wie in Abbildung 2.13 geändert wird, erhält man durch ein steileres Gradientenfeld (a) eine dünnere Schicht ( $\Delta z_a$ ) und durch ein schwächeres Gradientenfeld (b) eine dickere Schicht ( $\Delta z_b$ ).

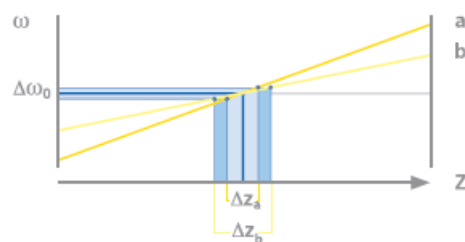


Abbildung 2.13: Bestimmung der Schichtdicke

Ein Bild entsteht aber nicht aus dem Messvorgang selber. Es werden vielmehr zuerst aus den empfangenen MR - Signalen Rohdaten erzeugt und daraus dann das eigentliche Bild berechnet.

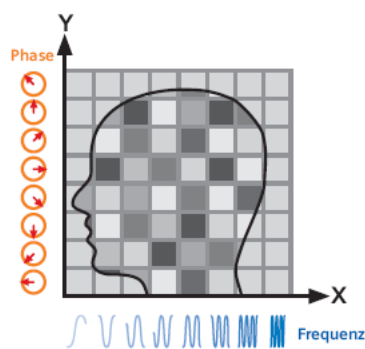
Die Bildberechnung erfolgt, indem aus dem gemessenen Echo des geschalteten Gradienten mit Hilfe der Frequenzkodierung zuerst die einzelnen Frequenzen

herausgefiltert werden und danach über eine Fourier-Transformation für jede Frequenz die entsprechende Signalstärke berechnet wird. Die jeweilige Signalstärke bestimmt den Grauwert des zugehörigen Pixels. Das Ergebnis ist ein Bildstreifen bzw. eine einzelne Bildzeile mit verschiedenen Grauwerten (siehe auch Abbildung 2.14 ).



*Abbildung 2.14: Einzelne Bildzeile mit verschiedenen Grau- Intensitäten*

Um ein Bild mit zwei Dimensionen (Zeilen und Spalten) zu erzeugen, wird noch die Phasenkodierung benötigt. Dazu muss zwischen dem HF-Puls und dem Echo ein Gradient in Y – Richtung geschaltet werden. Die Spins besitzen unmittelbar nach dem Abschalten des Gradienten verschiedene Phasenlagen. Durch ein Herausfiltern der Phasenlagen unter erneuter Zuhilfenahme der Fourier-Transformation erhält man die einzelnen Phasen. Auf diese Weise setzt sich aus Frequenzkodierung und Phasenkodierung eine Rohdaten-Matrix , wie sie in Abbildung 2.15 dargestellt ist, zusammen. Die Darstellung ist in diesem Beispiel auf 8 x 8 Pixel verkürzt worden.



*Abbildung 2.15: Rohdaten-Matrix, verkürzt auf 8x8 Pixel*

## 2.3 Herzuntersuchung am MRT

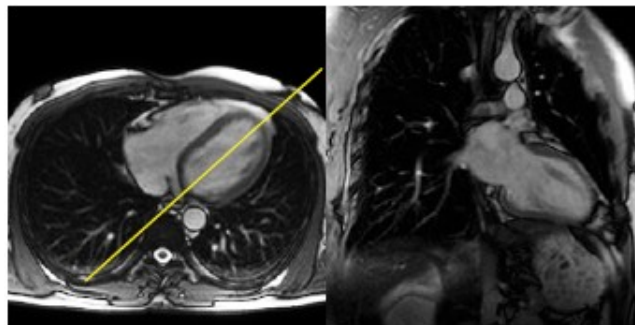
Der erste Schritt bei einer Untersuchung am MRT ist die Patientenregistrierung. Dabei werden alle patientenbezogenen Daten wie Name, Geburtsdatum, Alter, Größe, Gewicht und die zu untersuchende Region in ein entsprechendes Datenblatt eingegeben und gespeichert. Anhand der Merkmale können die MRT-Aufnahmen eindeutig einem Patienten zugeordnet werden.

Der erste Schritt bei der Herzuntersuchung ist das Auffinden des Herzens mit Hilfe des Lokalizers. Da die Lage des Herzens von Patient zu Patient stark variieren kann, ist eine sehr genaue Lokalisierung besonders wichtig. Je exakter die Lokalisierung erfolgt, desto genauer können Pathologien oder Funktionsstörungen in den gemessenen Bildern dargestellt werden.

Für diesen Vorgang wird die Lokalisierung in den folgenden Schritte durchlaufen:

- 2-Kammer-Lokalizier
- 4-Kammer-Lokalizier
- Kurzachsen-Lokalizier

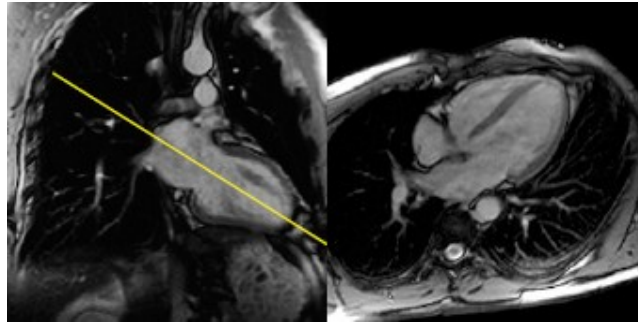
Der 2-Kammer-Blick zeigt die linke Herzkammer, die Vorder- und Hinterwand der linken Herzkammer, den linken Vorhof und die Mitralklappe. Siehe Abbildung 2.16. Mit dem 2-Kammer-Blick wird die vertikale Längsachse des Herzens dargestellt.



*Abbildung 2.16: 2-Kammer-Blick*

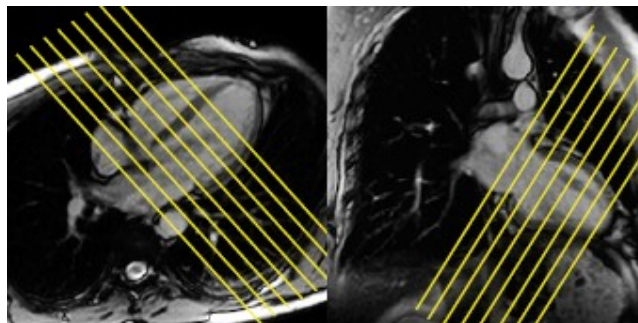
Anschließend folgt der 4-Kammer-Blick, mit welchem die horizontale Längsachse

dargestellt wird. Beim 4-Kammer-Blick sind die Seitenwände der linken Herzkammer sichtbar, sowie die Mitral- und Trikuspidalklappe. Er stellt die Länge der beiden Herzkammern sowie die zugehörigen Vorhöfe dar. Abbildung 2.17 zeigt einen 4-Kammer-Blick beim MRT.



*Abbildung 2.17: 4-Kammer-Blick*

Die Kurzachsen-Blicke zeigen gleichzeitig beide Herzkammern zusammen mit den Herzwänden. Eine Planung der zu messenden Herzschnitte erfolgt auf den Bildern des Kurzachsen-Lokalizers. Ein Beispiel für die Schicht-Planung auf einem Kurzachsen-Blick wird in Abbildung 2.18 dargestellt.



*Abbildung 2.18: Kurzachsen-Blick*

Nach dem erfolgreichen Abschluss der verschiedenen Lokalizier-Schritte kann die Messung der Herzschnittbilder gestartet werden. Diese Messung muss vom Patienten durch Luftanhalten unterstützt werden, um etwaige Veränderungen der Herzposition und einer damit verbundenen Verschlechterung der Bildqualität entgegenzuwirken.

### **2.3.1 Derzeitiges Auswertungsverfahren der aquirierten Schnittbilder**

Wenn die Messung aller Schichten des Herzens abgeschlossen ist, werden die Kurzachsen-Bilder für die Auswertung und Analyse der Herz-Leistungsfähigkeit in ein externes Programm (Argus) geladen. Hier wird von Hand jedes Schnittbild mit einer Kontur um die Linke Herzkammer (eine weitere Kontur um den Herzmuskel ist optional und wird zum Beispiel für die Masseberechnung des Myokard-Muskels verwendet) versehen um danach, basierend auf dem optischen Eindruck, die Phase von Diastole und Systole zu bestimmen.

Diese Prozedur ist sehr aufwändig und in keiner Weise robust, da jeder Anwender eine eigene Auslegung zur „richtigen Größe“ der linken Kammer hat. Es ist damit durchaus möglich, dass die Auswertungsergebnisse der gleichen Herzschnittbilder von einem Anwender zum Anderen unterschiedlich sind.

### **2.3.2 Verbesserung durch automatischen Konturfindung in den Bilddaten**

Aus der Beschreibung von Kapitel 2.3.1 wird auch das Problem der derzeitigen Methodik ersichtlich. Verschiedene Anwender kommen zu verschiedenen Ergebnissen. Weiterhin sind die Ergebnisse erst nach einer vollständigen Auswertung verfügbar. Der erhebliche Zeitaufwand und die Benutzerabhängigkeit stellen weitere negative Faktoren bei der bisherigen Vorgehensweise dar. Wünschenswert wäre eine Lösung, die direkt im Anschluss an eine MRT Messung ein brauchbares und reproduzierbares Ergebnis hervorbringt und dieses dann dem Anwender auf die bisherige bekannte Weise darstellt. Eine für die geschilderten Probleme anwendbare Lösung wird durch das in Kapitel 4.2 entwickelte Inline Cardio Modul repräsentiert.

## 3 Grundlagen der Softwareentwicklung

### 3.1 Lebenszyklusmodell von Software

Der Lebenszyklus moderner Software oder Softwarekomponenten beinhaltet alle Schritte, angefangen von der Problemdefinition bis hin zur Einführung und Wartung des fertigen Produktes. Der Softwarelebenszyklus wird nach Dumke wie folgt definiert:

*Der Software-Lebenszyklus (software life cycle) ist der Prozess der Entwicklung von Software-Produkten und kennzeichnet alle Phasen und Stadien dieser Produkte von ihrer Entwicklung, Einführung und Wartung bis zu ihrer Ablösung oder Beseitigung. [10, S.17]*

Der Lebenszyklus entspricht hierbei vielen Teilschritten bzw. Phasen, die sich genau voneinander abgrenzen lassen und so die Entwicklung einer komplexen Softwarelösung um ein Vielfaches vereinfachen. Durch diese Einteilung in mehrere einzelne Phasen kann nicht nur der Verlauf und der Fortschritt des Projektes besser beurteilt werden, sondern hilft vor allem bei der Projektplanung und der Entwicklung von problemorientierten Lösungen. Jede Phase besteht aus einem Anfangs- und einem Endzustand sowie einigen eigenen Ressourcen. Für die einzelnen Phase gibt es in der Praxis unterschiedliche Bezeichnungen (vgl. auch [10]).

1. Phase – Problemdefinition
2. Phase – Anforderungsanalyse
3. Phase – Spezifikation
4. Phase – Entwurf
5. Phase – Implementierung
6. Phase – Erprobung
7. Phase – Einführung

Vergleichbar zu Dumke wird der Softwarelebenszyklus, zum Beispiel von Balzert in der folgenden Weise beschrieben:

*Lebenszyklus: Gesamte Lebensdauer eines Produkts von seiner Entwicklung (Geburt) über seinen Betrieb bis hin zu seiner „Außer-Betriebnahme“ (Tod). [11, S.975]*

Balzerts Definition der einzelnen Phasen ist der von Dumke recht ähnlich

1. Phase – Planung
2. Phase – Definition
3. Phase – Entwurf
4. Phase – Implementierung
5. Phase - Abnahme/Einführung
6. Phase – Wartung/Pflege

Weiterführende Informationen zum Softwarelebenszyklus können unter anderem bei [22] und bei [23] entnommen werden.

### **3.2 Beschreibung einzelner Phasen des Softwarelebenszyklus**

Wie bereits in Kapitel 3.1 erwähnt, wird der Lebenszyklus einer Software in verschiedene Phasen eingeteilt. Die einzelnen Phasen dienen zur besseren Strukturierung und schaffen die Basis für einen guten Softwareentwicklungsprozess. Dumke beschreibt die Phase des Lebenszyklus wie folgt:

*Eine Lebenszyklus-Phase (life cycle phase) innerhalb des Softwareentwicklungs-, Anwendungs- und Wartungsprozesses ist ein zeitlich begrenzter Abschnitt mit relativ eigenständigen Ressourcen, für den eine Anfangssituation und ein bewertbarer Endzustand bestimmt werden können. [10, S.17]*

Die erste Phase im Software-Lebenszyklus stellt die Problemdefinition dar, in der es um eine Beschreibung der Anforderungen an ein Software-Produkt bzw. Software-System

geht. Die Anforderungen unterliegen dem gemeinsamen Standard IEEE 830 (Software Requirements Specification) bei der Softwareentwicklung. Darin werden die Anforderungen in der folgenden Weise klassifiziert:

- Funktionale Anforderungen
- Qualitätsanforderungen
- Systembezogene Anforderungen
- Prozessbezogene Anforderungen
- Sonstige Anforderungen

Nach Dumke (vgl. auch [10], S.23) ist die Problemdefinition eine zusammenfassende Beschreibung von Anforderungen an die Entwicklung von Software-Produkten. Daher werden zunächst die Anforderungen etwas genauer betrachtet.

Die funktionalen Anforderungen enthalten meist schon Angaben über die zu verarbeitenden Daten und mögliche Schnittstellen zum Anwender oder zum System.

Bei den Qualitätsanforderungen geht es um die verschiedenen Qualitätsgruppen von Software-Produkten. Dazu werden unterschiedliche Merkmale wie Leistung, Ressourcen und Systemverhalten zu einem gemeinsamen Gruppenmerkmal zusammengefasst (hier: Effizienz). In einem späteren Kapitel wird speziell auf einige Qualitätsgruppen näher eingegangen und versucht, die Qualität des hier entwickelten Software-Produktes einzustufen.

Unter dem Begriff „Systembezogene Anforderungen“ fallen alle diejenigen Anforderungen, die Aussagen oder Vorgaben zur Plattform und zu einer konkreten Programmiersprache geben. Diese Anforderungen werden oft allgemein umschrieben mit „PC-Netz“ und „objektorientierte Programmiersprache“.

Prozessbezogene Anforderungen beinhalten Angaben für einen zeitlichen und finanziellen Rahmen. Diese Anforderungen sind besonders richtungweisend für den Verlauf der Softwareentwicklung. Alle noch verbleibenden Anforderungen werden unter dem Begriff „Sonstige Anforderungen“ zusammengefasst. Hier sind

Anforderungen enthalten, die nicht oder nur schwer in die bisher genannten Anforderungsgruppen eingeordnet werden können.

Auf die Phase der Problemdefinition folgt die Phase der Anforderungsanalyse. Hier findet eine Kontrolle der Anforderungen an ein zu entwickelndes Software-System hinsichtlich Korrektheit, Vollständigkeit, Sachgerechtigkeit, Konsistenz und Machbarkeit statt. Bei Bedarf werden an dieser Stelle Bearbeitungen und Modifikationen, eventuell auch weitere Detaillierungen, der existierenden Anforderungen eingebracht. Das Ziel dieser Analyse ist eine formalisierte und nachprüfbare Ausarbeitung der Anforderungen.

Mit der Phase der Spezifikation beginnt der eigentliche Teil der Softwareentwicklung und wird nach Dumke in der folgenden Weise beschrieben:

*Die Spezifikation (specification) ist die Formulierung aller funktionalen und einiger qualitativer Anforderungen in einem Modell, welches die computerbezogenen und organisatorischen Systemkomponenten beschreibt.*  
[10, S.44]

In dieser Phase sollte die Umsetzung der Anforderungen in ein Modell erfolgen, dass die Funktionalität der zu entwickelnden Software vollständig beschreibt und damit eine spätere softwareseitige Realisierung sicherstellt.

Die Entwurfsphase werden alle hardware- und softwarebezogene Anforderungen umgesetzt. Diese werden häufig auch als systembezogene Anforderungen bezeichnet. Dabei werden die Anforderungen unterschieden in Systemvorgaben und Systemvoraussetzungen. Die Systemvorgaben geben Auskunft über die Anforderungen bei der Produktimplementierung zum Beispiel welche Programmiersprache verwendet werden soll, ob es sich um Desktop- oder um eine Web-Anwendung handelt oder welche Vorgaben zum Betriebssystem bestehen.

Ziel an dieser Stelle ist es, das in der Spezifikation erstellte Modell an die gegebenen

Hard- und Software-Plattform anzupassen.

In der Phase der Implementierung werden die verschiedenen Modelle und Entwürfe aus den vorangehenden Phasen in Software umgesetzt. Eine Definition dazu nach Dumke besagt:

*Die Implementation (implementation) ist die Umsetzung der Entwurfsergebnisse in ein programmiertes, auf spezieller Hardware oder Hardware-Klassen abarbeitbares System und vollzieht sich in den Phasen der Kodierung, Tests, Integration und Installation. [10, S.64]*

Direkt im Anschluss an die Implementierung, teilweise sogar während dieser Phase, folgt die Phase der Erprobung. Hier wird das entwickelte Softwareprodukt unter realen Bedingungen getestet und erprobt. Dumke definiert die Phase der Erprobung wie folgt:

*Die Erprobung eines Software-Systems (software acceptance testing) ist der Nachweis seiner Validität auf der Grundlage von Akzeptanzkriterien in einem ausgewählten Anwendungsfeld. [10, S.93]*

Diese Akzeptanzkriterien ermöglichen eine Bewertung der erzielten Ergebnisse des Software-Tests. Sollten während dieser Phase Fehler festgestellt werden, müssen unter Umständen die zuvor abgeschlossenen Phasen überarbeitet und ergänzt werden, um die im Test aufgetretenen Fehler zu beseitigen.

Die letzte Phase des Entwicklungs-Zyklus stellt die Wartung dar. In dieser Phase geht es nur bedingt um ein Ausbessern von noch enthaltenen Fehlern. Vielmehr finden notwendige Anpassungen an veränderte Systemsoftware statt, Austausch von einzelnen Komponenten des entwickelten Software-Produktes durch leistungsfähigere Entwicklungen, Erweiterungen durch neue Funktionalitäten oder Veränderungen der Nutzeroberfläche für eine besser Benutzerinteraktion. Dumke fasst die Phase der Wartung mit einer aussagekräftigen Definition zusammen:

*Die Software-Wartung (software maintenance) umfasst alle Aktivitäten der Erweiterung (extension), der Anpassung (adaptation), der Korrektur (correction), der Verbesserung (perfection) und der Vorbeugung (prevention) an einem Software-Produkt, um dessen fortgesetzte Anwendung über einen längeren Zeitraum zu gewährleisten [10, S.94]*

### **3.3 Vorgehensmodelle bei der Softwareentwicklung**

Die einzelnen in Kapitel 3.2 beschriebenen Phasen eines Software-Lebenszyklus lassen sich in zwei verschiedene Vorgehensmodelle einordnen.

Ein Vorgehensmodell ist eine Arbeitsanleitung, die beschreibt, wie und in welcher Reihenfolge Arbeiten bei der Organisation und Durchführung eines Projekts durchzuführen sind.

Dabei lassen sich die Vorgehensmodelle in zwei verschiedene Klassen einteilen. Auf der einen Seite findet man die „nicht-sequenziellen“ Modelle, zu denen vor allem die evolutionäre Softwareentwicklung und das Prototyping zählt. Diese Modelle werden auch als „zyklisches Modelle“ bezeichnet, da aus verschiedenen Gründen Rücksprünge zu vorherigen Phasen möglich sind. Die Rücksprünge, auch bezeichnet als Feedbacks, können sich bei den nicht-sequenziellen Modellen sogar über mehrere Phasen erstrecken. Das Vorgehen leitet sich aus den gewonnenen Erfahrungen und der Tatsache ab, dass eine Umsetzung aller Anforderungen in einem Software-Projekt nicht auf einmal erfolgen kann, sondern erst nach mehreren Zyklen erreicht wird. Eine spezielle Form des Prototypings ist das Rapid Prototyping. Bei diesem Modell soll der Prototyp die Machbarkeit von speziellen Anforderungen belegen und nachweisen. Es ist jedoch auch möglich, den entwickelten Prototypen teilweise oder auch ganz als Kernstück der neuen Softwareentwicklung einzusetzen. Abbildung 3.1 zeigt eine Darstellung des Entwicklungsvorgehens beim Rapid Prototyping (vgl. auch [9]).

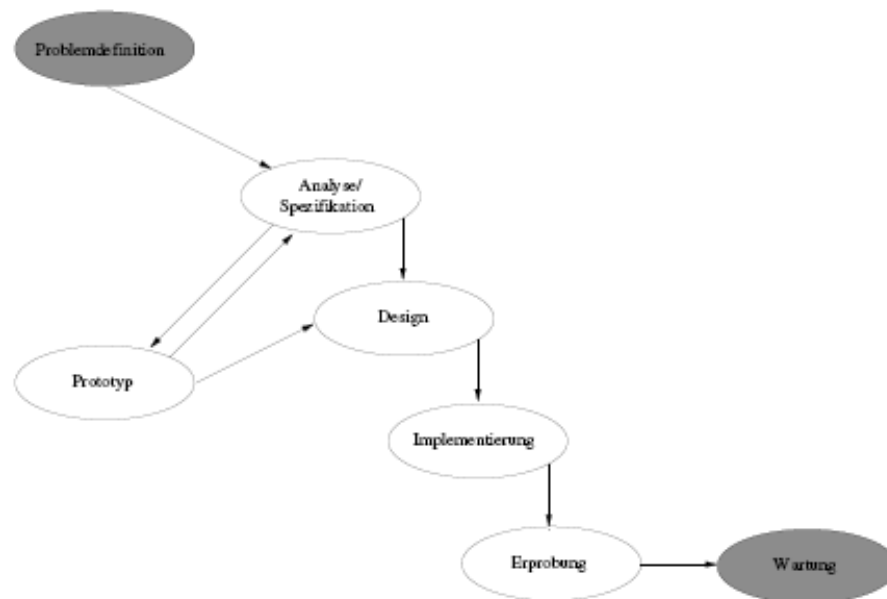


Abbildung 3.1: Prototyping nach Marciniak 1994 [9]

Die zweite Art der Software-Lebenszyklus Modelle sind die „sequentiellen“ Modelle. Bei diesen Modellen werden die einzelnen Phasen hintereinander abgearbeitet, wobei ein Rücksprung zur vorherigen Phase bei aufgetretenen Problemen zur Not immer noch möglich ist. So kann zum Beispiel eine schlechte oder nicht effiziente Implementierung eines Moduls durch ein Rücksprung zum Entwurf und der erneuten Auswahl einer besseren Lösung behoben werden. Die Ergebnisse der letzten Phase gehen als Ausgangsbasis in die aktuelle Phase über. Besonders nennenswert für die sequentiellen Modelle sind an dieser Stelle das Wasserfall-Modell und das V-Modell. In Abbildung 3.2 ist die schematische Darstellung des Wasserfall-Modells nach Royce (vgl. auch [13]) aufgezeigt.

Das dargestellte Wasserfall-Modell enthält alle Aktivitäten in streng sequenzieller Reihenfolge. Dieses Modell eignet sich, wenn sich die Arbeitsschritte deutlich voneinander abtrennen lassen. Alle Risiken müssen vor Projektbeginn ausgeschlossen werden können. Es ist angebracht, wenn die Mitarbeiteranzahl klein ist, da alle

Mitarbeiter gleichzeitig an einem Arbeitsschritt arbeiten können. Leider ist das Wasserfall-Modell für die meisten Aufgabenstellung unrealistisch und damit ungeeignet. Es setzt voraus, dass jede Aktivität auf Anhieb erfolgreich abgeschlossen werden kann. Ein bedeutender Nachteil entsteht meistens dann, wenn z.B. Anforderungen oft auch noch in späten Phasen des Projekts geändert bzw. erst dann richtig verstanden werden. In diesem Fall müssten frühere Aktivitäten wiederholt werden.

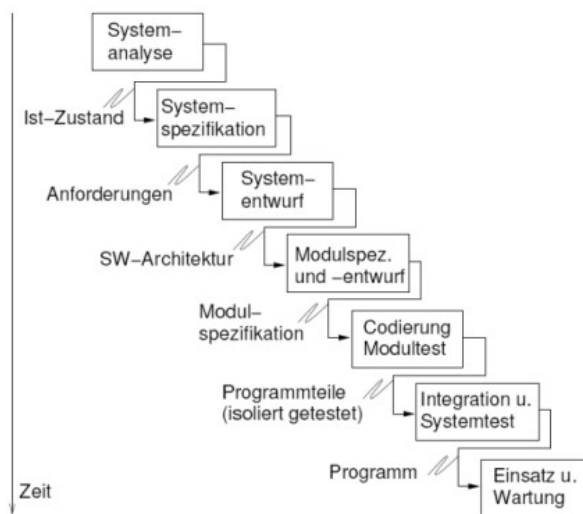
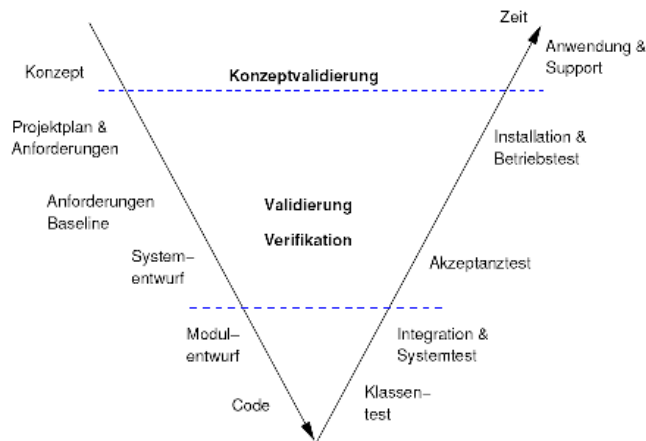


Abbildung 3.2: Wasserfallmodell nach Royce (1970)

Ein weiteres, oft für die Software Entwicklung verwendetes Modell, ist das V-Modell. Das V-Modell ist ein international anerkannter Entwicklungsstandard für IT-Systeme, der einheitlich und verbindlich festlegt, wie und wann die einzelnen Aufgaben durchzuführen sind, welche Mittel dafür eingesetzt werden und wann eine Aufgabe abgeschlossen ist. Das V-Modell verbessert die Produkt- und Prozessqualität, in dem es konkrete und für den Anwender leicht umsetzbare Handlungsanweisungen zur Durchführung der Aktivitäten gibt. Weiterhin wird durch das V-Modell eine im Wasserfall-Modell fehlende Qualitätssicherung in den Prozess der Softwareentwicklung integriert (vgl. auch [17] und [24]).

In Abbildung 3.3 wird die Struktur des V-Modells nach Boehm (vgl. dazu [12]) gezeigt.

Das V-Modell sieht sowohl Verifikationen als auch Validierungen vor. Dabei bedeutet die **Validierung** eine Eignung bzw. den Wert eines Produktes bezogen auf seinen Einsatzzweck. Die **Verifikation** drückt die Überprüfung der Übereinstimmung von Softwareprodukt und Spezifikation aus. (vgl. auch [17]).



*Abbildung 3.3: V-Modell nach Boehm [12]*

Für die Softwareentwicklung bei Siemens Medical Solutions wird seit langem erfolgreich das V-Modell verwendet. Die Vorteile des V-Modells liegen in der integrierten und detaillierten Darstellung von Systemerstellung, Qualitätssicherung, Konfigurationsmanagement und Projektmanagement. Das V-Modell stellt zudem ein generisches Vorgehensmodell dar, welches auf bestimmte projektspezifische Anforderungen zugeschnitten werden kann. Zudem ist es gut geeignet für große Projekte (large scale projects) und für eingebettete Systeme (vgl. auch [17]), wie sie bei Siemens entwickelt werden.

Die Softwareentwicklung beginnt mit der Anforderungsanalyse. Sie dient als Teil des Anforderungsmanagements dazu, alle Anforderungen an das zu entwickelnde Software-Modul zu ermitteln. Die Anforderungen bei diesem Projekt sind, ein automatisches Verfahren für die Volumenberechnung der linken Herzkammer zu implementieren und eine Ausführung unter dem Betriebssystem LINUX sicherzustellen. Mit dem Systementwurf, welche auch als Designspezifikation bezeichnet wird, folgt der nächste

Schritt im V-Modell. An dieser Stelle findet die Planung und das Design der zu implementierenden Software statt. Im Anschluss an das fertige Design erfolgt der Entwurf (Funktionale Spezifikation) des neuen Software-Moduls. Dafür werden die benötigten Funktionen und Klassen entsprechend den Vorgaben aus dem Design erstellt. Die letzte Stufe ist die eigentliche Implementierung und Entwicklung des erforderlichen Codes.

Die Firma Siemens verfügt über ein aufwendiges Informationssystem für Softwareentwickler, in welchem alle Phasen und Schritte bei der Softwareentwicklung hinterlegt und ausführlich beschrieben sind. Jeder Entwickler kann zu jeder Zeit auf dieses System zugreifen und sich einen detaillierten Überblick über einzelne Prozesse, z.B. den Prozess der Softwareentwicklung und seine derzeitige Rolle in diesem verschaffen. In Abbildung 3.4 ist zum besseren Verständnis ausschnittsweise der Prozessablauf der Softwareentwicklung bei Siemens abgebildet. Sichtbar sind die verschiedenen Aktivitäten, die sich bei Bedarf bis auf einzelne Rollen detaillieren lassen.

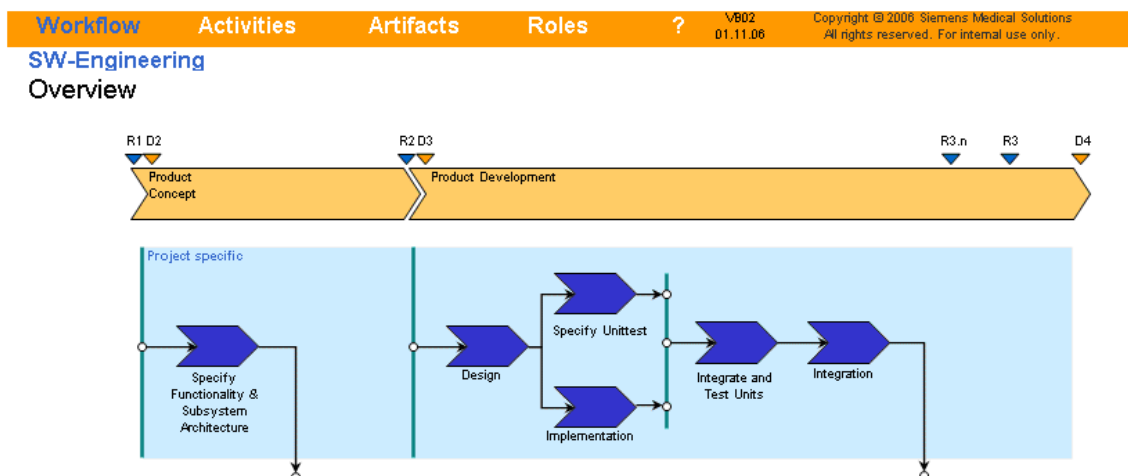


Abbildung 3.4: Übersicht über den Prozess der Softwareentwicklung

### 3.4 Anforderungen an die Software-Qualität

Bei jeder Art der Softwareentwicklung wird ein besonderes Augenmerk auf die Qualität der Entwicklung gelegt. Der Kunde vergibt den Auftrag und erwartet abschließend ein Produkt, welches den allgemein gültigen und letztendlich auch seinen eigenen Anforderungen gerecht werden soll. Um dieses zu gewährleisten, ist unter anderem der Begriff der Softwarequalität eingeführt worden. Aber was sagt das Wort „Qualität“ überhaupt aus? Der Begriff der Qualität oder auch Softwarequalität wird von Balzert [17] in fünf Ansätze aufgeteilt.

1. Transzendenter Ansatz : „[...] universell, absolut, einzigartig und vollkommen“
2. Produktbezogener Ansatz : „[...] messbare Eigenschaft des Produktes“
3. Benutzerbezogener Ansatz : „[...] Sicht des Produktbenutzers“
4. Prozessbezogener Ansatz : „[...] richtige Erstellung eines Produktes“
5. Kosten/Nutzenbezogener Ansatz : „[...] Funktion von Kosten und Nutzen“

Aber wie kann man nun diese Qualität beurteilen? Es gibt zwei grundlegende Arten von Qualität. Einmal die Qualität nach DIN 55350 - Teil 11 (vgl. auch [21]), welche die Gesamtheit von Eigenschaften und Merkmalen eines Produktes oder einer Tätigkeit, die zu einer entsprechenden Eignung führen, beschreibt. Auf der anderen Seite gibt es die Software-Qualität nach ISO<sup>17</sup> 9126 (vgl. auch [25]), die die Gesamtheit der Merkmale und Merkmalswerte eines Software-Produktes, die sich auf dessen Eignung beziehen.

Anhand der verschiedenen Qualitätsmerkmale von ISO 9126 (siehe Abbildung 3.5) lässt sich die Qualität der entwickelten Software genau einstufen und bewerten. Folgende sechs Haupt-Qualitätsmerkmale können bei der Bewertung der Softwarequalität unterschieden werden: Funktionalität, Zuverlässigkeit, Benutzbarkeit, Änderbarkeit, Effizienz und Übertragbarkeit.

---

<sup>17</sup> ISO ist die Abkürzung für Internationale Organisation für Normung

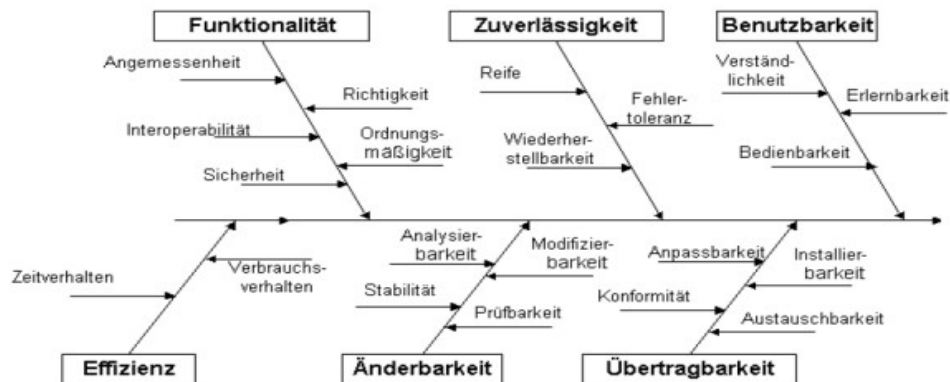


Abbildung 3.5: Qualitative Anforderungen bei der Softwareentwicklung [10]

Die Funktionalität beschreibt Funktionen mit festgelegten Eigenschaften, die die definierten Anforderungen erfüllen. Darunter fallen

- **Richtigkeit**, welche eine Lieferung der richtigen und vereinbarten Ergebnisse sicherstellt. (z.B.: eine bestimmte Genauigkeit von Werten)
- **Angemessenheit**, die für die Eignung von Funktionen für eine bestimmte Aufgabe zuständig ist (z.B.: Menge von aufgabenorientierten Funktionen)
- **Interoperabilität**, besitzt die Software die Fähigkeit, mit vorgegebenen Systemen zusammenzuarbeiten
- **Ordnungsmäßigkeit**, die Erfüllung von Vereinbarungen, Normen oder anwenderspezifischen Bestimmungen und Vorschriften
- **Sicherheit**, werden versehentliche oder absichtliche Zugriffe auf Programm oder Daten unterbunden

Durch die Zuverlässigkeit wird der Software die Fähigkeit bescheinigt, das Leistungsniveau unter bestimmten Bedingungen in einer festgelegten Zeit beizubehalten. Dabei unterscheidet man in

- **Reife**, welche eine Aussage über die Fehlerhäufigkeit trifft. Ziel ist ein geringes Auftreten von Fehlzuständen.

- **Fehlertoleranz**, die eine Fähigkeit ausmacht, bei der ein vorgegebenes Leistungsniveau durch aufgetretene Fehler nicht beeinträchtigt oder herabgesetzt wird
- **Wiederherstellbarkeit**, die bei einem Versagen der Software das vor dem Fehler herrschende Leistungsniveau wiederherstellt und gegebenenfalls betroffene Daten wiedergewinnt

Abschließend wird noch die Effizienz betrachtet. Sie gibt Auskunft über das Verhältnis zwischen dem Leistungsniveau der entwickelten Software und dem Umfang der eingesetzten Betriebsmittel. Zur Effizienz gehören das

- **Zeitverhalten**, das die Antwort- und Verarbeitungszeit sowie den Funktionsdurchsatz ausdrückt
- **Verbrauchsverhalten**, die Menge und Dauer der benötigten Betriebsmittel zur Erfüllung der Funktionen.

## **4 Entwurf und Design der Argus-Algorithmen und des Inline Cardio Moduls**

In der Entwurfs- und Design-Phase geht es um die Umsetzung von Hard- und Software-bezogenen Anforderungen für das Inline Cardio Modul, welches im weiteren als ICM bezeichnet wird. In dieser Phase wird die zu entwickelnde Software an die vorgegebene Plattform und die bereits vorhandenen Hard- und Softwareschnittstellen angepasst. Das Design ist die theoretische Umsetzung der vorliegenden Problemstellung und trägt grundlegend zur erfolgreichen Lösung bei. Dafür findet zuerst im Kapitel 4.1 eine Erläuterung der Segmentierung durch die Argus-Algorithmen statt. Anschließend beschäftigt sich Kapitel 4.2 mit dem Design des Inline Cardio Moduls für die automatische Auswertung der Schnittbilder sowie mit einer Beschreibung der dafür benötigten Funktionen.

### **4.1 Algorithmus zur Erkennung der linken Herzkammer**

Für eine Erkennung der linken Herzkammer wurde im Siemens-Forschungscenter in Princeton / USA (Siemens Cooperate and Research kurz: SCR) ein entsprechender Algorithmus (Argus-Algorithmus) entwickelt. Dieser Algorithmus ermittelt auf den gemessenen MRT-Schnittbildern die linke Herzkammer und markiert die entsprechende Region durch einzelne Punkte. Dieser Vorgang wird allgemein als Segmentierung bezeichnet.

Für den Einsatz unter einem 64Bit Linux Betriebssystem wie es bei Siemens verwendet wird, muss der von SCR bereitgestellte Algorithmus noch entsprechend angepasst werden. Diese Modifikation ist nötig, da der Quellcode bislang nur unter einem Windows System weiterentwickelt und getestet wurde. Ein Einsatz unter Linux ist bislang aus Gründen der Zweckmäßigkeit bei SCR nicht notwendig gewesen. Die Integration in das bestehende Laufzeitsystem von Siemens Medical macht es jedoch erforderlich, das die später erzeugte Bibliothek auf dem Bildrechner unter Linux

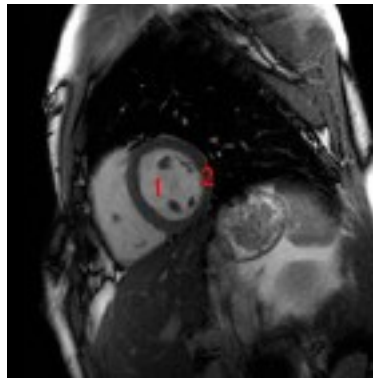
ausgeführt wird. Die Entscheidung für ein Linux System wird durch die Tatsache untermauert, dass eine Bereitstellung von mehr als 3 Gigabyte (GB) zusammenhängendem Speicher für die Rohdaten-Files derzeit nur unter Linux möglich ist. Damit ist gewährleistet, dass die erzeugten Daten nicht auf einzelne Blöcke verteilt im Speicher liegen und die Zugriffszeit optimiert wird.

In Folge dessen ist eine komplette Portierung der bestehenden Algorithmen unter Berücksichtigung der vom Betriebssystem abhängigen Gegebenheiten erforderlich. Das bedeutet in diesem Fall, dass jede Funktion und jede Klasse auf vorliegende Windows-Abhängigkeiten hin untersucht werden muss. Die wesentlichen Modifikationen der Segmentierungsalgorithmen sind ausführlich im Kapitel 5.1 beschrieben.

Der Algorithmus von SCR erhält als Eingangsgröße die aquirierten MR Bilddaten. Auf diesen wird unter Zuhilfenahme spezieller Verfahren die Position und die Größe der linken Herzkammer ermittelt, wobei der Kontrastunterschied zwischen der linken Kammer und dem Myokard-Muskel eine besonders wichtige Rolle spielt. Abbildung 4.1 zeigt deutlich die Abgrenzung von Kammer (1) und Muskel (2). Je besser dieser Kontrastunterschied erkennbar ist, um so genauer wird die Lokalisierung und die Flächenberechnung der Herzkammer. Sollten auf Grund einer schlechten Messung zum Beispiel bei einer falschen Auswahl von Parametern oder einer Veratmung<sup>18</sup> des Patienten der Kontrastunterschied zwischen Kammer und Muskel nicht deutlich genug sein, kann es bei der Bestimmung der Kammergrenze zu Abweichungen kommen. Die Folgen wären eine zu weite bzw. zu enge Kontur oder die Segmentierung einer völlig falschen Region. Der Segmentierungsalgorithmus ist nicht in der Lage, diesen Fehler zu erkennen und zu korrigieren. Das Resultat schlägt sich dann in einer falschen Flächen- und Volumenbestimmung und damit eine fehlerhaften Berechnung von EDV, ESV, SV und EF nieder. Im schlimmsten Fall hätte dies eine Falschdiagnose zur Folge. Um den Anwender auf eine mögliche falsche Kontur hinzuweisen, werden die medizinischen Bilder zusammen mit der ermittelten Kontur in einem Betrachtungsfenster angezeigt.

---

<sup>18</sup> Wenn der Patient während der Messung atmet, obwohl er nach Anweisung die Luft anhalten sollte, spricht man von „Veratmung“



*Abbildung 4.1: Abgrenzung der Linken Herzkammer vom Myokard*

Eine weitere Vertiefung in die Thematik der Herzkammersegmentierung ist nicht möglich, da das hierfür verwendete Verfahren der Vertraulichkeit bei Siemens unterliegt. Weitere Informationen zu diesem Thema können jedoch bei [18] und [19] entnommen werden.

## **4.2 Das Inline Cardio Modul (ICM)**

Seit Beginn der Untersuchung des Herzens gibt es Verfahren zur Auswertung der Herzvitalität. Diese Verfahren wurden bis zur heutigen Zeit immer weiter verbessert und lassen aussagekräftige Diagnosen zur Herz-Leistungsfähigkeit zu. Jedoch erfolgte diese Auswertung auf manuelle Weise, was, wie in den Kapiteln 1.3 und 3.5 beschrieben, entscheidende Nachteile bezüglich der Performance und der Reproduzierbarkeit mit sich bringt.

Für das Design findet wie im vorangegangenen Kapitel eine Betrachtung des derzeit vorliegende Software-Systems statt. Die Bildnachverarbeitung läuft auf dem Bildrechner unter Linux ab. Daher ist es notwendig, schon während des Designs darauf zu achten, dass keine Abhängigkeiten zu speziellen Windows Routinen enthalten sind. Damit das hier entwickelte Software-Modul später von der Bildverarbeitung verwendet werden kann, muss es an einer passenden Stelle innerhalb der Kette für

„nachverarbeitenden Algorithmen“ integriert werden. Dieses geschieht über einen Konfigurator, welcher das ICM auf Anforderung des Anwenders in die Kette ein- und aushängen kann. Dafür müssen Schnittstellen sowohl zum vorhergehenden Modul als auch zum nachfolgenden bereitgestellt werden, um medizinischen Bilder empfangen und abschließend weitergegeben werden zu können.

Zur Verarbeitung der Bilddaten ist es notwendig, nacheinander jedes Schnittbild in einer speziellen Speicherstruktur abzulegen, um diese dann an den Segmentierungsalgorithmus zu übergeben. Die Berechnung der richtigen Kontur erfolgt ausschließlich auf den Daten in dieser Struktur.

Für den internen Ablauf werden verschiedene Funktionen benötigt, um die Konturen von einem Bild zum anderen zu vererben. Dazu zählen:

- „SegmentationAutoAdjust“, um das Bild in der Idealschicht zu segmentieren (Hier findet keine Vererbung von Konturinformationen statt)
- „SegmentationUp“ für die Vererbung der Kontur zur vorherigen Schicht
- „SegmentationDown“ für eine Vererbung zur nächsten Schicht
- „SegmentationRight“ zur Segmentierung nach rechts (innerhalb einer Schicht)
- „SegmentationLeft“ für die Segmentierung zum linken Bild
- sowie weitere Funktionen für die interne Verarbeitung (siehe Abbildung 4.2)

Im Fall eines Segmentierungsfehlers müssen die an das ICM übergebenen Bilddaten unverzüglich an das nächste Modul weitergeleitet werden, ohne dass eine weitere automatische Segmentierung ausgeführt wird. Zu jedem Fehler wird ein entsprechender Eintrag ins Logfile geschrieben. Fehler bei der Segmentierung können durch Konturen hervorgerufen werden, die im Vergleich zur vorausgegangenen Kontur einen vorgegebenen Toleranzbereich über- oder unterschreiten und damit zu groß oder zu klein sind. Ein Fehler liegt auch dann vor, wenn keine Kontur vom Segmentierungsalgorithmus ermittelt werden konnte.

Nachdem die Verarbeitung aller Bilder abgeschlossen ist, werden die einzelnen Volumina für die Bestimmung der Herz-Leistungsfähigkeit berechnet und auf einem

leeren MR-Bild in übersichtlicher Form dargestellt. Auf diese Weise können die Auswertungsergebnisse sehr einfach in der Datenbank archiviert werden, indem das Ergebnisbild wie die vorhergehenden einfach weitergeleitet wird.

In Abbildung 4.2 ist das UML Diagramm für das ICM dargestellt. Darin sind die zu verwendende Methoden und Variablen für die spätere Implementierung aufgeführt.

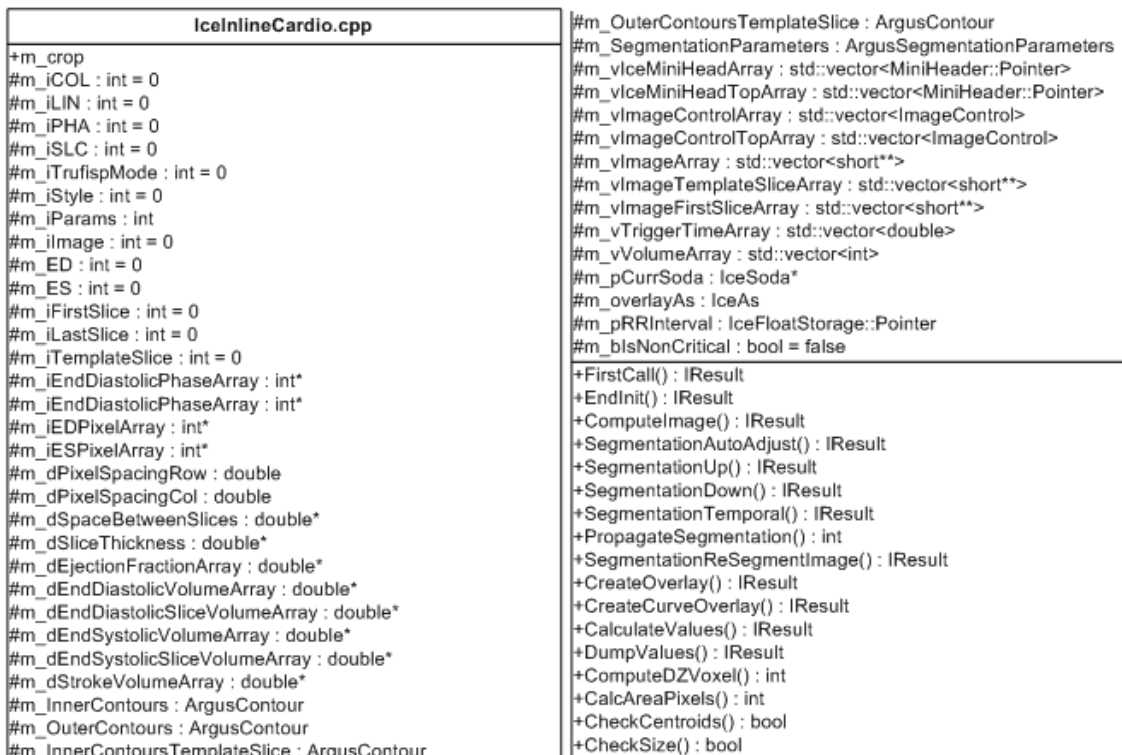


Abbildung 4.2: UML Diagramm für das ICM

### 4.3 Integration des ICM in die MR Bildverarbeitung

Jedes entwickelte Modul für die Bildverarbeitung an einem MRT kann durch den Anwender aktiviert werden und kommt somit zum Einsatz. Jedoch müssen gewisse Regeln existieren, die den Einsatz der installierten Module überwachen. Diese Regeln werden durch so genannte Messprogramme repräsentiert. Für eine Untersuchung einer oder mehrerer Geweberegionen bzw. Körperteile gibt es verschiedene Vorschriften, die definieren, welche Software-Module wann und wie lange zum Einsatz kommen. Das

Messprogramm enthält alle für den Messauftrag auszuführenden Module. Neben bereits standardmäßig auf jedem System installierten Messprogrammen wird dem Anwender auch die Möglichkeit gegeben, eigene Messprogramme zu erstellen und diese zum Einsatz zu bringen.

Um die Funktionalität des hier entwickelten Software-Moduls nutzen zu können, wird ein bereits bestehendes Messprogramm zur Untersuchung der linken Herzkammer um eine Checkbox<sup>19</sup> bzw. Schalter erweitert, über welchen sich die Inline Cardio Funktion bei Bedarf zuschalten lässt. Auf diese Weise kann der Anwender entscheiden, ob die automatische Auswertung der Herzschnittbilder zum Einsatz kommt. Die Erweiterung wirkt sich in keiner Weise negativ auf das bestehende Messprogramm aus. Wird eine Verwendung der Inline Cardio Funktion gewünscht, erweitert sich die Kette der verschiedenen Software-Module um das hier entwickelte ICMI. Im Verlauf der aktuellen Messung kann der Anwender das Resultat anhand der Bilder im Inline-Fenster betrachten. Sichtbar sind die anatomischen Herzbilder mit zusätzlich aufgeprägten Konturen der linken Herzkammer. Sollten diese Konturen nicht den Vorstellungen des Anwenders entsprechen, können die Bilder im Anschluss über das bisher verwendete Programm zur Herzvolumenauswertung nachträglich bearbeitet werden. Danach kann eine neue Auswertung der überarbeiteten Bilder erfolgen. Auf diese Weise ist sicher gestellt, dass eine fehlerhafte automatische Auswertung korrigiert werden kann.

---

<sup>19</sup> Eine Checkbox ist im softwaretechnischen Sinn eine Art Schalter, welcher entweder auf AN oder auf AUS steht (Aktiv oder Inaktiv)

## **5 Implementierung der Segmentierungsalgorithmen und des ICM**

### **5.1 Algorithmen für die Segmentierung**

Der Argus Algorithmus für eine Segmentierung der Linken Herzkammer stellt eine notwendige Basis für den Einsatz des ICM dar (siehe auch Kapitel 4.1).

Für die Implementierung in die bestehende Laufzeitumgebung ist es erforderlich, den Algorithmus von seiner derzeitigen Windows Architektur auf Linux zu portieren. Dieses Vorhaben ist jedoch mit einigen entscheidenden Problemen verbunden. Zum einen liegt der Quellcode nur in einer Microsoft Visual Studio Projektstruktur vor und kann damit nicht sofort in die neue Laufzeitumgebung übernommen werden.

Zum anderen wurde der bestehende Quellcode bislang nur unter Microsoft Windows übersetzt und ausgeführt. Es ist daher noch nicht bekannt, ob und mit welchem Aufwand eine entsprechende Portierung verbunden ist.

Aus den genannten Gründen wurde sich für das folgende Vorgehen entschieden.

1. Die derzeitige Projektstruktur wird aufgelöst und gemäß den aktuellen Richtlinien für die Softwareentwicklung bei Siemens Medical angepasst. Dazu muss eine eigenständige Bibliothek erzeugt werden.
2. Der Algorithmus wird in eine Linux-kompatible Form portiert.

#### **5.1.1 Anpassungen der Segmentierungsalgorithmen**

Um ein strukturiertes Vorgehen zu gewährleisten, wird zuerst der zu portierende Quellcode aus dem Gesamtpaket heraus getrennt. Daraus ergibt sich die folgende neue Struktur für den Quellcode (siehe auch Tabelle 5.1):

<i>Type</i>	<i>Directory</i>	<i>Extension</i>
Sourcecode	<b>/MrCardioAlgos/Segmentation/</b>	<b>*.cpp</b>
	<b>/MrCardioAlgos/BackEndCommon/</b>	<b>*.cpp</b>
Header	<b>/include/MrCardioAlgos/Segmentation/</b>	<b>*.h</b>
	<b>/include/MrCardioAlgos/BackEndCommon/</b>	<b>*.h</b>

*Tabelle 5.1: Neue Verzeichnisstruktur der zugeliferten Segmentierungsalgorithmen*

Daran anschließend wird für das Pakete ein eigenes make-File<sup>20</sup> erzeugt, in welchem alle Dateien des Segmentierungsalgorithmus sowie zusätzlich benötigte Bibliotheken enthalten sind. Die Abbildung 5.1 zeigt die grobe Struktur des verwendeten make-Files, welches vollständig im Anhang B enthalten ist.

Die zu erzeugende Bibliothek wird über das Steuerwort „LIB“ angegeben. Alle weiteren zugehörigen Dateien werden mittels „CPPSOURCESFROM“ bzw. „CPPSOURCES“ hinterlegt. Es bestehen für dieses Projekt keine weiteren Abhängigkeiten zu anderen Bibliotheken. Als letzten Schritt der Implementierung müssen die im Quellcode bestehenden #include - Direktiven auf die neue Struktur angepasst und gegebenenfalls einige Änderungen in den Dateien vorgenommen werden, um die Kompatibilität zum Linux-System zu gewährleisten. Diese Veränderungen beschränken sich vorwiegend auf die Schreibweise von verschiedenen Datentypen, welche in der Tabelle 5.2 dargestellt sind.

<sup>20</sup> Im make-File sind alle benötigten Dateien und weitere Bibliotheken aufgeführt, die für eine erfolgreiche Erstellung benötigt werden.

```
##-----  
## Copyright (C) Siemens AG 1999 All Rights Reserved. Confidential  
##-----  
##  
## Project: NUMARIS/4  
## File: \...\MrCardioAlgos\makefile.trc  
## Version:  
## Author: AUTOR  
## Date: n.a.  
##  
## Lang: make  
##  
## Descrip: Makefile fuer target MrCardioAlgos  
##-----  
  
##-----  
## enter include paths  
##  
INCLPATHS (-I /n4/pkg/MrServers/MrVista/include/)  
  
##-----  
## source files for Argus Inline Cardio functionality  
##  
CPPSOURCESFROM('Filename', Segmentation)  
...  
  
CPPSOURCESFROM('Filename', BackEndCommon)  
...  
  
##-----  
## enter target name  
##  
LIB(MrCardioAlgos)  
  
##-----  
## Copyright (C) Siemens AG 1999 All Rights Reserved. Confidential  
##-----  
VISTA_GCC_LINUX_AMD64(LDLIBS(stlport))  
VISTA_GCC_LINUX_AMD64(LDLIBS(ace))
```

*Abbildung 5.1: Aufbau eines make-Files für ein C++ Projekt*

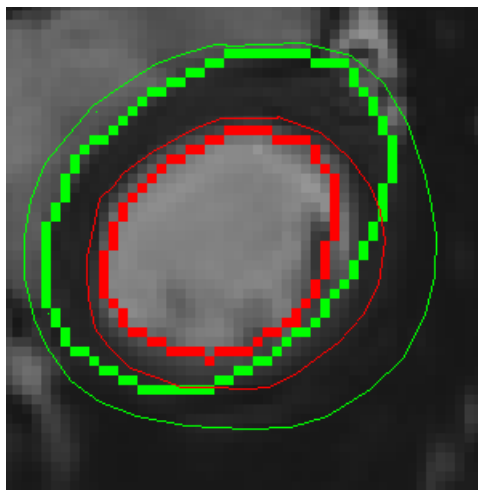
Bei der erzeugten Bibliothek handelt es sich um eine dynamisch Bibliothek (Dynamic-Link Library, kurz DLL), die bei ihrer ersten Verwendung in den Arbeitsspeicher geladen wird. Die bereitgestellte Bibliothek ist dabei nur einmal im Speicher verfügbar und kann von mehreren Programmen genutzt werden.

<i>Typ</i>	<i>Windows</i>	<i>Linux</i>
Boolean	BOOL	bool
True	TRUE	true
False	FALSE	false
...	...	...

*Tabelle 5.2: Verschiedene Schreibweisen von Datentypen unter Windows und Linux*

### 5.1.2 Plattformabhängige Besonderheiten

Nach einer erfolgreichen Übersetzung des neuen Projekts wurde die erzeugte Bibliothek sowohl unter Microsoft Windows als auch unter Linux getestet. Dabei wurde festgestellt, dass eine Segmentierung unter Windows erfolgreich verläuft, während der Test unter Linux fehlschlägt. Die Konturen bei einem Testlauf unter Windows stimmen genau mit der Größe der linken Herzkammer überein, bei der Ausführung unter Linux sind jedoch deutliche Einschnitte in den Herzkammer-Bereich sichtbar. Die dünne Linie in Abbildung 5.2 repräsentiert dabei die korrekte Kontur, während die dicke Linie das Ergebnis einer unter Linux durchgeführten Segmentierung zeigt. Der Unterschied zwischen beiden Konturen ist deutlich erkennbar.



*Abbildung 5.2: Falsche Kontur bei der Segmentierung unter Linux*

Aus dieser Beobachtung ergibt sich die folgende Fragestellung :

1. Wird das Ergebnis durch den Unterschied von einem 32Bit System zu einem 64Bit System hervorgerufen?
2. Verhält sich der Segmentierungsalgorithmus unter Linux anders als unter Windows?
3. Sind bei der Integration der bestehenden Algorithmen Fehler unterlaufen, die bei einem Einsatz unter der Linux Architektur zu einem Fehlverhalten führen?

Um eine schnelle Problemlösung herbeizuführen, wurden verschiedene Testszenarien zusammengestellt. Diese verlangen einen Funktionstest der Bibliothek auf einem 32Bit und einem 64Bit Windows sowie auf einem 64Bit Linux System. Ein 32Bit Linux System steht für diesen Test nicht zur Verfügung. Das Ergebnis kann Tabelle 5.3 entnommen werden. Darin ist ersichtlich, dass die Fehlerursache im Bereich der Architektur zu suchen ist und für einen funktionsfähigen Einsatz gefunden und behoben werden muss.

<i>Betriebssystem</i>	<i>Resultat</i>
Windows 32Bit	erfolgreich
Windows 64Bit	erfolgreich
Linux 32Bit	nicht verfügbar
Linux 64Bit	fehlgeschlagen

*Tabelle 5.3: Test der zugelieferten Algorithmen auf verschiedenen Betriebssystemen*

Für eine systematische Lösung dieses Problems wurde zuerst die Integration der bestehenden Algorithmen in der Linux Laufzeitumgebung auf Fehler hin überprüft. Dazu wurden alle Veränderungen der Klassen während der Integrationsphase auf mögliche Fehler untersucht und anschließend erneut getestet. Das Resultat der Überprüfung war negativ. Es konnte keine erfolgreiche Segmentierung unter einem Linux 64Bit System erreicht werden.

Ein zweiter Lösungsweg beschäftigt sich mit der Fehlersuche direkt im zu gelieferten Quellcode. Dabei spielt die Entstehung der Algorithmen eine nicht ganz unwesentliche

Rolle. Die ursprünglichen Algorithmen zur Erkennung und Segmentierung der linken Herzkammer wurden in der Programmiersprache Fortran geschrieben. Im Laufe der Entwicklung mussten diese Algorithmen in die von Siemens Medical vorgeschriebene Programmiersprache C bzw. C++ überführt werden. Dafür kam ein Tool zur Konvertierung des Fortran-Codes in C-Code zum Einsatz. Die Untersuchung einiger für die Berechnung relevanter Codesegmente führte zu dem Ergebnis, dass eine Funktion zur Berechnung des Absolutwerts fehlerhaft konvertiert wurde. (siehe Abbildung 5.3 ).

```
1 //Alte version aus Fortran Konvertierung
2
3 if ((d__1 = *xeval - x[1], abs(d__1)) <= eps)
4 {
5     *ieval = 1;
6     return 0;
7 }
```

Abbildung 5.3: *if(...)* Abfrage nach Fortran-to-C Konvertierung

*Die Variable „d\_\_1“ ist vom Typ double.*

Bei der Übersetzung des Quellcodes erkennt der Windows-Compiler den Fehler und ersetzt automatisch die Funktion `abs(...)`, die für die Berechnung von ganzen Zahlen vorgesehen ist, durch die richtige Funktion `fabs(...)` für Fließkomma Zahlen. Der Linux-Compiler erkennt diesen Fehler nicht und verwendet wie vorgesehen die Funktion `abs(...)` für die Berechnung. Dabei werden jedoch die Nach-Komma Stellen ignoriert und durch 0 ersetzt. Das Ergebnis dieser falschen Verwendung kann in Abbildung 5.4 betrachtet werden.

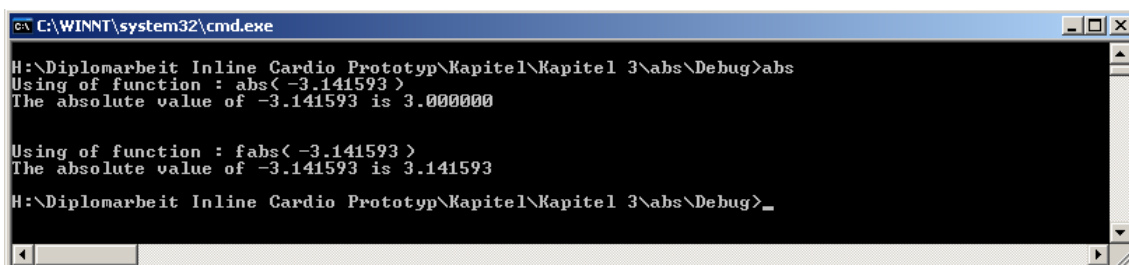


Abbildung 5.4: *Vergleich der Funktionen `abs()` und `fabs()`*

Deutlich sichtbar ist der Unterschied zwischen den Ergebniswerten 3.141593 und 3.000000. Dieser Fehler führte zu einem falschen Wert in der `if ( ... )` Anweisung und damit zu einem Fehlverhalten während der Laufzeit zwischen den Systemen Windows und Linux. Das hier zur Berechnung von `abs( )` und `fabs( )` verwendete Programm kann im Anhang C sowie auf der beigelegten CD eingesehen werden.

Nach einer Korrektur der Funktion `abs( )` zu `fabs( )` wie in Abbildung 5.5 dargestellt, wurde ein erfolgreicher Durchlauf sowohl unter Windows 32Bit und 64Bit als auch unter Linux 64Bit verzeichnet.

```
10 //Neue version nach Modifikation der Absolute Funktion für Windows und Linux
11
12 if ((d_1 = *xeval - x[1], fabs(d_1)) <= eps)
13 {
14     *ieval = 1;
15     return 0;
16 }
```

Abbildung 5.5: `if (...)` Abfrage modifiziert für eine Verwendung unter Linux

### 5.1.3 Erkennung und Vererbung von Geweberegionen

Der Segmentierungsalgorithmus besteht aus einer Funktion zur automatischen Erkennung der Außenkante der linken Herzkammer, der Außenkante des Papilar-Muskels (liegt um die Kammer herum) sowie einigen speziellen Funktionen für die Vererbung von Segmentierungsinformationen auf andere Schichtbilder. Diese Vererbung ist von besonderer Bedeutung, da die Bildinformationen nicht immer ausreichen, um bei der Herzkammer-Suche ein akzeptables Ergebnis hervorzubringen. Daher werden die Konturinformationen<sup>21</sup> von einem Bild zum Nächsten weitergegeben, um einen optimalen Zwischenwert basierend auf den weitergegebenen Konturen und den neuen Konturen zu berechnen. Dieser ist dann sowohl an den vorhergehenden als auch an den aktuellen Wert angenähert. Auf diese Weise kann zum Beispiel einer versehentlichen Segmentierung der Aorta entgegengewirkt werden. Der Algorithmus erkennt selbstständig, dass eine falsche Region als „angebliche“ linke Herzkammer erkannt wurde und korrigiert diesen Fehler durch die Region-Informationen aus den

---

<sup>21</sup> Die Kontur ist die begrenzende Line um eine bestimmte Region.

vorherigen Bilddaten. Somit ist fast immer gewährleistet, dass der Algorithmus die richtige Region erkannt hat und die Flächen- bzw. Volumenberechnung dadurch nicht negativ beeinflusst wird. Ein solcher Fehler könnte unter Umständen zu einer falschen Diagnose bezüglich der Gesundheit des Herzens führen.

Eine Segmentierung erfolgt in zwei Phasen. Die erste Phase besteht aus einer Initialisierung von benötigten Parametern, welche eine grundlegende Voraussetzung für die Segmentierung sind. In diesen Parametern sind unter anderem Informationen zum Typ des Bildes (MR oder CT Bild) sowie einige Parameter zur erfolgreichen Segmentierung und Vererbung enthalten. In der zweiten Phase wird mit den zuvor geladenen Informationen die Position der linken Kammer bestimmt. Die entstandenen Punkte werden später durch Linien verbunden und ergeben dann eine innere und eine äußere Kontur (innere Kontur liegt direkt um die linke Herzkammer, äußere Kontur liegt um den Papillar-Muskel) (vgl. auch Abbildung 5.7). Für das hier entwickelte Software-Modul und dessen auszuführende Berechnungen ist nur die innere Kontur von Bedeutung. Es werden jedoch auf Grund von Vorgaben aus dem Anwender-Bereich sowohl die innere als auch die äußere Kontur dargestellt und in die Datenbank geschrieben.

### **5.2 Einbindung des ICM in die MR Herzbild-Nachverarbeitung**

Für die Implementierung einer neuen Funktion oder eines neuen Moduls in ein bestehendes Softwaresystem müssen die vorliegenden Umgebungsmerkmale genau beachtet werden. Das hier entwickelte Software-Modul muss zusammen mit anderen bereits existierenden Modulen kooperieren und darf zu keinen Behinderungen oder Ausfällen führen. Dabei ist vor allem die nahtlose Integration in das bereits existierende Laufzeitsystem von besonderer Bedeutung.

### 5.2.1 Integration in das bestehende Softwaresystem

Ausgangsbasis für die zu entwickelnde Software ist ein leeres Software-Modul, das mittels eines Konfigurators in eine bestehende Kette von Modulen eingehängt wird. Die Struktur eines leeren Moduls kann im Anhang D eingesehen werden.

Über den Konfigurator erhält man Zugriff auf Übergabeparameter an das Modul und kann die genaue Position innerhalb der Kette bestimmen. In Abbildung 5.6 ist eine entsprechende Kette von Modulen dargestellt, in der die Software für das Inline Cardio schon eingehängt ist. Die am MRT gemessenen Bilder durchlaufen die gesamte Kette von nachverarbeitenden Algorithmen und ergeben am Ende das für den Anwender sichtbare Bild.

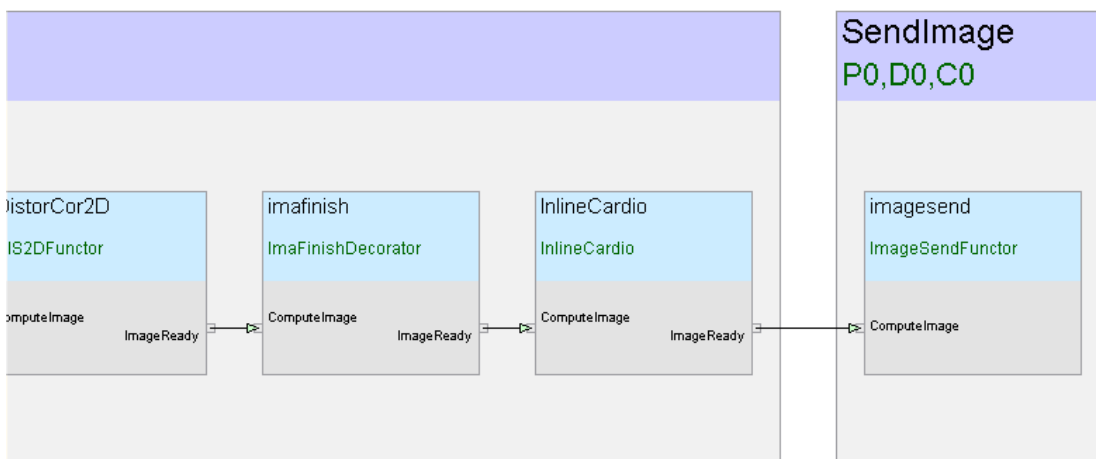


Abbildung 5.6: Kette von mehreren Modulen einschließlich des Cardio-Moduls

Das Software-Modul besteht aus einer Sourcecode- und einer Headerdatei wie in Tabelle 5.4 aufgelistet. Die Implementierung der Logik erfolgt nur in der Sourcecode-Datei. Die Headerdatei stellt die Schnittstelle des Software-Moduls nach außen hin dar. Funktionen und Variablen, öffentlich oder privat, werden hier deklariert.

<i>Type</i>	<i>Directory</i>	<i>Extension</i>
Sourcecode	<b>/ApplikationFunctor/InlineCardio/</b>	<b>*.cpp</b>
Header	<b>/include/Applicationfunctor/InlineCardio/</b>	<b>*.h</b>

Tabelle 5.4: Struktur des Software-Moduls

Innerhalb des Software-Moduls werden die folgenden einzelnen Stadien durchlaufen :

- Konstruktor : Einsprungspunkt in das Software-Modul
- FirstCall : Funktion die anschließend an den Konstruktor durchlaufen wird
- ComputeImage : Übergabe der Bilddaten an das Modul
- SendImage : Weiterleitung der Bilddaten an das nächste Modul
- EndOfJob : Funktion die ausgeführt wird, nachdem alle Bilddaten weitergeleitet wurden
- Destruktor : Ausstiegspunkt aus dem Software-Modul

Im Konstruktor werden alle benötigten Variablen-Initialisierungen vorgenommen. Die Aufgabe der Vorbelegung der Variablen und die Reservierung von Speicherplatz für Arrays oder Vektoren wird von der Funktion FirstCall übernommen. Sie wird im Anschluss an den Konstruktor durchlaufen.

Die Funktion ComputeImage ist das Kernstück des Moduls. Hier findet eine schrittweise Verarbeitung der Bilddaten statt. In erster Linie wird überprüft, ob während des letzten Durchlaufs ein Fehler aufgetreten ist. Sollte dieses der Fall sein, wird die Verarbeitung aller folgenden Bilddaten eingestellt und die original Bilder 1:1 an das nachfolgende Software-Modul weitergeleitet. Diese Maßnahme dient der Vermeidung von Folgefehlern.

Die Bilddaten werden an spezielle Funktionen des Segmentierungs-Algorithmus übergeben, welcher aus den Daten die Position und die Fläche der linken Herzkammer ermittelt. Die Positionsangaben der Konturen werden zusammen mit den bereits vorhandenen beschreibenden Bilddaten gespeichert und durch die Linienfarbe der

Kontur vervollständigt. Diese farbige Linie wird später auf den Bildern sichtbar sein und stellt damit eine Kontrollmöglichkeit für den Anwender dar. Die grafischen Darstellung werden als „Overlay“ bezeichnet, da sie wie eine Folie über das original Bild gelegt werden kann. So bleiben die vom MRT gemessenen medizinischen Bilder unverändert und können jederzeit als Original abgerufen werden. In Abbildung 5.7 ist ein MR-Bild mit aufgeprägten Konturen der linken Herzkammer zu sehen. Die rote Linie bezeichnet die Umrandung der Linken Kammer, die grüne Linie stellt den Rand den umgebenden Papilar-Muskels dar.

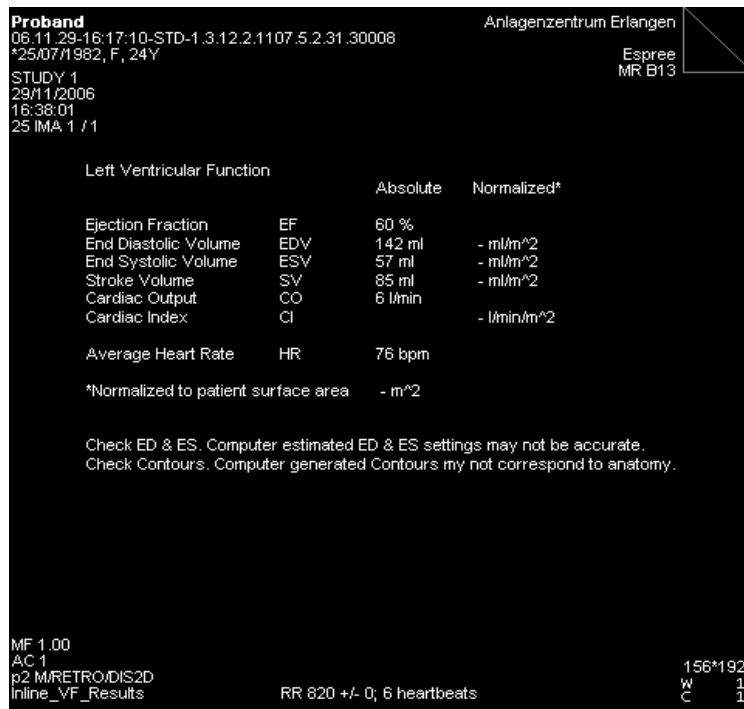


*Abbildung 5.7: Konturen um die Linke Kammer und Kammermuskel*

Die Funktion EndOfJob wird erst nach einer vollständigen Abarbeitung aller Bilddaten ausgeführt. Sie ist für die die Erstellung eines Bildes mit den ermittelten Resultaten zuständig. Dieses Bild soll dem Anwender die Möglichkeit geben, die ermittelten Herzwerte in einer übersichtlichen Darstellung zu beurteilen. Abbildung 5.8 zeigt das Beispiel eines Auswertungsergebnis.

Im Destruktor werden alle Variablen oder für die Verarbeitung der Daten belegter

Speicher wieder freigegeben. Unter C und C++ ist es im Vergleich zu Java oder C#<sup>22</sup> erforderlich, den über das Schlüsselwort „new“ reservierten Speicher wieder freizugeben. Dies kann bereits innerhalb der Funktion geschehen wenn das Objekt nicht mehr benötigt wird, spätestens jedoch im Destruktor.



Left Ventricular Function		Absolute	Normalized*
Ejection Fraction	EF	60 %	
End Diastolic Volume	EDV	142 ml	- ml/m <sup>2</sup>
End Systolic Volume	ESV	57 ml	- ml/m <sup>2</sup>
Stroke Volume	SV	85 ml	- ml/m <sup>2</sup>
Cardiac Output	CO	6 l/min	
Cardiac Index	CI		- l/min/m <sup>2</sup>
Average Heart Rate	HR	76 bpm	
*Normalized to patient surface area			- m <sup>2</sup>

Check ED & ES. Computer estimated ED & ES settings may not be accurate.  
Check Contours. Computer generated Contours may not correspond to anatomy.

MF 1.00  
AC 1  
p2 MRETRO/DIS2D  
Inline\_VF\_Results

RR 820 +/- 0; 6 heartbeats

156\*192  
W 1  
C 1

Abbildung 5.8: Ergebnissbild der vollautomatischen Auswertung

### 5.2.2 Funktionen des ICM

Die Aufgabe des Inline Cardio Moduls besteht in der Berechnung der Herz-Leistungsfähigkeit basierend auf dem Herz-Kammer-Volumen der linken Herzkammer zur enddiastolischen und endsystolischen Phase. Die beiden Volumina werden in ein bestimmtes Verhältnis zueinander gesetzt und ergeben die EF. Anhand dieses Wertes und der weiteren zur Laufzeit berechneten Kenngrößen kann ein entsprechend geschulter Anwender eine Aussage über die Vitalität des Herzens treffen. Der Vorteil

<sup>22</sup> C# (ausgeschrieben : C-Sharp) ist einer Weiterentwicklung von C++ und verfügt über ein automatisches Speichermanagement.

einer „Inline“ Berechnung, also eine Auswertung während des Datenverarbeitungsprozesses, liegt damit in der schnellen Verfügbarkeit der medizinischen Kenngrößen über den Gesundheitszustand des Herzens.

Das Herz wird bei der Untersuchung in mehreren Schichten zu verschiedenen Zeitpunkten gemessen. Daraus ergibt sich eine Matrix von Bildern, die auf der x-Achse in zeitlicher Reihenfolge und auf der y-Achse in den jeweils aufeinander folgenden Schichten angeordnet sind. Zur Verdeutlichung soll die Abbildung 5.9 dienen.



*Abbildung 5.9: Schichtbilder angeordnet nach Zeitlicher Auflösung (X-Achse) und entlang der Schichten durch das Herz (Y-Achse)*

Alle gemessenen Bilddaten durchlaufen die Kette der nachverarbeitenden Algorithmen und somit auch das Inline Cardio Modul. Der entwickelte Algorithmus übergibt die Bilddaten an den Segmentierungs-Algorithmus und erhält eine Menge an Koordinatenpunkten zurück, die die linke Herzkammer markieren. Aus den Koordinaten lässt sich die Fläche der Kammer jedes Bildes berechnen und daraus wiederum das Volumen der gesamten Herzkammer. Durch die zeitliche Auflösung

können zwei wichtige Volumina zur Verfügung gestellt werden: das Volumen zum enddiastolischen Zeitpunkt und das Volumen zum endsystolischen Zeitpunkt. Mit Hilfe der Formel zur Berechnung der EF (siehe dazu Kapitel 2.1.2) , welche auf dem Verhältnis von beiden Volumina basiert, kann am Ende eine Aussage über die Gesundheit des Herzens getroffen werden.

## 6 Test und Robustheitsanalyse

Die Abfolge der verschiedenen Testverfahren orientiert sich dabei an dem in Kapitel 3.3 vorgestellten V-Modell. Jeder Phase innerhalb der Entwicklungsperiode steht eine entsprechende Phase von Testaktivitäten gegenüber. Die Abbildung 6.1 verdeutlicht die einzelnen Stufen der Testaktivitäten, die in den folgenden Kapiteln ausführlicher beschrieben werden.

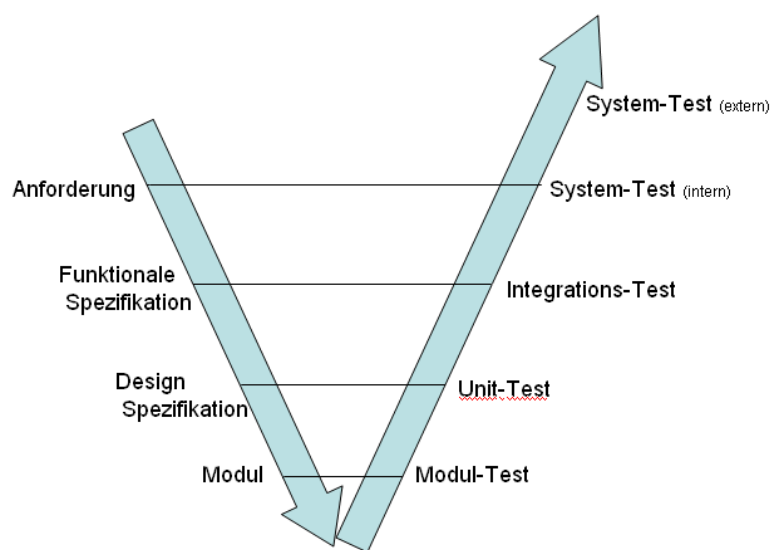


Abbildung 6.1: Gegenüberstellung von Entwicklungs- und Testphasen bei Siemens Medical

### 6.1 Test des Software Moduls

Im Anschluss an die Implementierung der zugelieferten Algorithmen und des ICM werden die einzelnen Funktionen sowie das implementierte Design auf Fehler und Ausfälle überprüft. Dabei kommt eine speziell für diesen Test entwickelte Software-Umgebung zum Einsatz. Diese Umgebung simuliert die Softwaregegebenheiten an einem Magnetresonanz-Tomographen lokal auf dem Desktop-PC des Software-Entwicklers. Einzelne Module mit weiteren nachverarbeitenden Algorithmen werden identisch zu einer Installation auf einem realem MRT in einer Kette, die auch als „Pipeline“ bezeichnet wird, eingehängt und nach dem Start der Simulationsumgebung

aufgerufen. Darunter befindet sich auch das für diese Arbeit entwickelte Software-Modul. Als Eingangsgröße dient ein bereits aufgezeichneter Datensatz mit MRT Schnittbildern vom Herzen, welcher direkt im Anschluss an eine MRT Untersuchung ohne jegliche Veränderung aufgezeichnet wurde. Dieser Rohdatensatz enthält die Bilddaten, auf denen die Segmentierung der linken Herzkammer ausgeführt werden soll.

Über einen Debugger, der zu den Standardanwendungen des Visual Studio© von Microsoft gehört, kann der gesamte Quellcode oder bei Bedarf auch jede einzelne Codezeile in dem Software-Modul durchlaufen werden. Auf diese Weise können die von der Simulations-Umgebung erzeugten oder berechneten Daten nicht nur Schritt für Schritt nachvollzogen, sondern bei Bedarf auch verändert werden. Die Simulationsumgebung bietet dem Softwareentwickler eine erste und schnelle Kontrollmöglichkeit, um die von ihm entwickelten Software zu überprüfen. Aufgetretene Fehler können rasch eingegrenzt und behoben werden, ohne viel Zeit und Kosten an einer realen Anlage zu verursachen.

Der Test soll vor allem zeigen, ob das Software-Modul korrekt in das bestehende System eingebunden ist, ob es angesprochen und durchlaufen wird und ob alle Funktionen innerhalb des Moduls gemäß ihrer Implementierung aufgerufen werden.

Probleme, die das Design betreffen sowie Fehler bei der softwaretechnischen Implementierung des Codes, sind auf schnelle und effektive Weise aufzufinden und zu beheben. Dieser Test lässt bereits Rückschlüsse auf eine korrekte und fehlerfreie Funktion der implementierten Algorithmen zu und ermöglicht schon zum derzeitigen Moment eine Aussage über die Integration des neuen Moduls in das bestehende Software-System. Weiterhin erhält der Entwickler einen ersten Eindruck von der Kommunikation bzw. der Zusammenarbeit der einzelnen Module untereinander. An dieser Stelle ist deutlich erkennbar, ob das Design des neu entwickelten Moduls in der vorliegenden Weise für den weiteren Einsatz geeignet ist oder gegebenenfalls verändert werden muss.

## 6.2 Unit-Test an einer Simulationsanlage

Die nächste Test-Instanz nach dem Modul-Test ist der Unit-Test. Hier wird das ICM an der Nachbildung eines MRTs getestet. Der Modul-Test findet an einer so genannten Simulationsanlage statt. Dieses recht aufwendige Testverfahren dient dem Auffinden und Beheben von Fehlern bei der Zusammenarbeit des Software-Moduls mit den vorhandenen hardwaretechnischen Gegebenheiten.

Genauer gesagt wird das MRT-Gerät, an dem die Software getestet werden soll, durch eine komplexe Hardware-Lösung nachgestellt. Dieses Vorgehen vereint gleich mehrere Vorteile. Auf der einen Seite entfällt die Bereitstellung eines originalen medizinischen Gerätes, das nicht nur viel räumlichen Platz für die Installation sondern auch diverses technisches Zubehör benötigen. Zum Anderen werden die Anschaffungskosten auf einen Bruchteil reduziert und man benötigt weder eine Versuchsperson noch eine entsprechende Schulung für die Verwendung des Geräts.

Die Hardware für eine MRT Simulationsanlage besteht aus einem Rausch-Generator und einem Softwaresystem. Eine wie in Abbildung 6.2 und 6.3 dargestellte Simulationsanlage ist in einem separaten klimatisierten Raum untergebracht. Erforderlich macht diese Abtrennung die akustische Belastung und die starke Wärmeentwicklung der Anlage. Durch die Klimaanlage kann der Raum auf eine Temperatur von im Schnitt 10°-12° herunter gekühlt werden. Ein Ausfall der Klimaanlage hätte im schlimmsten Fall einen Absturz des Systems und damit eine mögliche Beschädigung der Hardware zur Folge. Insgesamt enthalten beide Abbildungen vier getrennt von einander arbeitende Simulationsanlagen mit jeweils vier Rauschgeneratoren und vier Computersystemen. Die Anlagen werden durch wöchentliche Neuinstallation der Betriebssoftware immer auf einem aktuellen und unveränderten Softwarestand gehalten. Der Unterschied zwischen einer Simulationsanlage und einem realen MRT liegt in der Generierung von Bilddaten. Im Gegensatz zu einem MRT werden von der Simulationsanlage nur so genannte Rauschbilder erzeugt.

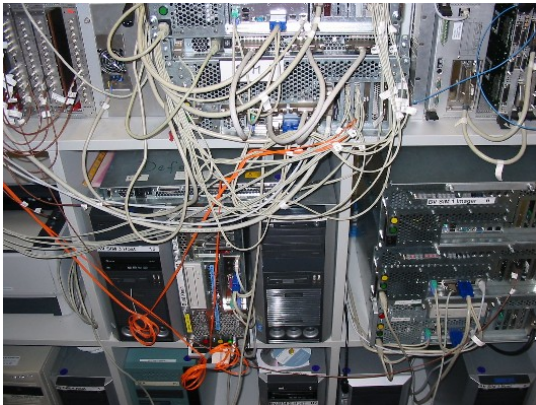


Abbildung 6.2: Simulationsanlage Teil 1

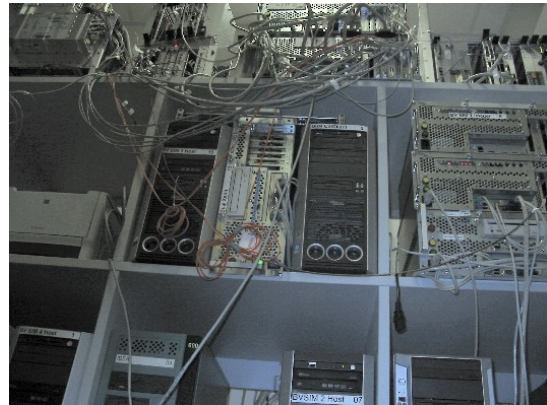


Abbildung 6.3: Simulationsanlage Teil 2

Das vom Rausch-Generator erzeugte Bild (siehe Abbildung 6.4) besteht dabei nur aus schwarzen und weißen Pixeln. Es wird also keinerlei Gewebe- oder Weichteilstruktur abgebildet.

Auf dem Computersystem ist identisch zu einem ausgelieferten MRT die System-Software installiert. Für einen Softwaretest werden die bereits installierten Module durch veränderte ersetzt bzw. neue Module in die Software integriert. Anschließend wird ein vom Anwender ausgewählter Messvorgang gestartet.

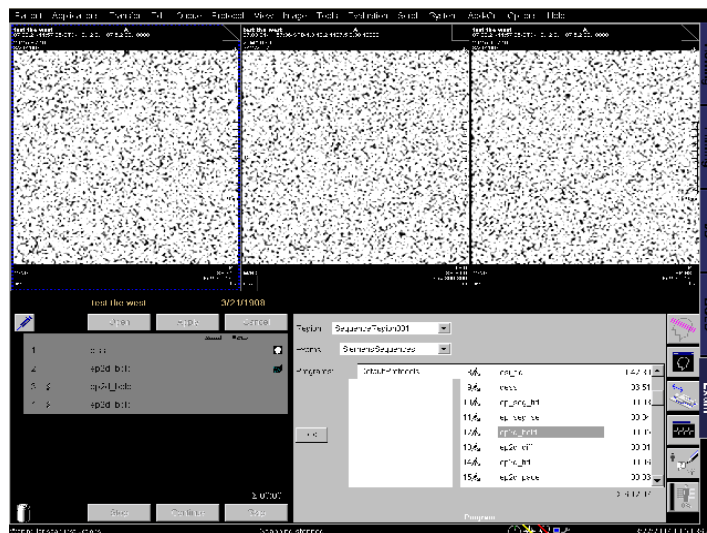


Abbildung 6.4: Erzeugtes Rauschbild während einer Messung an einer Simulationsanlage

Für die Fehlersuche bei der Bildverarbeitung dienen während des Messvorgangs erstellte Log-Files. Diese stellen eine wichtige Voraussetzung zur Fehlersuche dar und werden zudem für die Überprüfung der gesamten Systemsoftware benötigt. Es ist bei der Softwareentwicklung nicht immer davon auszugehen, dass alle Module und Funktionen reibungslos miteinander harmonieren. Selbst wenn dies zum Zeitpunkt X gegeben ist, kann eine Veränderung oder Erweiterung der bestehenden Module zu nicht vorhersagbaren Fehlern führen, die sich meistens erst im laufenden Betrieb zeigen. Die Praxis zeigt immer wieder, dass ein Software-Modul für sich allein gemäß seinen Bestimmungen und Vorgaben arbeitet. Wenn man jedoch ein oder mehrere Module kombiniert, kann es während der Ausführung zu einem Fehlverhalten oder zum Abbruch kommen. Um das Auftreten dieser Probleme zu verringern, werden alle Neuentwicklungen und alle Modifikationen zuerst ausführlich an einer Simulationsanlage getestet, bevor sie für eine weitere Verwendung freigegeben werden.

### **6.3 Integrations-Test am MRT**

Bei einem Integrations-Test wird das fehlerfreie Zusammenarbeiten der einzelnen Systemkomponenten überprüft. Für den Integrations-Test ist es erforderlich, dass bereits jede einzelne Komponente einem erfolgreichen Unit-Test unterzogen wurde. So werden Schritt für Schritt einzelne Komponenten zusammengeführt und immer wieder neu getestet. Dieses Vorgehen wird so lange wiederholt, bis das System vollständig integriert ist. Die Tests laufen nach einem bestimmten festgelegten Testplan ab und werden von speziellen Test-Teams an verschiedenen „realen“ MRTs durchgeführt. Im Vergleich zu anderen vorangegangenen Tests wird dieser nicht vom Entwickler selber durchgeführt. Das erhöht die Wahrscheinlichkeit, einen objektiven und unabhängigen Eindruck von der Software zu gewinnen und gegebenenfalls auch Möglichkeiten in den Test einzubeziehen, welche der Entwickler selber nicht in Betracht gezogen hat.

Für diesen Test wird das ICM an einem MRT installiert und im Anschluss an die Patientenregistrierung eine standardmäßige Herzuntersuchung gestartet, bei welcher das

ICM automatisch mit zum Einsatz kommt. Der Anwender kann während der Untersuchung im Inline-Display<sup>23</sup> die bereits verarbeiteten Bilder in Augenschein nehmen. Auf den angezeigten Bildern sind die innere und äußere Kontur erkennbar (vgl. dazu Abbildung 5.7). Anhand der Kontur kann man einen ersten Eindruck von der Funktionsfähigkeit des ICM gewinnen. Fehler bei der gemeinsamen Ausführung mit anderen Modulen lassen sich durch den Integrationstest schnell erkennen. Alle aufgetretene Fehler werden vom Test-Team bei jeder Messung protokollieren, um dem Entwickler im Anschluss ein möglichst ausführliches Feedback geben zu können.

Von den verschiedenen durchgeführten Tests des ICM ist ein Testfall besonders nennenswert. Dabei handelt es sich um die Überprüfung des Zeit- und Schichtpositionsstempels, welcher durch die Systemsoftware für jedes Bild einzeln erzeugt wird. Der Grund für diese Kontrolle ist eine Verarbeitung der Schichtbilder durch das ICM in einer modifizierter Reihenfolge. Dabei wird die Abfolge der zu verarbeitenden Schichten verändert, um ein bestmögliches Ergebnis bei der Segmentierung zu erzielen. Um dieses Vorgehen etwas zu verdeutlichen, wird die Verarbeitung der einzelnen Schichten in Tabelle 6.1 grob dargestellt. Die einzelnen Herzschnittbilder müssen nach der Segmentierung in der gleichen Reihenfolge vorliegen, wie das auch ohne den Einsatz des ICM geschehen wäre, bevor sie in die Datenbank geschrieben werden können.

<i>Eingangsreihenfolge</i>	<i>Verarbeitung im ICM</i>	<i>Ausgangsreihenfolge</i>
Schicht 1	Schicht 2	Schicht 1
Schicht 2	Schicht 1	Schicht 2
Schicht3	Schicht 3	Schicht 3
...	...	...
Schicht N	Schicht N	Schicht N

*Tabelle 6.1: Reihenfolge der einzelnen aquirierten Schichten bei der Verarbeitung*

<sup>23</sup> Das Inline-Display zeigt alle aufgezeichneten Bilder in zeitlicher Reihenfolge an.

## 6.4 System-Test

Nach dem erfolgreichen Abschluss des Integrations-Test folgt der Systemtest. Er stellt einen abschließenden Test von Software-Entwicklern und Qualitätsprüfer in einer realen Umgebung dar. Im Vergleich zum Integrations-Test sind beim System-Test nur noch die externen Schnittstellen des Systems oder das Benutzerinterface sichtbar. Die Grundlage für den System-Test sind Pflichtenheft, Produkt-Modell, Konzept der Benutzeroberfläche und das Benutzerhandbuch. (vgl. auch [17], S.537)

Der System-Test wird bei Siemens nochmals unterteilt in einen internen und einen externen System-Test. Der interne Test findet innerhalb des Unternehmens an speziell dafür vorgesehenen MRTs statt. An diesen Geräten ist jegliche Veränderung der System-Software untersagt, da sie zum Teil von Ärzten oder zu Kunden-Demonstrationen verwendet werden. Die hier installierte Software entspricht dem ausgelieferten Produktstand.

Der externe System-Test erfolgt nur noch bei ausgewählten Kunden in Krankenhäusern oder Medizinischen Einrichtungen („Work-in-Progress“) und soll Aufschluss über Fehlerquote, Robustheit und Reproduzierbarkeit bei der Untersuchung von gesunden und pathologischen Patienten im laufenden klinischen Betrieb geben. Er kann mit einem Abnahmetest gleichgesetzt werden (vgl. auch [17], S.542)

## 6.5 Wissenschaftliche Vergleichsdaten

Um eine geeignete Aussage über die korrekte Funktion des ICM treffen zu können, müssen die berechneten Ergebniswerte einem Vergleich mit bestehenden, wissenschaftlich fundierten Normalwerten unterzogen werden.

Die Normaldaten wurden in einer Studie von A.M. Maceira, S.K. Prasad und D.J. Pennell [3] erhoben und stellen die Bezugswerte für die Einordnung der automatisch ermittelten cardiovasculären Ergebnisse dar.

Die in Abbildung 6.5 und 6.6 aufgezeigten Werte sind getrennt nach männlichen und weiblichen Patienten und in Alterskategorien von jeweils zehn Jahren eingeteilt. Zu jeder Kategorie gibt es einen Idealwert (oberer Zahlenwert) und einen Toleranzbereich (unterer Zahlenwerte), der durch eine obere und untere Grenze repräsentiert wird. Für eine Bewertung der Gesundheit des Herzens werden alle ermittelten medizinischen Kenngrößen wie EDV, ESV, SV und EF mit den Normaldaten aus [3] verglichen.

Ein optimales Ergebnis wird erreicht, wenn sich der zu vergleichende Wert aus der automatischen Berechnung dem Idealwert annähert. Jedoch ist es auf Grund der verschiedenen Ausprägungen des menschlichen Herzens möglich, das nicht immer der Idealwert erreicht werden kann. Für diesen Fall genügt es, das sich der ermittelte Wert im Toleranzbereich befindet.

Males	20–29 years	30–39 years	40–49 years	50–59 years	60–69 years	70–79 years
<b>EDV [mL] SD 21</b>	167 (126, 208)	163 (121, 204)	159 (117, 200)	154 (113, 196)	150 (109, 191)	146 (105, 187)
<b>ESV [mL] SD 11</b>	58 (35, 80)	56 (33, 78)	54 (31, 76)	51 (29, 74)	49 (27, 72)	47 (25, 70)
<b>SV [mL] SD 14</b>	109 (81, 137)	107 (79, 135)	105 (77, 133)	103 (75, 131)	101 (73, 129)	99 (71, 127)
<b>EF [%] SD 4.5</b>	65 (57, 74)	66 (57, 75)	66 (58, 75)	67 (58, 76)	67 (58, 76)	68 (59, 77)
<b>Mass [g] SD 20</b>	148 (109, 186)	147 (109, 185)	146 (108, 185)	146 (107, 184)	145 (107, 183)	144 (106, 183)

Abbildung 6.5: Medizinische Kenngrößen von männl. Probanden (vgl. auch [3])

Females	20–29 years	30–39 years	40–49 years	50–59 years	60–69 years	70–79 years
<b>EDV [mL] SD 21</b>	139 (99, 179)	135 (94, 175)	130 (90, 171)	126 (86, 166)	122 (82, 162)	118 (77, 158)
<b>ESV [mL] SD 9.5</b>	48 (29, 66)	45 (27, 64)	43 (25, 62)	41 (22, 59)	39 (20, 57)	36 (18, 55)
<b>SV [mL] SD 14</b>	91 (63, 119)	89 (61, 117)	87 (59, 115)	85 (57, 113)	83 (56, 111)	81 (54, 109)
<b>EF [%] SD 4.6</b>	66 (56, 75)	66 (57, 75)	67 (58, 76)	68 (59, 77)	69 (60, 78)	69 (60, 78)
<b>Mass [g] SD 18</b>	105 (69, 141)	106 (70, 142)	107 (71, 143)	108 (72, 144)	109 (73, 145)	110 (74, 146)

Abbildung 6.6: Medizinische Kenngrößen von weibl. Probanden (vgl. auch [3])

Für den Vergleich mit den wissenschaftlichen Daten und um eine Aussage über die automatische Vorgehensweise des ICM treffen zu können, wurden verschiedene Probanden am MRT untersucht. Zur Verbesserung der Lesbarkeit ordnen sich die ermittelten Ergebnisse zusammen mit den der Studien von [3] in einer gemeinsamen Auswertung ein. Abbildung 6.7 zeigt diese Auswertung in entsprechend aufbereiteter Form. Dabei lässt sich erkennen, dass alle automatisch ermittelten Werte im gültigen Toleranzbereich liegen und die Werte von EF dem Idealwert angenähert sind.

Name	Geschlecht	EDV	ESV	SV	EF
Proband 1	f	<b>159.921</b> 139 (99, 179)	<b>62.293</b> 48 (29,66)	<b>97.628</b> 91 (63, 119)	<b>61.047</b> 66 (56, 75)
Proband 2	m	<b>135.656</b> 167 (126, 208)	<b>42.793</b> 58 (35, 80)	<b>92.862</b> 109 (81, 137)	<b>68.454</b> 65 (57,74)
Proband 3	m	<b>162.616</b> 163 (121, 204)	<b>67.367</b> 56 (33, 78)	<b>95.248</b> 107 (79, 135)	<b>58.572</b> 66 (57, 75)
Proband 4	m	<b>178.211</b> 167 (126, 208)	<b>71.091</b> 58 (35, 80)	<b>107.119</b> 109 (81, 137)	<b>60.108</b> 65 (57,74)
Proband 5	f	<b>167.583</b> 135 (94, 175)	<b>66.596</b> 45 (27, 64)	<b>100.987</b> 89 (61, 117)	<b>60.260</b> 66 (57, 75)
Proband 6	m	<b>174.357</b> 167 (126, 208)	<b>56.360</b> 58 (35, 80)	<b>117.997</b> 109 (81, 137)	<b>67.675</b> 65 (57,74)
Proband 7	m	<b>178.265</b> 167 (126, 208)	<b>66.022</b> 58 (35, 80)	<b>112.243</b> 109 (81, 137)	<b>62.964</b> 65 (57,74)
Proband 8	m	<b>147.014</b> 167 (126, 208)	<b>53.730</b> 58 (35, 80)	<b>93.283</b> 109 (81, 137)	<b>63.452</b> 65 (57,74)

Abbildung 6.7: Ergebnisse der automatischen Auswertung im direkten Vergleich mit wissenschaftlichen Daten von [3]

Dieses Ergebnis lässt auf eine erfolgreiche Implementierung und ein robustes Vorgehen des ICM bei der automatischen Herzauswertung schließen..

## **6.6 Beurteilung der Qualität des ICM nach ISO 9126**

Für eine Einordnung und Beurteilung der Qualität der für diese Arbeit entwickelten Software wurde das in Kapitel 3.5 beschriebene Qualitätsmodell nach ISO 9126 zu Hilfe genommen. Die Auswertung der Qualitätskriterien nach ISO 9126 kann detailliert in Tabelle 6.2 eingesehen werden. Dabei wird deutlich, dass die ausgewählten Qualitätsmerkmale von Funktionalität, Zuverlässigkeit und Effizienz vollständig erfüllt worden sind.

<i>Qualitätsmerkmal</i>	<i>Ja</i>	<i>Nein</i>	<i>Beschreibung</i>
<b>Funktionalität</b>			
Richtigkeit	x		Eine Bewertung der Richtigkeit ist in diesem Fall nur bedingt möglich, da keine direkten Vergleichsdaten existieren. Es kann höchstens eine Einordnung der Ergebnisse, basierend auf bereits durchgeführten Studien, stattfinden. Diese Einordnung war erfolgreich.
Angemessenheit	x		Es wurden speziell für diese Aufgabe Funktionen entwickelt bzw. bestehende Funktionen implementiert.
Interoperabilität	x		Die Software wurde als Erweiterung für ein vorgegebenes System entwickelt und auf dessen Bedarf und Funktionalität hin angepasst.
Ordnungsmäßigkeit	x		Die gewünschten Anforderungen wurden ordnungsgemäß umgesetzt.
Sicherheit	x		Es finden keine sicherheitskritischen Zugriffe statt. Veränderungen von Daten sind nur im Rahmen der Anforderungen möglich.
<b>Zuverlässigkeit</b>			
Reife	x		Fehlzustände werden durch qualitativ hochwertiges Bildmaterial (Messdaten) und durch ein intelligentes Handling der Daten minimiert.
Fehlertoleranz	x		Fehlzustände werden erkannt und entsprechend behandelt. Resultatswerte werden durch Fehler nicht negativ verändert.
Wiederherstellbarkeit	x		Bei einem aufgetretenen Fehler wird eine Weiterverarbeitung der Daten zurückgestellt und der Anwender informiert.
<b>Effizienz</b>			
Zeitverhalten	x		Der Zeitaufwand wird durch entsprechende Programmieretechnik so gering wie möglich gehalten.
Verbrauchsverhalten	x		Der Ressourcenverbrauch kann durch die Anwendung vom Garbage-Collector und wieder verwendbare Objekte stark minimiert werden.

Tabelle 6.2: Ergebnisse einzelner Qualitätsmerkmale von ISO 9126

## 6.7 Zusammenfassung

Eine Software zu entwickeln, vom Design bis zur fertigen Implementierung, ist eine komplexe und aufwendige Arbeit. Es steckt nicht nur viel Planungsarbeit darin, sondern auch der Ansporn alle Möglichkeiten korrekt bedacht und eventuell auftretende Fehlerfälle weitestgehend abgefangen zu haben. Um dieses Vorhaben so effektiv und erfolgreich wie möglich umzusetzen, wurden genau auf die Entwicklung zugeschnittene Tests ausgearbeitet und in den Entwicklungsprozess einbezogen. Keine Software würde auf dem freien Markt bestehen können, ohne dass sie zuvor ausführlichen und umfangreichen Tests unterzogen wurde. Die hier vorgestellten Testverfahren sind speziell auf Softwareentwicklungen bei Siemens Medical ausgelegt und richten sich wie die Entwicklung nach dem V-Modell. Um dieses recht aufwendige Test-Verfahren zu verdeutlichen, wird in Abbildung 6.8 das Entwicklungs-Modell bei Siemens dargestellt.

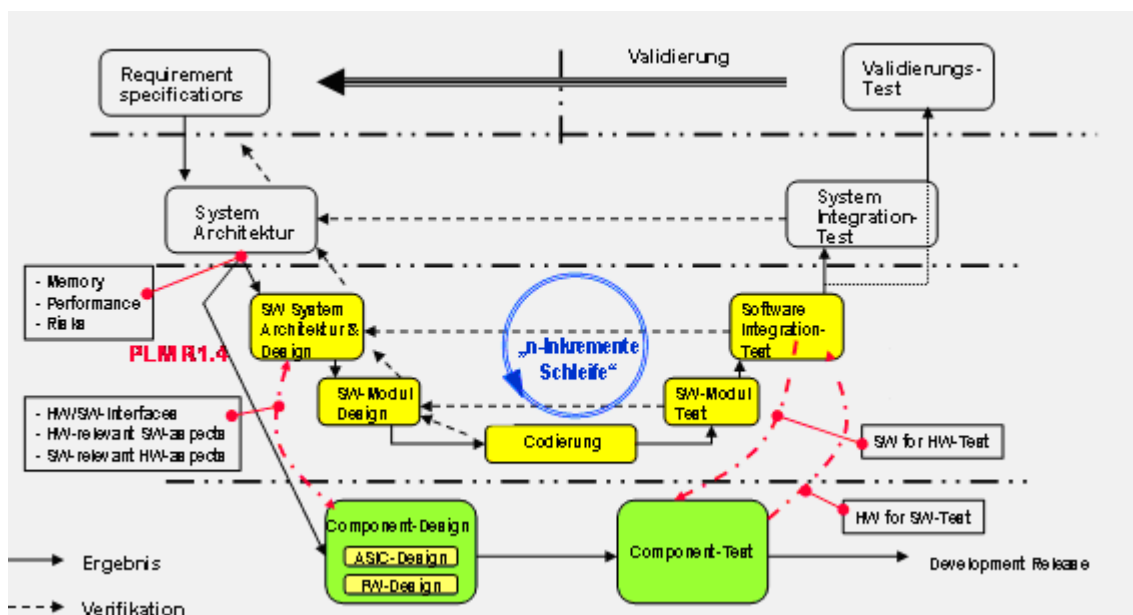


Abbildung 6.8: Modell für die Entwicklung und dem Test von Software bei Siemens

Die Fehlerquote in der ausgelieferten Software muss so gering wie möglich sein, um eine mögliche Fehldiagnose oder einen Systemabsturz zu verhindern. Aus diesem Grund werden alle Neuentwicklungen durch verschiedene Testinstanzen geprüft und

bewertet. Für das ICM konnte festgestellt werden, dass alle Testinstanzen bis zum internen System-Test erfolgreich durchlaufen wurden. Die während des Tests gefundenen Fehler wurden behoben und die Funktionalität erneut getestet. Die ermittelten Konturen der Herzschnittbilder sind durch geschulte MTAs bestätigt worden und können für die Erstellung von medizinischen Diagnosen zur Herz-Leistungsfähigkeit verwendet werden. Jedoch ist es zwingend erforderlich, weitere externe System-Tests im klinischen Betrieb durchzuführen, um eine robuste und reproduzierbare Auswertung sowohl bei gesunden als auch bei pathologischen Patienten zu erreichen.

## 7 Fazit und Ausblick

In der vorliegenden Diplomarbeit wurde die Implementierung eines voll-automatischen Verfahrens zu Bestimmung der Herz-Leistungsfähigkeit in der medizinischen Schnittbilddiagnostik dargelegt. Neben einer ausführlichen Beschreibung der medizinischen Grundlagen des Herzens und einer umfassenden Einführung in die Magnetresonanz-Tomographie wurden einzelne Bereiche der Softwareentwicklung und des Software-Managements diskutiert.

Dabei stand vor allem die Implementierung der Segmentierungsalgorithmen und des Inline Cardio Moduls (ICM) in ein bestehendes Software-System sowie verschiedene Testverfahren zur Überprüfung von Robustheit und Reproduzierbarkeit im Vordergrund. Die Realisierung des automatisch arbeitenden Software-Moduls geht an dieser Stelle weit über eine prototypische Implementierung hinaus. Die erfolgreiche Umsetzung wird nicht zuletzt durch die Übernahme des ICM in die offizielle Produktsoftware „Numaris VB15“ bestätigt. Damit steht dem Anwender ein verbessertes und schnelleres Verfahren bei der Auswertung von medizinischen Schnittbildern im Bereich der Herzdiagnostik mit MRT zur Verfügung. Weiterhin konnte mit dieser Arbeit gezeigt werden, dass eine automatische Auswertung der Herz-Leistungsfähigkeit im Bereich der Magnetresonanz-Tomographie möglich ist.

Die durch dieses Software-Auswerteprogramm ermittelten Ergebnisse konnten durch verschiedene Testverfahren und wissenschaftlichen Studien bestätigt werden und befinden sich derzeit in der klinischen Validierung.

Ziel dieser Arbeit war die Implementierung dieses automatischen Analyseverfahrens im Bereich der Magnetresonanz-Tomographie. Für die nähere Zukunft ist es vorstellbar, die Grundlagen aus der Diplomarbeit in einem erweiterten Rahmen für andere bildgebende Systeme in der Medizintechnik, wie zum Beispiel in der Computertomographie, zugänglich zu machen. Die Ergebnisse und die Struktur der abgeschlossenen Implementierung dieses Programms lassen sich durch den modularen Aufbau und die klaren Schnittstellen zu anderen Software-Komponenten sehr einfach

für weitere Forschungen auf dem Gebiet der automatischen Segmentierung verwenden.

Die rasante Entwicklung der Technik und die dadurch möglich werdende Verbesserung der Verfahren bei der Herzdiagnose lassen viele Ideen und Neuentwicklungen entstehen, von denen jede einzelne auf ihr Potenzial und ihre Möglichkeiten hin untersucht werden kann. Nur so wird es möglich sein, den Bereich der Medizintechnik immer weiter voranzutreiben und dem Patienten durch schnellere und verbesserte Diagnostik die bestmögliche Behandlung zukommen zu lassen.



## Literaturverzeichnis

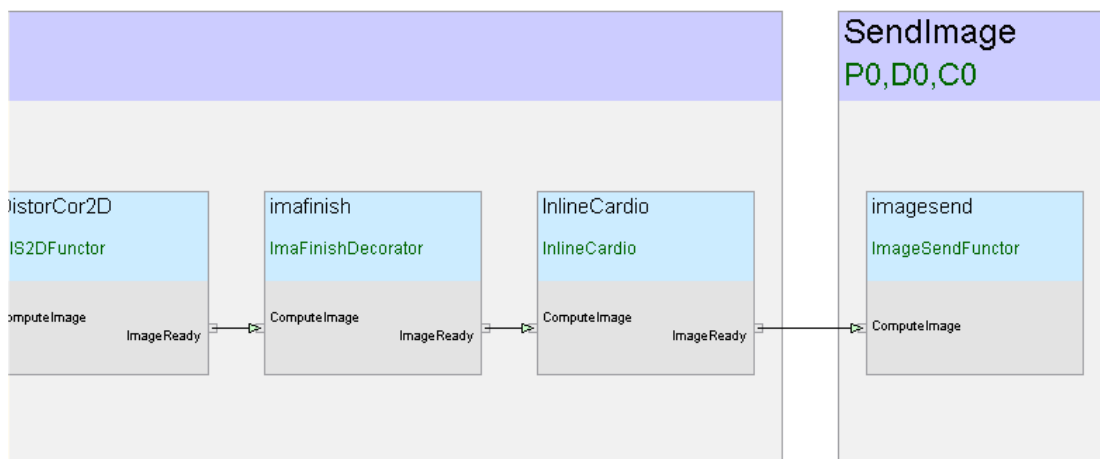
- [1] Lorenz C.H., Walker E.S., Morgan V.L., Klein S.S., Graham T.P.:  
Normal Human Right and Left Ventricular Mass, Systolic Functions and  
Gender Differences by Cine Magnetic Resonance Imagine,  
Journal of Cardiovascular Magnetic Resonance Imagine, 1(1); 7-22,  
1999. (A Guideline)
  
- [2] Ravi Kumar, SCR-IM:  
Cardiovascular Analysis Software – Ventricular Function, Functional  
Specification v2.0  
Siemens Medical Solutions, 2004.
  
- [3] Maceira A.M., Prasad S.K., Kahn M., Penell D.J.:  
Normalized Left Ventricular Systolic and Diastolic Function by Steady State Free  
Precession Cardiovascular Magnetic Resonance,  
Journal of Cardiovascular Magnetic Resonance 8; 417–426, 2006.
  
- [4] Netter F.H.: Atlas der Anatomie des Menschen, 1996.
  
- [5] Schäffler, A., Menche, N.: Mensch Körper Krankheit. 3. Auflage.  
Urban & Fischer Verlag, 1999.
  
- [6] Siemens Medical Solutions: Magnets, Spins, and Resonances  
Siemens AG, Erlangen 2003.
  
- [7] Lehmann T., Oberschelp W., Pelikan E., Repges R.: Bildverarbeitung für die  
Medizin  
Springer Verlag, 1997.
  
- [8] Royce, Dr. Winston W.: Managing the Development of Large Software Systems  
Concepts and Techniques, 1970.
  
- [9] Marciniak, John J. (Hrsg.): Encyclopedia of Software Engineering. Vol. I und II.  
John Wiley & Sons Inc., 1994.
  
- [10] Dumke, R.: Software Engineering – Eine Einführung für Informatiker und  
Ingenieure: Systeme, Erfahrungen, Methoden, Tools.  
2., überarbeitete und erweiterte Auflage.  
Wiesbaden : Vieweg, 2000.

- [11] Balzert H.: Lehrbuch der Software-Technik: Softwareentwicklung.  
Lehrbücher der Informatik. Heidelberg  
Spektrum Akademischer Verlag, 1996.
- [12] Boehm B. W.: Guidelines for verifying and validating software requirements and design specification.  
In: EURO IFIP, 1979.
- [13] Royce, Dr. Winston W.: Managing the Development of Large Software Systems.  
In: Proceedings Westcon,  
IEEE Press, 1970, S. 328–339
- [14] Morneburg H., Bildgebende Systeme für die medizinische Diagnostik.  
Siemens Aktiengesellschaft,  
3. wesentlich überarbeitete und erweiterte Ausgabe.  
Erlangen, 1995.
- [15] Positronen-Emissions-Tomographie  
<http://www.netdokter.de/untersuchungen/pet.htm>  
(Stand: 01.04.2007)
- [16] Elektrisches System des Herzens  
[http://www.guidant.de/Patient/Heart-BV-Basics/heart\\_electrical.aspx](http://www.guidant.de/Patient/Heart-BV-Basics/heart_electrical.aspx)  
(Stand: 01.04.2007)
- [17] Balzert H.: Lehrbuch der Software-Technik,  
Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung  
Spektrum Akademischer Verlag,  
Heidelberg 1998.
- [18] M. Wegner. Contour detection in mr cardiac images of the left ventricle.  
Technical report, CMR Brisbane, April 2002.
- [19] J. Weng, A. Singh, and M. Y. Chiu. Learning-based ventricle detection from cardiac MR and CT images. IEEE Transactions on Medical Imaging, 16, No. 4:378–392, August 1997.

- [20] H. Sillescu, Kernmagnetische Resonanz: Einführung in die theoretischen Grundlagen.  
Springer Verlag Berlin, Heidelberg, 1966.
- [21] Qualitäts-Management Lexikon  
<http://www.quality.de/lexikon/g0005009.htm> (Stand 12.04.2007)
- [22] Siemens AG: Prozesshandbuch für die Erstellung von Anwendersoftware und Realisierung von Projekten.  
1991.
- [23] Bröhl, A.P., Dröschel, W.: Das V-Modell  
Oldenbourg Verlag, 1995.
- [24] Boehm, B.W.: Software Engineering Economics  
Englewood Cliffs: Prentice Hall  
1981.
- [25] Internationale Organisation für Normung  
[www.iso.ch](http://www.iso.ch) (Stand 12.04.2007)

## Anhang A - Designspezifikation ICM

### Description



### *Pipe Service with Inline Cardio Functor as Singleton Pipe*

The Inline CardioFunctor performs unsupervised, fully-automatic left ventricle segmentation and calculates left ventricular function specific values during a running cardiac MR measurement. Thus, user-interactive image post processing typically performed offline in Argus Ventricular Function can be omitted in order to obtain values such as left ventricle ejection fraction. The inline process however does not replace the offline process (Argus) completely. Via an Eva-Protocol parameter (UI) the user can switch on or off the InlineCardio functionality.

The output of left ventricular function values like ejection fraction (EF), end-diastolic volume (EDV), end-systolic volume (ESV) and stroke volume (SV) is only visible after finish the measurement process in a “black image”.

During the segmentation an identity string (meaning) for the ventricle contours in every overlay will be set. In that way the automatic generated contours can be changed later in Argus Ventricular Function taskcard.

The functor uses the left ventricle segmentation algorithm from SCR (Princeton). There are functions available for finding a contour in the first slice and propagation of contours. The propagation of contours can occur between the phases of one slice (SegmentTemporal) and between two slices (SegmentUp, Segmentdown).

**Input-Parameter:**

In the user interface following parameters are available (these are in the WIP only):

- Patient weight: used to calculate the BSA value for normalized values.
- Patient height: used to calculate the BSA value for normalized values.

**Control**

MrProtocol/MRXProtocol:

- InlineCardio-Switch on/off

**Restrictions**

During a cardio measurement with usage of InlineCardio Functor there are the following restrictions:

- All slices must be planned and acquired within a single protocol (Breathhold method). The developed automatic algorithm only works correctly, if the measurement data delivers back-to-back and without interrupts of the protocol.
- A measurement must start at base and finish at apex, because the template slice for locating the left ventricle position must be near the base. This slice must be segmented first.

**Output**

All images in the Pipeline will be forward with contour data to the next Functor. After acquire the last slice, the functor creates a “black image” with ejection fraction, end-diastolic volume, end-systolic volume and stroke volume values. At the end of the measurement these values represent the final estimation, will be displayed in InlineDisplay and stored in database. The result text is only available in English. The native language support (NLS) will not be supported in this version but can be implemented later.

## Anhang B - Makefile

### Software-Modul Inline Cardio

```
##-----  
## Copyright (C) Siemens AG 1998 All Rights Reserved. Confidential  
##-----  
##  
## Project: NUMARIS/4  
## File: \n4\pkg\MrServers\MrVista\Ice\IceApplicationFuncutors\IceInlineCardio\makefile.tr  
##-----  
##-----  
## enter include paths  
##  
CPPFLAGS(-DICEAPPLICATIONFUNCTIONS_EXPORTS)  
##-----  
## enter source files  
##  
##CPPSOURCES( InlineCardioFuncutor)  
CPPSOURCES2TARGET( InlineCardioFuncutor, IceInlineCardio)  
##-----  
## enter link libraries  
##  
LDLIBS(IceAlgos)  
LDLIBS(IceBasic)  
LDLIBS(MrParc)  
LDLIBS(IceUtils)  
LDLIBS(MrBasicObjects)  
LDLIBS(MrCardioAlgos)  
##-----  
## enter target name  
##  
LIB(IceInlineCardio)  
EVPTLB(IceInlineCardio)  
##-----  
## Copyright (C) Siemens AG 1998 All Rights Reserved. Confidential  
##-----  
VISTA_GCC_LINUX_AMD64(LDLIBS(stlport))  
VISTA_GCC_LINUX_AMD64(LDLIBS(ace))
```

**Software-Modul Segmentierungs-Algorithmus (SCR)**

```

##-----
## Copyright (C) Siemens AG 1999 All Rights Reserved. Confidential
##-----
##
## Project: NUMARIS/4
## File: \n4\pkg\MrServers\MrVista\MrEvaAlgos\MrCardioAlgos\makefile.trc
##-----

##-----
# enter include paths
INCLPATHS(-I /n4/pkg/MrServers/MrVista/include/)

##-----
## enter local compiler flags
##
##CPPFLAGS(-DREAD_CONFIG_FILE)
##CPPFLAGS(-DWRITE_IMAGES)
##CPPFLAGS(-DWRITE_IMAGES_DATA)

## source files for Argus Inline Cardio functionality
CPPSOURCESFROM( ArgusAutoCropping, Segmentation)
CPPSOURCESFROM( ArgusAutoLocalization, Segmentation)
CPPSOURCESFROM( ArgusAutoLocalizationUtils, Segmentation)
CPPSOURCESFROM( argusclustering, Segmentation)
CPPSOURCESFROM( ArgusDijkstra, Segmentation)
CPPSOURCESFROM( argusdijkstrautils, Segmentation)
CPPSOURCESFROM( ArgusEdge, Segmentation)
CPPSOURCESFROM( ArgusEnergyImage, Segmentation)
CPPSOURCESFROM( ArgusGlobalLocalization, Segmentation)
CPPSOURCESFROM( argusgloballocalizationparamsset, Segmentation)
CPPSOURCESFROM( argusgraph, Segmentation)
CPPSOURCESFROM( arguslocaldeformationparamsset, Segmentation)
CPPSOURCESFROM( ArgusLocalDeformations, Segmentation)
CPPSOURCESFROM( arguslocaldeformationsutils, Segmentation)
CPPSOURCESFROM( argusreadtraining, Segmentation)
CPPSOURCESFROM( ArgusRectGraph, Segmentation)
CPPSOURCESFROM( ArgusRegion, Segmentation)
CPPSOURCESFROM( ArgusSegmentation, Segmentation)
CPPSOURCESFROM( ArgusSegmentationParameters, Segmentation)
CPPSOURCESFROM( arguswarp, Segmentation)
CPPSOURCESFROM( ArgusYBVolume, Segmentation)
CPPSOURCESFROM( gamma, Segmentation)
CPPSOURCESFROM( PointMatching, Segmentation)

```

```
CPPSOURCESFROM( arguscontour,                BackEndCommon)
CPPSOURCESFROM( arguscontour_Critical,        BackEndCommon)
CPPSOURCESFROM( arguscontourspline,          BackEndCommon)
CPPSOURCESFROM( arguscurvefit_Critical,       BackEndCommon)
CPPSOURCESFROM( argusimageprocutils,         BackEndCommon)
CPPSOURCESFROM( argusmemoryutils,            BackEndCommon)
CPPSOURCESFROM( argusnumericalanalysis,      BackEndCommon)

##-----
## enter target name
##
LIB(MrCardioAlgos)

VISTA_GCC_LINUX_AMD64(LDLIBS(stlport))
VISTA_GCC_LINUX_AMD64(LDLIBS(ace))
```

## Anhang C - Software für abs() / fabs() Funktion

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

void main( void )
{
    double dx = -3.141593, dy;

    //Using of function abs( );
    dy = abs( dx );
    printf( "Using of function : abs( -3.141593 )\n" );
    printf( "The absolute value of %f is %f\n", dx, dy);

    //Using of function fabs( );
    dy = fabs( dx );
    printf( "\n\nUsing of function : fabs( -3.141593 )\n" );
    printf( "The absolute value of %f is %f\n", dx, dy );
}
```

## Anhang D - Inline Cardio Software-Modul

### InlineCardioFunctor.cpp - File

Für eine bessere Lesbarkeit und der Tatsache, dass der vorliegende Quellcode der Vertraulichkeit der Firma Siemens unterliegt, wird an dieser Stelle nur das Gerüst des **.cpp** Files mit den benötigten Funktionsrümpfen abgebildet.

```
// -----  
// Copyright (C) Siemens AG 1998 All Rights Reserved.  
// -----  
//  
// Project: NUMARIS/4  
// File: IceInlineCardio\InlineCardioFunctor.cpp  
// -----  
  
// InlineCardioFunctor  
#include "IceInlineCardio/InlineCardioFunctor.h"  
  
// Class InlineCardioFunctor  
InlineCardioFunctor::InlineCardioFunctor()  
    : m_Variable(0)  
{  
    addCB(IFirstCall);  
}  
  
InlineCardioFunctor::~~InlineCardioFunctor()  
{  
}  
  
IResult InlineCardioFunctor::EndInit( IParcEnvironment* env )  
{  
    return I_OK;  
}  
  
IResult InlineCardioFunctor::endOfJob(IResult reason)  
{  
    return I_OK;  
}  
  
IResult InlineCardioFunctor::FirstCall(IceAs& srcAs, MiniHeader::Pointer & srcHeader,  
ImageControl& srcCtrl)  
{  
    return I_OK;  
}
```

```
}

IResult InlineCardioFunctor::ComputeImage(IceAs& srcAs, MiniHeader::Pointer & srcHeader,
ImageControl& srcCtrl)
{
    return I_OK;
}

IResult InlineCardioFunctor::SegmentationAutoAdjust(int pPhaseNumber)
{
    return I_OK;
}

IResult InlineCardioFunctor::SegmentationUp(int pTemplatePhaseNumber, int pPhaseNumber)
{
    return I_OK;
}

IResult InlineCardioFunctor::SegmentationDown(int pTemplatePhaseNumber, int pPhaseNumber)
{
    return I_OK;
}

IResult InlineCardioFunctor::SegmentationTemporal(int pPhaseNumber, bool pStopAtES)
{
    return I_OK;
}

IResult InlineCardioFunctor::PropagateSegmentation(int pTemplatePhaseNumber, int pPhaseNumber, int
pParamsMode, bool pUseTemplateArray)
{
    return I_OK;
}

IResult InlineCardioFunctor::SegmentationReSegmentImage(int pPhaseNumber)
{
    return I_OK;
}

void InlineCardioFunctor::AutoCropping()
{
    return I_OK;
}

IResult InlineCardioFunctor::CreateOverlay (int pSliceNumber, int pPhaseNumber, IceAs& pOverlayAs,
MiniHeader::Pointer &srcHeader)
{

```

```
    return I_OK;
}

bool InlineCardioFunctor::ComputeDZVoxel(int pSliceNumber, double *dz)
{
    return false;
}

double InlineCardioFunctor::calcAreaPixels(int pPhaseNumber)
{
    return 0.0;
}

bool InlineCardioFunctor::CheckCentroids(int pPhaseNumber)
{
    return false;
}

bool InlineCardioFunctor::CheckSize(int pPhaseNumber)
{
    return false;
}

bool InlineCardioFunctor::CheckApicalSlices()
{
    return false;
}

IResult InlineCardioFunctor::CalculateValues(int pSliceNumber)
{
    return I_OK;
}

bool InlineCardioFunctor::ComputeBV(int pSliceNumber, double *dBloodVolume, double *dz)
{
    return false;;
}

IResult InlineCardioFunctor::SaveOriginals(IceAs& srcAs, MiniHeader::Pointer & srcHeader,
ImageControl& srcCtrl)
{
    return I_OK;
}

void InlineCardioFunctor::SetFailure(int pSliceNumber, int pFailureType)
{
}
```

```

BEGIN_OBJECT_MAP()
  OBJECT_ENTRY(InlineCardioFunctor)
END_OBJECT_MAP()

```

### InlineCardioFunctor.h – File

Das Header – File (.h) enthält die Deklarationen aller verwendeter Funktionen und Variablen.

```

// -----
// Copyright (C) Siemens AG 1998 All Rights Reserved.
// -----
//
// Project: NUMARIS/4
// File: IceInlineCardio\InlineCardioFunctor.h
// -----

#ifndef InlineCardioFunctor_h
#define InlineCardioFunctor_h 1

class ICEAPPLICATIONFUNCTIONS_API InlineCardioFunctor : public IceImageReconFunctors
{
public:

  ICE_IMAGE_RECON_FUNCTOR(InlineCardioFunctor)

  BEGIN_PROPERTY_MAP(InlineCardioFunctor)
  PROPERTY_DEFAULT(SequenceType, "MEAS.ucSequenceType")
  PROPERTY_ENTRY(PatientWeight)
  PROPERTY_ENTRY(PatientHeight)
  END_PROPERTY_MAP()

  DECL_GET_SET_PROPERTY( int, i, SequenceType)
  DECL_GET_SET_PROPERTY( double, d, PatientWeight)
  DECL_GET_SET_PROPERTY( double, d, PatientHeight)

  // -----
  // - constructor / destructor -
  // -----

```

```

InlineCardioFunctor();
virtual ~InlineCardioFunctor();

// -----
// - callbacks -
// -----
virtual IResult FirstCall(IceAs& srcAs, MiniHeader::Pointer & srcHeader, ImageControl& srcCtrl);

// -----
// - event sink -
// -----
virtual IResult EndInit( IParcEnvironment* env );
virtual IResult endOfJob(IResult reason);
virtual IResult ComputeImage(IceAs& srcAs, MiniHeader::Pointer & srcHeader, ImageControl&
srcCtrl);
virtual IResult SegmentationAutoAdjust(int pPhaseNumber);
virtual IResult SegmentationUp(int pTemplatePhaseNumber, int pPhaseNumber);
virtual IResult SegmentationDown(int pTemplatePhaseNumber, int pPhaseNumber);
virtual IResult SegmentationTemporal(int pPhaseNumber, bool pStopAtES = false);
virtual IResult PropagateSegmentation(int pTemplateName, int pPhaseNumber, int pParamsMode,
bool pUseTemplateArray = false);
virtual IResult SegmentationReSegmentImage(int pPhaseNumber);
virtual IResult CreateOverlay (int pSliceNumber, int pPhaseNumber, IceAs& pOverlayAs,
MiniHeader::Pointer &srcHeader);
virtual IResult CalculateValues(int pSliceNumber);
virtual IResult SaveOriginals(IceAs& srcAs, MiniHeader::Pointer & srcHeader, ImageControl&
srcCtrl);

virtual void AutoCropping();
virtual bool ComputeDZVoxel(int pSliceNumber, double *dz);
virtual bool ComputeBV(int pSliceNumber, double *dBloodVolume, double *dz);
virtual bool CheckCentroids(int pPhaseNumber);
virtual bool CheckSize(int pPhaseNumber);
virtual bool CheckApicalSlices();
virtual double calcAreaPixels(int pPhaseNumber);
virtual void SetFailure(int pSliceNumber, int pFailureType);

Crop m_crop;

protected:
bool m_bIsNonCritical;
bool m_bIsCritical;
int m_iFirstSlice;
int m_iLastSlice;
int m_iLastSegmentedSlice;
int m_iCOL;
int m_iLIN;

```

```

int m_iPHA;
int m_iSLC;
int m_iTrufispMode;
int m_iStyle;
int m_iParams;
int m_iImage;
int m_ED;
int m_ES;
int m_iTemplateSlice;
double m_dSpaceBetweenSlices;
double m_dSliceThickness;
double m_dPixelSpacingRow;
double m_dPixelSpacingCol;
float m_fCurrentRR;

int m_iLengthDim0;
int m_iLengthDim1;
IceDim m_Dim0;
IceDim m_Dim1;

std::vector<ArgusContour> m_vInnerContours, m_vOuterContours;
std::vector<ArgusContour> m_vInnerContoursTemplateSlice, m_vOuterContoursTemplateSlice;
std::vector<MiniHeader::Pointer> m_vIceMiniHead;
std::vector<ImageControl> m_vImageControl;
std::vector<short**> m_vImage;
std::vector<short**> m_vImageTemplateSlice;
std::vector<short**> m_vImageFirstSlice;
std::vector<double> m_vTriggerTime;
std::vector<double> m_vNoOfPixels;
std::vector<double> m_vEjectionFraction;
std::vector<double> m_vEndDiastolicVolume;
std::vector<double> m_vEndSystolicVolume;
std::vector<double> m_vStrokeVolume;
std::vector<ArgusSegmentationParameters> m_vSegmentationParameters;
std::vector<float> m_vRRInterval;

IceSoda m_FirstSliceSoda;
IceSoda m_CurrentSliceSoda;
IceFloatStorage::Pointer m_pRRInterval;

private:
    InlineCardioFunctor(const InlineCardioFunctor &right);
    InlineCardioFunctor & operator=(const InlineCardioFunctor &right);
};

#endif

```

## Anhang E - Metrik Lines of Code

Auflistung der für die Softwaremodule verwendeten Files .

### Modul Inline Cardio

<i>Anzahl</i>	<i>Filename</i>	<i>Lines of Code (LoC)</i>
2	InlineCardioFunctor.cpp	2416
	InlineCardioFunctor.h	156

### Modul ArgusAlgos

<i>Anzahl</i>	<i>Filename</i>	<i>Lines of Code (LoC)</i>
42	ArgusAutoCropping.cpp	1187
	ArgusAutoCropping.h	74
	ArgusAutoLocalization.cpp	488
	ArgusAutoLocalization.h	329
	ArgusAutoLocalizationUtils.cpp	1592
	ArgusClustering.cpp	471
	ArgusClustering.h	385
	ArgusDijkstra.cpp	432
	ArgusDijkstra.h	295
	ArgusDijkstraUtils.cpp	625
	ArgusEdge.cpp	406
	ArgusEdge.h	80
	ArgusEnergyImage.cpp	755
	ArgusEnergyImage.h	331
	ArgusGlobalLocalization.cpp	763
	ArgusGlobalLocalization.h	245
	ArgusGlobalLocalizationParamsSet.cpp	93
	ArgusGlobalLocalizationParamsSet.h	322
	ArgusGraph.cpp	1022
	ArgusGraph.h	267
	ArgusLocalDeformationParamsSet.cpp	134
	ArgusLocalDeformationParamsSet.h	388
	ArgusLocalDeformations.cpp	1084
	ArgusLocalDeformations.h	372
	ArgusLocalDeformationsUtils.cpp	791
	ArgusReadTraining.cpp	12085

<i>Anzahl</i>	<i>Filename</i>	<i>Lines of Code (LoC)</i>
	ArgusRectGraph.cpp	998
	ArgusRectGraph.h	387
	ArgusRegion.cpp	325
	ArgusRegion.h	60
	ArgusSegmentation.cpp	2176
	ArgusSegmentation.h	452
	ArgusSegmentationParameters.cpp	485
	ArgusSegmentationParameters.h	292
	ArgusWarp.cpp	3078
	ArgusYBVolume.cpp	284
	ArgusYBVolume.h	173
	Gamma.cpp	133
	PointMatching.cpp	951
	PointMatching.h	99
	f2c.h	254
	RogueWaveEmulate.h	29
19	ArgusContour.cpp	2189
	ArgusContour.h	161
	ArgusConstants.h	84
	ArgusContour_Critical.cpp	448
	ArgusContourSpline.cpp	779
	ArgusCurvefit_Critical.cpp	1318
	ArgusCurvefit.h	39
	ArgusImageProcUtils.cpp	542
	ArgusImageProcUtils.h	42
	ArgusMemoryUtils.cpp	407
	ArgusMemoryUtils.h	48
	ArgusNLSUtils.h	87
	ArgusNonlinearCurvefit.cpp	1556
	ArgusNonlinearCurvefit.h	48
	ArgusNumericalAnalysis.cpp	160
	ArgusNumericalAnalysis.h	37
	ArgusPatientDataUtils.cpp	96
	ArgusPatientDataUtils.h	54
	ArgusPatientDataUtils_Critical.cpp	64

## **Selbständigkeitserklärung**

Hiermit erkläre ich, dass ich die vorliegende Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und erlaubten Hilfsmittel benutzt habe. Weiter erkläre ich, die Diplomarbeit in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt zu haben.

**Magdeburg, 18. April 2007**

**Matthias Ludewig**

